Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming

Virgolin, Marco

DOI
10.4233/uuid:03641b5f-f8f6-4ff9-be7f-11948f6d3cc7

Publication date
2020

Document Version
Final published version

Citation (APA)
Virgolin, M. (2020). *Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming*. [Dissertation (TU Delft), Delft University of Technology].
https://doi.org/10.4233/uuid:03641b5f-f8f6-4ff9-be7f-11948f6d3cc7

Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming

Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof.dr.ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended publicly on
Monday 8 June 2020 at 12:30 o'clock

by

Marco Virgolin

Master of Science in Computer Engineering, University of Trieste, Italy,
born in Monfalcone, Italy.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. P.A.N. Bosman,	Delft University of Technology, promotor
Prof. dr. C. Witteveen,	Delft University of Technology, promotor
Dr. T. Alderliesten,	Leiden University Medical Center / Delft University of Technology, copromotor
Independent members:	
Prof. dr. R. Babŭska	Delft University of Technology
Prof. dr. K. Krawiec	Poznan University of Technology, Poland
Dr. U.-M. O'Reilly	Massachusetts Institute of Technology, USA
Prof. dr. L.J.A. Stalpers	Amsterdam University Medical Centers, University of Amsterdam

The research reported in this dissertation was funded by Stichting Kinderen Kankervrij (KiKa), with project No. 187.

SIKS Dissertation Series No. 2020-13.
The research reported in this dissertation has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Keywords: evolutionary algorithms, genetic programming, machine learning, pediatric cancer, radiotherapy

Printed by: Ipskamp Printing

Front & Back: Cover art by Macs Gallo (https://artstation.com/macs-gallo).

ISBN 978-94-6384-138-2

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

We have a hunger of the mind which asks for knowledge of all around us, and the more we gain, the more is our desire; the more we see, the more we are capable of seeing.

Maria Mitchell

Contents

Summary

Machine learning is impacting modern society at large, thanks to its increasing potential to efficiently and effectively model complex and heterogeneous phenomena. While machine learning models can achieve very accurate predictions in many applications, they are not infallible. In some cases, machine learning models can deliver unreasonable outcomes. For example, deep neural networks for self-driving cars have been found to provide wrong steering directions based on the lighting conditions of street lanes (e.g., due to cloudy weather). In other cases, models can capture and reflect unwanted biases that were concealed in the training data. For example, deep neural networks used to predict likely jobs and social status of people based on their pictures, were found to consistently discriminate based on gender and ethnicity–this was later attributed to human bias in the labels of the training data.

The aforementioned issues typically concerned so-called black-box models, i.e., machine learning models which are too complex to be explained, such as, in fact, deep neural networks. Consequently, scientists and policy makers have increasingly started to agree that, for a responsible use of machine learning and Artificial Intelligence (AI), it is important to be able to *explain why* a model behaves the way it does: to have explanations about the reasoning of a model enables to track potential issues, and solve them. Therefore, algorithms are needed that can help explain why a model behaves in a certain why, or that can directly generate models that are human-interpretable.

Genetic Programming (GP) is a meta-heuristic that can be used to generate machine learning models in the form of human-readable computer programs, i.e., sequences of program instructions. GP algorithms work by stochastic search inspired by natural evolution. A population of random programs is iteratively evolved by recombining instructions into new programs, and by survival of the fittest, i.e., discarding the worst performing programs in the population. The program instructions are typically human-written and human-interpretable. This fact enables the possibility that the entire program is human-interpretable as well. In an attempt to increase the chances of obtaining human-interpretable programs, the work presented in this thesis is mostly focused on scenarios where programs need to contain a limited number of instructions.

While there is promise in using GP to obtain interpretable machine learning models, GP algorithms typically fall short in terms of efficiency when compared to many other machine learning algorithms. A major cause of inefficiency can be attributed to how the search steps are performed, i.e., the way program instructions are recombined, and what mechanisms are in place to keep good programs and discard bad programs. In particular, the recombination of instructions into new programs is typically done randomly and without any adaptive method to improve the effectiveness of recombination over time.

Recent research in GP has attempted to improve the speed and quality of the search. Most successful methods to date, however, achieve improvements by (repeatedly) stacking relatively large blocks of instructions. This leads to obtaining programs so large that any

chance of human-interpretability is ultimately lost. So, currently, a gap still exists: designing competent search mechanisms for GP that focus on obtaining programs of restricted size. This then immediately leads to the main goal of this thesis: **improving GP by the design and application of algorithms that perform more efficient and effective search, particularly when the total number of instructions needs to be limited**.

To reach our main goal, concepts of modern model-based evolutionary algorithms called *Optimal Mixing Evolutionary Algorithms* (OMEAs) from discrete optimization are brought to GP, and tested on benchmark and real-world problems. OMEAs are a type of EAs that are of particular interest because in these EAs recombination is configured to dynamically adapt based on information that emerges during the search, so as to improve efficiency and effectiveness. More specifically, OMEAs attempt to learn, on-line, what building blocks of solution components (in the case of programs: what instructions) belong together and should be preserved during recombination. By identifying and recombining building blocks, OMEAs can obtain knock-on effects in performance. This has already enabled OMEAs in other domains than GP to quickly solve high-dimensional problems that other EAs cannot solve in a reasonable time.

This thesis advances the state of knowledge about GP by presenting the following major contributions:

1. A new GP algorithm is introduced called *GP-GOMEA*, which builds upon the *Gene-pool Optimal Mixing Evolutionary Algorithm* (GOMEA) that was originally introduced for discrete optimization. The search procedure in GP-GOMEA is dynamically adapted by identifying what program instructions are interdependent and potentially constitute building blocks, and by subsequently recombining building blocks (Chapter 2).

2. Limitations of GP-GOMEA for supervised learning problems of non-trivial dimensionality (specifically for symbolic regression) are presented and tackled by proposing improvements that enable GP-GOMEA to also work well in these scenarios (Chapter 3). We further show that another type of GP algorithm (using so-called *semantic backpropagation-based approximately geometric variation*) does not scale to realistic symbolic regression problems, and propose improvements that overcome this (Chapter 4).

3. Beyond the use of GP-GOMEA to directly synthesize interpretable machine learning models, we consider the possibility to combine GP-GOMEA (and other GP algorithms) with another machine learning algorithm. We study whether models that different machine learning algorithms can generate can be made to have a higher chance of being explainable without incurring a significant performance loss by changing the feature space that the models are trained upon. In particular, we use GP-GOMEA and other search algorithms to automatically construct few salient and small features. We show that for several classification and regression problems and machine learning algorithms, it is in fact possible to construct features that enable achieving similar performance with the same machine learning algorithms. In some cases, performance can even improve. Furthermore, because discovered features are particularly small, they are themselves likely to be interpretable (we provide examples). Moreover, because we focus on finding particularly few (i.e., two) features,

it becomes possible to plot and visualize the predictions of the machine learning model, and hence obtain a comprehensive and intuitive representation of its behavior (Chapter 5).

4. We finally use GP-GOMEA to synthesize regression models in the form of readable mathematical expressions for a problem of real-world interest. In particular, we consider the estimation of radiation dose delivered to long-term childhood cancer survivors who were subject to radiation therapy when no 3D anatomy imaging was yet introduced in clinical practice. Obtaining 3D estimations (or related metrics) of the dose to (subvolumes of) organs is important to be able to study how radiation relates to adverse effects that appear decades after the treatment. Unfortunately, 3D dose estimations cannot be obtained in a straightforward manner because of the lack of 3D anatomy imaging.

   First, we study the feasibility of applying machine learning for the goal of estimating 3D anatomical metrics using scarce information available from patient records and 2D radiographs (Chapter 6). Second, we develop a method capable of generating a surrogate 3D anatomy for a patient, given again scarce information. This pipeline internally employs machine learning models to predict, using a database of 3D organ segmentations and CT scans, how to assemble a personalized 3D surrogate anatomy. GP-GOMEA is compared with other GP algorithms and machine learning algorithms of a different nature, as well as with state-of-the-art heuristics for surrogate anatomy construction. GP-GOMEA is found to deliver overall the most accurate models, which are arguably likely to be interpretable for many people (Chapter 7). Finally, alongside information on the patient, we propose to also include information about the treatment plan to be used as input features. By doing so, we show that it is possible to use GP-GOMEA to find models capable of directly predicting 3D dose-volume metrics useful for the study of adverse effects, without the need of using a surrogate anatomy (Chapter 8).

Essentially, this thesis shows that leveraging key principles of OMEAs can lead to more efficient and effective discovery of GP programs. Moreover, OMEAs can find programs that perform well while being particularly compact in terms of number of instructions. We show that this is generally not the case for other state-of-the-art GP algorithms, and we provide concrete results on real-world symbolic regression problems, including a clinical application.

We conclude that OMEAs for GP can be considered to be an important method for the automatic synthesis of small, and thus likely to be interpretable, machine learning models. Therefore, these algorithms have the potential to bring explainable machine learning models into practice in many sensitive applications of societal interest.

Samenvatting

Machine learning heeft invloed op de moderne samenleving als geheel, dankzij de toenemende potentie om complexe en heterogene fenomenen efficiënt en effectief te modelleren. Hoewel machine learning-modellen in veel toepassingen zeer nauwkeurige voorspellingen kunnen doen, zijn ze niet onfeilbaar. In sommige gevallen kunnen machine learning-modellen onwenselijke resultaten opleveren. Er is bijvoorbeeld vastgesteld dat diepe neurale netwerken voor zelfrijdende auto's tot verkeerde stuuracties kunnen leiden, afhankelijk van de lichtomstandigheden op de rijbaan (bijvoorbeeld vanwege bewolkt weer). In andere gevallen kunnen modellen ongewenste vooroordelen vastleggen en weerspiegelen die in de trainingsgegevens waren verborgen. Bijvoorbeeld, diepe neurale netwerken die werden gebruikt om te voorspellen wat waarschijnlijk de baan en sociale status van mensen zijn op basis van hun foto's, bleken consistent te discrimineren op basis van geslacht en etniciteit - dit werd later toegeschreven aan menselijke vooringenomenheid in de labels van de trainingsgegevens.

De bovengenoemde kwesties betroffen typisch zogenaamde *black-box*-modellen, die te complex zijn om te worden verklaard, zoals in feite diepe neurale netwerken. Hierdoor zijn wetenschappers en beleidsmakers het er in toenemende mate over eens geworden dat het voor een verantwoord gebruik van machine learning en *Artificial Intelligence* (AI) belangrijk is om te kunnen *verklaren waarom* een model zich op een bepaalde manier gedraagt: het kunnen geven van een verklaring van de redenering van een model maakt het mogelijk potentiële problemen op te sporen en op te lossen. Daarom zijn algoritmen nodig die kunnen helpen verklaren waarom een model zich op een bepaalde manier gedraagt, of die direct modellen kunnen genereren die door mensen kunnen worden geïnterpreteerd.

Genetic Programming (GP) is een meta-heuristiek die kan worden gebruikt om machine learning-modellen te genereren in de vorm van door mensen leesbare computerprogramma's, ofwel reeksen programma-instructies. GP-algoritmen werken door middel van stochastisch zoeken, geïnspireerd op natuurlijke evolutie. Een populatie van willekeurige programma's wordt iteratief geëvolueerd door instructies te combineren om zo tot nieuwe programma's te komen, en door het toepassen van het paradigma dat de sterksen overleven, dat wil zeggen, het verwijderen van de slechtst presterende programma's in de populatie. De programma-instructies zijn meestal door mensen geschreven en door mensen interpreteerbaar. Dit feit maakt het mogelijk dat het hele programma ook door mensen interpreteerbaar is. In een poging tot het vergroten van de kansen op het verkrijgen van door mensen interpreteerbare programma's, is het werk dat in dit proefschrift wordt gepresenteerd voornamelijk gericht op scenario's waarin programma's een beperkt aantal instructies moeten bevatten.

Hoewel het veelbelovend is om GP te gebruiken voor het verkrijgen van interpreteerbare modellen voor machine learning, schieten GP-algoritmen doorgaans tekort in termen van efficiëntie in vergelijking met veel andere algoritmen voor machine learning. Een belangrijke oorzaak van die inefficiëntie kan worden toegeschreven aan de manier waarop

zoekstappen worden uitgevoerd, dat wil zeggen, de manier waarop programma-instructies worden gecombineerd en welke mechanismen er zijn om goede programma's te behouden en slechte programma's te verwijderen. In het bijzonder wordt de constructie van nieuwe programma's middels recombinatie van instructies in bestaande programma's typisch willekeurig gedaan en zonder enige adaptieve methode om de effectiviteit van recombinatie te verbeteren.

In recent onderzoek op het gebied van GP is geprobeerd de snelheid en kwaliteit van het zoekproces te verbeteren. In de meest succesvolle methoden tot nu toe wordt dit bereikt door (herhaaldelijk) relatief grote instructies aan elkaar te knopen. Dit leidt tot programma's die zo groot zijn dat elke kans op menselijke interpreteerbaarheid uiteindelijk verloren gaat. Momenteel bestaat er dus nog steeds een kloof: het ontwerpen van competente zoekmechanismen voor GP die gericht zijn op het verkrijgen van programma's van beperkte omvang. Dit leidt meteen tot het hoofddoel van dit proefschrift: **GP verbeteren door het ontwerp en de toepassing van algoritmen die efficiënter en effectiever zoeken, met name wanneer het totale aantal instructies moet worden beperkt**.

Om ons hoofddoel te bereiken, worden concepten van moderne modelgebaseerde evolutionaire algoritmen genaamd *Optimal Mixing Evolutionary Algorithms* (OMEA's) uit discrete optimalisatie naar GP gebracht en getest op benchmark- en praktijkproblemen. OMEA's zijn een type EA's die in het bijzonder van belang zijn omdat in deze EA's recombinatie is geconfigureerd om zich dynamisch aan te passen op basis van informatie die tijdens het zoeken naar voren komt, gericht op het verbeteren van de efficiëntie en effectiviteit. Meer specifiek proberen OMEA's tijdens het zoekproces te leren welke bouwstenen in de vorm van meendere oplossingscomponenten (in het geval van programma's: welke instructies) bij elkaar horen en bewaard moeten blijven tijdens recombinatie. Het identificeren en combineren van bouwstenen vindt zijn doorslag in OMEA's in de vorm van verhoodge efficiëntie. Dit heeft OMEA's voor andere domeinen dan GP reeds in staat gesteld om hoog-dimensionale problemen snel op te lossen die andere EA's niet binnen een redelijke tijd kunnen oplossen.

Dit proefschrift bevordert de kennis op het gebied van GP door de volgende belangrijke bijdragen te presenteren:

1. Er wordt een nieuw GP-algoritme geïntroduceerd met de naam *GP-GOMEA*, dat voortbouwt op het *Gene-pool Optimal Mixing Evolutionary Algorithm* (GOMEA) dat oorspronkelijk werd geïntroduceerd voor discrete optimalisatie. De zoekprocedure in GP-GOMEA wordt dynamisch aangepast door te identificeren welke programma-instructies van elkaar afhankelijk zijn en mogelijk bouwstenen vormen, en door vervolgens bouwstenen te combineren (Hoofdstuk 2).

2. Beperkingen van GP-GOMEA voor gesuperviseerde leerproblemen van niet-triviale dimensionaliteit (specifiek voor symbolische regressie) worden gepresenteerd en aangepakt door verbeteringen voor te stellen waardoor GP-GOMEA ook goed kan werken in deze scenario's (Hoofdstuk 3). We laten verder zien dat een ander type GP-algoritme (met behulp van zogenaamde *semantic backpropagation-based approximately geometric variation*) niet schaalt naar realistische symbolische regressieproblemen en stellen verbeteringen voor die dit oplossen (Hoofdstuk 4).

3. Naast het gebruik van GP-GOMEA om interpreteerbare machine learning-modellen rechtstreeks te synthetiseren, bekijken we de mogelijkheid om GP-GOMEA (en andere GP-algoritmen) te combineren met een ander machine learning-algoritme. We onderzoeken of modellen die door verschillende machine learning-algoritmen kunnen worden gegenereerd, een grotere kans hebben om verklaard te worden zonder een aanzienlijk prestatieverlies op te lopen door de kenmerkenruimte (*feature space*) waarop de modellen zijn getraind te wijzigen. In het bijzonder gebruiken we GP-GOMEA en andere zoekalgoritmen om automatisch enkele opvallende en kleine kenmerken te bouwen. We laten zien dat het voor verschillende classificatie- en regressieproblemen en machine learning-algoritmen in feite mogelijk is om kenmerken te bouwen die vergelijkbare prestaties met dezelfde machine learning-algoritmen mogelijk maken. In sommige gevallen kunnen de prestaties zelfs verbeteren. Omdat ontdekte kenmerken bijzonder klein zijn, zijn ze zelf waarschijnlijk interpreteerbaar (we geven voorbeelden). Omdat we ons richten op het vinden van bijzonder weinig (bijvoorbeeld twee) kenmerken, maken ze het bovendien mogelijk om de voorspellingen van het machine learning-model in kaart te brengen en te visualiseren, en dus een uitgebreide en intuïtieve weergave van zijn gedrag te verkrijgen (Hoofdstuk 5).

4. Tenslotte gebruiken we GP-GOMEA om regressiemodellen te synthetiseren in de vorm van leesbare wiskundige uitdrukkingen voor een probleem van maatschappelijk belang. In het bijzonder beschouwen we de schatting van de stralingsdosis langdurig overlevenden van kinder kanker werden bloot gesteld vanwege een behandeling met radiotherapie toen er nog geen 3D-anatomische beeldvorming was geïntroduceerd in de klinische praktijk. Het verkrijgen van 3D-schattingen (of gerelateerde meetwaarden) van de dosis aan (subvolumes) van organen is belangrijk om te kunnen bestuderen hoe straling verband houdt met nadelige effecten die tientallen jaren na de behandeling optreden. Helaas kunnen schattingen van 3D-doses niet op een eenvoudige manier worden verkregen vanwege het ontbreken van 3D-beeldvorming van de anatomie.

   Eerst bestuderen we de haalbaarheid van het toepassen van machine learning met als doel het schatten van 3D-anatomische metrieken met behulp van schaarse informatie die beschikbaar is uit patiëntendossiers en 2D-röntgenfoto's (Hoofdstuk 6). Dan ontwikkelen we een methode die in staat is om een surrogaat 3D-anatomie voor een patiënt te genereren, opnieuw gegeven schaarse informatie. De bijkehorende pijplijn maakt intern gebruik van machine learning-modellen om met behulp van een database van 3D-orgaansegmentaties en CT-scans te voorspellen hoe een gepersonaliseerde 3D-surrogaatanatomie wordt samengesteld. GP-GOMEA wordt vergeleken met andere GP-algoritmen en machine learning-algoritmen van een andere aard, evenals met state-of-the-art heuristieken voor surrogaatanatomieconstructie. GP-GOMEA blijkt over het algemeen de meest nauwkeurige modellen te leveren, die waarschijnlijk voor veel mensen interpreteerbaar zijn (Hoofdstuk 7). Ten slotte stellen we voor om naast informatie over de patiënt ook informatie over het behandelplan op te nemen ter gebruik als kenmerken. Door dit te doen, laten we zien dat het mogelijk is om GP-GOMEA te gebruiken om modellen te vinden die in staat zijn om 3D-dosisvolumemetingen, die nuttig zijn voor de studie van nodelige effecten,

zonder de noodzaak om een surrogaatanatomie te gebruiken, direct te voorspellen (Hoofdstuk 8).

Dit proefschrift laat in essentie zien dat het gebruik van sleutelprincipes onderliggend aan OMEA's kan leiden tot een efficiëntere en effectievere ontdekking van GP-programma's. Bovendien kunnen OMEA's programma's vinden die goed presteren en toch bijzonder compact zijn in termen van aantal instructies. We laten zien dat dit over het algemeen niet het geval is voor andere state-of-the-art GP-algoritmen, en we bieden concrete resultaten voor symbolische regressieproblemen uit de praktijk, waaronder een klinische toepassing.

We concluderen dat OMEA's voor GP kunnen worden beschouwd als een belangrijke methode voor de automatische synthese van kleine, en dus waarschijnlijk interpreteerbare, machine learning-modellen. Daarom hebben deze algoritmen de potentie om verklaarbare modellen voor machine learning in de praktijk te brengen in gevoelige toepassingen die van maatschappelijk belang zijn.

1

Introduction

Machine learning is changing the world. Its applications range from commodities to improve one's comfort and entertainment, to crucial decision support for healthcare and finance. This thesis regards a particular form of machine learning: Genetic Programming (GP). GP is interesting because it has the possibility to create human-understandable machine learning models. Enabling human understanding is important to gain new knowledge as well as to prevent undesirable consequences. However, GP is computationally expensive. This chapter introduces the main goal of this thesis: improving the efficiency and effectiveness of GP. Firstly, an introduction to the need for explanations in machine learning, and for more efficient and effective GP algorithms, is presented (Sec. 1.1). Next, a learning task that mostly recurs in this thesis is provided, i.e., symbolic regression (Sec. 1.2). The workings of classic GP are then described, together with an example of its application for a simple symbolic regression problem (Sec. 1.3). Reasons are given as to why GP can be considered computationally expensive, and a key aspect of GP that could be improved to overcome this limitation is presented: variation*, i.e., the way GP takes search steps in the space of programs (Sec. 1.4). In the same section a small review on the state-of-the-art with respect to variation in GP is given, along with respective limitations, which motivate the research described in this thesis. Finally, the research questions that constitute the stepping stones of this thesis are presented (Sec. 1.5). Of these, the first half concerns the design of a new GP algorithm, and its tailorization to deal with symbolic regression problems; the second half concerns the application of such algorithms, to shed light on otherwise unintelligible machine learning models, as well as to find transparent machine learning models for a clinical application.*

1

1.1. Machine learning and the need for explanations

Machine learning is a broad term that stands for the study and application of algorithms that can infer, or "*learn*", how to perform a task automatically, in contrast to be explicitly programmed for it [1]. As such, machine learning has revolutionized the way humans can tackle the modeling of complex phenomena. Until a few decades ago, people interested in modeling a phenomenon could only rely on their ingenuity, and would first need to understand the phenomenon in depth. Now, in many situations, this is no longer necessary: powerful algorithms can automatically detect subtle, non-linear patterns from data, and infer accurate models for us [2].

Modern machine learning has been found to be competitive with, or even superior to, human performance in many applications. Examples range from medical applications (detection of skin cancer [3], detection of Parkinson's dyskinesia [4]) to natural language processing (text generation [5], synthesis of regular expressions [6]), from software and electrical engineering (Android apps crash correction [7], large circuit synthesis [8]) to gaming (mastering the game of Go [9], playing Atari games [10]).

Due to its appeal and practical usefulness, machine learning is pervading society rapidly and vastly, and affects the daily life of virtually everyone in the civilized world. Popular hand held devices such as smartphones and smartwatches are coming with all sort of machine learning-based enhancements, such as vocal assistants, face recognition, and camera super resolution[1]. What content and what advertisements are proposed on social media are tailored automatically by machine learning-based profiling [11]. The transportation industry is investing in machine learning to shape what the transportation of goods and people will look like in the future [12, 13]. Machine learning models are also becoming more popular in finance, health care, and even criminal justice, to suggest, respectively, what people are reliable for loan granting, what particular treatment should be administered to whom, and who is likely to have committed a crime [14–16].

Modeling enabled by machine learning is thus having considerable societal impact. However, not all of this impact is necessarily positive. In recent years, scientists, practitioners, and policy makers alike, are becoming increasingly concerned about possible misuses of this powerful technology [14, 17–19]. For example, in social profiling, models trained upon biased data that discriminate against particular groups of people may reflect these discriminations in their predictions, and accentuate the problem [20, 21]. Similarly, in health care, models that are not sufficiently comprehensive because they were trained on a small sample, or on outdated information, may provide suggestions that could end up being harmful for patients [22]. Because of these sorts of concerns, there is a wide agreement that it is important that automatic decision support systems, many of which increasingly adopt machine learning models, provide *explanations* of how and why they reach particular outcomes. In other words, there is a growing need to enable human-interpretability when dealing with processes that rely upon machine learning [23].

Ideally, one could acquire explanations of the predictions of a machine learning model by inspecting the model itself, and by following the logic wired into it [24]. However, machine learning models can be very complex, to the point of becoming unintelligible, and receiving the appellation "black boxes" [15, 24]. Very popular machine learning models

[1] `https://www.techradar.com/news/what-does-ai-in-a-phone-really-mean`

such as ensembles of decisions trees [25, 26] and neural networks [27] are typically considered to be such black boxes [15, 22]. The former typically builds (at least) hundreds of decision trees which, if taken singularly, could in principle be interpreted. Yet, due to the sheer number of these trees, it is essentially impossible to understand the joint effect of the ensemble. Classic neural networks such as multi-layer perceptrons as well as modern convolutional networks for image recognition normally use a very small number of non-linear function types, or even a single one (e.g., the rectified linear unit [28]). These functions are instanced in multiple network nodes, which in turn are arranged into layers that are densely connected to each other by weighted edges. The number of weighted edges can range from thousands to billions, making it impossible for a human to understand how the computations that are performed, relate to outcomes.

While the flexibility of neural networks comes from using a massive number of weights in conjunction with a few types of non-linear functions, it can be imagined that the number of weights can be reduced if more types of functions are adopted (linear and non-linear). In other words, model flexibility might arise from being able to instantiate a wide range of function compositions. For example, if summation, multiplication, division, and constant scalars are provided, an outcome of such a procedure could be a Taylor approximation [29]. Such models can be very interesting with respect to the need for explainable machine learning. In fact, if the functions to be composed are human-interpretable, and the compositions are not excessively involved, then the entire model may well be interpretable. In this light, Genetic Programming (GP) represents an interesting class of algorithms, because it precisely operates by automating the composition of human-provided functions [30, 31].

Since its popularization in the early '90s, GP has been proven to be a competitive approach to other machine learning algorithms [32], and has led to several creative, and sometimes unexpected, outcomes: many human-competitive results have been obtained by GP so far[2] [33]. Moreover, GP has been found to be capable of delivering human-interpretable programs [29, 34], and contemporary surveys on explainable machine learning list GP among the types of algorithm that can shed light on machine learning processes, by either inferring understandable models directly, or by approximating and explaining black box models [15, 24].

The potential of GP to search and discover understandable machine learning models comes, however, with a notable drawback: finding the best instruction (or function) composition is a non-convex optimization problem with many symmetries and a priori unspecified dimensionality, which often requires a large amount of computation effort to achieve results on par with other popular machine learning algorithms (see Sec. 1.4). For this reason, the main goal of this thesis is to explore the design of GP algorithms that search for programs (or machine learning models) in a more efficient and effective manner. Since having a small number of instructions can be considered a necessary condition to improve the chance of human-interpretability, focus is put on restraining the number of instructions to compose GP programs with. Furthermore, in this thesis the application of the designed GP algorithms is explored in two ways. First, the capability of GP to work in synergy with other machine learning algorithms is studied, in an attempt to obtain more explainable models from those machine learning algorithms. Second, the designed

[2] http://www.human-competitive.org

GP algorithms are used for a real-world clinical problem concerning radiation dose reconstruction for childhood cancer survivors. The aim is to obtain machine learning models that perform well and that, if desired, can be inspected to understand more about their behavior.

1.2. Symbolic regression

Symbolic regression is a fundamental machine learning problem that recurs in this thesis. Symbolic regression benchmark problems are considered in contributions related to the design of a new GP algorithm, and tasks concerning the clinical application will be cast to symbolic regression problems. Before delving into what *symbolic* means in this context, a short introduction to traditional regression follows.

1.2.1. Traditional regression

Regression is the problem of identifying relationships between variables, i.e., how one (or more) variable can be expressed as a function of one (or more) other variable. We consider the case where one variable can be expressed in terms of several other variables. Let y be a variable that is *believed* to depend on some other m variables $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$. Each $x^{(j)}$ ($j = 1, \ldots, m$) is called an independent variable, or feature, and y is called the dependent variable, or target. In regression, data is available in the form of n samples of features and of the target: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{x}_i = \{x_i^{(1)}, \ldots, x_i^{(m)}\} \in \mathbb{R}^m$, and $y_i \in \mathbb{R}$.

Let f be a function form that is desired to be used to capture the underlying relationship between $\mathbf{x}$ and y (e.g., linear, quadratic, logarithmic). The function f is defined in terms of a collection of k free parameters $\theta \in \mathbb{R}^k$. Regression concerns finding the optimal collection of parameter values $\theta^\star$, such that the approximation $y \approx f(\mathbf{x}, \theta^\star)$ is as good as possible.

To evaluate the quality of candidate parameter values θ, a loss function (or cost function) $\mathcal{L}$ is employed that measures the distance between y and $f(\mathbf{x}, \theta)$, e.g.:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} |y_i - f(\mathbf{x}_i, \theta)|^p. \tag{1.1}$$

Typical values of p are 1 and 2, with the latter choice penalizing larger errors more than the former. Formally, $\mathcal{L}$ depends on the data $\mathbf{x}_i, y_i$, and the chosen f as well as on θ. Here we consider only θ to be an argument of $\mathcal{L}$ to highlight the fact that only θ is subject to optimization (while the other variables are fixed).

Since the number of observations n can be relatively small with respect to the complexity of the chosen f, finding the true minimum of a loss function may not be the best choice. A particular phenomenon to be aware of is *overfitting*, i.e., the possibility that the inferred $f(\mathbf{x}, \theta)$ nicely fits the original n observations, but will not fit well new observations, that come from the same source distribution of the first n. To detect and combat overfitting, appropriate validation (e.g., assessing the loss on data that was held-out during optimization) and regularization (e.g., using the L1 norm of the weights and/or using early stopping) may be needed [1].

1.2.2. From traditional to symbolic regression

Hypotheses regarding the appropriate form of f to use for a particular application can be hard to make. For example, prior knowledge on the relationship between features and the target may not be available, or hard to infer due to, e.g., having too many features (i.e., dimensions) to allow direct plotting and visualization of the data [30, 35]. Symbolic regression aims at tackling this issue, by attempting to recover the entirety of f at once (and not only a collection of some real-valued parameters θ) [30]. In other words, symbolic regression entails finding the optimal $f^\star$ in a (sub)space of functions $\mathfrak{F}$. A loss function for symbolic regression can be formulated as:

$$\mathcal{L}(f) = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(\mathbf{x}_i)|^p. \tag{1.2}$$

In this formulation, f is subject to optimization in its entirety. A collection of parameters θ is not explicitly stated because these parameters are part of f, i.e., we consider two functions that differ in some scalar coefficients to be different functions from $\mathfrak{F}$.

Symbolic regression algorithms attempt to discover the entire formula from scratch, starting from pre-defined functions (including constant functions, potentially initialized at random) called *primitives*, that are provided by the user. These algorithms combine the primitives and optimize θ to form candidate functions f, which are evaluated according to a loss similar to Equation 1.2. It is clear that the search space of symbolic regression is necessarily larger than the one of traditional regression. Similar to traditional regression, ways to detect and prevent overfitting need to be applied as well.

Although different types of symbolic regression algorithms exist (e.g., [29, 36, 37]), the most common algorithms are forms of GP [32]. In the following section, the main characteristics of classic GP are described, in particular to tackle symbolic regression problems.

1.3. Classic genetic programming

GP is a popular metaheuristic for the automatic synthesis of programs (or, equivalently, computable functions), typically from examples of desired behavior [30, 31]. Once the set of primitives and a loss function, in GP called the *fitness function*, have been defined, GP synthesizes programs by loosely mimicking the concept of Darwinian evolution, i.e., by iterative *selection* and *variation* of a *population* of programs [31]. Selection represents survival of the fittest, to promote the proliferation of promising programs. Next, in the variation phase, offspring programs are created by changing the order, position, and type of parent programs' instructions. These iterations of selection and variation are called *generations*. Figure 1.1 shows a typical evolution scheme for GP.

Different types of GP algorithms exist, where the way programs are represented is specific to that algorithm. In this thesis, the classic and most popular type of representation is considered: the tree-based encoding [31]. To illustrate how this representation works, let us assume to need a program that encodes Newton's well-known law of gravitation [38]:

$$F = G\frac{m_1 m_2}{r^2}, \tag{1.3}$$

where F is the force of gravity, G is the gravitational constant, m_1 and m_2 are the masses of two bodies, and r is the distance between them. An example of a possible tree-based

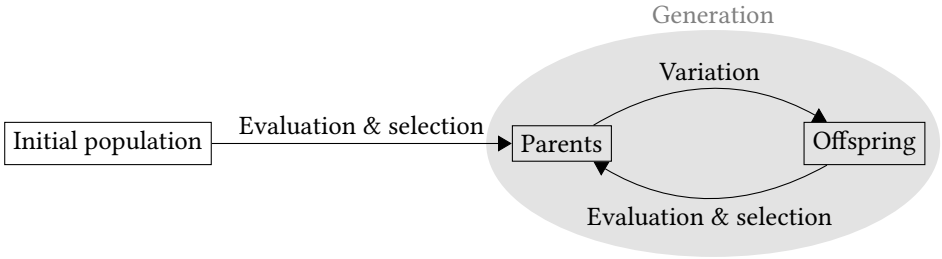

Figure 1.1: High-level illustration of typical workings of GP. First, an initial population of programs is sampled at random. The fitness of each program is evaluated, and the programs that are currently most successful are selected with larger probability to enter the pool of parent programs. An offspring population is created by variation (i.e., recombination and mutation of the parent pool). A generation is composed of fitness evaluation, selection, and variation. Generations are repeated until a termination criterion is met, and the best program found is ultimately returned.

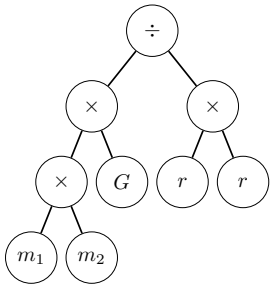

Figure 1.2: Example of a tree encoding a program that computes the law of gravitation.

encoding that represents a program that computes Equation 1.3 is shown in Figure 1.2. In the example, primitive instructions used to compose the tree are multiplication ($\times$) and division ($\div$), as are the interacting variables (G, m_1, m_2, r).

To ground the explanation on the workings of classic GP to a familiar example, the recovery of the aforementioned law of gravitation from data of F, m_1, m_2 and r is considered (the constant value of G needs to be found in $\mathbb{R}$), which is a symbolic regression problem. In this setting, for GP to find a program that explains how F is related to the other variables, means to find a function $f : \mathbb{R}^3 \to \mathbb{R}$ (three features are considered: m_1, m_2, and r).

To run a GP algorithm, firstly a set of primitives needs to be defined. The primitives will be composed to form programs. In tree-based GP, programs are represented with trees, and the nodes of the trees implement the primitives. In general, to facilitate the discovery of a well-performing program, any instruction (function) that is suspected to be part of the phenomenon should be included among the primitives. For example, consider the case where measurements of a complex and unintelligible circuit are taken and collected as data, and a model is sought that approximates the behavior of the original circuit. If sub-

circuits are known to be part of the total circuit, and these sub-circuits are known, then instructions that model such sub-circuits should be included as primitives. However, even in scenarios where no such information is available, GP can still perform competitively to black-box machine learning algorithms by adopting rather generic primitives [32].

For a typical symbolic regression task, two types of primitives can be identified in GP: primitives that require inputs, and primitives that do not. Commonly, the set containing the first type of primitives is called *function set*, while the set containing the second type of primitives is called *terminal set*. The function set $\mathcal{F}$ for a symbolic regression problem typically comprises linear and non-linear functions, e.g., $\mathcal{F} = \{+(\cdot,\cdot), -(\cdot,\cdot), \times(\cdot,\cdot), \div(\cdot,\cdot), \exp(\cdot)\}$. The terminal set $\mathcal{T}$ typically contains identity functions for each feature $x^{(j)}$, as well as constants, e.g., $\mathcal{T} = \{x^{(1)}, \ldots, x^{(m)}, -1.0, 1.0, \pi\}$. Primitives from the function set constitute non-leaf nodes of the trees, while primitives from the terminal set constitute leaf nodes (see Fig. 1.2).

Once the primitives have been chosen, the initial population of candidate programs must be sampled (see Fig. 1.1). This population is typically initialized randomly, i.e., by generating random trees. Several methods exists to achieve this [31]. For example, the *Grow* method is illustrated in Algorithm 1.1, that returns a random tree of a height that is limited by a user-specified parameter, given the function set and the terminal set.

Algorithm 1.1 Pseudo-code for the creation of a random GP tree.
Input: H: max. height; h: current height (initially 0); $\mathcal{F}$: function set; $\mathcal{T}$: terminal set.
Output: Root node N.

```
1 function SampleRandomTree(H, h, F, T)
2     if h = H then
3         N ← SampleNodeFrom(T)
4     else
5         N ← SampleNodeFrom(F ∪ T)
6         for i ∈ GetNumberOfExpectedInputs(N) do
7             C ← SampleRandomTree(H, h + 1, F, T)
8             AppendChildToParentNode(C, N)
9     return N
```

Once the initial population is sampled, the fitness of each program will be evaluated. In symbolic regression, the fitness is typically computed using Equation 1.2, with $p = 1$ (mean absolute error) or $p = 2$ (mean squared error). To obtain $f(\mathbf{x}_i)$, i.e., the scalar output of the program for the i-th data sample, the program must be executed with respect to the input $\mathbf{x}_i$. The execution of the program in tree-based encoding works as follows. The output is initially requested at the level of the root of the tree. Now, if the root is a terminal node, i.e., it has no inputs, it can immediately return. If the terminal represents the j-th feature, then the output is the scalar $x_i^{(j)}$. If the terminal represents a constant, then the output is that constant. Instead, if the root represents a function (e.g., $+$), then the root will request the output of its child nodes, and apply the function it represents on those outputs (e.g., will sum them). The same procedure holds for the child nodes, in a recursive fashion. Eventually, since leaves are terminal nodes, this recursion terminates, and the output of intermediate nodes flows from the bottom to the top of the tree.

After the fitness of each program has been computed, more fit programs are selected with larger probability, to become the parents that will breed the offspring population (see Fig. 1.1). Selection is normally performed based on fitness ranks, and the most popular method is called *tournament selection* [31]. Tournament selection works by randomly picking s (a parameter called tournament size) programs from the population (with replacement), and selecting the most fit one. This is typically repeated until the size of the selection is the same as the size of the population.

Next is the creation of offspring programs using the pool of parents. This is achieved by the use of variation operators. Two classic and still very popular variation operators in tree-based GP are subtree crossover and subtree mutation [31]. The hypothesis motivating subtree crossover is that fit parent programs (trees) contain important program subroutines (subtrees), therefore it is reasonable to attempt to obtain better offspring by recombination of these subroutines. Subtree crossover works as follows (see Fig. 1.3): two nodes in the parent trees are picked (uniformly at random or using some heuristic [30, 39]), after which two offspring programs are created by swapping the subtree rooted in those nodes. Subtree mutation works similarly to subtree crossover, with the difference that a random change is enforced, e.g., to perform an explorative step in the search space. A mutated offspring is made by replacing a randomly picked subtree with a new subtree, that is generated entirely at random (e.g., using Alg. 1.1).

Normally, a number of offspring programs equal to the population size is generated, and the offspring population is then used as a basis for the next generation. Generations are repeated until a satisfactory result is obtained (e.g., by imposing a fitness to reach), or a budget is exhausted (e.g., generation limit, time limit).

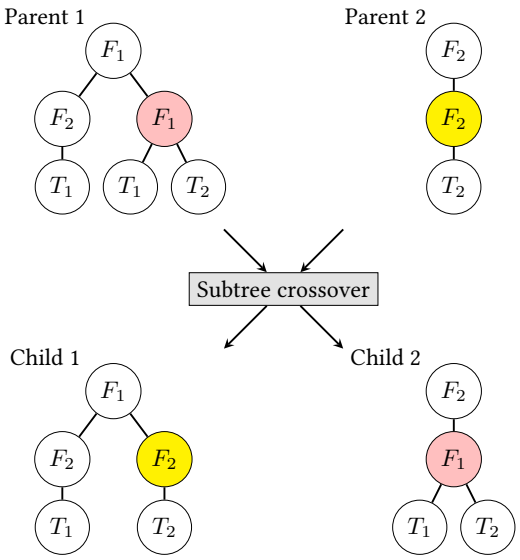

Figure 1.3: Illustration of subtree crossover. $F_{\#}$ and $T_{\#}$ represent generic function and terminal nodes. Highlighted nodes are the roots of the subtrees that are swapped.

1.3.1. An example of a GP run to recover Newton's law

As an illustrative example, we consider the task of regressing the right-hand side of the gravitation law, i.e., how F is determined in Equation 1.3. A dataset is generated by sampling $n = 1000$ observations, with a fictitious gravitational constant $G = 6.674$ (the scaling by 10^{-11} is ignored to avoid numerical instability problems), and masses and radii sampled between 1 and 10^2 by $10^{2\times u}$ with u uniformly distributed between 0 and 1. To simulate the presence of noise in the measurements, a normal error $\varepsilon \sim \mathcal{N}(0, 1)$ term is added to the right-hand side of the equation when generating the observations.

A GP using the scheme of Figure 1.1 is considered, with rather standard parameters. The population size is set to 500, and the trees are generated using Algorithm 1.1. The fitness is computed using the mean squared error (Eq. 1.2 with $p = 2$), and selection is performed with tournaments of size 4. The function set contains $\{+, -, \times, \tilde{\div}, \exp, \tilde{\log}\}$. Tilde operators include protections against numerical errors: $\tilde{\div}(a, b) := \text{sign}(b) \times \frac{a}{|b|+\kappa}$; $\tilde{\log}(a) := \log(|a| + \kappa)$. Here, $\kappa = 10^{-2}$ is used. The terminal set contains the three variables at play, i.e., m_1, m_2, r, as well as an Ephemeral Random Constant (ERC) [31]. ERC terminals have no specific value until a respective node is instantiated (e.g., sampled from the terminal set to be part of a tree in Alg. 1.1). The value is set to a scalar sampled randomly from a certain distribution. Here, the value is sampled uniformly at random between 0 and 10 with up to one decimal. Evolution is performed for a total of 25 generations. Trees with more than 15 nodes are discarded to avoid producing overly long, hard to read, programs. Assuming that the nature of the gravitational force F is unknown, consistency of operations with respect to units of measurements is not enforced.

Figure 1.4 shows, for a particular run, the fitness of the best program found at each generation, along with the number of nodes composing its tree, and some examples of the mathematical expression it represents. Over time, very different programs are found, of different size and involving different functions. In the last generation, a well-performing program is found of which its expression closely resembles the true law of gravitation (Eq. 1.3), apart from the constant G being imprecise. Notably, the program expression fits the data decently, it is extremely easy to read, and it is possible to interpret the program.

1.4. From classic to modern genetic programming and beyond by de-randomizing variation

Compared to other machine learning algorithms, perhaps the main disadvantage of GP is its computational expensiveness. For example in regression, to apply ordinary least squares to determine the coefficient of a linear model, the cost is $O(m^2 n)$ with m being the number of features/variables, and n the number of observations. To build a decision tree, $O(mn \log n)$ operations are required [25]. Further computation times for popular supervised machine learning algorithms are reported at: `https://bit.ly/2PG0xse`.

In GP, it is not straightforward to define an overall computation cost. How difficult it is to obtain programs with satisfactory performance largely depends on the problem to be solved, on the choice of primitives, and on the quality of the variation and selection methods. If it is assumed that a population size P and a number of generations G can lead to satisfactory results (for a given problem, set of primitives, and variation and selection methods), then the cost of GP will be $O(PGmn)$. The mn term is a crude estimation of

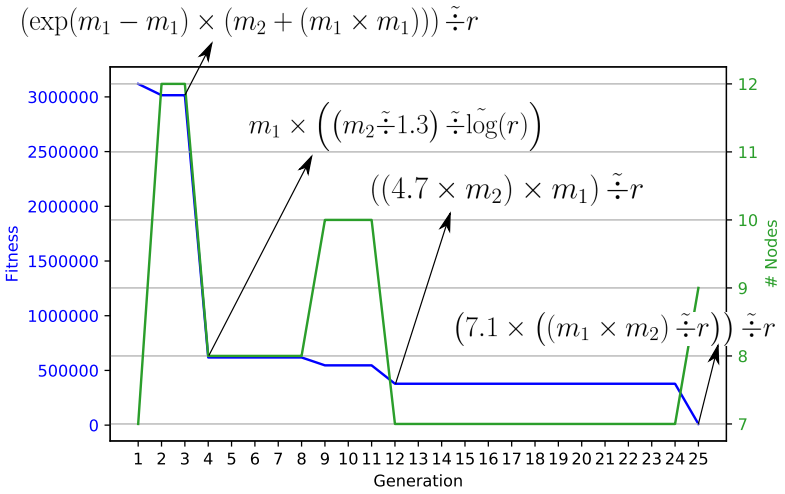

Figure 1.4: Fitness, number of nodes, and mathematical expression associated with the best program found by GP along 25 generations.

the cost of evaluating the fitness of programs with some sort of decent performance. To be more specific, to get the output of a program and calculate its fitness, each instruction needs to be evaluated on the n observations, and it is reasonable to expect fit programs to have a number of instructions that depends on the number of features m. Because the mn term in $O(PGmn)$ is essentially fixed by the task, it is PG that should be minimized to improve the efficiency of GP. In other words, we want to *achieve more* (quality of final programs) *with less* (evolutionary budget). The problem at hand is given, and the choice of primitives is usually dictated by knowledge of the problem. To minimize PG, one can thus attempt to improve the effectiveness of variation and selection.

Key to the success and efficiency of an evolutionary algorithm in general is the use of competent variation. It has long been known that evolutionary algorithms perform well when variation is capable of combining the right solution components sufficiently fast, i.e., before the population is taken over by more fit, yet sub-optimal, solutions made of sub-optimal components [40, 41]. As for any other evolutionary algorithm, this very much holds for GP as well, and how to improve variation is one of the most important open questions in the field.

Classic variation operators do not attempt to harness any sort of information to enhance their chances of leading to better programs. Rather, they typically act completely at random. For this reason, they are oftentimes referred to as "*blind*". The vast majority of variation operators employed in different forms of GP is blind. In classic tree-based GP, as explained before, subtree crossover swaps two *random* subtrees between two respective solutions (Fig. 1.3). Subtree mutation swaps a *random* subtree with a *random* new subtree. One-point mutation modifies *random* nodes with other *random* nodes [31]. *Cartesian GP* represents programs with directed graphs, and mutates node connections [42], at *random*. *Push GP* is often used to handle strongly-typed programs [43], and grammatical evolution can enforce very particular constraints on the interactions of program instructions,

by encoding programs with fixed-length binary solutions that are interpreted according to a grammar [44]. In both cases, the typical variation operators applied remain highly stochastic, and do not attempt to harness and exploit information that depends on the problem, or that may emerge while the search progresses.

The goal of this thesis is therefore to *design (and apply) a novel GP algorithm which provides a more principled way of performing variation to reduce the amount of computational effort required to obtain accurate programs.* Furthermore, as will be described below, for learning tasks such as symbolic regression, the research panorama of GP lacks suitable methods to efficiently evolve programs in scenarios where a small number of program instructions is desired, to enhance the chance of obtaining human-interpretable models. Clearly, obtaining small programs is not a sufficient condition for interpretability, but it can often be considered a necessary one. The positive aspects as well as the limitations of state-of-the-art contributions to variation in GP are described in more detail next.

1.4.1. Early studies on biasing variation and more recent ones on geometric semantic variation

The first works on improving variation in GP mostly focused on studying the effect of different biases in the recombination and mutation of subtrees. To name a few examples, in one of the seminal works in GP [30], it is recommended to select nodes for subtree crossover and subtree mutation with larger probability if they are functions rather than terminals (90% and 10% respectively) to limit *bloat*, i.e., excessive growth of the number of program instructions with limited effect on the fitness. For subtree crossover, the introduction of biases related to the positions of the subtrees to swap as well as on the size of the subtrees have also been explored [45, 46]. Essentially, early works on variation in GP focused on biasing variation at the level of program syntax, i.e., *how the programs look*, e.g., in terms of node types, subtree position, and subtree size for tree-based GP.

The last decade has seen the rise of studies on so-called *geometric semantic* variation operators. At the Genetic and Evolutionary Computation COnference (GECCO), the premiere conference on evolutionary computation, papers on geometric semantic variation in GP were nominated for, or won the best paper award, in 2013 [47], 2015 [48], 2016 [49], 2017 [50], and 2018 [51]. Thanks to works such as [52], many researchers came to the realization that looking beyond syntax is crucial to improve variation in GP. In fact, program modifications that can be considered small in terms of program syntax are not guaranteed to lead to changes that can be considered small in terms of program output [53]. For example, picture an arbitrary tree of which the nodes implement Boolean logic gates such as `AND`, `OR`, and `NOT`, and `ID` (the identity function). From a syntactic perspective, changing the value of a few nodes can be considered a "small" modification. Yet, if the node to be changed is the root, and the change swaps `ID` with `NOT`, then, for any set of inputs, the output of the program will be the opposite of the one before the change.

For the aforementioned reason, geometric semantic variation focuses on the effect variation will produce at the level of the program semantic, i.e., *what the programs do* in terms of the outputs they produce when executed [54–56]. For a task such as symbolic regression, where n samples are given, the program output (or semantic) is the n-dimensional vector of transformations the program performs upon the input samples $\{\mathbf{x}_i\}_{i=1}^{n}$. The term *geometric* comes from the fact that geometric semantic variation at-

tempts to ensure that the output of an offspring program is geometrically close to the output of its parent programs, according to a metric defined in the space of the outputs. For example, this may mean that the output of an offspring program will be placed on the hyper-plane that passes through the points represented by the outputs of the parent programs, or within the hyper-cube that has the outputs of the parent programs as opposing vertices [52]. Generally, any variation operator that establishes (or attempts to establish) some sort of geometric relationship between a parent output and any other set of outputs (e.g., the target variable y can be considered [57]), is called a geometric semantic variation operator [39, 58].

Geometric semantic variation is a meaningful approach in terms of effective and efficient search because to control how search is directed in the space of program outputs means to directly control the program quality, since in many tasks the output of a program directly determines its fitness. In machine learning problems such as symbolic regression, this property is especially interesting because the loss (fitness) function is often convex (e.g., Eq. 1.2 for $p = 2$) and therefore relatively easy to minimize if the right type of variation is used [52]. For example, consider the well-studied *exact* geometric semantic variation operators that were introduced in [52]: the Geometric Semantic Crossover (GSX), and the Geometric Semantic Mutation (GSM). GSX and GSM are called exact because they guarantee that the output of an offspring program will be close to the output of its parent programs. More specifically, GSX enforces the output of an offspring program to be bounded within a hyper-cube defined by the output of the two parent programs. Let f_1 and f_2 be the functions represented by the first and second parent program respectively. Then, for, e.g., a symbolic regression data set of n observations and m features, recall that the output of a program is $\{f(\mathbf{x}_i)\}_{i=1}^{n} \in \mathbb{R}^n$. GSX produces an offspring by combining program instructions (e.g., tree nodes) that perform the following operation:

$$GSX(f_1, f_2) := g \times f_1 + (\mathbf{1} - g) \times f_2, \tag{1.4}$$

where g is the function represented by a randomly sampled program, with codomain in $[0, 1]^n$, and $\mathbf{1}$ is a vector of ones in $\mathbb{R}^n$. For example, g can be generated by sampling a random tree with Algorithm 1.1, and appending a softmax node on top of the root of that tree. The aforementioned version of GSX is called the Manhattan version. Another version of GSX exists, the Euclidean one, where a linear combination of the parent programs is produced (g is then an n-dimensional vector with constant values in $[0, 1]$).

GSM works similarly to GSX: it produces an offspring from one parent with output that is bounded to be within a hyper-cube with side length r centered on the output of the parent. If f is the function represented by the parent program, then GSM is defined as:

$$GSM(f) := f + r(g - h), \tag{1.5}$$

where g and h are functions represented by two respectively randomly sampled programs, with codomain in $[0, 1]^n$. The value of r is a hyper-parameter to be chosen by the user.

GP equipped with exact geometric semantic variation operators has been shown to be competitive with other machine learning methods in terms of final prediction errors on several real-world supervised learning problems, including, e.g., street construction [59], energy forecasting [60], health care [61], and pharmacokinetics [62].

Unfortunately, along with its advantages in terms of search properties, the aforementioned exact geometric semantic variation operators come with a burdening limitation: their use generally results in programs growing to be very large in the number of instructions. In fact, note that both GSX and GSM re-use the entire function as represented by the parent program(s) (see Eq. 1.4 and Eq. 1.5). This can only be achieved by preserving the entire structure (tree) that represents the programs. Because of this, the repeated application of GSX results in exponentially larger offspring, while GSM introduces a linear growth factor [51, 52].

Ways to reduce program size have been explored for exact geometric semantic variation. Arithmetic simplification of programs is NP-hard, and heuristics have shown limited effect [52]. Because GSX and GSM essentially perform linear combinations [63], recent work assessed the possibility to keep track of what unique non-linear function compositions emerge when using these operators, to then re-arrange them in a compact linear sum [51]. Still, the program size then does not reduce below thousands of instructions in practical applications, even on moderately sized datasets. So far, techniques to stop the search early, i.e., as soon as programs with satisfactory performance are found, seem to work best to contain the issue [64].

Approximate geometric semantic variation operators differ from exact variation operators (like GSX and GSM) in that they lose the guarantee on the position that the offspring program's output will have, but, importantly, they can modify parent programs internally (e.g., at the level of subtrees) [39, 58, 65]. This means that, in principle, they have a chance to produce substantially shorter programs than exact geometric variation operators. Literature confirms this hypothesis: approximate geometric variation operators typically induce programs much smaller than their exact counterpart. However, the typical number of instructions can still be of the order of hundreds or thousands [39]. This means that the programs are still too large to allow interpretability.

In summary, geometric semantic variation operators have been found to make GP search more efficient and effective, but at the cost of program size. When programs have a very large number of instructions, any chance of interpreting them is lost. This means that GP is essentially producing black box models. In such scenarios, the very use of GP itself becomes questionable, because competitive models can be acquired by other machine learning algorithms in a fraction of the time taken by GP [32]. Therefore, there is a need for new variation paradigms that find small, accurate programs, in an efficient manner.

1.4.2. Between syntax and semantics: model-based variation

Another type of variation for GP that has been studied in the last twenty years, is *model-based* variation [66–68]. In this context, the term *model* should not be confused with the end-product of a machine learning algorithm or GP. Rather, it refers to a statistical model that contains information on what program instructions, at what positions, and with which inter-dependencies, are associated with well-performing programs.

GP using model-based variation is called model-based GP (or probabilistic model building GP). In model-based GP, a statistical model is inferred, typically every generation, to capture the type, location, and inter-dependency of instructions for instance in the form of a fitting probability distribution (e.g., using a minimum descriptor length metric) [68]. This is normally done using a portion of fit programs that is chosen, e.g., by tournament selection. At variation time then, the variation operator queries the statistical model to gain information on how to modify the programs.

The hypothesis behind a model-based approach is that, if salient instruction patterns can be detected from above-average fit programs, and the information they carry can be exploited during variation, progress will be more efficient than by making new combinations blindly. Model-based variation can therefore be considered to be somewhat in between the purely syntactic and purely semantic variation approaches: the statistical model captures information at the level of the syntax of programs, yet, this is done on well-performing programs, which possess favorable subroutines that contribute positively to their semantic.

The most widely studied form of model-based GP inherits and extends concepts of Estimation-of-Distribution Algorithms (EDAs) from binary optimization [66–68]. In EDA-GP, the variation operator samples instructions depending on their position in the encoding used for program representation, based on the information captured by the statistical model. To use EDAs in GP, firstly an issue must be solved involving the representation of programs: normally, GP adopts non-fixed length representations. For example, in tree-based GP, trees can grow to any shape. Consequently, it is not trivial to establish what information should be measured in what positions of the trees, so that this information shares a consistent meaning for any tree in the population.

One of the earliest approaches, the Probabilistic Incremental Program Evolution (PIPE) [69], tackled the problem of having meaningful representations that can be used in EDAs for GP by adopting a so-called *prototype tree*, i.e., a template tree in which the nodes contain probabilities that drive the sampling of possible instructions. To generate a new program tree, the prototype tree is parsed top-down, and instructions (i.e., the nodes for the program tree) are sampled according to the probabilities specified at the nodes of the prototype tree. Hence, the prototype tree embodies a statistical model, in which nodes are random variables that are associated with a specific and semantically consistent node position for the whole population. The sample space of each random variable composing the statistical model is the set of possible node types (functions and terminals). In PIPE, the statistical model employed is the simplest possible: a univariate factorization of the node types appearing at each position. This means that each random variable is independent, and no interactions between nodes is modeled.

Following PIPE, different approaches have been proposed, considering more complex statistical models. For example, the use of marginal product models was considered, estimating these models every generation by greedily optimizing the minimum description length [70]. To model even more complex relationships, the use of Bayesian networks was explored as well [71]. One work considered the possibility to sample n-grams of nodes connected by parenthood relationships, after having discovered that these types of relationships were among the most recurrent patterns emerging from multiple runs of classic GP [72]. Many other EDA-GP algorithms generate stochastic models that capture infor-

mation on so-called *indirect representations*, such as a set of rules (typically, a context-free grammar) that specifies how a tree can be constructed [68]. Then, these rules are sampled, and subsequently queried to generate new trees of different shape and size [73–76]. This latter type of works belongs to the sub-field called *grammar-based GP*.

Many experimental results from the mentioned literature suggest that model-based variation is a promising approach to efficiently find programs that are smaller than the ones found by blind and geometric semantic variation, and that can be very accurate [72]. Even though model-based variation does not exert the same level of control that geometric semantic variation has in terms of program semantic, it increases the probability of performing changes that can be, at the same time, syntactically smaller and semantically more impactful than by using traditional blind variation operators. This has been shown for a number of synthetic problems (e.g., in [77]), and for symbolic regression of small known functions (e.g., in [72]).

All in all, model-based variation may seem a promising approach to efficient and effective variation without a compromise in program size. However, experimental results on model-based variation mostly focused on benchmark problems, and considered only small-dimensional problems [68]. A reasonable explanation for this is that EDA-GP algorithms seem not to scale well to high-dimensional problems. For symbolic regression, none of the cited works considered a realistic dataset where the true underlying phenomenon is unknown, and (at least) dozens of features are present. Rather, only small dimensional synthetic functions are typically investigated (e.g., up to 5 features in [76]). One realistic task concerning the symbolic regression of a known physics law was considered, yet this law consists of only 5 features [78]. The fact that EDA-GP algorithms do not work well on large dimensional problems may not be surprising: the larger the number of primitives (functions and terminals), the larger the sample space of the random variables composing the stochastic model. This means that the population size must also be large, before accurate and helpful estimations can be obtained for variation [72].

Summarizing, work on EDA-GP algorithms has indicated that model-based evolution holds potential to evolve programs in a principled, more effective and efficient manner. Moreover, the programs found can be rather compact with respect to their performance [72]. However, EDA-GP algorithms estimate entire (joint) probability distributions, and, to work well, they need very large population sizes to tackle high-dimensional problems. This hinders their ability to scale. Originating from EDA research, the last decade has seen the rise of a new type of model-based evolutionary algorithm: *optimal mixing evolutionary algorithms*. These algorithms were first introduced for binary optimization, and were consistently found to outperform EDAs while requiring smaller population sizes [79–83]. It is therefore natural to wonder whether bringing this type of algorithms to GP may improve upon the state-of-the-art.

1.4.3. Optimal mixing evolutionary algorithms

Optimal Mixing Evolutionary Algorithms (OMEAs) have been first introduced in genetic algorithms for binary optimization [79]. OMEAs are a subclass of model-based evolutionary algorithms, that attempt to provide a better alternative to EDAs [80].

Given a generic evolutionary algorithm with fixed-length solution representation, EDAs attempt to estimate *both* inter-dependencies between positions, i.e., what the (conditional)

dependencies between random variables are, *and* the values that should be assumed in those positions. Consequently, large population sizes are required to capture information with a sufficient level of accuracy to bring benefit. This is a potential source of inefficiency. Furthermore, an EDA samples an entire solution at once, and only then the fitness of the solution is computed. If several sub-solutions exist in a whole solution that have different impact on the fitness, it is hard to separate these contributions, and exploit their individual contributions. From now on, these sub-solutions, if they have an above-average contribution to a solution's quality, will be generally referred to as *building blocks*[3].

OMEAs can already work well with smaller population sizes compared to EDAs for two key reasons. First, the models employed by OMEAs, called *linkage models*, only attempt to establish which positions are inter-dependent, i.e., exhibit strong *linkage*, and do not attempt to model which values should be assumed in what positions. The idea is to identify salient patterns of solution components (i.e., building blocks) that need to be propagated together when performing variation, so as to avoid the disruption of their positive joint effect. Linkage models often consist of multiple linkage sets that are often arranged in hierarchical levels. Second, the variation operator of OMEAs, called *optimal mixing*, mixes solutions by explicitly attempting to preserve the candidate building blocks as indicated by the interdependencies captured by the linkage model, and immediately tests for the impact of each mixing event.

The name optimal mixing comes from the theory of genetic algorithms, where it is known that the best-possible recombination operator is the one that mixes *en block* components that constitute building blocks [41]. In more detail, optimal mixing works by iteratively performing recombination according to blocks of highly dependent components indicated by the linkage model, and, after each modification, by immediately calculating the contribution of that modification in terms of fitness. If the fitness becomes worse, then the change is reverted. Otherwise, the change is kept. This behavior explicitly enforces selection pressure to focus on the contribution of potential building blocks, something that is not considered by EDAs, where solutions are modified entirely, and only then evaluated [79, 80]. This also causes improvements to be accepted based on whether individual solutions improve directly, rather than requiring a population-wide selection step. The latter causes also partial non-improving changes to be selected, which, from a mixing viewpoint, is not optimal (leading to the name optimal mixing).

The literature on binary optimization shows that OMEAs consistently outperform EDAs on difficult optimization problems [79, 81, 83]. Interestingly, however, no OMEA has ever been proposed for GP. Therefore, an investigation into the design of an OMEA for GP may well provide new insights and methods to obtaining small and accurate programs efficiently. The research questions that lead to the design of an OMEA for GP, as well as its application, are presented in the next section.

1.5. Research questions

This thesis comprises five research questions, the answers to which are aimed at supporting the achievement of the main goal of this thesis: improving the efficiency and

[3] An informal and generic definition is used here because several, and more or less formal, definitions of *building blocks* exist, typically varying depending on the type of evolutionary algorithm considered.

effectiveness of GP, in particular in the scenario where programs are of limited size, to increase the chances they are interpretable. Two of the research questions concern the design of an OMEA for GP, and two concern its application to two particular problems, in an effort to provide tangible successful outcomes. One more question is formulated regarding efficiency and effectiveness of approximate geometric semantic variation for symbolic regression. In the following, these research questions are listed and described.

Research question 1. How can an OMEA be designed for GP, and how does it fare against state-of-the-art GP algorithms?
The possibility of designing a competent OMEA for GP, and an evaluation on how such an algorithm fares against the state-of-the-art, is addressed by this question. The goal is to assess whether OMEAs for GP can, in fact, make GP more efficient and effective. Benchmark problems where an optimal program is sought are considered for this purpose.

In Chapter 2, for the first time, an OMEA for GP is presented. In particular, an adaptation of the binary Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) is presented, called *GP-GOMEA* [79].

The contents of this chapter are based on the following publication: **M. Virgolin**, T. Alderliesten, C. Witteveen, and P.A.N. Bosman. Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*, pp. 1041-1048, ACM (2017).

Research question 2. Does GP-GOMEA work well on realistic cases of symbolic regression?
EDAs for GP do not scale well on realistic problems such as symbolic regression for real-world data. Following the promising results on benchmark problems in Chapter 2, we more closely investigate whether OMEAs for GP (GP-GOMEA in particular), can perform better in these scenarios that are of high interest from a practical viewpoint.

In Chapter 3 possible limitations of some of GP-GOMEA's core mechanisms (the linkage learning in particular) that arise when dealing with realistic symbolic regression, and that hold in general for supervised learning tasks (e.g., classification) are identified. Following the identification of these limitations, methods to improve GP-GOMEA are proposed and tested in the scenario where symbolic expressions are sought that are sufficiently small to have a large chance of being interpretable.

The contents of this chapter are based on the following preprint: **M. Virgolin**, T. Alderliesten, C. Witteveen, and P.A.N. Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Accepted for publication in Evolutionary Computation. Preprint arXiv:1904.02050*, arXiv (2019).

Research question 3. Does geometric semantic variation work well on realistic cases of symbolic regression?
After having shown that GP-GOMEA is a promising approach to deal with realistic symbolic regression problems, it is natural to wonder how approximate geometric semantic variation performs on similar tasks. In other words, with this research question it is assessed whether approximate geometric semantic variation is capable of improving the ef-

ficiency and effectiveness of GP in scenarios that are of interest from a practical viewpoint (symbolic regression on real-world data).

In Chapter 4 it is shown that, similar to the literature on EDA-GP algorithms, almost every work in approximate geometric semantic variation only considered small-dimensional problems. Subsequent experiments suggest that this method does not actually work well for symbolic regression of realistic datasets. Consequently, improvements are proposed to enable approximate geometric semantic variation to work competitively on realistic datasets.

The contents of this chapter are based on the following publication: **M. Virgolin**, T. Alderliesten, and P.A.N. Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*, pp. 1084-1092, ACM (2019).

Research question 4. Can the capability of GP-GOMEA to find small and accurate programs be used to make a machine learning algorithm generate transparent models?
With this question a first important application of GP-GOMEA is addressed. Under the assumption that GP-GOMEA can in fact be efficient and effective in finding small and accurate programs, it is interesting to assess whether this property can be exploited to enable the discovery of more transparent machine learning models trained by a chosen machine learning algorithm.

This question is addressed in Chapter 5, in which a synergistic combination of GP-GOMEA with a second machine learning algorithm is explored, where the first is used to evolve small and salient constructed features, and the second trains a small-dimensional model upon those features. The goal is to reduce the dimensionality of an original set of features down to only two small and readable constructed features (i.e., symbolic transformations of the original features) so that the behavior (i.e., the predictions for a large set of inputs) of the final machine learning model can be visualized. Moreover, we assess whether the accuracy of the model obtained this way is better or worse than the accuracy obtained by using the original (high-dimensional) set of features. This approach is tested on classification and regression real-world datasets.

The contents of this chapter are based on the publication: **M. Virgolin**, T. Alderliesten, and P.A.N. Bosman. On explaining machine learning models by evolving crucial and compact features. *Swarm and Evolutionary Computation* **53**, pp. 100640, Elsevier (2020).

Research question 5. Can GP-GOMEA find models that are accurate and likely to be interpretable for a clinical problem in pediatric radiation oncology?
To investigate the practical usefulness of GP-GOMEA in a societally-impactful problem, an application of GP-GOMEA to pediatric oncology is investigated at large. A medical application represents a suitable fit for the use of a machine learning algorithm that may deliver models that doctors can understand and trust. In particular, the application we study is *radiation treatment dose reconstruction* for pediatric cancer patients. Children with cancer that undergo multi-modality treatment including irradiation of the tumor tend to develop adverse effects. To improve pediatric radiation treatment, it is imperative to study how the radiation dose to sensitive organs is linked to adverse effects. Recent studies consider 3D dose distributions. However, this type of fine-grained information can oftentimes be

absent when *late* adverse effects are considered. This is because some late adverse effects can occur decades after the treatment, which might have happened before the advent of 3D imaging became common practice (e.g., prior to the 1990s in the Netherlands). Therefore, for children treated decades ago, 3D dose distributions need to be reconstructed (estimated as accurately as possible). We explored the use of machine learning, and GP-GOMEA in particular, to improve upon the limitations of existing methods employed in dose reconstruction. This application is tackled in three chapters.

Chapter 6 introduces the aforementioned problem, and explores the feasibility of using machine learning (specifically random forest [25]) to use the information that is typically available for patients treated in the past (i.e., without 3D imaging) to predict 3D anatomical information. This 3D information can then be used to craft (or select) a *phantom*, i.e., a 3D surrogate of the true patient anatomy, which can be used to simulate the delivery of the radiation treatment, to obtain estimations of the 3D dose distribution.

The contents of this chapter are based on the following publication: **M. Virgolin**, I.W.E.M. van Dijk, J. Wiersma, C.M. Ronckers, C. Witteveen, A. Bel, T. Alderliesten, and P.A.N. Bosman. On the feasibility of automatically selecting similar patients in highly individualized radiotherapy dose reconstruction for historic data of pediatric cancer survivors. *Medical Physics* **45** (4), pp. 1504-1517, Wiley (2018).

In Chapter 7 the concepts of the previous chapter are taken to the next level by presenting an automatic pipeline that can generate patient-specific 3D pediatric phantoms of the abdomen. A comparison is included between GP-GOMEA and other machine learning algorithms, which are used to provide models that predict, given the (non-3D) information available for the patient as an input, how 3D anatomical imaging of other patients should be assembled into a personalized phantom. The contents of this chapter are based on the following preprint: **M. Virgolin**, Z. Wang, T. Alderliesten, and P.A.N. Bosman. Machine learning for automatic construction of pseudo-realistic pediatric abdominal phantoms. *Submitted. Preprint arXiv:1909.03723*, arXiv (2019). The preprint extends the publication: **M. Virgolin**, Z. Wang, T. Alderliesten, and P.A.N. Bosman. Machine learning for automatic construction of pediatric abdominal phantoms. *In Proceedings of SPIE Medical Imaging 2020: Imaging Informatics for Healthcare, Research, and Applications*, International Society for Optics and Photonics (2020) (to appear).

Lastly, Chapter 8 presents the use of GP-GOMEA to perform dose reconstruction in a novel way. Instead of attempting to craft a phantom for a patient, machine learning models are trained to *directly* estimate key dose-volume metrics that are typically considered when studying the link between radiation treatment and adverse effects. To this end, the information available for the patient is used as input for the machine learning models *jointly* with the information available for the radiation treatment plan. The contents of this chapter are based on the following preprint: **M. Virgolin**, Z. Wang (shared first co-author), B.V. Balgobind, I.W.E.M. van Dijk, J. Wiersma, P.S. Kroon, G.O. Janssens, M. van Herk, D.C. Hodgson, L. Zadravec Zaletel, C.R.N. Rasch, A. Bel, P.A.N. Bosman, and T. Alderliesten. Surrogate-free machine learning-based organ dose reconstruction for pediatric abdominal radiotherapy. *Submitted. Preprint arXiv:2002.07161*, arXiv (2020).

1

References

[1] C. M. Bishop, *Pattern recognition and machine learning* (Springer, 2006).

[2] V. Mayer-Schönberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think* (Houghton Mifflin Harcourt, 2013).

[3] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, *Dermatologist-level classification of skin cancer with deep neural networks,* Nature **542**, 115 (2017).

[4] M. A. Lones, J. E. Alty, J. Cosgrove, P. Duggan-Carter, S. Jamieson, R. F. Naylor, A. J. Turner, and S. L. Smith, *A new evolutionary algorithm-based home monitoring device for Parkinson's Dyskinesia,* Journal of Medical Systems **41**, 176 (2017).

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners,* OpenAI Blog **1** (2019).

[6] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao, *Inference of regular expressions for text extraction from examples,* IEEE Transactions on Knowledge and Data Engineering **28**, 1217 (2016).

[7] S. H. Tan, Z. Dong, X. Gao, and A. Roychoudhury, *Repairing crashes in android apps,* in *International Conference on Software Engineering* (ACM, 2018) pp. 187–198.

[8] M. Češka, J. Matyáš, V. Mrazek, L. Sekanina, Z. Vasicek, and T. Vojnar, *Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished,* in *International Conference on Computer-Aided Design* (IEEE Press, 2017) pp. 416–423.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, *Mastering the game of Go with deep neural networks and tree search,* Nature **529**, 484 (2016).

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing Atari with deep reinforcement learning,* (2013), arXiv preprint arXiv:1312.5602.

[11] Y. Koren, *The bellkor solution to the Netflix grand prize,* Netflix prize documentation **81**, 1 (2009).

[12] J. Talamini, G. Scaini, E. Medvet, and A. Bartoli, *Selfish vs. global behavior promotion in car controller evolution,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Companion* (ACM, 2018) pp. 1722–1727.

[13] E. Brynjolfsson, D. Rock, and C. Syverson, *Artificial intelligence and the modern productivity paradox: A clash of expectations and statistics*, Tech. Rep. (National Bureau of Economic Research, 2017).

[14] Z. C. Lipton, *The mythos of model interpretability,* Queue **16**, 30:31 (2018).

[15] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, *A survey of methods for explaining black box models,* ACM Computing Surveys (CSUR) **51**, 93:1 (2018).

[16] A. W. Flores, K. Bechtel, and C. T. Lowenkamp, *False positives, false negatives, and false analyses: A rejoinder to machine bias: There's software used across the country to predict future criminals. And it's biased against blacks,* Federal Probation **80**, 38 (2016).

[17] D. Doran, S. Schulz, and T. R. Besold, *What does explainable AI really mean? A new conceptualization of perspectives,* (2017), arXiv preprint arXiv:1710.00794.

[18] B. Goodman and S. Flaxman, *European union regulations on algorithmic decision-making and a "right to explanation",* AI Magazine **38**, 50 (2017).

[19] F. Pasquale, *The black box society* (Harvard University Press, 2015).

[20] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq, *Algorithmic decision making and the cost of fairness,* in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2017) pp. 797–806.

[21] D. Pedreshi, S. Ruggieri, and F. Turini, *Discrimination-aware data mining,* in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, NY, USA, 2008) pp. 560–568.

[22] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, *Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission,* in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2015) pp. 1721–1730.

[23] Z. C. Lipton, *The doctor just won't accept that!* (2017), arXiv preprint arXiv:1711.08037.

[24] A. Adadi and M. Berrada, *Peeking inside the black-box: A survey on explainable artificial intelligence (XAI),* IEEE Access **6**, 52138 (2018).

[25] L. Breiman, *Random forests,* Machine Learning **45**, 5 (2001).

[26] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system,* in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016) pp. 785–794.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).

[28] V. Nair and G. E. Hinton, *Rectified linear units improve restricted Boltzmann machines,* in *International Conference on Machine Learning (ICML 10)* (2010) pp. 807–814.

[29] M. Schmidt and H. Lipson, *Distilling free-form natural laws from experimental data,* Science **324**, 81 (2009).

[30] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection* (MIT Press, 1992).

[31] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming* (Lulu Enterprises, UK Ltd, 2008).

[32] P. Orzechowski, W. La Cava, and J. H. Moore, *Where are we now?: A large benchmark study of recent symbolic regression methods,* in *Genetic and Evolutionary Computation Conference (GECCO) 2018* (ACM, 2018) pp. 1183–1190.

[33] J. R. Koza, *Human-competitive results produced by genetic programming,* Genetic Programming and Evolvable Machines **11**, 251 (2010).

[34] P. G. Espejo, S. Ventura, and F. Herrera, *A survey on the application of genetic programming to classification,* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **40**, 121 (2009).

[35] H. J. Bierens, *Kernel estimators of regression functions,* in *Advances in Econometrics: Fifth World Congress*, Vol. 1 (1987) pp. 99–144.

[36] T. McConaghy, *FFX: Fast, scalable, deterministic symbolic regression technology,* in *Genetic Programming Theory and Practice IX* (Springer, 2011) pp. 235–260.

[37] F. O. de França, *A greedy search tree heuristic for symbolic regression,* Information Sciences **442**, 18 (2018).

[38] I. Newton, *The Principia: mathematical principles of natural philosophy* (Univ of California Press, 1999).

[39] T. P. Pawlak, B. Wieloch, and K. Krawiec, *Semantic backpropagation for designing search operators in genetic programming,* IEEE Transactions on Evolutionary Computation **19**, 326 (2015).

[40] C. L. Bridges and D. E. Goldberg, *An analysis of reproduction and crossover in a binary-coded genetic algorithm,* in *International Conference on Genetic Algorithms and Their Application* (L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1987) pp. 9–13.

[41] D. Thierens and D. E. Goldberg, *Mixing in genetic algorithms,* in *International Conference on Genetic Algorithms* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993) pp. 38–47.

[42] J. F. Miller and S. L. Harding, *Cartesian genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2008) pp. 2701–2726.

[43] L. Spector and A. Robinson, *Genetic programming and autoconstructive evolution with the push programming language,* Genetic Programming and Evolvable Machines **3**, 7 (2002).

[44] M. O'Neill and C. Ryan, *Grammatical evolution,* IEEE Transactions on Evolutionary Computation **5**, 349 (2001).

[45] P. D'Haeseleer, *Context preserving crossover in genetic programming,* in *IEEE Conference on Evolutionary Computation (CEC) 1994* (IEEE, 1994) pp. 256–261.

[46] W. B. Langdon, *Size fair and homologous tree genetic programming crossovers,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (Morgan Kaufmann Publishers Inc., 1999) pp. 1092–1097.

[47] A. Moraglio and A. Mambrini, *Runtime analysis of mutation-based geometric semantic genetic programming for basis functions regression,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 989–996.

[48] R. Ffrancon and M. Schoenauer, *Memetic semantic genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2015) pp. 1023–1030.

[49] L. O. V. B. Oliveira, F. E. Otero, and G. L. Pappa, *A dispersion operator for geometric semantic genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2016) pp. 773–780.

[50] L. F. Miranda, L. O. V. B. Oliveira, J. F. B. S. Martins, and G. L. Pappa, *How noisy data affects geometric semantic genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2017) pp. 985–992.

[51] J. F. B. S. Martins, L. O. V. B. Oliveira, L. F. Miranda, F. Casadei, and G. L. Pappa, *Solving the exponential growth of symbolic regression trees in geometric semantic genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2018) pp. 1151–1158.

[52] A. Moraglio, K. Krawiec, and C. G. Johnson, *Geometric semantic genetic programming,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2012) pp. 21–31.

[53] J. McDermott, E. Galván-Lopéz, and M. O'Neill, *A fine-grained view of phenotypes and locality in genetic programming,* in *Genetic Programming Theory and Practice IX* (Springer, 2011) pp. 57–76.

[54] L. Vanneschi, M. Castelli, and S. Silva, *A survey of semantic methods in genetic programming,* Genetic Programming and Evolvable Machines **15**, 195 (2014).

[55] K. Krawiec and U.-M. O'Reilly, *Behavioral programming: a broader and more detailed take on semantic GP,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2014) pp. 935–942.

[56] J. Albinati, G. L. Pappa, F. E. Otero, and L. O. V. B. Oliveira, *The effect of distinct geometric semantic crossover operators in regression problems,* in *European Conference on Genetic Programming* (Springer, 2015) pp. 3–15.

[57] B. Wieloch and K. Krawiec, *Running programs backwards: instruction inversion for effective search in semantic spaces,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 1013–1020.

[58] Q. Chen, B. Xue, and M. Zhang, *Improving generalisation of genetic programming for symbolic regression with angle-driven geometric semantic operators,* IEEE Transactions on Evolutionary Computation **23**, 488 (2019).

[59] M. Castelli, L. Vanneschi, and S. Silva, *Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators,* Expert Systems with Applications **40**, 6856 (2013).

[60] M. Castelli, L. Vanneschi, and M. De Felice, *Forecasting short-term electricity consumption using a semantics-based genetic programming framework: the south Italy case,* Energy Economics **47**, 37 (2015).

[61] M. Castelli, L. Vanneschi, and S. Silva, *Prediction of the unified Parkinson's disease rating scale assessment using a genetic programming system with geometric semantic genetic operators,* Expert Systems with Applications **41**, 4608 (2014).

[62] L. Vanneschi, S. Silva, M. Castelli, and L. Manzoni, *Geometric semantic genetic programming for real life applications,* in *Genetic Programming Theory and Practice XI* (Springer, 2014) pp. 191–209.

[63] T. P. Pawlak, *Geometric semantic genetic programming is overkill,* in *European Conference on Genetic Programming* (Springer, 2016) pp. 246–260.

[64] I. Gonçalves, S. Silva, C. M. Fonseca, and M. Castelli, *Unsure when to stop?: ask your semantic neighbors,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2017) pp. 929–936.

[65] K. Krawiec and T. Pawlak, *Approximating geometric crossover by semantic backpropagation,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 941–948.

[66] M. Hauschild and M. Pelikan, *An introduction and survey of estimation of distribution algorithms,* Swarm and Evolutionary Computation **1**, 111 (2011).

[67] Y. Shan, R. I. McKay, D. Essam, and H. A. Abbass, *A survey of probabilistic model building genetic programming,* in *Scalable Optimization via Probabilistic Modeling* (Springer, 2006) pp. 121–160.

[68] K. Kim, Y. Shan, X. H. Nguyen, and R. I. McKay, *Probabilistic model building in genetic programming: a critical review,* Genetic Programming and Evolvable Machines **15**, 115 (2014).

[69] R. Salustowicz and J. Schmidhuber, *Probabilistic incremental program evolution,* Evolutionary Computation **5**, 123 (1997).

[70] K. Sastry and D. E. Goldberg, *Probabilistic model building and competent genetic programming,* in *Genetic Programming Theory and Practice* (Springer, 2003) pp. 205–220.

[71] Y. Hasegawa and H. Iba, *A Bayesian network approach to program generation,* IEEE Transactions on Evolutionary Computation **12**, 750 (2008).

[72] E. Hemberg, K. Veeramachaneni, J. McDermott, C. Berzan, and U.-M. O'Reilly, *An investigation of local patterns for estimation of distribution genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2012) pp. 767–774.

[73] Y. Shan, R. I. McKay, R. Baxter, H. Abbass, D. Essam, and H. Nguyen, *Grammar model-based program evolution,* in *IEEE Congress on Evolutionary Computation (CEC)*, Vol. 1 (IEEE, 2004) pp. 478–485.

[74] P. A. N. Bosman and E. D. de Jong, *Learning probabilistic tree grammars for genetic programming,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2004) pp. 192–201.

[75] P.-K. Wong, L.-Y. Lo, M.-L. Wong, and K.-S. Leung, *Grammar-based genetic programming with Bayesian network,* in *IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2014) pp. 739–746.

[76] L. F. D. P. Sotto and V. V. de Melo, *A probabilistic linear genetic programming with stochastic context-free grammar for solving symbolic regression problems,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2017) pp. 1017–1024.

[77] Y. Hasegawa and H. Iba, *Latent variable model for estimation of distribution algorithm based on a probabilistic context-free grammar,* IEEE Transactions on Evolutionary Computation **13**, 858 (2009).

[78] A. Ratle and M. Sebag, *Avoiding the bloat with stochastic grammar-based genetic programming,* in *International Conference on Artificial Evolution (Evolution Artificielle)* (Springer, 2001) pp. 255–266.

[79] D. Thierens and P. A. N. Bosman, *Optimal mixing evolutionary algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2011) pp. 617–624.

[80] P. A. N. Bosman and D. Thierens, *Linkage neighbors, optimal mixing and forced improvements in genetic algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2012) pp. 585–592.

[81] D. Thierens and P. A. N. Bosman, *Hierarchical problem solving with the linkage tree genetic algorithm,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 877–884.

[82] B. W. Goldman and W. F. Punch, *Parameter-less population pyramid,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2014) pp. 785–792.

[83] S.-H. Hsu and T.-L. Yu, *Optimization by pairwise linkage detection, incremental linkage set, and restricted/back mixing: DSMGA-II,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2015) pp. 519–526.

2

Scalable Genetic Programming by Gene-pool Optimal Mixing

The Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) is a recently introduced model-based EA that has been shown to be capable of outperforming state-of-the-art alternative EAs in terms of scalability when solving discrete optimization problems. One of the key aspects of GOMEA's success is a variation operator that is designed to extensively exploit linkage models by effectively combining partial solutions. Here, we bring the strengths of GOMEA to Genetic Programming (GP), introducing GP-GOMEA. Under the hypothesis of having little problem-specific knowledge, and in an effort to design easy-to-use EAs, GP-GOMEA requires no parameter specification. On a set of well-known benchmark problems we find that GP-GOMEA outperforms standard GP while being on par with more recently introduced, state-of-the-art EAs. We furthermore introduce Input-space Entropy-based Building-block Learning (IEBL), a novel approach to identifying and encapsulating relevant building blocks (subroutines) into new terminals and functions. On problems with an inherent degree of modularity, IEBL can contribute to compact solution representations, providing a large potential for knock-on effects in performance. On the difficult, but highly modular Even Parity problem, GP-GOMEA+IEBL obtains excellent scalability, solving the 14-bit instance in less than 1 hour.

The contents of this chapter are based on the following publication: **M. Virgolin**, T. Alderliesten, C. Witteveen, and P.A.N. Bosman. Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*, pp. 1041-1048, ACM (2017).

2.1. Introduction

When a problem's structure has some inherent degree of modularity, being able to efficiently and effectively exploit this modularity in an Evolutionary Algorithm (EA), e.g., by recombining partial solutions, can lead to better solutions much faster than when using only blind variation operators [1]. The term *schemata* is often used in Genetic Algorithms (GAs) to refer to such partial solutions, which can be moderately to completely independent from each other.

In Black-Box Optimization (BBO), it is unknown how schemata are encoded, hence it is not possible to design any specific recombination operator beforehand that prevents their disruption when mixing solutions. In an attempt to learn and exploit problem structure, model-based EAs use a model to capture such structure [2]. In the case of BBO, model instances are inferred from the *genotype* (i.e., the encoding) of promising solutions.

In Genetic Programming (GP), the term *Building Blocks* (BBs) typically refers to connected parts of the genotype (i.e., connected nodes in tree-based GP) that represent useful subroutines. Whereas solutions in GAs have a fixed size and the main focus is to avoid the disruption of schemata, solutions in GP are typically free to grow. Therefore, many studies have explored steps to re-use BBs by encapsulating them into compact representations. With one of the first attempts, the Automatically Defined Functions (ADFs) [3], it has been shown that the re-use of BBs can be extremely beneficial, making GP capable of tackling very difficult, yet highly modular problems such as Even Parity. Many different approaches have been proposed in the last 25 years (see Sec. 5.2 of [4] for an overview). However, none of them has shown clear superiority in systematically identifying salient BBs [5]. Some works even synthesize BBs from randomly chosen subtrees [6, 7]. Other proposals relax the BBO hypothesis substantially to synthesize BBs from successful runs on smaller problem instances [8, 9].

Our purpose is to introduce novel, general, and principled ways to identify and exploit problem structure in tree-based GP. As a first contribution, we bring key strengths of the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) to tree-based GP, resulting in GP-GOMEA. GOMEA is a model-based EA which performs a memetic variation of solutions by extensively exploiting linkage information, i.e., strong interdependencies between parts of the genotype [10]. Our second contribution is a novel method to identify and encapsulate BBs into new terminals and functions in Boolean problems, thereby enhancing the search space with atomic representations of partial solutions. We call this method Input-space Entropy-based Building-block Learning (IEBL). IEBL is inspired by information theory and construction heuristics for classification trees [11], and can potentially be applied to a number of GP algorithms. To the best of our knowledge, no similar approach to identify salient BBs in GP has ever been proposed. Finally, under the hypothesis of no knowledge on the problem and for the sake of usability, we set out to design this algorithm to require no parameter specification.

2.2. GP-GOMEA

The current closest GOMEA implementation on which this GP version is based on is described in [10]. The general GOMEA outline is depicted in Algorithm 2.1. At the top level, GOMEA has the characteristics of any EA with population initialization and the

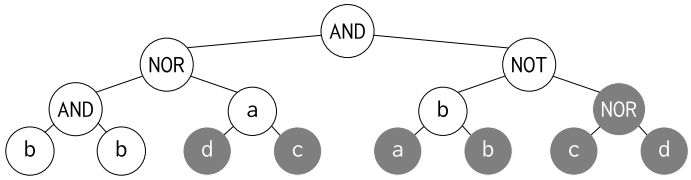

Figure 2.1: GP tree encoded by the fixed-length string of size 15 "&+&bbadc¬bab+cd". Gray nodes are introns.

generational loop that continues until a termination criterion is met (e.g., population convergence, evaluations limit, time limit). A generation consists of the learning of a linkage model F (which may be provided beforehand if the problem structure is known a priori) and the applying of the variation operator GOM to each solution in the population, which extensively exploits F to improve a solution.

Algorithm 2.1 GOMEA general outline

```
1 procedure RUNGOMEA(n)
2     P ←initializePopulation(n)
3     while ¬shouldTerminate() do
4         F ← buildLinkageModel(P)
5         for P_i ∈ P do
6             O_i ← GOM(P_i, F, P)
7         P ← O = {O_1, ..., O_n}
```

2.2.1. Genotype

Although the original implementation of GOMEA works on fixed-length strings of binary variables, handling variables of higher cardinality is straightforward. This representation is the first step in using GOMEA for GP, as we use it to map discrete values to program functions and program inputs. Like in Standard GP (SGP), solutions in GP-GOMEA are trees of variable size composed of terminal and function nodes. Trees can be encoded as fixed-length strings using preorder tree traversal (Figure 2.1). All nodes but the ones at maximum depth always have r child nodes, with r the maximum arity (i.e., number of expected inputs) of the function nodes. We make it possible for GOMEA to work with variable-size trees even though they are encoded with fixed-length strings. Trees always have a maximum height. Syntactically, trees are always full, but semantically they are not. If a terminal appears in an internal node, the subtrees below it are disregarded. Moreover, for function nodes with arity lower than r, only the leftmost child nodes are evaluated. Hence, some nodes are *introns*, i.e., they will be ignored during the evaluation of the tree.

2.2.2. Linkage models

As in the original GOMEA, GP-GOMEA uses the Family Of Subsets (FOS) as linkage model. The FOS is a set of sets which contain *loci*, i.e., indices representing positions in the geno-

type. Each one of these sets specifies which parts of the genotype should be replaced *en bloc* during variation. Note that a FOS containing all and only singletons of each locus, i.e., $F = \{\{0\}, \{1\}, \ldots, \{l-1\}\}$, with l the length of the genotype, models complete independence among loci. We call this FOS that allows only the variation of one locus at a time *Univariate* (U). In this chapter, we analyze the contribution of three different FOSs: U, *Linkage Tree* (LT) and *Random Tree* (RT).

We consider LT as it has so far been found to lead to the best performance on a number of different BBO problems [10]. A key strength of this model is that it can capture at the same time multiple levels of dependency (linkage) among loci. An LT can be seen as a tree where the leaves are singletons (i.e., U) while its internal nodes are built by merging sets in an iterative fashion, up to reaching the root, i.e., the set that contains all loci. An LT may be fixed a priori, but especially in a BBO setting, it is learned from the population at each generation (line 4 of Algorithm 2.1). Specifically, a measure of linkage between pairs of loci is measured using mutual information, i.e., the measure of mutual dependence between two variables in information theory. New sets are iteratively built by merging sets with the highest mutual information. Using only combinations of mutual information between pairs of variables, the hierarchical structure of dependencies expressed by an LT can be efficiently learned in $O(|\mathcal{P}|l^2)$, with $|\mathcal{P}|$ the population size [12].

Lastly, RT is built like LT, but using random information instead of measured linkage. This FOS enables the variation of multiple parts of the genotype like LT, but does not assume that specific parts of the genotype should be kept intact. RT is thus a model that enables blind variation that differs from the classic GP subtree crossover in that any configuration of nodes can be swapped.

2.2.3. Gene-pool optimal mixing

The variation operator GOM, that also incorporates selection, always generates an offspring that is at least as fit as the parent. Different from standard crossover in GP where entire subtrees are swapped, GOM mixes potentially unconnected parts (i.e., tree nodes) of the fixed-size genotype. Moreover, instead of generating two new solutions from two parents, it creates an offspring by iteratively mixing a parent solution with multiple other solutions.

The procedure is shown in Algorithm 2.2. Given a solution s, an identical offspring o is made, and each set F_i of the FOS F is used to try to improve o. Given F_i, a donor d is randomly picked from the population, and the symbols of o at the loci specified by F_i are replaced with the symbols of d at the same loci (the j-th symbol of o can be replaced *only* with the j-th symbol of d). Replacement is not done if F_i contains all loci, since that would mean to fully replace o with d, nor if all loci in F_i identify introns of o, because the semantic of o would not change. If the mixing results in a syntactical change of o, then this new solution is evaluated. The changes are kept only if the fitness of o does not worsen.

If o never changes during this first phase or no new best fitness has been found in the last $1 + \log_{10}(|\mathcal{P}|)$ generations, then the *forced improvement phase* starts. In this phase Optimal Mixing is performed by mixing o only with s^{elitist}, the best solution ever found. Here, changes to o are accepted only in case of strict fitness improvement. Moreover, upon an accepted change, the procedure stops. If even this does not lead to a change of o, then o becomes a copy of s^{elitist}.

Because root nodes can only be exchanged with root nodes, in the classic ramped half-and-half generation of the initial population the first symbol is always initialized to represent a function.

Algorithm 2.2 Gene-pool Optimal Mixing

```
1  function GOM(s, F, P)
2      o ← s; fitness[o] ← fitness[s]
3      b ← o; fitness[b] ← fitness[o]
4      I ←inactiveNodes(o)
5      R ←randomPermutation({0, 1, ..., |F − 1|})
6      c ← 0
7      for i ∈ {0, 1, ..., |F − 1|} do
8          F_i ← F[R[i]]
9          if |F_i| ≠ |o| & F_i ⊈ I then
10             d ←randomDonorSolution(P)
11             o_{F_i} ← d_{F_i}
12             if o ≠ d then
13                 evaluateFitness(o)
14                 if fitness[o] ≥ fitness[b] then
15                     b_{F_i} ← o_{F_i}; fitness[b] ← fitness[o]
16                     I ←inactiveNodes(o)
17                     c ← 1
18                 else
19                     o_{F_i} ← b_{F_i}; fitness[o] ← fitness[b]
20     if c = 0 | noImprovementsStretch() then
21         forcedImprovementOM(o, F)
22     return(o)
```

2.2.4. Partial evaluations

To enhance the speed of evaluating solutions a simple mechanism can be used in tree-based GP to perform partial evaluations. We use this also for GP-GOMEA. This is done by maintaining the output of all tree nodes (i.e., string symbols) in memory. Note that introns do not have any output. During GOM, track is kept of which nodes are changed. Consequently, only subtrees where at least one (active) node changed, need to be re-evaluated, whereas the roots of unchanged subtrees can immediately return their cached output.

2.2.5. Interleaved multistart scheme

The task of sizing a problem-specific population and genotype (string length or, equivalently, tree height) is crucial in many EAs. Tuning such parameters is often tedious and time-consuming but also necessary to ensure efficiency and to guarantee the successful discovery of (near-)optimal solutions. Setting these parameters wrong can give a vastly wrong impression of an algorithm's capabilities. For this reason, we designed GP-GOMEA so that it does not require the user to specify any parameter. A similar scheme as the one

Table 2.1: BB identification in IEBL. Columns are the fitness cases, O^* is the desired output, O the observed output for the tree depicted in Figure 2.1. Entropy is measured on red cells.

a	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
b	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
c	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
d	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
O	1	**1**	**1**	1	0	**0**	**0**	0	0	**0**	**0**	0	**0**	0	0	**0**
O^*	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1

proposed in [13] is adopted, where multiple runs of the algorithm with different parameter settings are interleaved. We call this scheme Interleaved Multistart Scheme (IMS).

Specifically, every g generations of a GP-GOMEA run, another run with double the population size performs 1 generation. This is repeated recursively. Similarly, the maximum tree height (and thus the encoding string length) increases by 1 every 2 runs.

The first run is initialized with a population size of 2 in ramped half-and-half, and with a maximum tree height such that full trees have a number of nodes at maximal depth equal or bigger than the number of inputs of the problem (i.e., for a problem with n inputs and functions of maximum arity r, it is $\lceil \log_r(n) \rceil$).

A copy of s^{elitist}, the best solution ever found by any run so far, is stored and used by all runs in the forced improvement phase of GOM (Algorithm 2.2, line 21). If a new best solution s^{elitist} is found the size of which (i.e., maximum tree height or, equivalently, string length) is smaller than the size of solutions evolved by a run R, then a copy of s^{elitist} is made for R that has the same size of the solutions of R and in which empty loci are filled with random introns. If the new best solution s^{elitist} has a larger size than the one of solutions evolved by R, then R is immediately terminated.

Other criteria for the termination of a run R are the following: (i) the population of R converged to all identical solutions; (ii) a run R' with larger population achieved a better average fitness than R or than a run R'' with bigger population size than R. Finally, the whole multi-run scheme can be terminated at a specific threshold by specifying a maximum number of evaluations or seconds.

2.3. IEBL

We here describe a novel method to identify and encapsulate useful, small trees into new terminals and functions, called Input-space Entropy-based Building-block Learning (IEBL). In this context, we use the term BBs to refer to such small trees. IEBL is aimed at improving the search process on Boolean problems which exhibit a degree of decomposability, i.e., for which meaningful BBs exist.

The identification of salient BBs is based on fitness cases (pairs of input and desired output values) and is inspired by information theory and heuristics to build classification trees. To the best of our knowledge, no similar approach exists in GP. While the identification method attempts to find those BBs that represent partial solutions to the problem, the encapsulation of BBs changes the search space by providing the EA with compact representations of higher-level functionalities that can be used in the search process. Moreover, IEBL can be applied iteratively, using encapsulated BBs to generate higher-order BBs.

Here, we show how IEBL can be used in GP-GOMEA, but its salient concepts can straightforwardly be used in a number of other GP paradigms. The following sections explain the method in detail.

2.3.1. Identification of BBs

A dedicated population of small trees is used. This is due to some early experiments, confirming literature [5], where we observed that the frequency of known good subtructures in the population (i.e., the `XOR` and `XNOR` functions for the Even Parity problem) does not necessarily increase during optimization. To generate the dedicated population, we use a slight variation of the ramped half-and-half method. Specifically, roots are always functions and for each tree a subset of the set of all terminal nodes $\mathcal{T}$ is used, with cardinality between 2 and $|\mathcal{T}|$. This increases the redundancy of terminals contained in candidate BBs, increasing the probability of generating complex interactions between few terminals.

Let $\mathcal{I}$ be the set of input variables. Given a BB b, we say that b embodies an input variable i if there exists at least one non-intron terminal node that represents i in b. Let $\mathcal{J} \subset \mathcal{I}$ be the set of input variables not embodied by b. We only consider BBs for which $\mathcal{J} \neq \emptyset$, as they represent partial solutions using part of the inputs. Let $\mathcal{E}$ be the set of fitness cases for which the execution of b returns a wrong Boolean output. If $\mathcal{E} = \emptyset$, b is a solution to the problem and the EA is terminated. A quality-score is assigned to b by looking at the values taken by the input variables of $\mathcal{J}$ in $\mathcal{E}$. Specifically, the joint entropy of the values assumed by the inputs of $\mathcal{J}$ in $\mathcal{E}$ is measured. Lower entropy is considered to be better because this means that the fitness cases that are still wrong have more regularities and thus represent a less complex problem to be solved.

For example, consider the task of regressing a circuit that, given 4 bits, returns 1 when an even number of bits are set to 1 (4-bits Even Parity). The BB consisting of the tree in Figure 2.1 outputs 1 only when the input variables `a` and `b` are 0. Also, $\mathcal{J} = \{\mathsf{c}, \mathsf{d}\}$. $\mathcal{E}$ contains 8 cases. Table 2.1 shows the configurations of the input values of `c` and `d` over which the entropy is computed in red. Since "01" and "10" appear each in 3 out of 8 cases, while "00" and "11" appear each 1 out of 8 cases, the entropy is $E = -\sum p \log p = -2\frac{3}{8}\log(\frac{3}{8}) - 2\frac{1}{8}\log(\frac{1}{8})$.

Some BBs are discarded during this procedure, namely those (i) that have the same output of another BB, but higher or equal entropy; (ii) for which $\mathcal{J} = \emptyset$; (iii) whose output is *always-false* or *always-true*; (iv) for which $\mathcal{I} - \mathcal{J} = \{i\}$, since the only realizable functions of i are *always-false*, *always-true*, identity and negation.

2.3.2. Encapsulation of BBs — terminal nodes

After the identification method has computed the entropy of BBs, we encapsulate into new terminals the best (i.e., with lowest entropy) $|\mathcal{I}|$ BBs, thus doubling the number of terminals. Expanding the terminal set effectively changes the search space. We limit the number of new terminals to avoid an excessive complication of the search space. If more than $|\mathcal{I}|$ BBs are found with minimal entropy, then random $|\mathcal{I}|$ ones are kept and the others are discarded.

To enable running IEBL when earlier executions already identified and encapsulated new BBs, we keep track of which input variables are embodied. This allows to always define the set $\mathcal{J}$ needed to compute the entropy.

2.3.3. Encapsulation of BBs — function nodes

The best $|\mathcal{I}|$ BBs are also used for encapsulation into new function nodes. Let r be the arity of a BB, i.e., the number of different (non-intron) terminal nodes in it. The functional encapsulation of this BB is achieved by generating a function node that accepts r children and, given one of the 2^r possible binary configurations of the inputs, returns the output of the BB for that configuration. We discard BBs leading to duplicate function nodes, i.e., those whose arity and output are identical to an already-encapsulated one or to a function from the starting set $\mathcal{F}$. Further, we discard BBs which realize *always-false*, *always-true* and functions of arity 1.

Similarly to what is done for terminal nodes, we impose a fixed limit of $|\mathcal{F}|$ new function nodes to expand $\mathcal{F}$. If more than $|\mathcal{F}|$ BBs have minimal entropy, then $|\mathcal{F}|$ at random are kept.

2.3.4. Implementation of IEBL in GP-GOMEA

To alleviate users from having to choose the dedicated population size and tree height for IEBL, we propose a scheme to include IEBL in GP-GOMEA that requires no parameter specification.

IEBL is applied at the start of each new GP-GOMEA run to expand the terminal and function sets. In particular, for the i-th run, IEBL is iterated i times consecutively to discover higher-order BBs. This means that the nodes created by the j-th iteration of IEBL are used (together with the starting functions and terminals) for the generation of the dedicated population of the $j+1$-th iteration of IEBL. For the first GP-GOMEA run, the dedicated population size for IEBL is set to $|\mathcal{F}|(|\mathcal{F}|+t)^r$, which corresponds to the number of possibilities for the first two levels of an r-ary tree with any function as root, and with any function or terminal, among t different ones, as children of the root. For the function set of the Boolean benchmark problems it is $r = 2$, and we fixed $t = 4$ to ensure starting from a moderate dedicated population size (i.e., 256 trees). The dedicated population size is doubled for each new GP-GOMEA run, and the height of the trees constituting the candidate BBs is initially set to 2 and is incremented by 1 every 4 runs. In other words, IEBL is applied *once* before the first run of GP-GOMEA, using a dedicated population size of 256 with trees of height 2; IEBL is applied i times before the i-th run, using a dedicated population size of $256 \times 2^{i-1}$ in each iteration, with trees of height $2 + \lfloor i/4 \rfloor$. If the i-th IEBL finds a BB with an entropy of 0, then the number of iterations for all next IEBLs is frozen to i. Because IEBL provides nodes that inherently embody trees of a certain height, we lower by 1 the starting tree height of the IMS of GP-GOMEA. To avoid an unbounded growth of the genotype, we limit the learning of new node functions to BBs with arity 2 only. Finally, unless a BB entropy of 0 is reached, if no lower entropy is found after four consecutive uses of IEBL (i.e., four new runs of GP-GOMEA), then IEBL is disabled: all current GP-GOMEA runs are terminated (the elite solution is also forgotten), and subsequent runs will no longer use IEBL to learn and use BBs.

2.4. Experimental setup

The performance of GP-GOMEA[1] is tested in terms of scalability on well-known GP benchmark problems. This enables us to compare our algorithm with SGP, but also with state-of-the-art work. All experiments were executed on a machine mounting 2 Intel® Xeon® CPU E5-2699 v4 @ 2.20GHz and 630 GB of RAM. Each experiment consists of 30 independent runs and only successful experiments are considered. Runs exceeding a time limit of 24 hours or a memory limit of 500 GB are aborted.

2.4.1. Benchmark problems

We consider two sets of benchmark problems. The first set is composed of artificial problems often used to assess the performance of model-based EAs, because of their focus on obtaining specific substructures in the genotype. The second set considers well-known regression problems of Boolean circuits, which differ from the problems in the first set in that the solution encoding is much more redundant, so that very different solutions have the same output. In all problems, maximization of the fitness is sought.

Artificial problems

Order is a GP version of the well-known OneMax problem in GA research and is known to be easy to solve for SGP [14]. Given a problem size n, the terminal set consists of $2n$ node variables X_i and their complement $\bar{X}_i$. The function set contains only one node which is a placeholder for a function of arity 2 with no semantic meaning. The output O of a tree is a list of its inputs derived from the inorder traversal of its nodes, such that there are no duplicates and only one of the two complementary inputs is present, depending on which is encountered first in the traversal. For example, if $\{X_3, \bar{X}_0, X_1, X_1, X_0, \bar{X}_2\}$ are the inputs appearing in the inorder traversal, the tree outputs $O = \{\bar{X}_0, X_1, \bar{X}_2, X_3\}$. The fitness is $f^n_{\text{order}} = |O \setminus \{\bar{X}_0, \bar{X}_1, \ldots, \bar{X}_{n-1}\}|$. In other words, the optimal solution is a tree where, for each i, X_i is present and $\bar{X}_i$ is not, or the latter appears after X_i according to inorder traversal.

The problem Trap employs the same terminal and function sets of Order, but is considered a hard problem for SGP [14]. This is because of its deceptive fitness function defined for blocks of k variables, which is inspired by the well-known deceptive trap functions in GA research. The deceptive attractor corresponds to the number of $\bar{X}_i$ in the output, while the needle in the haystack is the optimum of Order. Specifically, for a block of k variables,

$$f^k_{\text{Trap}} = \begin{cases} 1 & \text{if } f^k_{\text{order}} = k \\ 0.75\left(1 - \frac{f^k_{\text{order}}}{k-1}\right) & \text{if } f^k_{\text{order}} < k \end{cases} \tag{2.1}$$

We denote with Trap-3 and Trap-4 the problem with trap length k of 3 and 4 respectively. The problem size n is the number of traps.

Like Order and Trap, the tunable benchmark problem introduced in [15], here denoted with KÜ (from the authors' surnames), uses binary trees, but with a predefined maximum height. The aim is to synthesize a tree in which function nodes are arranged according to positional constraints. The size of the problem n corresponds to the maximum tree height.

[1] The code is available at `https://bit.ly/3ajDUBY`

The terminal set contains only one node, while the function set contains functions of arity 2, i.e., $\mathcal{F} = \{\mathcal{F}_0, \mathcal{F}_1, \dots\}$. $\mathcal{F}$ can be changed together with n to tune problem difficulty. In this work, we always consider $\mathcal{F} = \{\mathcal{F}_0, \dots, \mathcal{F}_{2n-1}\}$. The output of a tree is the output of its root. A function node outputs the weighted sum of its 2 children's outputs, determined according to the constraints: (i) the index of the parent function must be lower than that of its children; (ii) the index of child 1 must be smaller than the index of child 2. The terminal has no index and does not violate constraints. If child 1 violates the first constraint, then its output is penalized with a weight of η, and similarly for child 2. If the second constraint is violated, then the output of both children is penalized with η (we use $\eta = 0.25$). Since the maximum tree height *is* the problem size n, for this problem the maximum tree height in the IMS is fixed to n.

Boolean problems

These problems are defined with fitness cases, i.e., pairs of input and desired output values. Different from the artificial problems, inputs are now binary. In all these problems the aim is to synthesize a tree that realizes a Boolean circuit which satisfies all fitness cases, giving the correct output for any input configuration. The fitness of a solution is the number of correct fitness outputs. Boolean circuits we consider are: Comparator, Even Parity, Majority and Multiplexer. The terminal set contains a terminal for each input, and the function set contains the logic functions `AND`, `OR`, `NAND`, and `NOR`.

A circuit realizes the n-inputs Comparator if it outputs *true* only if the first $\lceil n/2 \rceil$ bits represent a number that is lower than the one encoded by the second $\lfloor n/2 \rfloor$ bits. The Even Parity problem of n inputs expects the output *true* when the number of input bits set to 1 is even and *false* otherwise. In Majority, the solution must return *true* only when at least $\lceil n/2 \rceil$ out of a total of n input bits are set to 1. Finally, a Multiplexer has $n = m + 2^m$ input bits: the first m bits encode the address, the second 2^m bits encode the data. A circuit satisfies the Multiplexer problem if it always outputs the value of the data bit encoded by the address.

2.4.2. Standard GP and state-of-the-art

We compare the scalability of GP-GOMEA with SGP and two state-of-the-art algorithms. For Order and Trap, the model-based algorithm extended Compact Genetic Programming (eCGP) is considered [14]. For the Boolean benchmark problems, we consider recent algorithms based on Semantic Backpropagation (SB): the Iterated Local Tree Improvement (ILTI) [16] and GP using the Random Desired Operator (RDO) [17]. These two algorithms inherently embody the partial evaluations method with which GP-GOMEA is also equipped, since they require the memorization of each node output. Differently from GP-GOMEA and SGP, these former 2 algorithms always require the definition of fitness cases.

For eCGP, we consider the scalability reported in [14]. The performance is obtained with an empirically pre-computed good population size and on a predefined maximum tree height.

Both ILTI and (GP with) RDO rely on SB, a recent technique in which improvements at the level of single fitness cases are sought. SB is applied top-down (from the root to the leaves) and computes, for each node, a desired output O^*. For the root, O^* is the vector that satisfies all fitness cases of the problem. The desired output of a child of the root

is computed by inverting the root function, using O^* and the current output of its other children as arguments. The design of inverted functions is not always straightforward, as sometimes a solution does not exist (e.g., looking for an input for AND with desired output 1 when the other input is 0) or can assume any value (e.g., looking for an input for OR with desired output 1 when the other input is 1). Among several differences, ILTI is an (1,1)-EA, while RDO is population based. Here ILTI is used in the best performing configuration, that is with a (maximum, when more are possible) library size of 450 full trees of height 2. Similarly, for RDO we adopt the best-performing configuration on Boolean problems (named RDO_p in [17]), which uses a dynamic library of semantically-unique trees taken from the subtrees in the population instead of a fixed-size library.

We propose two configurations for SGP: SGP_{param}, with hand-picked population size and initial tree height as typically done in literature, and SGP_{IMS}, enclosed in a scheme similar to the IMS of GP-GOMEA. For the former, we set the initial tree height to 6 and the maximum allowed one to 12. The population size for Order is set to 2^{n+2} (easy problem), for Trap-3 and KÜ to 2^{n+7} (difficult problem), and for Trap-4 to 2^{n+9} (very difficult problem), with n the problem size. As to the Boolean problems, the population size for Comparator and Majority is set to 2^{n+5} (medium difficulty). For Even Parity, known to be difficult for SGP [3], the population size is 2^{n+7}. For the 3, 6, and 11-bits Multiplexer the population size is set to 256, 1024, and 4096, respectively. SGP_{IMS} works with no parameter specification, within a IMS scheme that differs from the one of GP-GOMEA in the following aspects: (i) there is no common elitist solution among runs, nor a stopping criteria related to it; (ii) a run performs 1 generation every 8 generations of the smaller run; (iii) the initial tree height h is computed as the maximum tree height of GP-GOMEA (Section 2.2.5), and the maximum height is $h+4$. We experimentally found that increasing the intervals of the IMS is beneficial for SGP_{IMS}, since it performs much less evaluations than GP-GOMEA (with any of the 3 linkage models) per generation. Furthermore, we set a maximum tree height bigger than the initial tree height because the standard crossover and mutation swap and generate subtrees of arbitrary height. For both SGP configurations, we set tournament selection size to 4, probability of crossover to 0.9, probability of mutation to 0.1, and reproduction of the best solution. Finally, we equip SGP with the same caching method of node outputs used in GP-GOMEA, to perform partial evaluations.

2.5. Results & discussions

Scalability graphs are provided in Figure 2.2 for the artificial problems, and Figure 2.3 for the Boolean problems. GP-GOMEA, SGP, ILTI and RDO use partial evaluations. When IEBL is applied, evaluations of BBs candidates are also counted, and the number of nodes is obtained by recursively unwrapping encapsulated BBs.

Artificial problems

Results show that GP-GOMEA with LT as the linkage model ($\text{GP-GOMEA}_{\text{LT}}$) is generally the best performing algorithm in all metrics: number of evaluations, time, and size of the final solution. On the easy Order problem, for which we run a limited number of instances, no marked difference between both SGP configurations and GP-GOMEA with any linkage model is observed. However, the more difficult the problem, the more $\text{GP-GOMEA}_{\text{LT}}$ shows superior performance. On Trap-3, SGP_{param} markedly outperforms SGP_{IMS}, and

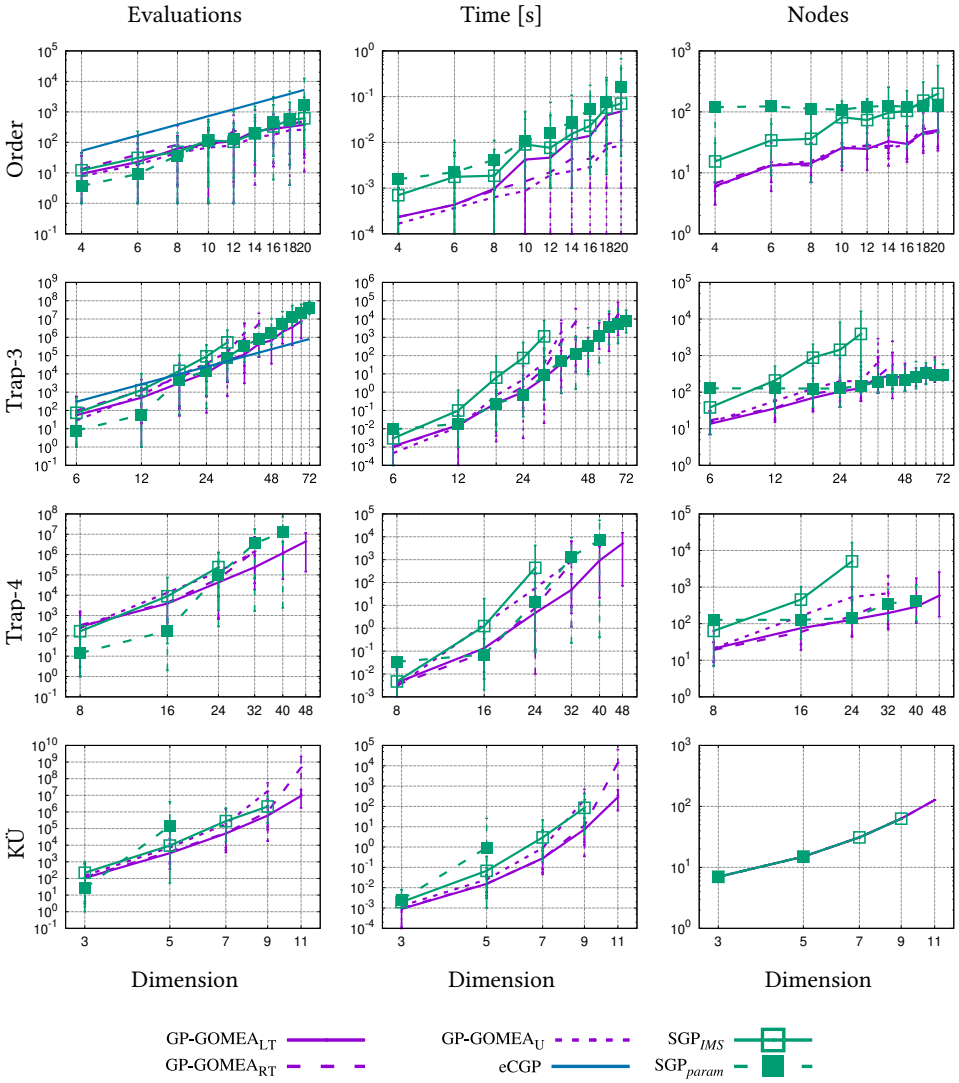

Figure 2.2: Average, maximum and minimum number of evaluations, time, and number of nodes in the final solution for the artificial problems. The problem dimension for Order, Trap-3, and Trap-4 is defined in terms of possible terminals. For KÜ, the problem dimension is defined in terms of possible functions.

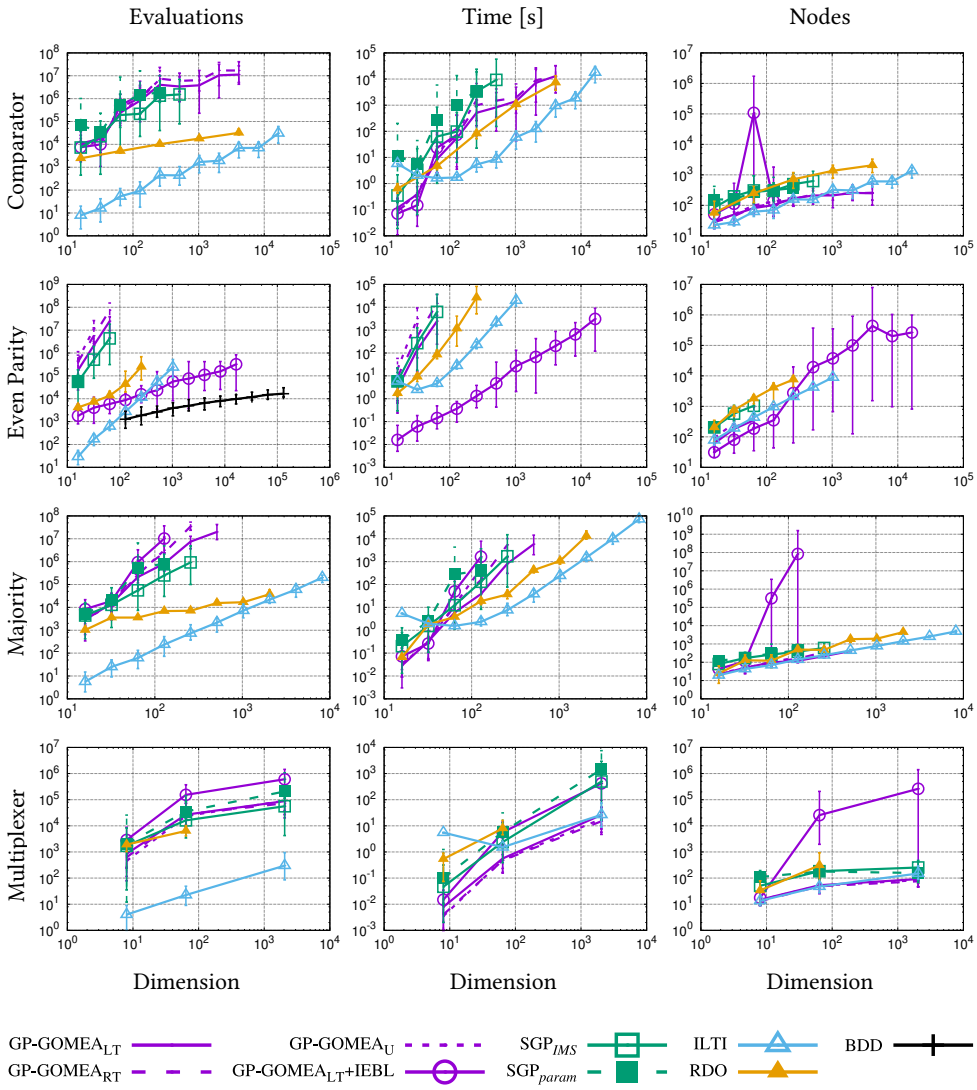

Figure 2.3: Average, maximum and minimum number of evaluations, time, and number of nodes in the final solution for the Boolean problems. The problem dimension is defined in terms of number of fitness cases.

performs slightly better than GP-GOMEA$_{\mathrm{LT}}$. This result is possibly due to the immediate employment of a big population size in SGP$_{param}$ and a good setting of the initial tree height. On Trap-4 and KÜ, however, GP-GOMEA$_{\mathrm{LT}}$ scales better than any other algorithm, showing an effective capability of learning and exploiting the problem structure. On the evaluations of Order and Trap-3, the scalability of eCGP of $O(n^{2.86})$ and $O(n^{3.18})$ respectively, as reported in [14], is shown. Although it may appear that GP-GOMEA$_{\mathrm{LT}}$ performs slightly worse than eCGP, it is important to remember that the performance of eCGP is obtained on an empirically pre-computed good population size and on a predefined initial tree height, whereas GP-GOMEA runs according to the IMS.

Boolean problems

On the Boolean problems, the difference between the linkage models LT and RT used for GP-GOMEA is far less pronounced, suggesting that the LT is not capable of modeling key linkage information to help increase efficiency substantially. This may well be because of the high redundancy in the representation of solutions, and the consequential fact that locus-based dependence is not the most important source of problem structure.

GP-GOMEA$_{\mathrm{LT}}$ and SGP$_{IMS}$ show similar scalability overall, with exception of Majority, where SGP$_{IMS}$ performs best. Nonetheless, in all other cases GP-GOMEA$_{\mathrm{LT}}$ reaches the optimal solution faster and evolves much smaller solutions. This is also reflected when comparing scalability in terms of time: partial evaluations are much more beneficial for the operators of SGP than for GOM (e.g., Figure 2.4 shows a comparison on how time scalability is affected by partial evaluations in Majority). This is because GOM exchanges multiple nodes at the same time which are not necessarily connected, requiring to re-compute the output of the chain of parent nodes scattered in the solution. Different output caching methods may thus be much more beneficial for GOM (e.g., storing hash-output of the most recurrent subtrees). Moreover, GP-GOMEA typically needs much smaller populations than SGP$_{IMS}$ thanks to the extensive mixing trials performed by GOM, which is also much less prone to bloat. These aspects are very beneficial in terms of memory usage: on difficult problems like Trap-4 some runs of SGP$_{IMS}$ need even 100 times more memory than GP-GOMEA$_{\mathrm{LT}}$.

Another crucial observation from our results comes from the comparison of SGP$_{IMS}$ with SGP$_{param}$. Whereas the former has inherent overhead due to multiple runs with increasing population and tree size, the second sometimes fails to find the optimal solution on complex problems. Although it is arguable that our parameter choice for SGP$_{param}$ is not optimal, some runs converge to a local minimum and are unable to escape it with the sole aid of mutation. Instead, the employment of multiple runs, each starting on random genetic material, dramatically increases the chances of finding the optimal solution. This also explains the success of GP-GOMEA.

The state-of-the-art algorithms ILTI and RDO generally run faster and require less evaluations than GP-GOMEA and SGP. It is worth noticing that these algorithms rely on a fitness function which is *de facto* different: in the semantic fitness, improvements at the level of single fitness cases are sought, whereas in its original specification the fitness is defined as the sum of all correct fitness cases. Therefore, SB needs a decomposition of the fitness to be defined, as well as the design of inverted functions, which compromises the applicability of this powerful technique to general problems. Moreover, whereas SGP$_{param}$

uses sub-optimal parameters, and GP-GOMEA and SGP_{IMS} run with the IMS, ILTI and RDO use the best performing configurations reported. When comparing scalability, ILTI and RDO are superior on Even Parity and Majority, but not on Comparator and Multiplexer. In Comparator, the SB-based operators cause solutions to bloat even worse than the classic operators of SGP, ultimately harming runtime. RDO has particularly poor performance on Multiplexer, being unable to escape from a local minimum within 24 hours in some runs of the 11-bits instance.

Lastly, we observe the effect of IEBL combined with the on average well-performing $\text{GP-GOMEA}_{\text{LT}}$, and see that it is either detrimental or very helpful. By encapsulating BBs into new terminal and function nodes, the search space is increased consistently. If such nodes represent partial solutions and can be readily combined into bigger partial solutions, then the search improves. Otherwise, the search is harmed. In the Multiplexer problem, we found that IEBL is not capable of learning any reasonably useful BBs, suggesting that the entropy-based identification method cannot grasp the complex relationships between bits present in this problem. In Comparator, IEBL does catch some relevant relationships: e.g., in the 6-bits instance, which outputs *true* when $\mathsf{b}_0\mathsf{b}_1\mathsf{b}_2 < \mathsf{b}_3\mathsf{b}_4\mathsf{b}_5$, the BB which returns 1 only if $\mathsf{b}_i = 0$ and $\mathsf{b}_{i+3} = 1$ is often identified. This BB represents the 2-bits Comparator between equally significant bits. As another example, the BB returning 1 when $\mathsf{b}_1\mathsf{b}_2$ are set to 1 is learned, which is part of the solution for the case $011 < 100$. However, the expansion of the search space is so big that even learning seemingly-useful material ends up being more detrimental than helpful. Similar considerations hold for the learning of new function nodes. In Majority there is no specific pattern to learn, since no specific relationships between bits are needed: the circuit outputs *true* as long as the majority of bits is set to one. Finally, in Even Parity, IEBL really leads to excellent performance, where in fact partial solutions can be combined to form bigger ones, e.g., two 2-bits Even Parity solutions can be combined with `XNOR`, which is the function performing a 2-bits Even Parity, to form the 4-bits one. Although the learning of BBs is noisy, resulting in a big performance difference among best and worst runs, the scalability is the best among all algorithms, with the number of evaluations increasing sublinearly with the number of fitness cases.

A downside of IEBL is the size of solutions, which may be lowered by implementing mechanisms to prefer shorter BBs. In the plot of the evaluations of Even Parity, we also report the best performance we are aware of, obtained by the Binary Decision Diagrams (BDD) [18]. BDD scales even better than $\text{GP-GOMEA}_{\text{LT}}$+IEBL in evaluations. However, this EA is specifically designed for Boolean problems, with a dedicated genotype (diagrams assuming a fixed variable ordering of inputs) and particular parameter settings (population size of 5, mutation-only), while $\text{GP-GOMEA}_{\text{LT}}$+IEBL is a combination of a much more general EA with a Boolean-dedicated mechanism for learning and exploiting BBs. For future work, it would be interesting to attempt to automatically detect when the addition of IEBL is useful so that highly negative effects in the performance of GP-GOMEA are prevented.

2.6. Conclusions

In this chapter we presented GP-GOMEA, a novel, scalable, model-based approach to GP. Our algorithm requires no parameter specification, which is important for making fair

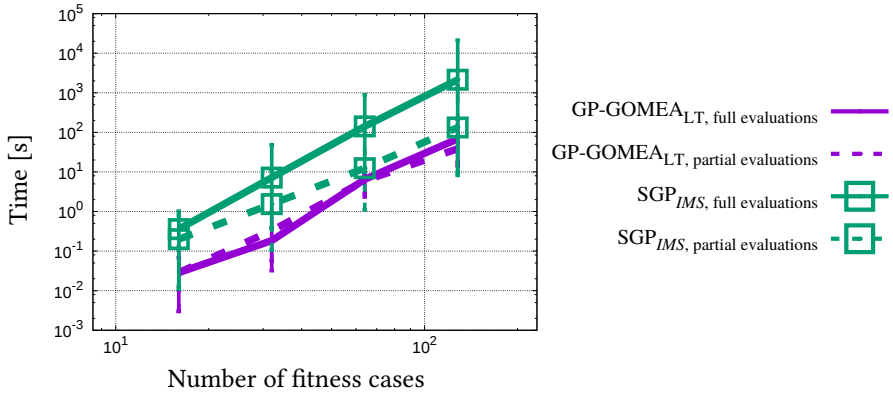

Figure 2.4: Example of the contribution of partial evaluations in terms of time for the Majority problem.

comparisons and for ease-of-use by practitioners. Even though GP-GOMEA is inherently not-tuned, it shows competitive scalability when compared with the latest state-of-the-art algorithms, based on semantic backpropagation. Moreover, while these algorithms need fitness cases and inverted functions to be defined, GP-GOMEA does not have these requirements, making it more generally applicable. Compared to SGP, GP-GOMEA exhibits superior performance on structured problems, while, in general, it evolves much smaller solutions and requires much less memory and time.

We further introduced a novel method to identify and encapsulate useful BBs into new terminals and functions, termed IEBL. The novelty of this method is that it tries to harvest information from the input-space on which fitness cases are defined. The combination of IEBL with GP-GOMEA has been shown to be detrimental in non-modular problems, but extremely efficient on a modular problem like Even Parity. In fact, IEBL helps tackling the complexity of Even Parity to a point where the scalability of GP-GOMEA becomes less than linear, which ultimately leads to solving the 14-bits instance in less than 1 hour.

Acknowledgments

The authors acknowledge the Kinderen Kankervrij foundation for personnel financial support (project #187), and the Maurits and Anna de Kock foundation for financing a high performance computing system. We thank prof. Tomasz P. Pawlak and prof. Krzysztof Krawiec for assisting with reproducing experiments of RDO.

References

[1] R. A. Watson and T. Jansen, *A building-block royal road where crossover is provably essential,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2007) pp. 1452–1459.

[2] Y. Chen, T.-L. Yu, K. Sastry, and D. E. Goldberg, *A survey of linkage learning techniques in genetic and evolutionary algorithms,* IlliGAL report **2007014** (2007).

[3] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection* (MIT Press, 1992).

[4] L. O. V. B. Oliveira, F. E. B. Otero, G. L. Pappa, and J. Albinati, *Sequential symbolic regression with genetic programming,* in *Genetic Programming Theory and Practice XII*, edited by R. Riolo, W. P. Worzel, and M. Kotanchek (Springer International Publishing, Cham, 2015) pp. 73–90.

[5] A. Dessi, A. Giani, and A. Starita, *An analysis of automatic subroutine discovery in genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (Morgan Kaufmann Publishers Inc., 1999) pp. 996–1001.

[6] P. J. Angeline and J. B. Pollack, *Coevolving high-level representations,* in *Sante Fe Institute Studies in the Sciences of Complexity*, Vol. 17 (Addison-Wesley Publishing CO, 1994) pp. 55–55.

[7] S. Roberts, D. Howard, and J. Koza, *Evolving modules in genetic programming by subtree encapsulation,* in *Genetic Programming* (Springer, 2001) pp. 160–175.

[8] D. Jackson and A. P. Gibbons, *Layered learning in Boolean GP problems,* in *European Conference on Genetic Programming* (Springer, 2007) pp. 148–159.

[9] M. Keijzer, C. Ryan, and M. Cattolico, *Run transferable libraries – learning functional bias in problem domains,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2004) pp. 531–542.

[10] D. Thierens and P. A. N. Bosman, *Hierarchical problem solving with the linkage tree genetic algorithm,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 877–884.

[11] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees* (Wadsworth, 1984).

[12] I. Gronau and S. Moran, *Optimal implementations of UPGMA and other common clustering algorithms,* Information Processing Letters **104**, 205 (2007).

[13] J. C. Pereira and F. G. Lobo, *Java implementation of a parameter-less evolutionary portfolio,* (2015), arXiv preprint arXiv:1506.08867.

[14] K. Sastry and D. E. Goldberg, *Probabilistic model building and competent genetic programming,* in *Genetic Programming Theory and Practice* (Springer, 2003) pp. 205–220.

[15] E. E. Korkmaz and G. Üçoluk, *Design and usage of a new benchmark problem for genetic programming*, in *International Symposium on Computer and Information Sciences* (Springer, 2003) pp. 561–567.

[16] R. Ffrancon and M. Schoenauer, *Memetic semantic genetic programming*, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2015) pp. 1023–1030.

[17] T. P. Pawlak, B. Wieloch, and K. Krawiec, *Semantic backpropagation for designing search operators in genetic programming*, IEEE Transactions on Evolutionary Computation **19**, 326 (2015).

[18] R. M. Downing, *Evolving binary decision diagrams using implicit neutrality*, in *IEEE Congress on Evolutionary Computation (CEC)*, Vol. 3 (IEEE, 2005) pp. 2107–2113.

3

Improving Model-based Genetic Programming for Symbolic Regression

The Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) is a model-based EA framework that has been shown to perform well in several domains, including Genetic Programming (GP). Differently from traditional EAs where variation acts blindly, GOMEA learns a model of interdependencies within the genotype, i.e., the linkage, to estimate what patterns to propagate. In this chapter, we study the role of Linkage Learning (LL) performed by GOMEA in Symbolic Regression (SR). We show that the non-uniformity in the distribution of the genotype in GP populations negatively biases LL, and propose a method to correct for this. We also propose approaches to improve LL when ephemeral random constants are used. Furthermore, we adapt a scheme of interleaving runs to alleviate the burden of tuning the population size, a crucial parameter for LL, to SR. We run experiments on 10 real-world datasets, enforcing a strict limitation on solution size, to enable interpretability. We find that the new LL method outperforms the standard one, and that GOMEA outperforms both traditional and semantic GP. We also find that the small solutions evolved by GOMEA are competitive with tuned decision trees, making GOMEA a promising new approach to SR.

The contents of this chapter are based on the following preprint: **M. Virgolin**, T. Alderliesten, C. Witteveen, and P.A.N. Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Accepted for publication in Evolutionary Computation. Preprint arXiv:1904.02050*, arXiv (2019).

3.1. Introduction

Symbolic Regression (SR) is the task of finding a function that explains hidden relationships in data, without prior knowledge on the form of such function. Genetic Programming (GP) [1] is particularly suited for SR, as it can generate solutions of arbitrary form using basic functional components.

Much work has been done in GP for SR, proposing novel algorithms [2–4], hybrids [5, 6], and other forms of enhancement [7, 8]. What is recently receiving a lot of attention is the use of so-called *semantic-aware* operators, which enhance the variation process of GP by considering intermediate solution outputs [9–11]. The use of semantic-aware operators has proven to enable the discovery of very accurate solutions, but often at the cost of complexity: solution size can range from hundreds to billions of components [9, 12]. These solutions are consequently impossible to interpret, a fact that complicates or even prohibits the use of GP in many real-world applications because many practitioners desire to understand what a solution means before trusting its use [13, 14]. The use of GP to discover uninterpretable solutions can even be considered to be questionable in many domains, as many alternative machine learning algorithms exist that can produce competitive solutions much faster [15].

We therefore focus on SR when GP is *explicitly constrained* to generate small-sized solutions, i.e. mathematical expressions consisting of a small number of basic functional components, to increase the level of interpretability. With size limitation, finding accurate solutions is particularly hard. It is not without reason that many effective algorithms work instead by growing solution size, e.g., by iteratively stacking components [11, 16].

A recurring hypothesis in GP literature is that the evolutionary search can be made effective if *salient patterns*, occurring in the representation of solutions (i.e., the genotype), are identified and preserved during variation [17]. It is worth studying if this holds for SR, to find accurate small solutions.

The hypothesis that salient patterns in the genotype can be found and exploited is what motivates the design of Model-Based Evolutionary Algorithms (MBEAs). Among them, the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) is recent EA that has proven to perform competitively in different domains: discrete optimization [18, 19], real-valued optimization [20], but also grammatical evolution [21], and, the focus of this chapter, GP [22, 23]. GOMEA embodies within each generation a model-learning phase, where *linkage*, i.e. the inter-dependency within parts of the genotype, is modeled. During variation, the linkage information is used to propagate genotype patterns and avoid their disruption.

The aim of this chapter is to understand the role of linkage learning when tackling SR, and consequently improve the GP variant of GOMEA (GP-GOMEA), to find small and accurate SR solutions for realistic problems. We present three main contributions. First, we propose an improved linkage learning approach, that, differently from the original one, is unbiased with respect to the way the population is initialized. Second, we analyze how linkage learning is influenced by the presence of many different constant values, sampled by Ephemeral Random Constant (ERC) nodes [17], and explore strategies to handle them. Third, we introduce improvements to GP-GOMEA's Interleaved Multistart Scheme (IMS), a scheme of multiple evolutionary runs of increasing evolutionary budget that executes them in an interleaved fashion, to better deal with SR and learning tasks in general.

The structure of this chapter is as follows. In Section 3.2 we briefly discuss related work on MBEAs for GP. In Section 3.3, we explain how GP-GOMEA and linkage learning work. Before proceeding with the description of the new contributions and experiments, Section 3.4 shows general parameter settings and datasets that will be used along the chapter. Next, we proceed by interleaving our findings on current limitations of GP-GOMEA followed by proposals to overcome such limitations, and respective experiments. In other words, we describe how we improve linkage learning one step at a time. In particular, Section 3.5 presents current limitations of linkage learning, and describes how we improve linkage learning. Strategies to learn linkage efficiently and effectively when ERCs are used are described in Section 3.6. We propose a new IMS for SR in Section 3.7, and use it in Section 3.8 to benchmark GP-GOMEA with competing algorithms: traditional GP, GP using a state-of-the-art semantic-aware operator, and the very popular decision tree for regression [24]. Lastly, we discuss our findings and draw conclusions in Section 3.9.

3.2. Related work

We differentiate today's MBEAs into two classes: Estimation-of-Distribution Algorithms (EDAs), and Optimal Mixing EAs (OMEAs). EDAs work by iteratively updating a probabilistic model of good solutions, and sampling new solutions from that model. OMEAs attempt to capture linkage, i.e., inter-dependencies between parts of the genotype, and proceed by variating solutions with mechanisms to avoid the disruption of patterns with strong linkage.

Several EDAs for GP have been proposed so far. References [25] and [26] are relatively recent surveys on the matter. Two categories of EDAs for GP have mostly emerged in the years: one where the shape of solutions adheres to some template to be able to estimate probabilities of what functions and terminals appear in what locations (called *prototype tree* for tree-based GP) [27–30], and one where the probabilistic model is used to sample grammars of rules which, in turn, determine how solutions are generated [31–34]. Research on EDAs for GP appears to be limited. The review of [26] says, quoting: "*Unfortunately, the latter research [EDAs for GP] has been sporadically carried out, and reported in several different research streams, limiting substantial communication and discussion*".

Concerning symbolic regression, we crucially found no works where it is attempted on realistic datasets (we searched among the work reported by the surveys and other recent work cited here). Many contributions on EDAs for GP have been validated on hard problems of artificial nature instead, such as *Royal Tree* and *Deceptive Max* [35]. Some real-world problems have been explored, but concerning only a limited number of variables [36, 37]. When considering symbolic regression, at most synthetic functions or small physical equations with only few (≤ 5) variables have been considered (e.g., by [34, 38]).

The study of OMEAs has emerged the first decade of the millennium in the field of binary optimization, where it remains mostly explored to date [39–42]. As to GP, GOMEA is the first OMEA ever brought to GP. GP-GOMEA was first introduced in [22] (Chapter 2), to tackle classic yet artificial GP benchmark problems (including some of the ones mentioned before), where the optimum is known. The IMS, largely inspired on [43], was also proposed, to relieve the user from the need of tuning the population size. Population sizing is particularly crucial for MBEAs: the population needs to be big enough for probability or linkage models to be reliable, yet small enough to allow efficient search [44].

GP-GOMEA has also seen a first adaptation to SR, to find small and accurate solutions for a clinical problem where interpretability is important [23]. There, GP-GOMEA was engineered for the particular problem, and no analysis of what linkage learning brings to SR was performed. Also, instead of using the IMS, a fixed population size was used. This is because the IMS was originally designed to enable benchmark problems to be solved to optimality (Chapter 2). No concern on generalization of solutions to unseen test cases was incorporated.

As to combining OMEAs with grammatical evolution, [21] also employed GOMEA, to attempt to learn and exploit linkage when dealing with different types of pre-defined grammars. In that work, only one synthetic function was considered for symbolic regression, among other four benchmark problems.

There is a need of assessing whether MBEAs can bring an advantage to real-world symbolic regression problems. This work attempts to do this, by exploring possible limitations of GP-GOMEA and ways to overcome them, and validating experiments upon realistic datasets with dozens of features and thousands of observations.

3.3. Gene-pool optimal mixing evolutionary algorithm for GP

Three main concepts are at the base of (GP-)GOMEA: solution representation (genotype), linkage learning, and linkage-based variation. These components are arranged in a common outline that encompasses all algorithms of the GOMEA family.

Algorithm 3.1 shows the outline of GOMEA. As most EAs, GOMEA starts by initializing a population $\mathcal{P}$, given the desired population size n^{pop}. The generational loop is then started and continues until a termination criterion is met, e.g., a limit on the number of generations or evaluations, or a maximum time. Lines 4 to 8 represent a generation. First, the linkage model is learned, which is called Family of Subsets (FOS) (explained in Sec. 3.3.2). Second, each solution $\mathcal{P}_i$ is used to generate an offspring solution $\mathcal{O}_i$ by the variation operator Gene-pool Optimal Mixing (GOM). Last, the offspring replace the parent population. Note the lack of a separate selection operator. This is because GOM performs variation and selection at the same time (see Sec 3.3.3).

For GP-GOMEA, an extra parameter is needed, the tree height (or, equivalently, tree depth) h. This is necessary to determine the representation of solutions, as described in the following Section 3.3.1.

Algorithm 3.1 Outline of GOMEA

```
1 procedure runGOMEA(n^pop)
2     P ←initializePopulation(n^pop)
3     while terminationCriteriaNotMet() do
4         F ←learnFOS(P)
5         O ← ∅
6         for i ∈ {1, ..., n^pop} do
7             O_i ←GOM(P_i, P, F)
8         P ← O
```

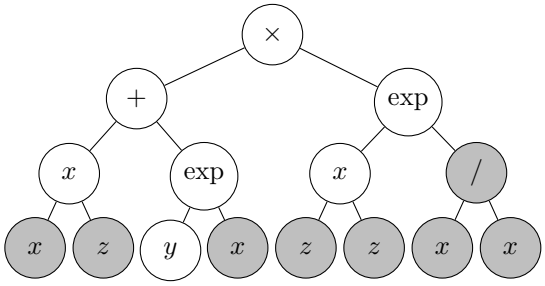

Figure 3.1: Example of tree for GP-GOMEA with $h = 3$ and $r = 2$. While 15 nodes are present, the nodes that influence the output are only 7: the gray nodes are introns.

3.3.1. Solution representation in GP-GOMEA

GP-GOMEA uses a modification of the tree-based representation [1] which is similar to the one used by [27]. While typical GP trees can have any shape, GP-GOMEA uses a fixed template, that allows linkage learning and linkage-based variation to be performed in a similar fashion as for other, fixed string-length versions of GOMEA.

All solutions are generated as *perfect* r-ary trees of height h, i.e., such that all non-leaf nodes have exactly r children, and leaves are all at maximum depth h, with r being the maximum number of inputs accepted by the functions (arity) provided in the function set (e.g., for $\{+, -, \times\}$, $r = 2$), and h chosen by the user. Note that, for any node that is not at maximum depth, r child nodes are appended anyway: no matter if the node is a terminal, or if it is a function requiring less than r inputs (in this case, the leftmost nodes are used as inputs). Some nodes are thus *introns*, i.e., they are not executed to compute the output of the tree. It follows that while trees are *syntactically* redundant, they are not necessarily *semantically* so. All trees of GP-GOMEA have the same number of nodes, equal to $\ell = \sum_{i=0}^{h} r^i = \frac{r^{h+1}-1}{r-1}$. Figure 3.1 shows a tree used by GP-GOMEA.

3.3.2. Linkage learning

The linkage model used by GOMEA algorithms is called the Family of Subsets (FOS), and is a set of sets:

$$F = \{F_1, \ldots, F_{|F|}\}, F_i \subseteq \{1, \ldots, l\}. \tag{3.1}$$

Each F_i (called FOS subset) contains indices representing locations in the genotype. For GP-GOMEA, these indices represent node locations. It is sufficient to choose a parsing order to identify the same node locations in all trees, since trees share the same shape.

In GOMEA, linkage learning corresponds to building a FOS. Different types of FOS exist in literature, however, the one recommended as default is the *Linkage Tree* (LT), by, e.g., [22, 40]. The LT captures linkage on hierarchical levels. An LT is learned every generation, from the population. To assess whether linkage learning plays a key role, i.e. whether it is better than randomly choosing linkage relations, we also consider the Random Tree (RT) [22].

Linkage tree

The LT arranges the FOS subsets in a binary tree structure representing hierarchical levels of linkage strength among genotype locations. The LT is built bottom-up, i.e., from the leaves to the root. The bottom level of the LT, i.e., the leaves, assume that all genotype locations are independent (no linkage), and is realized by instantiating FOS subsets to singletons, each containing a genotype location $i, \forall i \in \{1, \ldots, \ell\}$.

To build the next levels, mutual information is used as a proxy for linkage strength. Mutual information is a sensible choice to represent linkage strength because it expresses, considering e.g. the pair (i, j) of genotype locations as random variables, the amount of information gained on i given observations on j (and vice versa). In this light, the population can be considered as a set of realizations of the genotype. In particular, the realizations of each genotype location i are what *symbols* appear at location i in the population. In a binary genetic algorithm, symbols are either '0' or '1', while in GP, symbols correspond to the types of function and terminal nodes, e.g., '$+$','$-$','x_1','x_2'. In other words, random variables can assume as many values as there are possible symbols in the instruction set[1].

Now, the next step is to compute the mutual information between each and every pair of locations in the genotype of the entire population. Mutual information between a pair of locations can be computed after measuring entropy for single locations $\mathrm{H}(i)$, and the joint entropy for locations pairs, $\mathrm{H}(i, j)$ (this aspect will be used in Sec. 3.5):

$$\begin{aligned} \mathrm{MI}(i, j) &= \mathrm{H}(i) + \mathrm{H}(j) - \mathrm{H}(i, j), \text{where} \\ \mathrm{H}(i) = -\sum \mathrm{P}_i \log \mathrm{P}_i, &\quad \mathrm{H}(i, j) = -\sum \mathrm{P}_{ij} \log \mathrm{P}_{ij}, \end{aligned} \tag{3.2}$$

and P_i (P_{ij}) is the (joint) probability distribution over the symbols at location(s) i (i, j), which can be estimated by counting occurrences of symbol types in the population genotype. This requires to loop over the entire population, and to use nested loops over location pairs $i \in \{1, \ldots, \ell\}$ and $j \in \{i, \ldots, \ell\}$, leading to a time complexity of $O(n^{\mathrm{pop}} \ell^2)$. The contribution to the entropy of null probability cases ($-0 \log 0$) is set to 0.

Given mutual information between location pairs, we approximate linkage among higher orders of locations using the Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [45]. To ease understanding, we now provide an explanation of how UPGMA is used to build the rest of the LT that is primarily meant to be intuitive. In practice, we do not use an implementation that strictly adheres to the following explanation, but we use a more advanced algorithm that achieves the same result while having lower time complexity, called the Reciprocal Nearest Neighbor algorithm (RNN). For details on RNN, see [45].

UPGMA operates in a recursive, hierarchical fashion. Consider each singleton containing a different genotype location i as a cluster C_i, and the mutual information between location pairs as a measure of similarity S between clusters, i.e., $S(C_i, C_j) := \mathrm{MI}(i, j)$. Let $\mathcal{C}$ be the collection of clusters to be parsed, initially containing all location singletons. Every iteration, firstly a new cluster $C_{i^\star} \cup C_{j^\star}$ is formed by joining the clusters $C_{i^\star}, C_{j^\star}$ that have maximal similarity. Secondly, $C_{i^\star}$ and $C_{j^\star}$ are removed from $\mathcal{C}$, and $C_{i^\star} \cup C_{j^\star}$ is inserted in $\mathcal{C}$. When this happens, a FOS subset is added in the LT that corresponds to

[1] More symbols can be possible than the number of instructions in case ERCs are used, since instantiating an ERC in a solution results in a constant being randomly sampled.

(contains the same locations of) $C_{i^\star} \cup C_{j^\star}$, as parent of the subsets that represent $C_{i^\star}$ and $C_{j^\star}$. Thirdly, the similarity between $C_{i^\star} \cup C_{j^\star}$ and every other cluster C_k is computed, with:

$$S(C_k, C_{i^\star} \cup C_{j^\star}) = \frac{|C_{i^\star}|}{|C_{i^\star}| + |C_{j^\star}|} S(C_k, C_i) + \frac{|C_{j^\star}|}{|C_{i^\star}| + |C_{j^\star}|} S(C_k, C_j). \tag{3.3}$$

Iterations are repeated until no more merging is possible, i.e., $\mathcal{C} = \emptyset$. This necessarily happens in $2\ell - 1$ iterations. Note that the last iterations sets the root of the LT, i.e., the subset that contains all genotype locations: $\{1, \ldots, \ell\}$. Note also that the structure of the LT is related to the structure of the tree-like genotype of GP solutions only in the sense that the LT contains $2\ell - 1$ FOS subsets and the genotype has length ℓ, but it is not a one-to-one match to the structure of the genotype.

With the efficient implementation of UPGMA by RNN, the time complexity to build the LT remains bounded by $O(n^{\text{pop}}\ell^2)$.

RANDOM TREE

While linkage learning assumes an inherent structural inter-dependency to be present within the genotype that can be captured in an LT, such hypothesis may not be true. In such a scenario, using the LT might be not better than building a similar FOS in a completely random fashion. The RT is therefore considered to test this. The RT shares the same tree-like structure of the LT, but is built randomly rather than using mutual information (taking $O(\ell)$). We use the RT as an alternative FOS for GP-GOMEA.

Algorithm 3.2 Pseudocode of GOM

```
1  procedure GOM(P_i, P, F)
2      B_i ← P_i; f_{B_i} ← f_{P_i}; O_i ← P_i
3      F ←randomShuffle(F)
4      for F_j ∈ F do
5          D ←pickRandomDonor(P)
6          O_i ←overrideNodes(O_i, D, F_j)
7          if O_i ≠* B_i then
8              f_{O_i} ←computeFitness(O_i)
9              if f_{O_i} ≤ f_{B_i} then          #Assumption: minimization of f
10                 B_i ← O_i; f_{B_i} ← f_{O_i}
11             else
12                 O_i ← B_i; f_{O_i} ← f_{B_i}
13         else
14             B_i ← O_i
```

3.3.3. GENE-POOL OPTIMAL MIXING

Once the FOS is learned, the variation operator GOM generates the offspring population. GOM varies a given solution $\mathcal{P}_i$ in iterative steps, by overriding the nodes at the locations specified by each F_j in the FOS, with the nodes in the same locations taken from random

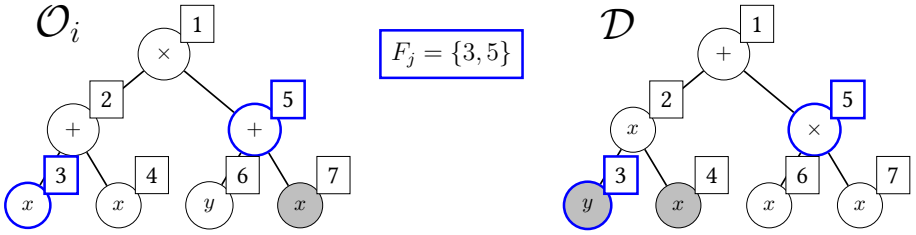

Figure 3.2: Example of variation step performed by GOM for trees with $h = 2$. Squares on top of each node indicate the node location according to pre-order traversal (depth-first). GOM replaces the nodes of $\mathcal{O}_i$ of which the location is specified by F_j, with the homologous nodes of $\mathcal{D}$ (blue contour).

donors in the population. Selection is performed within GOM in a hill-climbing fashion, i.e., variation attempts that result in worse fitness are undone.

The pseudo-code presented in Algorithm 3.2 describes GOM in detail. To begin, a backup $\mathcal{B}_i$ of the parent solution $\mathcal{P}_i$ is made, including its fitness, and similarly an offspring solution $\mathcal{O}_i = \mathcal{P}_i$ is created. Next, the FOS F is shuffled randomly: this is to provide different combinations of variation steps along the run and prevent bias. For each set of node locations F_j, a random donor $\mathcal{D}$ is then picked from the population, and $\mathcal{O}_i$ is changed by replacing the nodes specified by F_j with the homologous ones from $\mathcal{D}$. This process is exemplified in Figure 3.2. It is then assessed whether at least one (syntactic) non-intron node of the tree has been changed by variation (indicated by $\neq^\star$ in line 7). When that is not the case, $\mathcal{O}_i$ will have the same behavior as $\mathcal{B}_i$, thus the fitness is necessarily identical. Otherwise, the new fitness $f_{\mathcal{O}_i}$ is computed: if not worse than the previous one, the change is kept, and the backup is updated, otherwise the change is reversed.

Note that if a change results in $f_{\mathcal{O}_i} = f_{\mathcal{B}_i}$, the change is kept. This allows random walks in the neutral fitness landscape [46, 47]. Note also that differently from traditional subtree crossover and subtree mutation [1], GOM can change unconnected nodes at the same time, and keeps tree height limited to the initially specified parameter h. Finally, GOM *does not consider* any FOS subset that contains all node locations, i.e., $F_j = \{1, \ldots, \ell\}$, as using such subset would mean to entirely replace $\mathcal{O}_i$ with $\mathcal{D}$.

3.4. General experimental settings

We now describe the general parameters that will be used in this chapter. Table 3.1 reports the parameter settings which are typically used in the following experiments, unless specified otherwise. The notation $\mathbf{x}$ represents the matrix of feature values. We use the Analytic Quotient (AQ) [48] instead of protected division. This is because the AQ is continuous in 0 for the second operand: $x_1 \div_{\text{AQ}} x_2 := x_1/\sqrt{1 + x_2^2}$. Albeit continuity is not needed by many GP variation operators (including GOM), it is useful at prediction time: [48] show that using the AQ helps generalization (whereas using protected division does not). However, the AQ may be considered relatively hard to interpret.

As mentioned in the introduction, we focus on the evolution of solutions that are constrained to be small, to *enable* interpretability. We choose $h = 4$ because this results

Table 3.1: General parameter settings for the experiments.

Parameter	Setting
Function set	$\{+, -, \times, \div_{\text{AQ}}\}$
Terminal set	$\mathbf{x} \cup \{\text{ERC}\}$
ERC bounds	$[\min \mathbf{x}, \max \mathbf{x}]$
Initialization for GP-GOMEA	Half-and-Half as in [23]
Tree height h	4
Train-validation-test split	50%–25%–25%
Experiment repetitions	30

in relatively balanced trees with up to 31 nodes (since $r = 2$). We consider this size limitation a critical value: for the given function set, we found solutions to be already borderline interpretable for us (this is discussed further in Sec. 3.9). Larger values for h would therefore play against the aim of this study. When benchmarking GP-GOMEA in Section 3.8, we also consider $h = 3$ and $h = 5$ for completeness.

We consider 10 real-world benchmark datasets from literature [12] that can be found on the UCI repository[2] [49] and other sources[3]. The characteristics of the datasets are summarized in Table 3.2.

We use the linearly-scaled Mean Squared Error (MSE) to measure solution fitness [7], as it can be particularly beneficial when evolving small solutions. This means a fast (cost $O(n)$ with n number of dataset examples) linear regression is applied between the target y and the solution prediction $\tilde{y}$ prior to computing the MSE. We present our results in terms of variance-Normalized MSE (NMSE), i.e., $\frac{\text{MSE}(y,\tilde{y})}{var(y)}$, so that results from different datasets are on a similar scale.

To assess statistical significance when comparing the results of multiple executions of two algorithms (or configurations) on a certain dataset, we use the Wilcoxon signed-rank test [50]. This test is set up to compare competing algorithms based on the same prior conditions. In particular, we employ pairs of executions where the dataset is split into identical training, validation, and test sets for both algorithms being tested. This is because the particular split of data determines the fitness function (based on the training set), and the achievable generalization error (for the validation and test sets). We consider a difference to be significant if a smaller p-value than $0.05/\beta$ is found, with β the Bonferroni correction coefficient, used to prevent false positives. If more than two algorithms need to be compared, we first perform a Friedman test on mean performance over all datasets [50]. We use the symbols ▲, ▼ to respectively indicate significant superiority, and inferiority (absence of a symbol means no significant difference). The result *next* to the symbol ▲ (▼) signifies a result being better (worse) than the result obtained by the algorithm that has the same color of the symbol. Algorithms and/or configurations are color coded in each table reporting results (colors are color-blind safe).

[2] https://archive.ics.uci.edu/ml/index.php
[3] https://goo.gl/tn6Zxv

Table 3.2: Regression datasets used in this work.

Name	Abbreviation	# Features	# Examples
Airfoil	Air	5	1503
Boston housing	Bos	13	506
Concrete compres. str.	Con	8	1030
Dow chemical	Dow	57	1066
Energy cooling	EnC	8	768
Energy heating	EnH	8	768
Tower	Tow	25	4999
Wine red	WiR	11	1599
Wine white	WiW	11	4898
Yacht hydrodynamics	Yac	6	308

3.5. Improving linkage learning for GP

In previous work on GP-GOMEA, learning the LT was performed the same way it is done for any discrete GOMEA implementation, i.e. by computing the mutual information between pairs of locations (i, j) in the genotype (Eq. 3.2) [22]. However, the distribution of node types is typically not uniform when a GP population is initialized (e.g., function nodes never appear as leaves). In fact, this depends on the cardinality of the function and terminal sets, on the arity of the functions, and on the population initialization method (e.g., *Full*, *Grow*, *Half-and-Half*, *Ramped Half-and-Half* [51]). Note that it does not depend on the particular dataset in consideration (except in that the number of features determines the size of the terminal set). The lack of uniformity in the distribution leads to the emergence of mutual information between particular parts of the genotype. Crucially, this mutual information is natural to the solution representation, the sets of symbols and the initialization process.

If mutual information is used to represent linkage, then linkage will already be observed at initialization. However, it is reasonable to expect no linkage to be present in an initialized population, as evolution did not take place yet. Figure 3.3 shows the mutual information matrix between pairs of node locations in an initial population of $1,000,000$ solutions with maximum height $h = 2$, using *Half-and-Half*, a function set of size 4 with maximum number of inputs $r = 2$, and a terminal set of size 6 (no ERCs are used). Each tree contains exactly 7 nodes. We index node locations with pre-order tree traversal, i.e., 1 is the root, 2 its first child, 5 its second child, $3, 4$ are (leaves) children of 2, and $6, 7$ are (leaves) children of 5. Nodes at locations 2 and 5 can be functions only if a function is sampled at node 1. It can be seen that the mutual information matrix of location pairs (correctly) captures the non-uniformity in the initial distribution (i.e., larger mutual information values are present between non-leaf nodes). Using mutual information directly as a proxy for linkage may be undesirable.

3.5.1. Biasing mutual information to represent linkage

We propose to overcome the aforementioned problem by measuring linkage with a modified version of the mutual information, such that no linkage is measured at initialization.

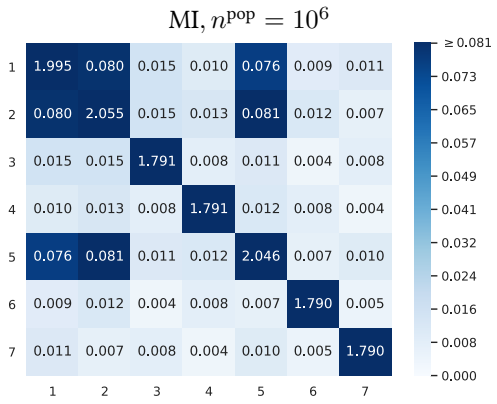

Figure 3.3: Mutual information matrix between pairs of locations in the genotype (x and y labels). Darker blue represents higher values. The matrix is computed for an initialized population of size 10^6. The values suggest the existence of linkage even though no evolution has taken place yet.

Our hypothesis is that, if we apply such a correction so that no patterns are identified at initialization, the truly salient patterns will have a bigger chance of emerging during evolution, and better results will be achieved.

Let us consider the scenario where, at initialization, symbols are uniformly distributed. For example, this typically happens in binary genetic algorithms. The mutual information between pairs of genotype locations that is expected at initialization, i.e., at generation $g = 1$ before variation and selection, will then correspond to the identity matrix: $\text{MI}^g|_{g=1} = I$ (assuming binary symbols and mutual information in bits as well as a sufficiently large population size). This mutual information matrix is suitable to represent linkage as no linkage should be present at initialization.

We propose to adopt a biased mutual information matrix $\text{MI}_b(i,j)$ to represent the linkage between a pair of genotype locations (i,j), that has the property:

$$\text{MI}_b^g(i,j)|_{g=1} = I, \tag{3.4}$$

no matter the actual distribution of the initial population.

To this end, we use Equation 3.2, i.e., we manipulate the entropy terms, to represent maximal randomness to be present at initialization for each genotype location. In particular, we propose to use biased entropy metrics such that $\text{H}_b^g(i)|_{g=1} = 1$ and $\text{H}_b^g(i,j)|_{g=1} = 2$ (for $i \neq j$) , since

$$\begin{aligned} \text{MI}_b^g(i,j)|_{g=1} &= \left(\text{H}_b^g(i) + \text{H}_b^g(j) - \text{H}_b^g(i,j)\right)|_{g=1} \\ &= 1 + 1 - 2 = 0 \qquad\qquad (\text{for } i \neq j, \text{ else } 1). \end{aligned} \tag{3.5}$$

We propose to use linear biasing coefficients β_i ($\beta_{i,j}$) to have the general biased entropy for any generation g as $\text{H}_b^g(i) = \beta_i \text{H}(i)$ and $\text{H}_b^g(i,j) = \beta_{i,j}\text{H}(i,j)$, with $\beta_i = \left(\text{H}_b^g(i)|_{g=1}\right)^{-1}$ and $\beta_{i,j} = 2\left(\text{H}^g(i,j)|_{g=1}\right)^{-1}$ to enforce maximal randomness at initialization.

To determine the beta coefficients *exactly* means to know the true distribution inferred by the sampling process used to sample the initial population, and thus the true initial entropy for each genotype location. However, this is generally not trivial to determine for GP, since a number of factors need to be considered. For example, if the *Ramped Half-and-Half* initialization method is used, what symbol is sampled at a location depends on the chance to use *Full* or *Grow*, the chance to pick the function or the terminal set based on the depth, the size of these sets, and possibly other problem-specific factors. Hence, we propose to simply approximate the β coefficients by using the $\mathrm{H}^g(i)|_{g=1}$ measured on the initial population, assuming the population to be large enough.

Summing up, the pairwise linkage estimation we propose to use at generation g, for a pair of locations (i, j), will be:

$$\mathrm{MI}^g_{\tilde{b}}(i,j) = \beta_i \mathrm{H}^g(i) + \beta_j \mathrm{H}^g(j) - \beta_{i,j} \mathrm{H}^g(i,j). \tag{3.6}$$

The tilde in $\tilde{b}$ is to remark that this is an approximation.

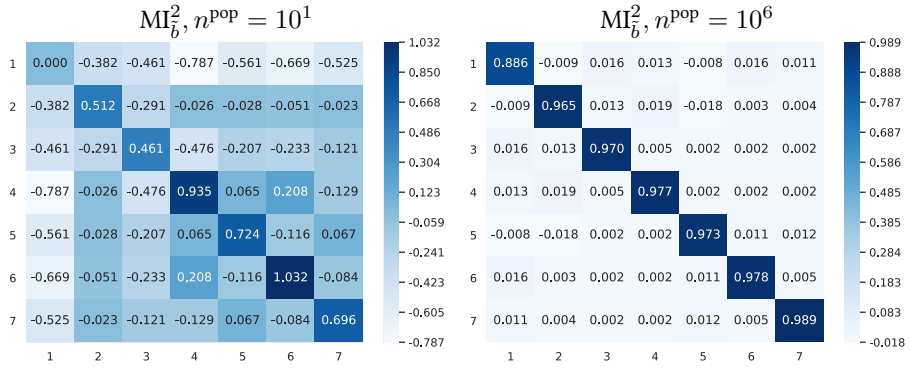

Figure 3.4: Mutual information matrices at the second generation using our biasing method to better represent linkage, with population size of 10 (left), and of 10^6 (right) for a particular run of GP-GOMEA. The rightmost matrix is closest to the identity I. A different color scaling is used in the two images.

3.5.2. Estimation of linkage by $\mathrm{MI}_{\tilde{b}}$

As a preliminary step, we observe what linkage values are obtained between pairs of genotype locations by using $\mathrm{MI}_{\tilde{b}}$. For conciseness, in the following we denote $\mathrm{MI}^g_{\tilde{b}}(i,j)|_{g=\Gamma}$ with $\mathrm{MI}^\Gamma_{\tilde{b}}(i,j)$. We show the MI matrix computed at the second generation of a GP-GOMEA run on the dataset Yac ($\mathrm{MI}^1_{\tilde{b}} = I$ by construction). We do this for two population sizes, $n^{\mathrm{pop}} = 10$ and $n^{\mathrm{pop}} = 10^6$. We expect that, the bigger n^{pop} is, the closer $\mathrm{MI}^2_{\tilde{b}}$ is to I.

We use the parameters of Table 3.1, a terminal set of size 6 (the features of Yac, no ERC) and $h = 2$, i.e. $\ell = 7$ nodes per tree. Figure 3.4 shows the biased mutual information matrix between location pairs, for the two population sizes. It can be seen that the values can be lower than 0 or bigger than 1. However, while this is particularly marked for $n^{\mathrm{pop}} = 10$, with minimum of -0.787 and maximum of 1.032, it becomes less evident for $n^{\mathrm{pop}} = 10^6$, with minimum of -0.018 and maximum of 0.989. The fact that $\mathrm{MI}^2_{\tilde{b}} \approx I$ for $n^{\mathrm{pop}} = 10^6$

is because, with such a large population size, considerable diversity is still present in the second generation.

3.5.3. Experiment: LT–MI$_{\tilde{b}}$ vs. LT–MI vs. RT

We now test the use of MI$_{\tilde{b}}$ over the standard MI for GP-GOMEA with the LT. We denote the two configurations with LT–MI$_{\tilde{b}}$ and LT–MI. We also consider the RT to see if mutual information drives variation better than random information.

We set the population size to 2000 as a compromise between having enough samples for linkage to be learned, and meeting typical literature values, which range from hundreds to a few thousands. We use the function set of Table 3.1, and a tree height $h = 4$ (thus $\ell = 31$). We set a limit of 20 generations, which corresponds to approximately 1200 generations of traditional GP, as each solution is evaluated up to $2\ell - 2$ times (size of the LT minus its root and non-meaningful changes, see Sec. 3.3.2 and 3.3.3).

The training and test NMSE performances are reported in Table 3.3. The Friedman test results in significant differences along training and test performance. GP-GOMEA with LT–MI$_{\tilde{b}}$ is clearly the best performing algorithm, with significantly lower NMSE compared to LT–MI on 8/10 datasets when training, and 7/10 at test time. It is always better than using the RT when training, and in 9/10 cases when testing. The LT–MI is comparable with the RT for these problems.

The result of this experiment is that the use of the new MI$_{\tilde{b}}$ to build the LT simply enables GP-GOMEA to perform a more competent variation than the use of MI. Also, using the LT this way leads to better results than when making random changes with the RT. Figure 3.5 shows the evolution of the training NMSE for the dataset Yac. It can be seen that the LT–MI$_{\tilde{b}}$ allows to quickly reach smaller errors than the other two FOS types. We observed similar training patterns for the other datasets (not shown here).

In the remainder, when we write "LT", we refer to LT–MI$_{\tilde{b}}$.

Table 3.3: Median NMSE of 30 runs for GP-GOMEA with LT–MI$_{\tilde{b}}$, LT–MI, and RT.

	Training			Test		
Dataset	LT–MI$_{\tilde{b}}$	LT–MI	RT	LT–MI$_{\tilde{b}}$	LT–MI	RT
Air	29.9 ▲▲	31.2 ▼	32.7 ▼	31.8 ▲▲	34.8 ▼	34.0 ▼
Bos	15.4 ▲▲	15.4 ▼▲	17.5 ▼▼	24.0 ▼▼	23.0 ▲	22.5 ▲
Con	17.5 ▲▲	18.5 ▼▲	19.0 ▼▼	18.7 ▲▲	19.6 ▼▲	20.1 ▼▼
Dow	20.9 ▼▲	20.3 ▲▲	24.0 ▼▼	22.6 ▼▲	21.1 ▲▲	26.0 ▼▼
EnC	8.42 ▲▲	9.68 ▼▼	9.09 ▼▲	9.18 ▲▲	10.7 ▼▼	10.3 ▼▲
EnH	6.24 ▲▲	6.44 ▼▼	6.40 ▼▲	6.50 ▲▲	7.10 ▼▼	6.70 ▼▲
Tow	12.5 ▼▲	12.5 ▲▲	13.1 ▼▼	13.0 ▲	12.8 ▲	13.2 ▼▼
WiR	60.3 ▲▲	60.9 ▼▲	61.2 ▼▼	62.5 ▲▲	63.0 ▼	63.1 ▼
WiW	68.1 ▲▲	68.4 ▼	68.7 ▼	69.1 ▲▲	69.7 ▼▼	69.5 ▼▲
Yac	0.34 ▲▲	0.37 ▼	0.36 ▼	0.58 ▲▲	0.62 ▼▼	0.62 ▼▲

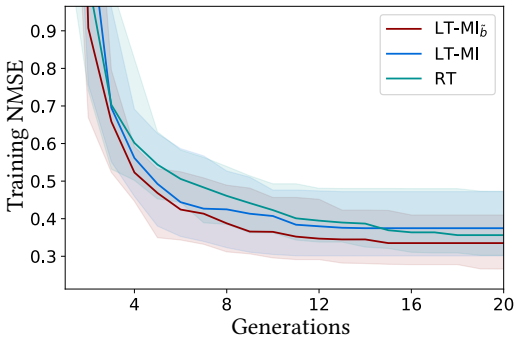

Figure 3.5: Median fitness of the best solution of 30 runs on Yac, for LT–MI$_{\tilde{b}}$, LT–MI, and RT (10^{th} and 90^{th} percentiles in shaded area).

3.5.4. Experiment: assessing propagation of node patterns

The previous experiment showed that using linkage-driven variation (LT) can be favorable compared to random variation (RT). This seems to confirm the hypothesis that, in certain SR problems, salient underlying patterns of nodes exist in the genotype that can be exploited. Another aspect that can be considered with respect to such hypothesis is how final solutions look: if linkage learning leads to the propagation of salient node patterns, different runs might result in similar solutions.

Therefore, we now want to assess whether the use of the LT has a bigger chance to lead to the discovery of a particular best-of-run solution, compared to the use of the RT. We use the same parameter setting as described in Section 3.5.3, but perform 100 repetitions. While each run uses a different random seed (e.g., for population initialization), we fix the dataset split, as changing the training set results in changing the fitness function. We repeat the 100 runs on 5 random dataset splits, on the smallest dataset Yac. Together with $n^{\text{pop}} = 2000$ as in the previous experiment, we also consider a doubled $n^{\text{pop}} = 4000$.

Table 3.4 reports the number of best found solutions that have at least one duplicate, i.e. their genotype is semantically equivalent (e.g., $x_1 + x_2 = x_2 + x_1$), along different runs for 5 random splits of Yac (semantic equivalence was determined by automatic tests[4] followed by manual inspection). It can be seen that the LT finds more duplicate solutions than the RT, by a margin of around 30% (difference between averages). Figure 3.6 shows the distribution of solutions found for the second dataset split with $n^{\text{pop}} = 4000$, i.e. where both the LT and the RT found a large number of duplicates. The LT has a marked chance of leading to the discovery of a particular solution, up to one-fourth of the times. When the RT is used, a same solution is found only up to 6 times out of 100.

This confirms the hypothesis that linkage-based variation can propagate salient node patterns more than random variation should such patterns exist, enhancing the likelihood of discovering particular solutions.

[4]Including the use of symbolic simplification with `https://andrewclausen.net/computing/deriv.html`

Table 3.4: Percentage of best solutions with duplicates found by GP-GOMEA with LT and RT for different splits of Yac.

	$n^{\text{pop}} = 2000$		$n^{\text{pop}} = 4000$	
Split	LT	RT	LT	RT
1	36	18	44	15
2	42	12	49	21
3	40	7	43	8
4	43	8	45	25
5	36	16	49	16
Avg.	39	12	46	17

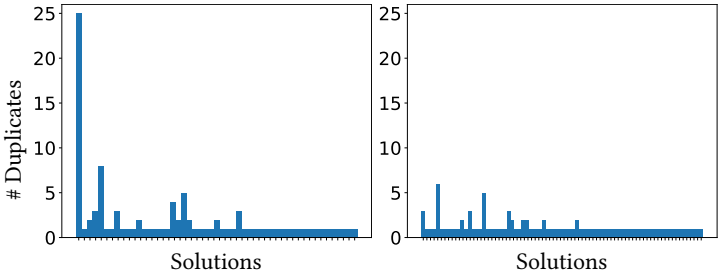

Figure 3.6: Distribution of best found solutions for 100 runs by using the LT (left) and the RT (right) with $n^{\text{pop}} = 4000$ on the second dataset split of Yac.

3.6. Ephemeral random constants & linkage

In many GP problems, and in particular in SR, the use of ERCs can be very beneficial [17]. An ERC is a terminal which is set to a constant only when instantiated in a solution. In SR, this constant is commonly sampled uniformly at random from a user-defined interval.

Because every node instance of ERC is a different constant, linkage learning needs to deal with a large number of different symbols. This can lead to two shortcomings. First, a very large population size may be needed for salient node patterns to emerge. Second, data structures used to store the frequencies of symbols grow really big and become slow (e.g., hash maps). We explore three strategies to deal with this:

- all-const: Ignore the shortcomings, and consider all different constants as different symbols during linkage learning;
- no-const: Skip all constants during linkage learning, i.e. set their frequency to zero. This approximation is reasonable since all constants are unique at initialization, and the respective frequency is almost zero. However, during evolution some constants will be propagated while others will be discarded, making this approximation less and less accurate over time;
- bin-const: Perform on-line binning. We set a maximum number γ of constants to consider. After γ different constants have been encountered in frequency counting, any further constant is considered to fall into the same bin as the closest constant among the first γ. The closest constant can be determined with binary search in

$\log_2(\gamma)$ steps. Contrary to strategy no-const, we expect the error of this approximation to lower over time, because selection lowers diversity, meaning that the total number of different constants will be reduced as generations pass.

3.6.1. Experiment: linkage learning with ERCs

We use the same parameter setup of the experiment in Section 3.5.3, this time adding an ERC terminal to the terminal set. We compare the three strategies to handle ERCs when learning the LT. For this experiment and in the rest of the chapter, we use $\gamma = 100$ in bin-const. We observed that for problems with a small number of features (e.g., Air and Yac), i.e., where ERC sampling is more likely and thus more constants are produced, this choice reduces the number of constant symbols to be considered by linkage learning in the first generations by a factor of ~ 50. We also report the results obtained with the RT as a baseline, under the hypothesis that using ERCs compromises linkage learning to the point that random variation becomes equally good or better.

The results of this experiment are shown in Table 3.5 (training and test NMSE) and Table 3.6 (running time). The Friedman test reveals significant differences among the configurations for train, test, and time performance. Note that the use of ERCs leads to lower errors compared to not using them (compare with Table 3.3).

In terms of training error, the RT is always outperformed by the use of the LT, no matter the strategy. The all-const strategy is significantly better than no-const in half of the problems, and never worse. Overall, bin-const performs best, with 6 out of 10 significantly better results than all-const. The fact that all-const can be outperformed by bin-const supports the hypothesis that linkage learning can be compromised by the presence of too many constants to consider, which hide the true salient patterns. Test results are overall similar to the training ones, but less comparisons are significant.

In terms of time, all-const is almost always significantly worse than the other methods, and often by a consistent margin. This is particularly marked for problems with a small number of features (i.e., Air, Yac). There, more random constants are present in the initial population, since the probability of sampling the ERC from the terminal set is inversely proportional to the number of features.

Table 3.5: Median training NMSE and median test NMSE of 30 runs for GP-GOMEA with the LT using the three strategies all-const, no-const, bin-const, and with the RT.

	Training NMSE				Test NMSE			
Dataset	all-const	no-const	bin-const	RT	all-const	no-const	bin-const	RT
Air	27.7 ▼▲	28.0 ▼▲	27.5 ▲▲▲	31.4 ▼▼▼	28.7 ▲▼▲	29.6 ▼▼▲	27.8 ▲▲▲	32.5 ▼▼▼
Bos	15.2 ▼▲	15.3 ▲	15.0 ▲ ▲	17.6 ▼▼▼	24.2 ▼▼	23.2 ▼▲	21.8 ▲▲▲	24.2 ▲▲▼
Con	17.2 ▲▼▲	17.2 ▼▼▲	17.0 ▲▲▲	18.5 ▼▼▼	18.5 ▲	18.7 ▲	18.8 ▲	19.8 ▼▼▼
Dow	21.4 ▼▲	21.1 ▲	20.7 ▲ ▲	24.5 ▼▼▼	22.8 ▼ ▲	21.9 ▲▲▲	22.5 ▼▲	25.5 ▼▼▼
EnC	5.51 ▲▲▲	5.72 ▼ ▲	5.76 ▼ ▲	6.44 ▼▼▼	6.18 ▲ ▲	6.34 ▼▼▲	6.00 ▲▲	6.77 ▼▼▼
EnH	3.00 ▲▼▲	3.14 ▼▼▲	2.80 ▲▲▲	4.10 ▼▼▼	3.28 ▲▼▲	3.33 ▼▼▲	3.11 ▲▲▲	4.67 ▼▼▼
Tow	12.3 ▼▲	12.2 ▲	12.3 ▲ ▲	13.2 ▼▼▼	12.9 ▲	12.8 ▲	12.8 ▲	13.5 ▼▼▼
WiR	60.3 ▲▲	60.2 ▲	60.2 ▼ ▲	61.2 ▼▼▼	63.6 ▲	62.9	62.9 ▲	63.2 ▼ ▼
WiW	67.6 ▲▲▲	68.1 ▼▼▲	68.0 ▼▲▲	68.5 ▼▼▼	68.9 ▲	69.0 ▲	69.4 ▲	69.9 ▼▼▼
Yac	0.32 ▲▲▲	0.35 ▼▼▲	0.34 ▼▲▲	0.38 ▼▼▼	0.55 ▲	0.61 ▼▲	0.52 ▲▲	0.63 ▼▼▼

Table 3.6: Median time of 30 runs for GP-GOMEA with the LT using the three strategies all-const, no-const, bin-const, and with the RT.

	Time (s)			
Dataset	all-const	no-const	bin-const	RT
Air	355.4 ▼▼▼	71.4 ▲▲▲	80.0 ▲▼▲	80.1 ▲▼▼
Bos	63.4 ▼▼▼	29.4 ▲▲▼	30.9 ▲▼▼	24.5 ▲▲▲
Con	154.9 ▼▼▼	56.7 ▲▲	59.8 ▲▼	58.4 ▲
Dow	53.8 ▼▲▼	51.7 ▲▼	54.9 ▼▼▼	37.7 ▲▲▲
EnC	147.2 ▼▼▼	40.5 ▲▲▲	43.5 ▲▼▲	45.6 ▲▼▼
EnH	145.0 ▼▼▼	45.8 ▲▲	49.4 ▲▼▼	45.7 ▲ ▲
Tow	255.9 ▼▼▼	246.6 ▲ ▼	245.6 ▲ ▼	233.9 ▲▲▲
WiR	126.1 ▼▼▼	67.7 ▲▲	80.2 ▲▼▼	70.1 ▲ ▲
WiW	285.0 ▼▼▼	213.3 ▲▲▲	237.2 ▲▼▼	224.1 ▲▼▲
Yac	236.5 ▼▼▼	23.9 ▲▲▼	24.8 ▲▼▼	22.8 ▲▲▲

Interestingly, despite the lack of a linkage-learning overhead, using the RT is not always the fastest option. This is because random variation leads to a slower convergence of the population compared to the linkage-based one, where salient patterns are quickly propagated, and less variation attempts result in changes of the genotype that require a fitness evaluation (see Sec. 3.3.3). The slower convergence caused by the RT can also be seen in Figure 3.5 (for the previous experiment), and was also observed in other work, in terms of diversity preservation [52].

Between the LT-based strategies, the fastest is no-const, at the cost of a bigger training error. Although consistently slower than no-const, bin-const is still quite fast, and achieves the lowest training errors. We found bin-const to be preferable in test NMSE as well. In the following, we always use bin-const, with $\gamma = 100$.

3.7. Interleaved multistart scheme

The Interleaved Multistart Scheme (IMS) is a wrapper for evolutionary runs largely inspired by [43]. It works by interleaving the execution of several runs of increasing resources (e.g., population size). The main motivation for using the IMS is to make an EA much more robust to parameter settings, and alleviate the need for practitioners to tinker with them. In fact, the whole design of GP-GOMEA attempts to promote ease-of-use and robustness: the EA has no need for parameters that specify how to conduct variation (e.g., crossover or mutation rates), nor how to conduct selection (e.g., tournament size). The IMS or similar schemes are often used with MBEAs [41, 53], where population size plays a crucial role in determining the quality of model building. Note that although the IMS has potential to be parallelized, here it is used in a sequential manner.

An IMS for GP-GOMEA was first proposed in Chapter 2, and its outline is as follows. A collection of parameter settings σ_{base} is given as input, which will be used in the first run R_1. The IMS runs until a termination criterion is met (e.g., number of generations, time budget). The run R_i performs one generation if no run that precedes it exists (e.g., because it is the first run or because all previous runs have been terminated), or if the previous run R_{i-1} has executed g generations. The first time R_i is about to execute a generation, it is initialized using the parameter settings σ_{base} scaled by the index i. For example, the

population size can be set to $2^{i-1}n_{\text{base}}^{\text{pop}}$ (i.e., doubling the population size of the previous run). Finally, when a run completes a generation, a check is done to determine if the run should be terminated (explained below).

3.7.1. An IMS for supervised learning tasks

The first implementation of the IMS for GP-GOMEA was designed to deal with GP benchmark problems of pure optimization (Chapter 2). That implementation therefore scaled both the population size and the height of trees in an attempt to find the optimal solution (of unknown size). In this work, we use the IMS as follows. (i) *Scaling of parameter settings*: We scale only the population size. For run R_i, the population size is set to $n_i^{\text{pop}} = 2^{i-1}n_{\text{base}}^{\text{pop}}$. (ii) *Run termination*: A run R_i is terminated if the fitness of its best solution is worse than the one of a run R_j initialized later, i.e., with $j > i$, or if it converges to all identical solutions. Differently from Chapter 2, we no longer scale the tree height h because in SR, and in supervised learning tasks in general, no optimum is known beforehand, and it is rather desired to find a solution that generalizes well to unseen examples. Moreover, h bounds the maximum solution size, which influences interpretability. Hence h is left as a parameter for the user to set, and we recommend $h \leq 4$ to increase the chance that solutions will be interpretable (see Sec. 3.9).

We set the run termination criteria to be based upon the fitness of best solutions instead of mean population fitness as done by [43] and in Chapter 2, because in SR it can happen that the error of a few solutions becomes so large that it compromises the mean population fitness. This can trigger the termination criteria even if solutions exist that are competitive with the ones of other runs. Also different from the other versions of the IMS, when terminating a run, we do not automatically terminate all previous runs. Indeed, some runs with smaller parameter settings may still be very competitive (e.g., due to the fortunate sampling of particular constants when using ERCs).

We lastly propose to exploit the fact that many runs are performed within the IMS to tackle a central problem of learning tasks: generalization. Instead of discarding the best solutions of terminating runs, we store them in an archive. When the IMS terminates, we re-compute the fitness of each solution in the archive using a set of examples different from the training set, i.e. the validation set, and return the new best performing, i.e., the solution that generalized best. The final test performance is measured on a third, separate set of examples (test set).

3.8. Benchmarking GP-GOMEA

We compare GP-GOMEA (using the new LT) with tree-based GP with traditional subtree crossover and subtree mutation (GP-Trad), tree-based GP using the state-of-the-art, semantic-aware operator Random Desired Operator (GP-RDO) [9], and Decision Tree for Regression (DTR) [24].

We consider RDO because, as mentioned in the introduction, semantic-aware operators have been studied with interest in the last years. Several works either built upon RDO, or used RDO as a baseline for comparison (see, e.g., [10, 54, 55]). Yet, consistently large solutions were found. It is interesting to assess how RDO fares when rather strict solution size limits are enforced. Because of such limits, we remark we cannot consider another

popular set of semantic-aware operators, i.e., the operators used by Geometric Semantic Genetic Programming (GSGP) [11]. These operators work by stacking entire solutions together, necessarily causing extremely large solution growth (even if smart simplifications are attempted [12]).

We consider DTR because it is considered among the state-of-the-art algorithms to learn interpretable models [14, 56]. We remark that DTR ensembles (e.g., [16, 57]) are typically markedly more accurate than single DTRs, but are considered not interpretable.

3.8.1. EXPERIMENTAL SETUP

For the EAs, we use a fixed time limit of $1,000$ seconds[5]. We choose a time-based comparison because GP-GOMEA performs more evaluations per generation than other GP algorithms (up to $2\ell-2$ evaluations per generation with the LT), and so that the overhead of learning the LT (which does not involve evaluations) is taken into account.

We consider maximum solution sizes $\ell = 15, 31, 63$ (tree nodes), i.e. corresponding to $h = 3, 4, 5$ respectively, for full r-ary trees. The EAs are run with a typical fixed population size $n^{\text{pop}} = 1000$ and also with the IMS, considering three values for the number of generations in between runs g: 4, 6, and 8. For the fixed population size, if the population of GP-GOMEA converges before the time limit, since there is no mutation, it is randomly re-started. Choices of g between 4 and 8 are standards from literature [20, 22].

Our implementation of GP-Trad and GP-RDO mostly follows the one of [9]. The population is initialized with the *Ramped Half-and-Half* method, with tree height between 2 and h. Selection is performed with tournament of size 7. GP-Trad uses a rate of 0.9 for subtree crossover, and of 0.1 for subtree mutation. GP-RDO uses the population-based library of subtrees, a rate of 0.9 for RDO, and of 0.1 for subtree mutation. Subtree roots to be variated are chosen with the *uniform depth mutation* method, which makes nodes of all depths equally likely to be selected [9]. Elitism is ensured by cloning the best solution into the next generation. All EAs are implemented in C++, and the code is available at: `https://goo.gl/15tMV7`.

For GP-Trad we consider two versions, to account for different types of solution size limitation. In the first version, called GP-Trad^h, we force trees to be constrained within a maximum height ($h = 3, 4$), as done for GP-GOMEA. This way, we can see which algorithm searches better in the same representation space. In the second version, GP-Trad$^\ell$, we allow more freedom in tree shape, by only bounding the number of tree nodes. This limit is set to the maximum number of nodes obtainable in a full r-ary tree of height h ($\ell = 15$ for $h = 3$, $\ell = 31$ for $h = 4$). As indicated by previous literature [58, 59], and as will be shown later in the results, GP-Trad$^\ell$ outperforms GP-Trad^h. We found that the same holds also for GP-RDO, and present here only its best configuration, i.e., a version where the number of tree nodes is limited like for GP-Trad$^\ell$.

We use the Python Scikit-learn implementation of DTR [60], with 5-fold cross-validation grid-search over the training set to tune the following hyper-parameters: *splitter* $\in$ {'*best*','*random*'}; *max_features* $\in \{\frac{1}{2}, \frac{3}{4}, 1\}$; *max_depth* $\in$ {3,4,5,6} (documentation available at `http://goo.gl/hbyFq2`). We do not allow larger depth values because, like for GP solutions, excessively large decision trees are uninterpretable. The best generalizing model found by cross-validation is then used on the test set.

[5] Experiments were run on an Intel® Xeon® Processor E5-2650 v2.

Table 3.7: Median validation and test NMSE of 30 runs with $\ell = 15$ for GP-GOMEA (G), GP-Trad^h (T$^\ell$), GP-Trad^h (T$^\ell$), GP-RDO (R) with $n^{\text{pop}} = 1000$ and IMS with $g \in \{4, 6, 8\}$, and DTR. Significance is assessed within each population scheme w.r.t. GP-GOMEA. The last row reports the number of times the EA performs significantly better (B) and worse (W) than GP-GOMEA.

Validation $\ell = 15$

	$n^{\text{pop}} = 1000$				IMS $g = 4$				IMS $g = 6$				IMS $g = 8$				
	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	D
Air	39.2	40.6▼	35.0▲	44.0▼	34.7	38.3▼	31.4▲	42.5▼	34.9	39.7▼	33.6▲	42.0▼	34.4	39.4▼	32.0▲	42.3▼	31.1▲
Bos	21.1	23.4	25.3▼	25.7▼	18.2	21.2▼	19.0▼	20.8▼	19.2	21.2▼	20.4	20.9▼	19.4	21.6▼	19.9▼	22.5▼	22.9▼
Con	23.2	25.3▼	23.4	27.0▼	20.3	23.1▼	19.4▲	26.4▼	20.2	23.3▼	19.9▲	26.2▼	19.4	23.2▼	19.3	26.9▼	22.7▼
Dow	26.7	28.5▼	27.5	30.6▼	24.2	26.8▼	24.2	32.3▼	24.6	26.4▼	24.8▼	31.0▼	24.5	26.3▼	25.2▼	31.0▼	30.6▼
EnC	8.72	10.6▼	7.34	11.0▼	5.86	10.2▼	6.49▼	10.7▼	6.01	10.3▼	6.24	10.5▼	5.87	10.2▼	6.10▼	10.8▼	4.23▲
EnH	4.95	7.45▼	3.83▲	7.65▼	3.33	7.19▼	3.74▼	7.34▼	3.28	7.30▼	3.76▼	7.42▼	3.23	7.24▼	3.72▼	7.54▼	0.43▲
Tow	12.9	14.4▼	13.9▼	20.1▼	12.8	13.6▼	13.6▼	20.5▼	12.7	14.0▼	13.5▼	20.4▼	13.0	14.0▼	13.4▼	20.1▼	11.2▲
WiR	65.3	64.8	64.9	66.5▼	63.9	64.7▼	64.4	65.1▼	63.6	63.9▼	64.4▼	64.9▼	63.9	63.9▼	64.2	65.7▼	71.7▼
WiW	71.4	71.3	70.9	72.6▼	70.8	71.2▼	70.7▼	72.3▼	70.7	71.5▼	70.8▼	72.6▼	71.2	71.4▼	71.2▼	72.6▼	72.2▼
Yac	1.25	1.22	0.70▲	0.96▲	0.89	1.04▼	0.61▲	0.67▲	0.92	1.01▼	0.61▲	0.73▲	0.95	1.03▼	0.62▲	0.76▲	0.88▲
B/W	–	0/6	3/2	1/9	–	0/10	3/5	1/9	–	0/10	3/5	1/9	–	0/10	2/6	1/9	5/5

Test $\ell = 15$

	$n^{\text{pop}} = 1000$				IMS $g = 4$				IMS $g = 6$				IMS $g = 8$				
	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	D
Air	38.5	40.7▼	35.3▲	44.1▼	35.8	39.5▼	32.5▲	43.3▼	35.2	39.3▼	33.2▲	42.4▼	35.4	39.7▼	32.8▲	42.8▼	30.8▲
Bos	22.7	23.3▼	24.3▼	26.7▼	22.5	23.1	23.3▼	22.9▼	21.7	23.6▼	22.6▼	25.0▼	22.1	22.5▼	23.6▼	23.3▼	26.1▼
Con	23.1	26.1▼	23.9	27.0▼	20.8	23.9▼	19.3▲	27.7▼	21.2	23.9▼	19.9▲	26.4▼	20.4	24.3▼	19.3▲	27.8▼	21.3▼
Dow	26.3	27.5▼	26.4	31.0▼	24.8	26.1▼	24.7▲	30.7▼	24.5	26.6▼	24.5	30.1▼	24.3	26.8▼	25.1	31.6▼	28.0▼
EnC	9.72	11.2▼	7.86▲	11.8▼	6.36	10.6▼	6.80▼	11.5▼	6.37	10.5▼	6.18	10.9▼	6.02	10.5▼	6.33▼	11.7▼	4.47▲
EnH	5.03	7.19▼	4.04▲	7.85▼	3.45	7.57▼	3.88▼	7.62▼	3.28	7.64▼	3.88▼	7.59▼	3.51	7.56▼	3.86▼	7.65▼	0.33▲
Tow	13.4	14.4▼	13.9▼	20.3▼	13.0	14.1▼	14.0▼	20.8▼	12.9	14.1▼	13.7▼	20.7▼	13.0	14.3▼	13.3▼	20.5▼	11.2▲
WiR	63.1	63.7▼	62.4	64.6▼	63.3	63.4	63.2	64.4▼	63.6	63.5	63.2▲	64.3▼	63.4	63.8	63.3	63.7▼	72.6▼
WiW	70.5	70.5	70.1	71.3▼	70.4	70.0▼	70.3▼	71.6▼	69.7	70.5▼	70.1▼	71.0▼	70.2	70.5▼	70.3▼	71.8▼	72.2▼
Yac	1.23	1.23	0.78▲	0.95▲	1.16	1.24▼	0.73▲	0.77▲	1.17	1.24	0.73▲	0.77▲	1.17	1.23▼	0.71▲	0.86▲	0.91▲
B/W	–	0/8	4/2	1/9	–	0/8	4/5	1/9	–	0/8	4/4	1/9	–	0/9	3/5	1/9	5/5

Table 3.8: Median validation and test NMSE of 30 runs with $\ell = 31$. Details as in Table 3.7.

Validation $\ell = 31$

	$n^{\text{pop}} = 1000$				IMS $g = 4$				IMS $g = 6$				IMS $g = 8$				
	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	D
Air	26.4	33.8▼	32.1▼	36.0▼	24.9	25.9▼	23.2▲	37.0▼	25.0	27.1▼	24.9	37.6▼	24.8	28.4▼	24.8	37.1▼	31.1▼
Bos	22.3	21.1	19.2	24.8▼	16.7	18.8▼	16.2▼	20.4▼	17.4	18.3▼	17.3▼	20.6▼	17.3	18.4▼	17.6▼	20.4▼	22.9▼
Con	17.3	18.6▼	17.9▼	20.8▼	16.0	17.6▼	16.7▼	20.5▼	16.6	18.1▼	16.4▼	20.1▼	16.1	18.3▼	17.2▼	20.2▼	22.7▼
Dow	21.3	22.6▼	22.6	24.3▼	19.4	21.6▼	19.2	25.6▼	19.4	21.2▼	19.4▼	25.4▼	19.2	21.9▼	20.1▼	25.8▼	30.6▼
EnC	5.14	5.60▼	4.99▲	7.62▼	4.62	5.51▼	4.82▼	8.04▼	4.35	6.04▼	4.56▼	8.48▼	4.37	5.65▼	4.73▼	7.81▼	4.23▲
EnH	2.29	2.54▼	1.75▲	6.21▼	1.95	3.05▼	1.72▲	4.97▼	2.00	2.84▼	1.65▲	5.93▼	1.88	3.10▼	1.62	6.11▼	0.43▲
Tow	12.0	13.0▼	12.6▼	17.5▼	11.8	12.3▼	11.9	17.8▼	11.7	12.2▼	12.2▼	16.6▼	12.0	12.4▼	11.8	17.6▼	11.2▲
WiR	64.2	64.7▼	64.7▼	65.9▼	62.8	62.4▼	62.6	64.5▼	62.3	63.6▼	62.1	64.1▼	62.6	62.9▼	61.8	64.6▼	71.7▼
WiW	70.2	70.4▼	70.9	71.4▼	69.6	69.7	69.7	71.1▼	70.0	70.2▼	70.0	71.0▼	70.0	70.1▼	69.6	71.2▼	72.2▼
Yac	0.46	0.59▼	0.42▲	0.57▼	0.37	0.51▼	0.40▼	0.59▼	0.38	0.56▼	0.38	0.54▼	0.40	0.54▼	0.42▼	0.52▼	0.88▼
B/W	–	0/9	3/4	0/10	–	0/9	2/4	0/10	–	0/10	1/5	0/10	–	0/10	0/5	0/10	3/7

Test $\ell = 31$

	$n^{\text{pop}} = 1000$				IMS $g = 4$				IMS $g = 6$				IMS $g = 8$				
	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	G	T^h	T$^\ell$	R	D
Air	26.4	33.5▼	30.8▼	37.1▼	25.9	26.5▼	23.3▲	37.6▼	24.9	27.1▼	24.7	39.2▼	24.9	28.8▼	26.1	38.2▼	30.8▼
Bos	21.4	22.8	21.6▼	26.2▼	20.1	21.3	21.8▼	23.4▼	20.9	21.2▼	22.2▼	23.2▼	20.2	22.3▼	22.6▼	26.0▼	26.1▼
Con	17.6	18.7▼	17.8▼	21.5▼	16.9	18.1▼	17.1▼	21.2▼	16.7	18.8▼	16.9	21.1▼	17.2	18.3▼	17.0	21.5▼	21.3▼
Dow	20.3	21.9▼	22.2▼	24.4▼	19.2	20.7▼	19.1	24.4▼	18.9	21.4▼	18.6	24.4▼	18.7	22.2▼	20.2▼	25.5▼	28.0▼
EnC	5.28	5.91▼	4.76▲	7.00▼	4.43	5.76▼	4.79▼	7.69▼	4.44	6.05▼	4.71	8.73▼	4.60	5.62▼	4.77▼	7.94▼	4.47▲
EnH	2.29	2.49▼	1.83▲	5.98▼	2.05	3.20▼	1.58▲	5.12▼	2.10	3.07▼	1.75▲	5.77▼	2.00	2.91▼	1.55▲	6.51▼	0.33▲
Tow	12.2	13.2▼	13.1▼	18.7▼	12.1	12.6▼	12.0▲	18.2▼	12.1	12.4▼	12.3	16.8▼	12.2	12.7▼	12.0	17.2▼	11.2▲
WiR	62.1	63.1	62.1	63.5▼	62.7	63.1▼	61.9	63.9▼	62.4	62.9▼	63.3▼	64.2▼	61.9	63.0▼	62.9▼	63.4▼	72.6▼
WiW	69.0	69.7▼	69.8▼	70.2▼	69.4	69.3	69.2	70.6▼	69.1	69.4▼	69.2▼	70.7▼	69.1	69.6▼	69.3▲	70.5▼	72.2▼
Yac	0.52	0.66▼	0.49▲	0.66▼	0.50	0.58▼	0.47	0.67▼	0.50	0.64▼	0.48▲	0.63▼	0.53	0.63▼	0.48	0.70▼	0.91▼
B/W	–	0/8	3/6	0/10	–	0/8	3/3	0/10	–	0/10	2/3	0/10	–	0/10	2/4	0/10	3/7

3.8.2. Results: benchmarking GP-GOMEA

We consider validation and test NMSE. We now show validation rather than training error because the IMS returns the solution which better generalizes to the validation set among the ones found by different runs (same for DTR due to cross-validation). Tables 3.7, 3.8, and 3.9 show the results for maximum sizes $\ell = 15, 31, 63$ ($h = 3, 4, 5$) respectively. On each set of results, the Friedman test reveals significant differences among the algorithms. As we are only interested in benchmarking GP-GOMEA, we test whether significant performance differences exist only between GP-GOMEA and the other algorithms (with Bonferroni-corrected Wilcoxon signed-rank test).

We begin with some general results. Overall, error magnitudes are lower for larger values of ℓ. This is not surprising: limiting solution size limits the complexity of relationships that can be modeled. Another general result is that errors on validation and test set are generally close. Likely, the validation data is a sufficiently accurate surrogate of the test data in these datasets, and solution size limitations make over-fitting unlikely. Finally, note that the results for DTR are the same in all tables.

We now compare GP-GOMEA with GP-Trad^h, focusing on statistical significance tests (see rows "B/W" of the tables), over all size limit configurations. Recall that these two algorithms work with the same type of limitation, i.e., based on maximum tree height. No matter the population sizing method, GP-GOMEA is almost always significantly better than GP-Trad^h. GP-GOMEA relies on the LT with improved linkage learning, which we showed to be superior to using the RT, i.e., blind variation, in the previous series of experiments (Sec. 3.5.3, 3.6.1). Subtree crossover and subtree mutation are blind as well, and can only swap subtrees, which may be a limitation.

GP-GOMEA and GP-Trad$^\ell$ are compared next. Recall that GP-Trad$^\ell$ is allowed to evolve any tree shape, as long as the limit in number of nodes is respected. Having this extra freedom, GP-Trad$^\ell$ performs better than GP-Trad^h (not explicitly reported in the tables), which confirms previous literature results [58, 59]. No marked difference exists between GP-GOMEA and GP-Trad$^\ell$ along different configurations. By counting the number of times one EA is found to be significantly better than the other along *all* 240 comparisons, GP-GOMEA beats GP-Trad$^\ell$ by a small margin: 87 significantly lower error distributions vs. 65 (88 draws).

For the traditional use of a single population ($n^{\text{pop}} = 1000$), GP-Trad$^\ell$ is slightly better than GP-GOMEA for $\ell = 15$ (Table 3.7), slightly worse for $\ell = 31$ (Table 3.8), and similar for $\ell = 63$ (Table 3.9), on both validation and test errors. The performance of the two (and also of the other EAs) improves when using the IMS. Although not explicitly shown in the tables, using the IMS is typically significantly better than not using it. When using a single fixed population size and a single run, only a single best-found solution is found. Depending on the configuration of that run, in particular the size of the population, that final solution may be underfitted or overfitted. When using a scheme such as the IMS, multiple solutions are marked best in the different interleaved runs. These solutions can subsequently be compared more in terms of generalization merits, i.e., by observing the associated performance on the validation set. The best performing solution can then ultimately be returned. Essentially, this thus provides a means to mitigate to some extent the problem of underfitting or overfitting. It should be noted, however, that the extent to which the setup of the IMS, particularly in terms of growing population sizes, contributes

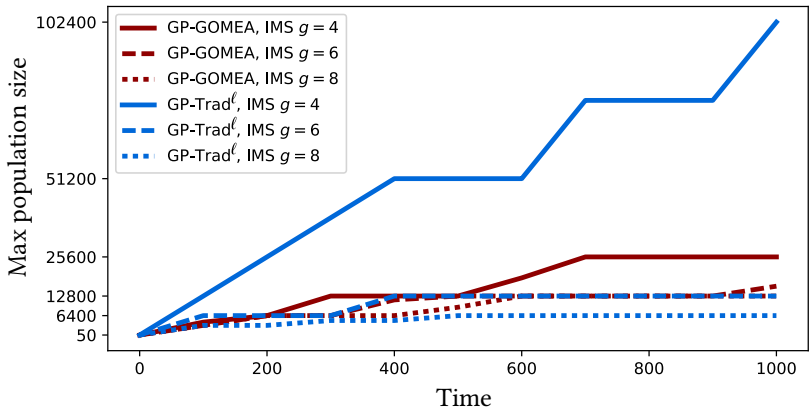

Figure 3.7: Maximum population size reached (vertical axis) in time (seconds, horizontal axis) with the IMS for GP-GOMEA ($h = 4$ limit) and GP-Trad$^\ell$ ($\ell = 31$ limit), for $g \in \{4, 6, 8\}$. The median among problems and repetitions is shown.

to this is not immediately clear. This could be studied by comparing with a scheme in which multiple runs are also performed, but all with a single population size. The final results of these runs can then also first be tested against the validation set. Likely, the use of a scheme like the IMS has an advantage because multiple population sizes will be tried. Therefore, likely a larger variety of results will be produced to test against the validation set, but a closer examination of this impact is left for future work.

The comparisons between GP-Trad$^\ell$ and GP-GOMEA tend to shift in favor of the latter when using the IMS, particularly for larger values of g. For $g = 4$, outcomes are still overall mixed along different ℓ limits. For $g = 8$, GP-GOMEA is preferable, with moderately more significant wins for $\ell = 15$, several more wins for $\ell = 31$, and slightly more wins for $\ell = 63$.

To investigate further the comparison between GP-GOMEA and GP-Trad$^\ell$, we consider the effect of g of the IMS for $\ell = 31$ (similar results are found for the other size limits). Figure 3.7 shows the median maximum population size reached by the IMS for different values of g in GP-GOMEA and GP-Trad$^\ell$. As can be expected, the bigger g, the less runs and the smaller populations at play. GP-Trad$^\ell$ reaches much bigger population sizes than GP-GOMEA when $g = 4$ (on average 3 times bigger). This is because GP-Trad$^\ell$ executes generations much faster than GP-GOMEA: it does not learn a linkage model, and performs n^{pop} evaluations per generation. GP-GOMEA performs $(2\ell - 2)n^{\text{pop}}$ variation steps (size of LT excluding its root times the population size) and up to $(2\ell - 2)n^{\text{pop}}$ evaluations per generation (only meaningful variation steps are evaluated).

GP-Trad$^\ell$ performs well for small values of g due to huge populations being instantiated with trees of various shape, i.e., expensive random search. Note that this behavior may be problematic when limited memory is available, especially if caching mechanisms are desirable to reduce the number of expensive evaluations (e.g., caching the output of each node as in [9, 22]). On the other hand, GP-GOMEA works fairly well with much smaller populations, as long as they are big enough to enable effective linkage learning

Table 3.9: Median validation and test NMSE of 30 runs with $\ell = 63$. Details as in Table 3.7.

Validation $\ell = 63$

	$n^{\text{pop}} = 1000$				IMS $g = 4$				IMS $g = 6$				IMS $g = 8$				
	G	T^h	T^ℓ	R	G	T^h	T^ℓ	R	G	T^h	T^ℓ	R	G	T^h	T^ℓ	R	D
Air	22.6	25.3▼	25.0▼	33.0▼	20.6	22.4▼	20.8	35.1▼	20.7	23.3▼	21.3▼	34.3▼	20.8	24.5▼	20.1	34.2▼	31.1▼
Bos	21.1	19.5	21.9	22.1	16.5	16.8▼	15.7	19.7▼	16.3	17.6▼	15.4▲	18.9▼	16.2	18.5▼	16.7▼	21.2▼	22.9▼
Con	16.6	17.4▼	16.6	18.5▼	15.2	16.1▼	15.7▼	18.5▼	15.5	16.5▼	15.6▼	19.6▼	15.3	16.3▼	15.9▼	19.0▼	22.7▼
Dow	18.6	19.0	18.8▼	21.7▼	17.4	17.8▼	16.7▲	24.1▼	17.7	18.2▼	17.0▲	24.3▼	17.8	19.8	17.6▲	22.4▼	30.6▼
EnC	4.66	5.15▼	4.26▲	5.55▼	3.67	4.37▼	4.14▼	6.92▼	3.85	4.50▼	4.02▼	7.08▼	3.76	4.88▼	3.99▼	6.78▼	4.23▼
EnH	1.65	1.52▼	1.13▲	2.63▼	0.69	1.54▼	0.84▼	4.02▼	0.92	1.78▼	1.02▼	3.81▼	0.87	1.78▼	0.80	3.68▼	0.43▲
Tow	11.5	11.7▼	11.7▼	15.7▼	11.3	10.9	11.1	16.1▼	11.4	11.3	10.9▲	17.0▼	11.3	11.9▼	11.2▲	16.6▼	11.2▲
WiR	64.4	64.6▲	65.2▼	64.3▲	63.0	62.4	62.5	63.8▼	62.3	62.9	62.8	64.6▼	62.7	62.9▼	62.5▼	64.5▼	71.7▼
WiW	70.1	70.1	68.8▲	70.9▼	69.2	69.2	68.7▲	71.1▼	68.9	69.3▼	69.4	71.6▼	69.1	69.7▼	69.6	71.4▼	72.2▼
Yac	0.46	0.45	0.37▲	0.46	0.32	0.38▼	0.33	0.40▼	0.32	0.39▼	0.33	0.44▼	0.33	0.40▼	0.33▲	0.48▼	0.88▼
B/W	–	1/5	4/4	1/7	–	0/7	2/3	0/10	–	0/8	3/4	0/10	–	0/9	3/4	0/10	2/8

Test $\ell = 63$

	$n^{\text{pop}} = 1000$				IMS $g = 4$				IMS $g = 6$				IMS $g = 8$				
	G	T^h	T^ℓ	R	G	T^h	T^ℓ	R	G	T^h	T^ℓ	R	G	T^h	T^ℓ	R	D
Air	23.0	25.5▼	25.9▼	31.5▼	21.1	22.5▼	19.6	34.9▼	21.7	22.4▼	21.9▼	33.8▼	21.2	23.4▼	21.6	34.1▼	30.8▼
Bos	22.0	20.0	21.2	21.9	19.2	20.7▼	20.4▼	24.1▼	21.5	20.3	20.3▲	25.0▼	19.8	19.7▼	21.4▼	25.8▼	26.1▼
Con	15.9	17.1▼	16.5▼	18.3▼	15.3	16.2▼	15.5	19.1▼	15.3	16.3▼	15.8▼	19.9▼	15.3	16.6▼	16.1▼	18.9▼	21.3▼
Dow	18.3	18.6▼	17.4▲	22.3▼	17.5	17.9	17.0▲	23.7▼	17.6	18.2▼	17.2▲	24.6▼	17.7	18.2	17.9	22.6▼	28.0▼
EnC	4.49	4.70▼	4.24▲	5.63▼	3.77	4.37▼	3.99▼	6.94▼	3.93	4.42▼	3.95▼	7.42▼	3.95	4.85▼	4.20▼	7.37▼	4.47▼
EnH	1.60	1.59▼	1.12▲	2.74▼	0.80	1.52▼	0.89▼	3.73▼	0.88	1.67▼	0.94	4.12▼	0.89	1.92▼	0.93▼	3.71▼	0.33▲
Tow	11.6	12.2▼	12.1▼	15.9▼	11.5	11.4	11.4	16.8▼	11.6	11.5	11.2▲	16.7▼	11.4	12.2▼	11.4	17.1▼	11.2▲
WiR	63.1	63.0	64.4▼	62.9	62.9	62.5▼	61.7	62.5▼	62.5	63.0	62.3	63.6▼	62.7	63.0	61.8▲	63.2▼	72.6▼
WiW	68.7	69.0	68.0▲	69.9▼	68.3	68.6	68.3▲	70.2▼	69.1	69.4	68.2▲	70.6▼	68.2	69.3▼	69.0	70.3▼	72.2▼
Yac	0.44	0.46	0.40▲	0.46	0.41	0.49▼	0.40▲	0.45▼	0.41	0.45▼	0.42	0.52▼	0.46	0.46	0.44▲	0.50▼	0.91▼
B/W	–	0/6	5/4	0/7	–	0/7	3/3	0/10	–	0/6	4/3	0/10	–	0/7	2/4	0/10	2/8

(the fixed $n^{\text{pop}} = 1000$ is smaller than the population sizes reached with the IMS). Despite the disadvantage of adhering to a specific tree shape, GP-GOMEA is typically preferable than GP-Trad$^\ell$ for larger values of g. Furthermore, Figure 3.7 shows that GP-GOMEA population scaling behaves sensibly with respect to g, i.e., it does not grow abruptly when g becomes small, nor shrink excessively when g becomes larger. This latter aspect is because in GP-GOMEA populations ultimately converge to a same solution, and are terminated, allowing for bigger runs to start. In GP-Trad$^\ell$ this is unlikely to happen, because of the use of mutation and stochastic (tournament) selection, stalling the IMS. For the larger $g = 8$, GP-GOMEA reaches on average 1.6 times bigger populations than GP-Trad$^\ell$.

GP-RDO, although allowed to evolve trees of different shape like GP-Trad$^\ell$, performs poorly on all problems, with all settings. It performs significantly worse than GP-GOMEA almost everywhere. GP-RDO is known to evolve big solutions, and it is reasonable to expect that GP-RDO actually benefits from solutions growing big, as these provide more subtree diversity for its library. The strict size limitation basically breaks GP-RDO. However, we remark that this EA was never designed to work under these circumstances, and when solution size is not strictly limited, GP-RDO is known to work well [9].

DTR is compared with GP-GOMEA using the IMS with $g = 8$. Although GP-GOMEA is not optimized (e.g., by tuning the function set), it performs on par with tuned DTR for $\ell = 15$, and better for $\ell = 31, 63$, on both validation and test sets. Where one algorithm outperforms the other, the magnitude of difference in errors are relatively large compared to the ones between EAs. This is because GP and DTR synthesize models of completely different nature (decision trees only use if-then-else statements).

Tower:

$4668.49 - 3.56((662.77 + x_{21})x_{12} \div_{\text{AQ}} x_{16} - x_1 - x_{15} + x_5 + 4x_{12} - x_{23}(x_6 \div_{\text{AQ}} x_1 + 1))$

Yacht:

$0.73 + 33004.40\left(((x_6^2 \div_{\text{AQ}} (x_5 x_2)) \div_{\text{AQ}} (x_3 x_2 \div_{\text{AQ}} (x_2 \div_{\text{AQ}} x_1)))(x_6 + 0.30)x_6^5 x_5\right)$

Figure 3.8: Examples of best solution found by GP-GOMEA ($\ell = 31$, IMS $g = 8$).

3.9. Discussion & conclusion

We built upon previous work on model-based GP, in particular on GP-GOMEA, to find accurate solutions when a strict limitation on their size is imposed, in the domain of SR. We focused on small solutions, in particular much smaller solutions than typically reported in literature, to prevent solutions becoming too large to be (easily) interpretable, a key reason to justify the use of GP in many practical applications.

A first limitation of this work is that to truly *achieve* interpretability may well require different measures. Interpretation is mostly subjective, and many other factors besides solution size are important, including the intuitiveness of the subfunctions composing the solution, potential decompositions into understandable repeating sub-modules, the number of features considered, and the meaning of these features [13, 56]. Nonetheless, much current research on GP for SR is far from delivering any interpretable results precisely because the size of solutions is far too large (see, e.g., the work of [12]).

We considered solution sizes up to $\ell = 63$ (corresponding to $h = 5$ for GP-GOMEA with subfunctions of arity ≤ 2). In our opinion, the limit of $\ell = 31$ ($h = 4$) is particularly interesting, as interpreting some solutions at this level can already be non-trivial at times. For example, we show the (manually simplified) best test solution found by GP-GOMEA (IMS $g = 8$) for Tower and Yacht, i.e. the biggest and smallest dataset respectively, in Figure 3.8. The solution for Tower is arguably easier to understand than the one for Yacht. We found solutions with $\ell = 63$ ($h = 5$) to be overly long to attempt interpreting, and solutions with $\ell = 15$ ($h = 3$) to be mostly readable and understandable. We report other example solutions at: `http://bit.ly/2IrUFyQ`.

We believe future work should address the aforementioned limitation: effort should be put towards reaching some form of interpretability notions, that go beyond solution size or other custom metrics (e.g., [61]). User studies involving the end users of the model (e.g., medical doctors for a diagnosis model) could guide the design of notions of interpretability. If an objective that represents interpretability can be defined, the design of multi-objective (model-based) GP algorithms may lead to very interesting results.

Another limitation of this work lies in the fact that we did not study how linkage learning behaves in GP for SR in depth. In fact, it would be interesting to assess when linkage learning is beneficial, and when it is superfluous or harmful. To this end, a regime of experiments where linkage-related outcomes are predefined, such as emergence of specific patterns, needs to be designed. Simple problems where the true function to regress is known may need to be considered. Studies of this kind could provide more insights on how to improve linkage learning in GP for SR (and other learning tasks), and are an interesting direction for future work.

Another crucial point to base future research upon is enabling linkage learning and linkage-based mixing in GP with trees of arbitrary shape. In fact, GP-GOMEA was not found to be markedly better than GP-Trad$^{\ell}$, and a large performance gap was found between GP-Trad$^{\ell}$ and GP-Trad^{h}. This is indicative that there is added value to perform evolution directly on non-templated trees, which, from this perspective, may be considered a limitation of GP-GOMEA. Going beyond the use of a fixed tree template, while still enabling linkage identification and exploitation, is a challenging open problem that could bring very rewarding results. On the other hand, we believe it is interesting to see that when GP-GOMEA and GP-Trad are set to work on the same search space, i.e., when GP-Trad^{h} is used, then GP-GOMEA performs markedly better.

In summary and conclusion, we have identified limits and presented ways to improve a key component of a state-of-the-art model-based EA, i.e. *GP-GOMEA*, to competently deal with realistic SR datasets, when small solutions are desired. This key component is linkage learning. We showed that solely and directly relying on mutual information to identify linkage may be undesirable, because the genotype is not uniformly distributed in GP populations, and we provided an approximate biasing method to tackle this problem. We furthermore explored how to incorporate ERCs into linkage learning, and found that on-line binning of constants is an efficient and effective strategy. Lastly, we introduced a new form of the IMS, to relieve practitioners from setting a population size, and from finding a good generalizing solution. Ultimately, our contributions proved successful in improving the performance of GP-GOMEA, leading to the best overall performance against competing EAs, as well as tuned decision trees. We believe our findings set an important first step for the design of better model-based GP algorithms capable of learning interpretable solutions in real-world data.

Acknowledgments

The authors thank the Foundation Kinderen Kankervrij for financial support (project no. 187), and SURFsara for granting access to the Lisa Compute Cluster.

References

[1] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection* (MIT Press, 1992).

[2] K. Krawiec, *Behavioral Program Synthesis with Genetic Programming*, 1st ed. (Springer Publishing Company, Incorporated, 2015).

[3] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, *Multifactorial genetic programming for symbolic regression problems,* IEEE Transactions on Systems, Man, and Cybernetics: Systems , 1 (2018).

[4] V. V. De Melo, *Kaizen programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2014) pp. 895–902.

[5] J. Žegklitz and P. Pošík, *Symbolic regression algorithms with built-in linear regression,* (2017), arXiv preprint arXiv:1701.03641.

[6] I. Icke and J. C. Bongard, *Improving genetic programming based symbolic regression using deterministic machine learning,* in *IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2013) pp. 1763–1770.

[7] M. Keijzer, *Improving symbolic regression with interval arithmetic and linear scaling,* in *European Conference on Genetic Programming* (Springer, 2003) pp. 70–82.

[8] Q. Chen, B. Xue, and M. Zhang, *Generalisation and domain adaptation in gp with gradient descent for symbolic regression,* in *IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2015) pp. 1137–1144.

[9] T. P. Pawlak, B. Wieloch, and K. Krawiec, *Semantic backpropagation for designing search operators in genetic programming,* IEEE Transactions on Evolutionary Computation **19**, 326 (2015).

[10] Q. Chen, B. Xue, and M. Zhang, *Improving generalization of genetic programming for symbolic regression with angle-driven geometric semantic operators,* IEEE Transactions on Evolutionary Computation **23**, 488 (2018).

[11] A. Moraglio, K. Krawiec, and C. G. Johnson, *Geometric semantic genetic programming,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2012) pp. 21–31.

[12] J. F. B. S. Martins, L. O. V. B. Oliveira, L. F. Miranda, F. Casadei, and G. L. Pappa, *Solving the exponential growth of symbolic regression trees in geometric semantic genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2018) pp. 1151–1158.

[13] Z. C. Lipton, *The mythos of model interpretability,* Queue **16**, 30:31 (2018).

[14] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, *A survey of methods for explaining black box models,* ACM Computing Surveys (CSUR) **51**, 93:1 (2018).

[15] P. Orzechowski, W. La Cava, and J. H. Moore, *Where are we now?: A large benchmark study of recent symbolic regression methods,* in *Genetic and Evolutionary Computation Conference (GECCO) 2018* (ACM, 2018) pp. 1183–1190.

[16] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system,* in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016) pp. 785–794.

[17] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming* (Lulu Enterprises, UK Ltd, 2008).

[18] D. Thierens and P. A. N. Bosman, *Optimal mixing evolutionary algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2011) pp. 617–624.

[19] N. H. Luong, H. La Poutré, and P. A. N. Bosman, *Multi-objective gene-pool optimal mixing evolutionary algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2014) pp. 357–364.

[20] A. Bouter, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, *Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2017) pp. 705–712.

[21] E. Medvet, A. Bartoli, A. De Lorenzo, and F. Tarlao, *GOMGE: Gene-pool optimal mixing on grammatical evolution,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2018) pp. 223–235.

[22] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, *Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning,* in *Genetic and Evolutionary Computation Conference (GECCO) 2017* (ACM, New York, NY, USA, 2017) pp. 1041–1048.

[23] M. Virgolin, T. Alderliesten, A. Bel, C. Witteveen, and P. A. N. Bosman, *Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2018) pp. 1395–1402.

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees* (Wadsworth, 1984).

[25] M. Hauschild and M. Pelikan, *An introduction and survey of estimation of distribution algorithms,* Swarm and Evolutionary Computation **1**, 111 (2011).

[26] K. Kim, Y. Shan, X. H. Nguyen, and R. I. McKay, *Probabilistic model building in genetic programming: a critical review,* Genetic Programming and Evolvable Machines **15**, 115 (2014).

[27] R. Salustowicz and J. Schmidhuber, *Probabilistic incremental program evolution,* Evolutionary Computation **5**, 123 (1997).

[28] K. Sastry and D. E. Goldberg, *Probabilistic model building and competent genetic programming,* in *Genetic Programming Theory and Practice* (Springer, 2003) pp. 205–220.

[29] K. Yanai and H. Iba, *Estimation of distribution programming based on Bayesian network,* in *IEEE Congress on Evolutionary Computation (CEC)*, Vol. 3 (IEEE, 2003) pp. 1618–1625.

[30] E. Hemberg, K. Veeramachaneni, J. McDermott, C. Berzan, and U.-M. O'Reilly, *An investigation of local patterns for estimation of distribution genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2012) pp. 767–774.

[31] Y. Shan, R. I. McKay, R. Baxter, H. Abbass, D. Essam, and H. Nguyen, *Grammar model-based program evolution,* in *IEEE Congress on Evolutionary Computation (CEC)*, Vol. 1 (IEEE, 2004) pp. 478–485.

[32] P. A. N. Bosman and E. D. de Jong, *Learning probabilistic tree grammars for genetic programming,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2004) pp. 192–201.

[33] P.-K. Wong, L.-Y. Lo, M.-L. Wong, and K.-S. Leung, *Grammar-based genetic programming with Bayesian network,* in *IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2014) pp. 739–746.

[34] L. F. D. P. Sotto and V. V. de Melo, *A probabilistic linear genetic programming with stochastic context-free grammar for solving symbolic regression problems,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2017) pp. 1017–1024.

[35] Y. Hasegawa and H. Iba, *Latent variable model for estimation of distribution algorithm based on a probabilistic context-free grammar,* IEEE Transactions on Evolutionary Computation **13**, 858 (2009).

[36] I. Tanev, *Genetic programming incorporating biased mutation for evolution and adaptation of snakebot,* Genetic Programming and Evolvable Machines **8**, 39 (2007).

[37] X. Li, S. Mabu, H. Zhou, K. Shimada, and K. Hirasawa, *Genetic network programming with estimation of distribution algorithms for class association rule mining in traffic prediction,* in *IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2010) pp. 1–8.

[38] A. Ratle and M. Sebag, *Avoiding the bloat with stochastic grammar-based genetic programming,* in *International Conference on Artificial Evolution (Evolution Artificielle)* (Springer, 2001) pp. 255–266.

[39] Y. Chen, T.-L. Yu, K. Sastry, and D. E. Goldberg, *A survey of linkage learning techniques in genetic and evolutionary algorithms,* IlliGAL report **2007014** (2007).

[40] D. Thierens and P. A. N. Bosman, *Hierarchical problem solving with the linkage tree genetic algorithm,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 877–884.

[41] B. W. Goldman and W. F. Punch, *Parameter-less population pyramid,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2014) pp. 785–792.

[42] S.-H. Hsu and T.-L. Yu, *Optimization by pairwise linkage detection, incremental linkage set, and restricted/back mixing: DSMGA-II,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2015) pp. 519–526.

[43] G. R. Harik and F. G. Lobo, *A parameter-less genetic algorithm,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (Morgan Kaufmann Publishers Inc., 1999) pp. 258–265.

[44] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, *The gambler's ruin problem, genetic algorithms, and the sizing of populations,* Evolutionary Computation **7**, 231 (1999).

[45] I. Gronau and S. Moran, *Optimal implementations of UPGMA and other common clustering algorithms,* Information Processing Letters **104**, 205 (2007).

[46] M. Ebner, M. Shackleton, and R. Shipman, *How neutral networks influence evolvability,* Complexity **7**, 19 (2001).

[47] K. L. Sadowski, P. A. N. Bosman, and D. Thierens, *On the usefulness of linkage processing for solving MAX-SAT,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 853–860.

[48] J. Ni, R. H. Drieberg, and P. I. Rockett, *The use of an analytic quotient operator in genetic programming,* IEEE Transactions on Evolutionary Computation **17**, 146 (2013).

[49] A. Asuncion and D. Newman, *UCI machine learning repository,* (2007), `http://archive.ics.uci.edu`.

[50] J. Demšar, *Statistical comparisons of classifiers over multiple data sets,* Journal of Machine Learning Research **7**, 1 (2006).

[51] S. Luke and L. Panait, *A survey and comparison of tree generation algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (Morgan Kaufmann Publishers Inc., 2001) pp. 81–88.

[52] E. Medvet, M. Virgolin, M. Castelli, P. A. N. Bosman, I. Gonçalves, and T. Tušar, *Unveiling evolutionary algorithm representation with DU maps,* Genetic Programming and Evolvable Machines **19**, 351 (2018).

[53] Y.-J. Lin and T.-L. Yu, *Investigation of the exponential population scheme for genetic algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2018) pp. 975–982.

[54] T. P. Pawlak and K. Krawiec, *Competent geometric semantic genetic programming for symbolic regression and boolean function synthesis,* Evolutionary Computation **26**, 177 (2018).

[55] M. Virgolin, T. Alderliesten, and P. A. N. Bosman, *Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2019).

[56] F. Doshi-Velez and B. Kim, *Towards a rigorous science of interpretable machine learning,* (2017), arXiv preprint arXiv:1702.08608.

[57] L. Breiman, *Random forests,* Machine Learning **45**, 5 (2001).

[58] C. Gathercole and P. Ross, *An adverse interaction between crossover and restricted tree depth in genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (MIT Press, 1996) pp. 291–296.

[59] W. B. Langdon and R. Poli, *An analysis of the MAX problem in genetic programming,* Genetic Programming **1**, 222 (1997).

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, *Scikit-learn: Machine learning in Python,* Journal of Machine Learning Research **12**, 2825 (2011).

[61] E. J. Vladislavleva, G. F. Smits, and D. den Hertog, *Order of nonlinearity as a complexity measure for models generated by symbolic regression via Pareto genetic programming,* IEEE Transactions on Evolutionary Computation **13**, 333 (2009).

4

Linear Scaling in Semantic Backpropagation-based Genetic Programming

Semantic Backpropagation (SB) is a recent technique that promotes effective variation in tree-based genetic programming. The basic idea of SB is to provide information on what output is desirable for a specified tree node, by propagating the desired root-node output back to the specified node using inversions of functions encountered along the way. Variation operators then replace the subtree located at the specified node with a tree for which the output is closest to the desired output, by searching in a pre-computed library. In this chapter, we propose two contributions to enhance SB specifically for symbolic regression, by incorporating the principles of Keijzer's Linear Scaling (LS). In particular, we show how SB can be used in synergy with the scaled mean squared error, and we show how LS can be adopted within library search. We test our adaptations using the well-known variation operator Random Desired Operator (RDO), comparing to its baseline implementation, and to traditional crossover and mutation. Our experimental results on real-world datasets show that SB enhanced with LS substantially improves the performance of RDO, resulting in overall the best performance among all tested GP algorithms.

The contents of this chapter are based on the following publication: **M. Virgolin**, T. Alderliesten, and P.A.N. Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*, pp. 1084-1092, ACM (2019).

4.1. Introduction

Semantic Backpropagation (SB) is a recent technique in tree-based Genetic Programming (GP) [1, 2] which enables the design of novel variation operators [3, 4]. For any tree node, given a target output for the tree, SB determines what the *desired output* for that node is. If the node were to be replaced with a subtree that delivers the desired output, then the outputs of the ancestor nodes would also change, ultimately making the root deliver the target output. The application of SB-based GP algorithms has been shown to be particularly effective in supervised learning applications such as Boolean circuit synthesis and symbolic regression [4–6].

SB-based variation operators modify trees by replacing nodes with subtrees that match desired outputs as closely as possible. The Random Desired Operator (RDO) is perhaps the most known among them, as it has been shown to perform best on a variety of problems [3, 4]. Key components of RDO are the use of a library of trees with pre-computed outputs, and a library search procedure to retrieve the tree which most closely matches the desired output.

As to the library, two traditional ways exist to build it [4]. The first way is to generate all possible trees within a maximum tree height, and to retain one tree for each unique output (the tree with less nodes is kept). Clearly, this method cannot scale with the number of dimensions, nor with the sampling of real-valued constants. In [4], for problems with a single feature, a maximum height of 3 already results in hundreds of thousands of trees. The second way is to dynamically refresh the library every generation, by including all subtrees with unique output as observed in the population. The downside of this approach is that the expressiveness of the library may be limited, as it is biased by how the population evolves.

Linear Scaling (LS) is an interesting existing technique to minimize the mean squared error of a GP tree by applying an optimal linear transformation to the output of the tree [7, 8]. While typically used to improve the fitness, LS can more generally be applied to scenarios where a (monotonic transformation of the Euclidean) distance between two outputs needs to be minimized. As SB-based GP operates by matching desired outputs, it stands to reason that some form of LS can be integrated to benefit the algorithm. This is precisely the topic of this chapter: we study how to best integrate and how to best observe the impact of LS on SB-based GP.

We propose, for the first time, the use of LS as (i) A separate, but synergistic mechanism, to work *with* SB; and (ii) A joint mechanism, to use *within* SB-based GP, namely during library search. Much previous work solely considered synthetic benchmark functions with few variables, and generational computation budgets (see Sec. 4.3). The latter choice arguably favors SB-based GP when compared to other forms of GP (e.g., traditional GP that swaps and mutates subtrees randomly [1]), in that the computational time taken by SB itself, library construction, and library search, is not considered [4]. For this reason, in our experiments, we assess the effectiveness of the proposed LS-enhanced, SB-based GP in terms of *both* number of generations and *time*. Moreover, we test the algorithms on realistic small- to medium-sized regression problems, using ten established real-world benchmark datasets.

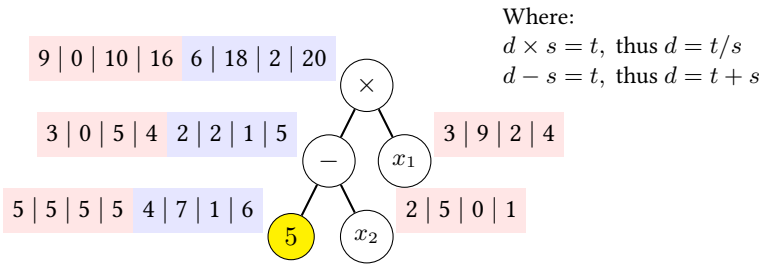

Figure 4.1: Example of SB for the yellow leaf. The current outputs of each node are in pink. The desired outputs are in blue. The desired output of the root is the (given) target output, and the others are computed by recursive inversion on the path down to the yellow leaf. The operations in the top right describe the inversions used in this example.

4.2. SEMANTIC BACKPROPAGATION

Given a target output t for the tree (i.e., for its root), SB computes a desired output d^N for one specific node N. This information can then be used to replace N with a subtree that has output as close as possible to d^N. It is expected that, the closer the output of the subtree is to d^N, the closer the output of the root will be to t.

Let D be the depth of N. Then N has D ancestors. Let A_k be the ancestor of N at depth k. Similarly, let $S_k = \{S_k^1, S_k^2, \dots\}$ be the (possibly empty) set of sibling nodes of A_k. For the sake of brevity, we now use the same notation used to refer to a node to also identify the function implemented by that node. For example, $A_k\left(x, S_{k+1}\right)$ represents the application of the function of node A_k on x, and on (the outputs of) the nodes S_{k+1}. Therefore, we can say that SB computes:

$$d^{A_{k+1}} = A_k^{-1}\left(d^{A_k}, S_{k+1}\right), \tag{4.1}$$

where A_k^{-1} represents the inversion of the function A_k. SB starts from the root by setting its desired output to the target, i.e. $d^{A_0} := t$. The recursive computation of the desired output for N (at depth D) then follows: $d^N = A_{D-1}^{-1}\left(d^{A_{D-1}}, S_D\right)$. Figure 4.1 shows an example.

Note that, if non-injective functions (e.g., $abs(\cdot)$) are included in the function set, each desired output vector will grow to represent different possible outcomes, i.e., $d \in \mathbb{R}^{\gamma \times n}$ and $d_i = \{d_i^1, \dots, d_i^\gamma\}$, with $i = 1, \dots, n$ the indices of training examples (from now on, for brevity, we drop the superscript N from d^N). Note that γ can be ∞, e.g., for $\sin$. Similarly, any value may satisfy some inversions: e.g., for $x = \times^{-1}(0,0)$, $0 \times x = 0, \forall x \in \mathbb{R}$. In such cases, we will indicate that any value is good with $*$. We describe how d_i with multiple and/or $*$ values is handled during library search in the following Section 4.2.1. Conversely, some functions are not invertible (in $\mathbb{R}$) in some points (e.g., $(\cdot^2)^{-1}(-1) = \sqrt{-1}$), thus some d_i^j may not exist. If SB is unfeasible, i.e., $\exists i : d_i = \emptyset$, we abort SB (as in [4]).

4.2.1. Library and library search

Given the desired output d, a subtree with similar output is sought. For this purpose, typical SB-based variation operators rely on a searchable library of pre-computed trees with unique outputs. As aforementioned in the introduction, a way to build the library is to pre-compute all possible trees up to a maximum height, but this becomes intractable with already few features (terminals) [4]. The other typical method is to collect all subtrees as observed in the population (updated every generation) [9, 10].

In the so-called "population-based" library, if multiple subtrees with the same output exist in the population, the one smallest in terms of the number of nodes is retained. Furthermore, subtrees with constant output are not considered (library search handles constants separately, see below).

The library search procedure parses the library to find the tree of which its output o minimizes the distance from d, e.g., in terms of *modifications* of the L1 or L2 distance (or any Minkowski distance). By modified version of the distances we mean that the multi-valued nature of d_i must be accounted for. The distance can be computed by finding the one d_i^j that minimizes $|d_i^j - o_i|^w, \forall w \in \{1, 2, \dots\}$ [4]. Furthermore, if $\exists j : d_i^j = *$, then the values of d_i do not matter, and it is defined to have $|* - o_i|^w = 0$. By pre-sorting the d_i^j in j, a linearly addressable library $\mathscr{L}$ can be parsed in $O(|\mathscr{L}| n \log \gamma)$ [4].

With trees with a constant output being typically excluded from the library, library search further considers the distance between a constant value and the desired output in a separate fashion. In [4], the values of d are considered to be candidate constant values, and the best one is picked. The best tree found in the library, or a single-node implementing the best constant, is finally returned by the library search procedure, depending on which is closest to d.

4.2.2. Random desired operator

RDO works by generating an offspring tree that differs from the parent in one subtree. The pseudocode of RDO is shown in Algorithm 4.1. First, the offspring (O) is created as a clone of the parent (P), and one of its nodes N is selected. In [4], it is proposed to select N with the *equal depth probability* criterion, which first samples the depth D to consider, and then samples N among the nodes with depth D, both uniformly at random. Second, SB is executed for N, by setting the target for the root to the dependent variable to regress, i.e. $t := y$. Note that, in general, t can be different (e.g., a crossover operator is proposed in [6] that sets the target output for one parent to the output of another parent). If SB is aborted because an unfeasible d is computed, RDO returns the clone of the parent. Otherwise, library search is performed, resulting in a tree T that has output with minimum distance from d. Finally, RDO returns the offspring, adapted by replacing its subtree at N with T.

4.2.3. Intermediate output caching

SB-based GP is particularly efficient if the output of subtrees are cached. In particular, each recursive iteration of SB requires to know the output of the sibling nodes, and library search requires the output of the library trees. Therefore, in [4] it is proposed to cache intermediate tree outputs, i.e., the outputs of all nodes.

Intermediate output caching not only speeds up SB-related methods, but also the traditional evaluation of trees. In fact, if a node is changed, it is sufficient to recompute the

Algorithm 4.1 Pseudocode of RDO.

```
1 function RDO(y, P, 𝒟)
2     O ← Clone(P)
3     N ← EqualDepthProbability(O)
4     d ← SemanticBackpropagation(N, y)
5     if ∃i : d_i = ∅ then return O
6     T ← LibrarySearch(d, 𝒟)
7     O ← ReplaceSubtree(N, T)
8     return O
```

outputs only along the chain of ancestors, i.e., from the parent of that node upwards, to get the output of the root. While these *partial evaluations* can be very effective, especially for high-dimensional outputs, they take a toll in terms of memory (see, e.g., the discussion on scalability in Chapter 2).

4.3. Related work

Two research lines are mostly related to this chapter, namely the one on LS, and the one on SB. As to LS, the most cited work to date is [7], which shows how LS can dramatically improve the performance of GP, for synthetic functions with up to three variables. In [8], theoretical motivations for the added value of LS are given. LS was successfully used for practical applications in, e.g., [11–13].

To the best of our knowledge, no contribution has been made that proposes modifications of LS itself. This is not surprising, as LS is quick (i.e., $O(n)$), and the scaling and translation coefficients are optimal with respect to the dependent variable y (see the description of LS in Sec. 4.4.1). Perhaps more interestingly, we also could not find any work where LS is combined with another method in a truly synergistic way, i.e., having LS and/or the other method sharing information with each other. For example, in [14], LS is used together with a particular mating scheme, but the two methods co-exist independently from each other. Here, we consider for the first time a use of LS that is deeply intertwined with another method, i.e., SB.

As to SB, it was first introduced together with RDO in [3], and much research work has followed. Perhaps one of the most comprehensive contributions is [4], which compares several variation operators, two of which are SB-based (RDO, and approximately geometric crossover [6]). RDO is shown to outperform most of the other operators, on both Boolean and regression problems.

While RDO uses SB to replace a subtree, the Forward Propagation Mutation (FPM) operator proposed in [9] does the opposite: it preserves the subtree, and replaces the remaining part of the tree, called the *context*. A new context is built by determining a new root, and another subtree to append to the root, which is a sibling to the preserved one. This new subtree is retrieved by library search using cosine similarity, and is rescaled by an optimal constant (determined in $O(n)$). The authors claim that an alternative could be to use LS to also determine a translation coefficient during library search for FPM. This is indeed investigated in our work, for RDO.

Very recently, a variant of FPM has been proposed in [15] where the target is set to a random point in the segment between y and the output of the parent, and where LS is applied *after* library search. While this improves the fitting of the subtree to the context, it is less effective (but faster) compared to considering optimal translation coefficients *during* library search (as recognized by [9]).

Notwithstanding the novelty and advantages of the aforementioned and of other work on SB-based GP (e.g., [5, 16]), as we mentioned in the introduction, mostly synthetic functions have been considered so far, in the domain of regression. These functions have up to three variables only. Moreover, comparisons have only been framed in terms of number of generations, thus ignoring much of the computational expensiveness of SB-based GP.

To the best of our knowledge, only in [15] and [10] four and two real-world benchmarks are respectively considered, for GP using RDO (and a variant) and the aforementioned variant of FPM. Yet again, only generational budgets are considered, except for the supplementary material of [15], where experiments using a time limit are reported. Although those results undeniably bring additional insight, we believe it remains hard to assess what the impact of using SB-based operators is on computation time, because of two reasons. Firstly, a relatively small population size of 100 is used, meaning that population-based libraries will also be small and quick to parse. Secondly, the considered GP algorithms have several differences (e.g., selection schemes), and always employ other variation operators together with the SB-based ones.

In this chapter, not only do we consider how LS can be combined with SB-based GP, but we also attempt to address the main limitations of the related work. We adopt ten real-world benchmark datasets for regression with dozens of features, as they are arguably more representative of practical problems, and we attempt to frame algorithmic comparisons in terms of both generations and time limits to also observe the potential overhead of adopting SB-based GP.

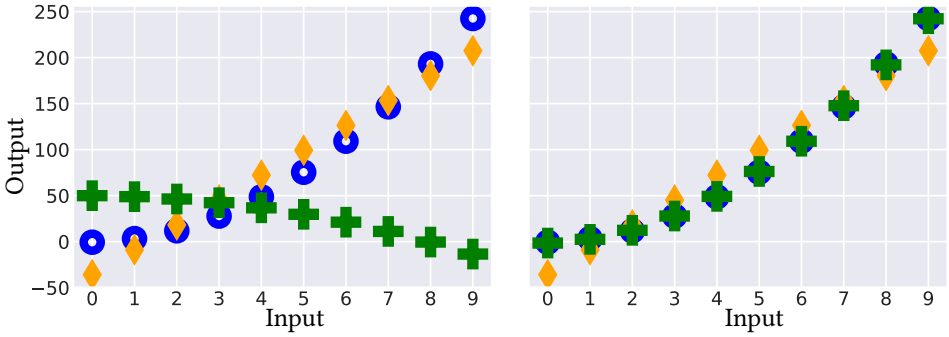

Figure 4.2: Example of the effect of LS. Blue circles represent the output of the function to approximate, while orange diamonds and green crosses are the output of two trees. Left: The orange diamonds are closest to the blue circles. Right: The application of LS substantially improves the green crosses, making them become the best match.

4.4. Linear scaling *with* SB-based GP

We now describe the first contribution of this chapter, i.e., how SB can work together with LS, in synergy. The main concept behind LS is to allow GP to focus on the "shape" of the function to approximate, by providing translation and scaling coefficients that minimize the training Mean Squared Error (MSE) [7] (see, e.g., Fig. 4.2).

In principle, LS and SB can work independently, without making changes to the two methods. In RDO, SB works by setting the target output t (i.e., the desired output for the root) to y. To back-propagate this information means to directly try to optimize the tree towards delivering exactly y, without exploiting the fact that LS helps scaling the output of the root. In other words, in this setting, LS acts solely as a "patch" on top of SB-based GP, as it attempts to correct for the residual error that the algorithm normally makes.

We argue that it is reasonable to attempt to make LS and SB work in synergy to reduce the error, in particular by informing SB on what the effect of LS is. Indeed, we hypothesize that if the transformation applied by LS is also backpropagated to determine the desired output, the subsequent variation will be more effective, as it will attempt to correct the error that remains *after* LS is applied.

4.4.1. Linear scaling

LS works as follows. Let the MSE between the dependent variable y and the tree output o be the fitness function for regression:

$$\text{MSE}(y, o) = \frac{1}{n} \sum_{i=1}^{n} (y_i - o_i)^2.$$

LS introduces a *scaled* version of the MSE, where respectively a translation coefficient a and a scaling coefficient b are used within the computation of the MSE, in order to minimize it:

$$\text{MSE}^{a,b}(y, o) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (a + bo_i))^2.$$

The optimal a and b that minimize the error are (see [7]):

$$\begin{aligned} a &= \bar{y} - b\bar{o}, \\ b &= \sum_{i=1}^{n} \frac{(y_i - \bar{y})(o_i - \bar{o})}{(o_i - \bar{o})^2} = \frac{cov(y, o)}{var(o)}. \end{aligned} \tag{4.2}$$

The implementation of Equation 4.2 takes $O(n)$.

4.4.2. Linear scaling in synergy with semantic backpropagation

We now describe how LS and SB can work in synergy. To begin, we point out that using the $\text{MSE}^{a,b}$ is equivalent to using the traditional MSE on a tree where the addition of a and multiplication by b are encoded within the tree itself, with suitable nodes placed on top of the root. For example, consider the rightmost tree of Figure 4.3: ignore the nodes in white, and imagine the plus node to be the actual root. That tree is essentially one where the effect of LS is incorporated in its structure (with pink nodes). For such a tree, it is

straightforward to compute SB (as described in Sec. 4.2). In particular, we immediately see that for a target output t, the desired output for the original root (top green node) will be:

$$d_i = \frac{t_i - a}{b}, \forall i. \tag{4.3}$$

Therefore, whenever SB needs to be performed, we can calculate a and b based on t and the current tree output o (or, if $t = y$ like in RDO, a and b can be cached after they are computed for $\text{MSE}^{a,b}(y, o)$). Then we can compute d for the root using Equation 4.3 as starting point for SB, and then we can proceed with Equation 4.1 as usual.

We remark that the computation overhead for including LS in SB this way can be considered negligible. If D is the depth of the node chosen for replacement, then SB needs to compute D inversions. If only injective functions are considered, this leads to $O(Dn)$ (it is $O(D\gamma^D n)$ if non-injective functions are present). In typical symbolic regression settings, D is bounded by a small constant and n is large, i.e. $D \ll n$, meaning that the bound is $O(n)$. Since to include LS in synergy with SB means to compute Equation 4.2 and to compute Equation 4.3, and since these computations bring only additional $O(n)$ contributions, the bound remains $O(n)$.

4.5. Linear scaling *within* SB-based GP

When performing library search, a tree T that has output o close to a desired output d is sought for. This situation is similar to the symbolic regression problem itself, where the output of the root node is expected to match y. Because LS is known to help in the latter scenario [7, 8], it is reasonable to expect that LS can improve the effectiveness of library search as well, as optimally scaled tree outputs will be considered.

4.5.1. Linear scaling during library search

Let L2 be the distance metric adopted by the library search procedure. Since L2 is a monotonic transformation of the MSE, the optimal coefficients a and b can be computed with Equation 4.2 (replacing y with d) to decrease the distance between d and o.

In practice, d needs to be in $\mathbb{R}^n$ (instead of in $\mathbb{R}^{\gamma \times n}$) to have a unique, well-defined way to compute a and b. For example, what is $\bar{d}$ if there exists some d_i with multiple values? Some criterion should be used to choose one of the values for d_i (e.g., the value closest to the mean given by considering the other d_j that have unique values). Restricting the multi-dimensionality of the desired output by choosing a single value for each d_i means that possibly better matching outputs present in the library will not be searched for. Alternatively, multiple scalings could be computed and the best one could be taken, but exponential possibilities could exist. In this chapter, we include a non-injective function in the function set of GP that is symmetric around zero. For its inversion, we choose to return only the positive value (see Sec. 4.6). Regarding $*$ values, we make the assumption that, if present, they are few, and can be ignored when computing a and b.

We thus assess the effect of using LS within library search. Whenever library search is performed, for each tree in the library, we compute optimal a and b coefficients that minimize the distance between the output of the tree and the desired output. Library search then returns the best matching tree, along with its a, b coefficients. When this tree needs to be appended by the variation operator, four nodes are added on top of its root,

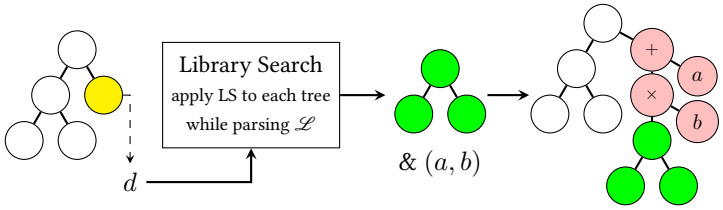

Figure 4.3: Illustration of SB-based GP variation using LS within library search. From left to right: the desired output d is computed for a node (in yellow). Library search retrieves the tree (in green) with output that, scaled by a, b, best matches d. The yellow node is replaced, and a structure is incorporated to account for the scaling (in pink).

namely two constants with value a and b respectively, an addition and a multiplication node, to effectively incorporate the scaling in the structure of the tree. Figure 4.3 illustrates this procedure.

The computation time taken by LS is $O(n)$, and it is additive with respect to the time taken to compute the distance between the output of a tree in the library and d. Therefore, the library search bound remains $O(|\mathscr{L}|n)$. In practice, some adjustments can be made to reduce computations. Once the library is created, the mean of each tree output $\bar{o}$ can be cached, as well as the terms $(o_i - \bar{o})$ (see Eq. 4.2). Furthermore, the mean of the desired output $\bar{d}$, and the terms $(d_i - \bar{d})$ can be computed only once, before starting the library search. This way, the only operation with cost linear in n that is left to do when searching is the computation of the numerator of b in Equation 4.2.

Lastly, when using LS within library search, we also change the way a competing constant is computed: we set the constant to the optimal value, i.e., $\bar{d}$.

4.6. Experimental setup

The parameter settings for GP are reported in Table 4.1, and are typical settings used in literature ([2], related work of Sec. 4.3). The function $\div_{\text{AQ}}$ is the analytic quotient [17], which allows for smooth division with no discontinuities (the denominator can never become 0). The inversions for the functions considered are reported in Table 4.2. Note that in the inversion of $\div_{\text{AQ}}$ for a_j, we return only the positive value. The terminal set includes an Ephemeral Random Constant (ERC) [2], that has the effect of generating nodes with randomized constant output. These constant outputs are sampled uniformly in the interval defined by the minimum and maximum value (available at training time) of the features.

Together with SB-based GP using RDO, we consider as a baseline GP with standard subtree crossover and subtree mutation operators (SGP) [1, 2], respectively applying them on 90% and 10% of the population every generation. Like for RDO, the nodes to swap/mutate are chosen with equal depth probability, as in [4].

The operators of SGP take much less computation time compared to RDO (essentially $O(1)$), in particular because the latter requires to build the library of trees, and performs SB and library search. Therefore, we consider both a limit of 100 generations and a time-dependent limit of 1000 seconds. As time-based comparisons can very much depend on

Table 4.1: Parameter settings of GP.

Parameter	Setting
# Generations / time limit	100 / 1000 s
Population size	500
Function set	$\{+, -, \times, \div_{\text{AQ}}\}$
Terminal set	Features $\cup$ ERC
ERC sample method	$\mathbb{U}[\min(\text{Features}), \max(\text{Features})]$
Initialization	Ramped Half-Half 2–6
Maximum tree height	12
Maximum number of nodes	500
Selection	Tournament 4 & Elitism 1
Variation	RDO with rate 1.0
Intermediate output caching	Active

implementation details, we attempt to boost their fidelity by developing all algorithms in the same C++ code base, which can be found at: `https://goo.gl/UbFFSU`.

We consider ten real-world benchmark regression datasets, with variable numbers of examples and features, as reported in Table 4.3. The datasets Dow chemical and Tower are recommended as benchmarks in [18]. The others are often used in GP literature and come from the UCI machine learning repository[1]. These datasets can be considered "well-behaved", in that overfitting to the training typically happens only if very complicated models are learned, or functions with discontinuities are used (e.g., protected division [1]). We adopt a typical 75%-25% random splitting of the examples into training and test set for a given run.

Each experiment consists of 30 independent runs. To assess if the results of one experiment are significantly better or worse than the ones of another, we use the non-parametric Wilcoxon signed-rank test [19], pairing runs by random seed. The random seed determines the train-test split and the sampling of the initial population. We say a result is significant if the p-value of the statistical test is below a threshold τ. We use $\tau = 0.05$, and further apply the Bonferroni correction method, to prevent false positive claims [19].

We run the experiments on a machine with two Intel® Xeon® CPU E5-2699 v4 @ 2.20GHz, and 630 GB of RAM. Big amounts of memory are needed to use the intermediate output caching, as single runs can already employ a few GB of memory.

Table 4.2: Functions considered and their inversions.

Function	Direct	Inversion(s)
$+$	$a_i + a_j = o$	$a_i = o - a_j$
$-$	$a_i - a_j = o$	$a_i = o + a_j, a_j = a_i - o$
$\times$	$a_i \times a_j = o$	$a_i = o/a_j$ if $o, a_j \neq 0$
		$*$ if $o, a_j = 0$
		impossible if $o \neq 0, a_j = 0$
$\div_{\text{AQ}}$	$a_i/\sqrt{1 + a_j^2} = o$	$a_i = o \times \sqrt{1 + a_j^2}$
		$a_j = +\sqrt{(a_i/o)^2 - 1}$ if $o \neq 0, (a_i/o)^2 \geq 1$
		impossible if $o = 0$ or $(a_i/o)^2 < 1$

[1] `https://archive.ics.uci.edu/ml/index.php`

Table 4.3: Real-world benchmark datasets.

(Abbreviation) Name	Examples (n)	Features	Variance of y	Link
(A) Airfoil	1503	5	$4.756 \cdot 10$	goo.gl/uNMLv3
(B) Boston housing	506	13	$8.442 \cdot 10$	goo.gl/KxCnq1
(C) Concrete strength	1030	8	$2.788 \cdot 10^{2}$	goo.gl/Gjq9oN
(D) Dow chemical	1066	57	$1.228 \cdot 10^{-1}$	goo.gl/9D2z3b
(Ec) Energy cooling	768	8	$9.039 \cdot 10$	goo.gl/ANV6dW
(Eh) Energy heating	768	8	$1.017 \cdot 10^{2}$	goo.gl/ANV6dW
(T) Tower	4999	26	$6.518 \cdot 10^{-1}$	goo.gl/9D2z3b
(Wr) Wine red	1599	11	$7.842 \cdot 10^{-1}$	goo.gl/inDsCE
(Ww) Wine white	4899	11	$7.702 \cdot 10^{3}$	goo.gl/inDsCE
(Y) Yacht hydrodynamics	308	6	$2.291 \cdot 10^{2}$	goo.gl/cmkRor

4.7. Results

We proceed by showing the results of the following experiments. Firstly we consider whether using LS in synergy with SB is beneficial compared to using it independently. Secondly, we compare all configurations of SB-based GP with SGP, by fixing the maximum number of generations, and observing the time taken. Thirdly, we repeat the previous experiment, but this time using a fixed time budget, to take into account computational expensiveness.

4.7.1. Independent vs. synergistic linear scaling with semantic backpropagation

Table 4.4 shows the median error obtained by end-of-run best trees found using SB-based GP without LS (noLS), with LS but independently from SB (iLS), and with LS in synergy with SB (sLS), after 100 generations. The results are reported in terms of variance-Normalized MSE (NMSE), given by dividing the MSE by the variance of the dependent variable y, to have results of similar order of magnitude, and multiplying by 100.

Evidently, iLS has much better training and test performance compared to noLS. This is always significant with respect to training NMSE, and is also significant on all datasets but for Boston at test time. However, to use LS in synergy with SB is even better, significantly outperforming both noLS (all cases) and iLS on 8/10 datasets both at training and test time. Our hypothesis that using LS in synergy with SB is beneficial is therefore experimentally confirmed.

4.7.2. SB-based GP vs. standard GP

The next results regard the comparison between SB-based GP and SGP, with and without using LS to scale the error and within library search.

Budget of 100 generations

Figure 4.4 shows, for each dataset, the evolution of the best training fitness for SGP, SGP with LS (SGP$_{+LS}$), SB-based GP with traditional RDO (RDO), RDO using LS in synergy during backpropagation (RDO$_{+\text{LS}}$), RDO using LS within library search (RDO$^{\text{xLS}}$), and RDO using both LS in synergy with backpropagation and within library search (RDO$^{\text{xLS}}_{+\text{LS}}$).

Table 4.4: Training and test median NMSEs for SB-based GP without LS (noLS), with LS used independently (iLS), and with LS used in synergy with SB (sLS). Underlined results are best in that no other is significantly better.

	Train NMSE			Test NMSE		
	noLS	iLS	sLS	noLS	iLS	sLS
A	41	29	22	43	32	29
B	27	17	14	27	21	16
C	34	19	15	37	21	18
D	71	29	20	69	28	21
Ec	9.9	6.9	4.9	8.7	7.1	5.4
Eh	6.8	4.0	5.4	8.7	6.2	7.1
T	16	14	13	17	14	14
Wr	69	63	60	65	63	62
Ww	75	69	67	78	71	70
Y	.62	.44	.40	.94	.62	.61
# Best	0	2	10	0	2	10

RDO and SGP are complementary: one is better than the other on half cases. However, on Tower and Yacht, SGP has much larger errors. In some cases (Airfoil, Boston, Concrete, Wine white, Yacht), it is noticeable that the error of SGP levels off less markedly than the one of RDO, thus a larger generational budget may favor SGP. $\text{RDO}_{+\text{LS}}$ is better than SGP_{+LS} on all datasets but Yacht.

RDO^{xLS} and $\text{RDO}^{\text{xLS}}_{+\text{LS}}$ are consistently the best performing, with the second reaching slightly smaller errors than the first. Moreover, both algorithms have smaller variances than $\text{RDO}_{(+\text{LS})}$. This is because the use of LS within library search (xLS) dynamically adapts the library to the desired output that is searched. Without xLS, the expressiveness of the library is more aleatory as it completely depends on the subtrees from the population.

Table 4.5 shows training and test NMSEs of end-of-run best trees. The training errors reflect what already seen in Figure 4.4. Test errors are typically similar to training ones, for all the algorithms. $\text{RDO}^{\text{xLS}}_{+\text{LS}}$ is the best performing, while RDO^{xLS} is the second best. On Wine red at test time, $\text{SGP}_{+\text{LS}}$ is preferable over $\text{RDO}^{\text{xLS}}_{+\text{LS}}$, which indicates that the latter overfits (slightly).

Time taken by 100 generations

Figure 4.5 show the time taken to perform 100 generations for the algorithms. The difference between the times taken by SGP and SGP_{+LS} and the various configurations of RDO is very large. For Yacht, that has 308 examples, RDO takes around 20 times longer than $\text{SGP}_{(+\text{LS})}$; For Tower, that has 4999 examples, RDO takes around 100 times longer than $\text{SGP}_{(+\text{LS})}$. This result strongly motivates the need for a time-based comparison between SGP and RDO configurations, for fairness.

The use of LS in addition to RDO, or within library search, does, on average, increase running times. However, these running times are not too dissimilar if put in perspective to the times taken by $\text{SGP}_{(+\text{LS})}$. This is expected because +LS and xLS do not affect computational time bounds. RDO and $\text{RDO}_{+\text{LS}}$ have larger variations (some of the extreme time points of RDO are considered outliers). These variations in time are linked to the variations already seen in terms of fitness (e.g., see the Energy datasets in Fig. 4.4).

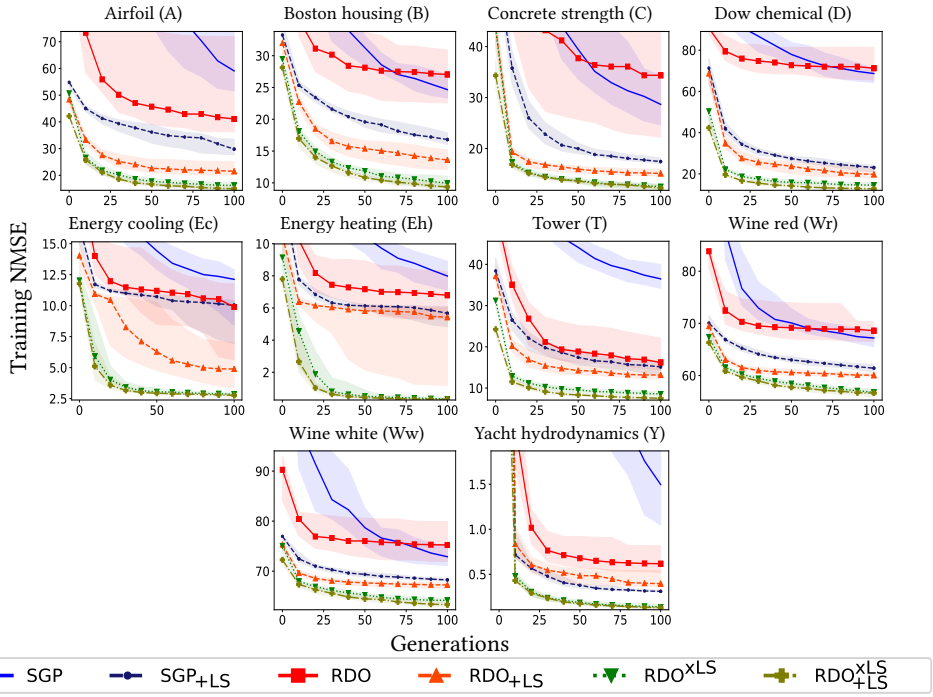

Figure 4.4: Median best training NMSE (25th and 75th percentiles within shaded area) in 100 generations.

Table 4.5: Training and test median NMSEs, for the experiments with a budget of 100 generations. Underlined results are best in that no other is significantly better.

	Train NMSE						Test NMSE					
	SGP	SGP$_{+LS}$	RDO	RDO$_{+LS}$	RDO^{xLS}	RDO$^{xLS}_{+LS}$	SGP	SGP$_{+LS}$	RDO	RDO$_{+LS}$	RDO^{xLS}	RDO$^{xLS}_{+LS}$
A	59	30	41	22	16	15	60	32	43	29	20	19
B	25	17	27	14	10	9.4	23	17	27	16	16	14
C	29	17	34	15	13	12	30	20	37	18	15	15
D	69	23	71	20	15	13	65	23	69	21	16	15
Ec	12	10	9.9	4.9	2.9	2.8	11	9.0	8.7	5.4	3.4	3.2
Eh	8.0	5.7	6.8	5.4	.35	.32	10	7.6	8.7	7.1	.50	.40
T	36	15	16	13	8.6	7.5	36	15	17	14	10	9.5
Wr	67	61	69	60	57	57	65	62	65	62	62	65
Ww	73	68	75	67	64	63	75	70	78	70	69	68
Y	1.5	.31	.62	.40	.14	.13	1.9	.40	.90	.60	.30	.30
# Best	0	0	0	0	1	10	0	1	0	1	4	8

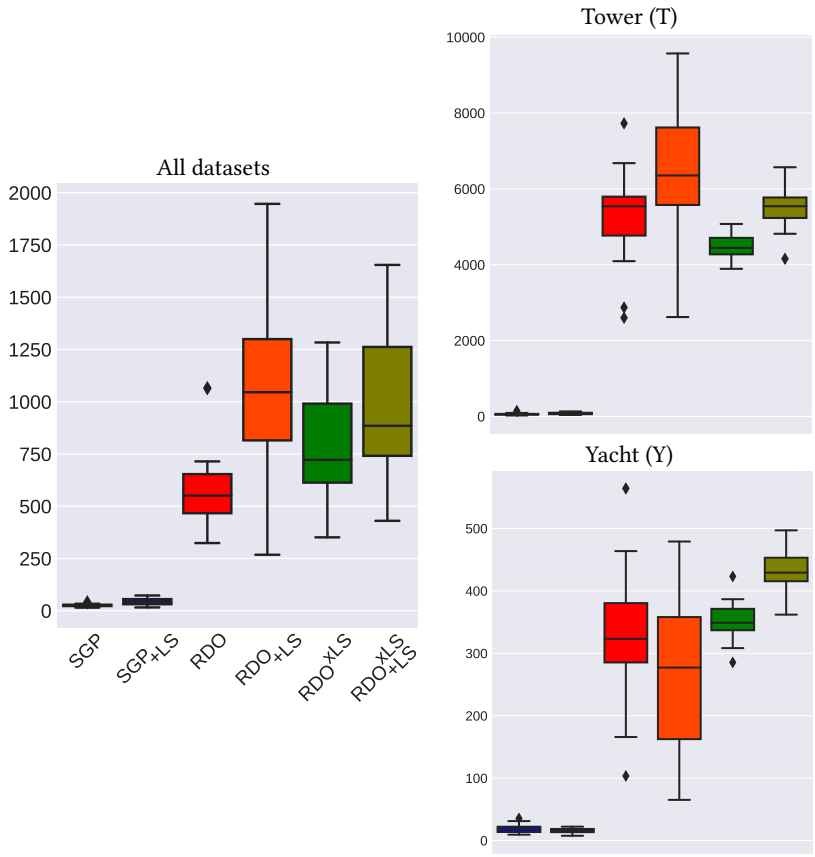

Figure 4.5: Time (seconds) to complete 100 generations. Left: Mean time over all datasets; Right: Time by the 30 runs on Tower (top), and Yacht (bottom); Boxes extend from the 25th to the 75th percentiles (inner bar is the 50th), whiskers from the 10th to the 90th. Diamonds are outliers.

Budget of 1000 seconds

The evolution of the best training fitness in time is reported in Figure 4.6, for each dataset and algorithm. The conclusions that can be drawn from these results are different from the ones based on a generational limit. For SGP, the use of a time limit of 1000 seconds seems more appropriate than the limit of 100 generations, since the fitness tends to plateau more in this case (this is particularly evident for the smallest dataset Yacht).

Now, RDO performs markedly worse than SGP, and $\text{RDO}_{+\text{LS}}$ is also worse than SGP_{+LS}, with the latter typically achieving close performance to RDO^{xLS}. While in a time-based comparison RDO performs quite poorly, it is interesting to see that, instead, RDO^{xLS} and $\text{RDO}^{\text{xLS}}_{+\text{LS}}$ still perform very well. Indeed, the inclusion of LS within library search makes variation so effective that, even if library search itself becomes slower, fitter trees are discovered sooner. While it is perhaps not surprising that xLS makes variation improve, it is interesting to see the extent of this improvement.

Table 4.6 summarizes both training and test NMSEs of end-of-run best trees. The tests for statistical significance confirm what already seen in the training fitness convergence plots of Figure 4.6: $\text{RDO}^{\text{xLS}}_{+\text{LS}}$ is the top performing algorithm, followed by RDO^{xLS}. In terms of error magnitudes, SGP_{+LS} is relatively close to RDO^{xLS} and $\text{RDO}^{\text{xLS}}_{+\text{LS}}$ (yet often significantly worse), compared to SGP and RDO w/o LS.

When it comes to generalization, $\text{RDO}^{\text{xLS}}_{+\text{LS}}$ is still preferable, as it is significantly worse than another algorithm only on 2 datasets, by relatively small magnitudes. SGP_{+LS} leads to very good generalization on 3 out of 10 datasets, indicating that $\text{RDO}^{\text{xLS}}_{+\text{LS}}$, which was better at training time, delivered slightly overfitted trees.

All in all, our results show that scaling the trees during library search is extremely valuable for RDO. In addition, to consider LS when backpropagating, i.e., $\text{RDO}^{\text{xLS}}_{+\text{LS}}$, gives a further edge.

4.8. Discussion

We showed that a comparison between RDO and SGP on real-world datasets strongly depends on how this comparison is framed. With a typical budget of 100 generations, the algorithms perform complementarily. Instead, when the comparison is framed in terms of time, RDO performs worse than SGP. Our proposal of incorporating LS within the mechanisms of RDO makes the algorithm much more effective even if extra computations take place, and makes it capable of outperforming all the other algorithms.

We now discuss some limitations of this chapter. To begin, we used typical parameter settings. One may wonder whether our findings do generalize to other configurations. Population sizing is perhaps the most interesting aspect to consider [20], especially when using a population-based library (as it uses all subtrees with unique output from the population). If a library is large enough, i.e., if it has enough representative power, the adoption of LS may become redundant. However, because LS applies a linear transformation that is *optimal*, we argue that populations and libraries will likely need to grow too big to be practically usable to compete with LS. As to other parameters, we believe that the magnitude of our results, as well as the small variances found along the runs, strongly indicate that the use of LS within library search will remain beneficial for many other parameter settings.

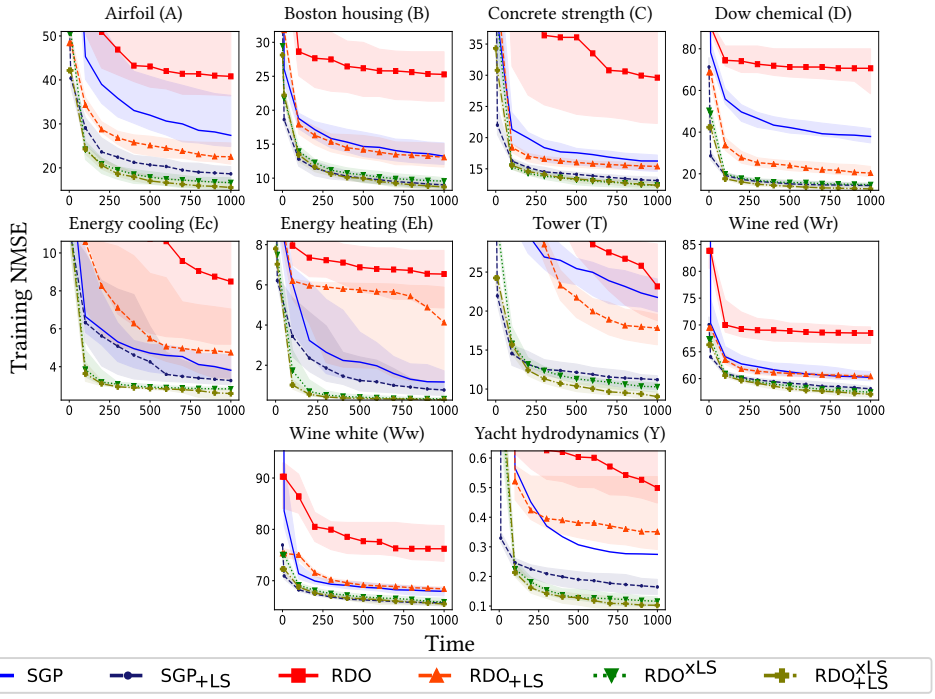

Figure 4.6: Median best training NMSE (25th and 75th percentiles within shaded area) in 1000 seconds.

Table 4.6: Training and test median NMSEs, for the experiments with a budget of 1000 seconds. Underlined results are best in that no other is significantly better.

	Train NMSE						Test NMSE					
	SGP	SGP$_{+LS}$	RDO	RDO$_{+LS}$	RDO^{xLS}	RDO$^{xLS}_{+LS}$	SGP	SGP$_{+LS}$	RDO	RDO$_{+LS}$	RDO^{xLS}	RDO$^{xLS}_{+LS}$
A	27	19	41	23	17	16	32	22	44	30	20	20
B	13	9.0	25	13	10	8.7	17	14	26	15	16	14
C	16	13	30	15	12	12	18	16	37	19	15	15
D	38	14	71	20	15	13	43	16	68	22	16	15
Ec	3.8	3.3	8.5	4.7	2.8	2.6	4.8	3.8	7.4	5.6	3.4	3.2
Eh	1.2	.77	6.5	4.1	.33	.28	1.5	.91	8.4	5.4	.45	.35
T	22	11	23	18	10	9.0	22	12	23	19	12	11
Wr	60	58	69	60	57	57	63	62	65	62	62	63
Ww	68	66	76	68	66	66	70	69	79	70	69	69
Y	.27	.16	.50	.35	.12	.10	.49	.33	.80	.55	.35	.33
# Best	0	0	0	0	2	10	0	3	0	2	4	8

Another limit of our approach, in particular of using LS within library search, is the growth of tree size. Any time a tree is retrieved from the library, four nodes are added to incorporate the effect of LS. Larger trees are slower to evaluate (and require more memory to cache intermediate outputs), and are also less likely to lead to interpretable expressions. Interpretability of machine learning models can be a relevant aspect for practical applications, e.g., in healthcare [13, 21]. By including LS within library search, we did find trees to grow bigger, with the best trees found by $\text{RDO}^{\text{xLS}}_{\text{+LS}}$ being on average 1.1 times larger than the the ones found by $\text{SGP}_{\text{+LS}}$ in the time-based comparison. We claim that this difference in size is largely unimportant. Both algorithms deliver quite big trees anyway, with approximately 325 nodes for $\text{SGP}_{\text{+LS}}$ and 360 nodes for $\text{RDO}^{\text{xLS}}_{\text{+LS}}$, on average. From a performance perspective, the increment in time taken to evaluate a larger tree is limited, but may become noticeable for much larger datasets than the ones we considered. As to interpretability, the algorithms deliver trees that are equivalently too large to result in interpretable expressions. Future work may therefore focus on reducing tree size, e.g., by exploring the inclusion of bloat control methods [22], or by expressing preference for smaller trees as a secondary objective [23]. However, if having trees with only a few dozen nodes is truly desired, we believe substantially different approaches to GP may need to be taken, such as modern model-based GP (Chapters 2 and 3).

Another aspect worth investigating is the use of more efficient data structures to implement the library. In [6], k-d trees are used [24, 25]. We did experiments with this data structure, and although searching k-d trees may not be quick for datasets with many examples [25], we did observe speed ups for the datasets we considered. Unfortunately, LS cannot be used within k-d tree search. This is because a k-d tree is built exploiting the fixed distribution of tree outputs, to cut exploration branches when searching. To apply LS means to dynamically change such a distribution.

We did experiments with adopting k-d trees jointly with the computation of *only* the optimal translation coefficient a. This can be achieved by (i) Subtracting the mean of the output o of each library tree prior to building the k-d tree; (ii) Subtracting the mean of the desired output d prior to k-d tree search; (iii) Incorporating the addition of $a = \bar{d} - \bar{o}$ to the tree returned by the search. This achieves the optimal translation (see Eq 4.2). However, we found this to be less effective than using LS (which also computes b) within traditional library search. To find an efficient data structure that enables the use of LS or of a similarly powerful method, as well as investigating code parallelization, may allow to use SB-based GP for large scale symbolic regression.

4.9. Conclusion

We presented the use of Linear Scaling (LS) in synergy with Semantic Backpropagation (SB), and within library search, in Genetic Programming (GP) for symbolic regression. We validated the proposed adaptations on ten real-world datasets, comparing various GP configurations using a generational and a time budget. We found that incorporating LS within SB-based GP leads to much lower errors in both cases, and outperforms the use of traditional variation operators. Lastly, the cost incurred by our adaptations is limited, as the asymptotic time bounds of SB-based GP remain unchanged.

Acknowledgments

Financial support for this work was provided by Stichting Kinderen Kankervrij (Children Cancer-free Foundation), project #187. We further thank the Maurits en Anna de Kock Foundation for financing a high-performance computing system, and SURFsara for the support in using the Lisa Compute Cluster. We also thank professor Krzysztof Krawiec from the Poznan University of Technology, Poland, for the insightful exchanges on semantic backpropagation-based genetic programming.

References

[1] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection* (MIT Press, 1992).

[2] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming* (Lulu Enterprises, UK Ltd, 2008).

[3] B. Wieloch and K. Krawiec, *Running programs backwards: instruction inversion for effective search in semantic spaces,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 1013–1020.

[4] T. P. Pawlak, B. Wieloch, and K. Krawiec, *Semantic backpropagation for designing search operators in genetic programming,* IEEE Transactions on Evolutionary Computation **19**, 326 (2015).

[5] R. Ffrancon and M. Schoenauer, *Memetic semantic genetic programming,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2015) pp. 1023–1030.

[6] K. Krawiec and T. Pawlak, *Approximating geometric crossover by semantic backpropagation,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013) pp. 941–948.

[7] M. Keijzer, *Improving symbolic regression with interval arithmetic and linear scaling,* in *European Conference on Genetic Programming* (Springer, 2003) pp. 70–82.

[8] M. Keijzer, *Scaled symbolic regression,* Genetic Programming and Evolvable Machines **5**, 259 (2004).

[9] M. Szubert, A. Kodali, S. Ganguly, K. Das, and J. C. Bongard, *Semantic forward propagation for symbolic regression,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2016) pp. 364–374.

[10] Q. Chen, M. Zhang, and B. Xue, *Geometric semantic genetic programming with perpendicular crossover and random segment mutation for symbolic regression,* in *Asia-Pacific Conference on Simulated Evolution and Learning* (Springer, 2017) pp. 422–434.

[11] A. Raja, R. M. A. Azad, C. Flanagan, and C. Ryan, *Real-time, non-intrusive evaluation of VoIP,* in *European Conference on Genetic Programming* (Springer, 2007) pp. 217–228.

[12] F. Archetti, I. Giordani, and L. Vanneschi, *Genetic programming for anticancer therapeutic response prediction using the NCI-60 dataset,* Computers & Operations Research **37**, 1395 (2010).

[13] M. Virgolin, T. Alderliesten, A. Bel, C. Witteveen, and P. A. N. Bosman, *Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2018) pp. 1395–1402.

[14] D. Costelloe and C. Ryan, *On improving generalisation in genetic programming,* in *European Conference on Genetic Programming* (Springer, 2009) pp. 61–72.

[15] Q. Chen, B. Xue, and M. Zhang, *Improving generalization of genetic programming for symbolic regression with angle-driven geometric semantic operators,* IEEE Transactions on Evolutionary Computation **23**, 488 (2018).

[16] Q. N. Huynh, S. Chand, H. K. Singh, and T. Ray, *Genetic programming with mixed-integer linear programming-based library search,* IEEE Transactions on Evolutionary Computation **22**, 733 (2018).

[17] J. Ni, R. H. Drieberg, and P. I. Rockett, *The use of an analytic quotient operator in genetic programming,* IEEE Transactions on Evolutionary Computation **17**, 146 (2013).

[18] D. R. White, J. Mcdermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly, and S. Luke, *Better GP benchmarks: community survey results and proposals,* Genetic Programming and Evolvable Machines **14**, 3 (2013).

[19] J. Demšar, *Statistical comparisons of classifiers over multiple data sets,* Journal of Machine Learning Research **7**, 1 (2006).

[20] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, *The gambler's ruin problem, genetic algorithms, and the sizing of populations,* Evolutionary Computation **7**, 231 (1999).

[21] Z. C. Lipton, *The mythos of model interpretability,* Queue **16**, 30:31 (2018).

[22] S. Luke and L. Panait, *A comparison of bloat control methods for genetic programming,* Evolutionary Computation **14**, 309 (2006).

[23] A. Ekárt and S. Z. Nemeth, *Selection based on the Pareto nondomination criterion for controlling code growth in genetic programming,* Genetic Programming and Evolvable Machines **2**, 61 (2001).

[24] J. L. Bentley, *Multidimensional binary search trees used for associative searching,* Communications of the ACM **18**, 509 (1975).

[25] J. H. Friedman, J. L. Bentley, and R. A. Finkel, *An algorithm for finding best matches in logarithmic time,* ACM Transactions on Mathematical Software **3**, 209 (1976).

5

Explainable Machine Learning by Evolving Crucial and Compact Features

Feature construction can substantially improve the accuracy of Machine Learning (ML) algorithms. Genetic Programming (GP) has been proven to be effective at this task by evolving non-linear combinations of input features. GP additionally has the potential to improve ML explainability since explicit expressions are evolved. Yet, in most GP works the complexity of evolved features is not explicitly bound or minimized though this is arguably key for explainability. In this chapter, we assess to what extent GP still performs favorably at feature construction when constructing features that are (1) Of small-enough number, to enable visualization of the behavior of the ML model; (2) Of small-enough size, to enable interpretability of the features themselves; (3) Of sufficient informative power, to retain or even improve the performance of the ML algorithm. We consider a simple feature construction scheme using three different GP algorithms, as well as random search, to evolve features for five ML algorithms, including support vector machines and random forest. Our results on 21 datasets pertaining to classification and regression problems show that constructing only two compact features can be sufficient to rival the use of the entire original feature set. We further find that a modern GP algorithm, GP-GOMEA, performs best overall. These results, combined with examples that we provide of readable constructed features and of 2D visualizations of ML behavior, lead us to positively conclude that GP-based feature construction still works well when explicitly searching for compact features, making it extremely helpful to explain ML models.

The contents of this chapter are based on the following publication: **M. Virgolin**, T. Alderliesten, and P.A.N. Bosman. On explaining machine learning models by evolving crucial and compact features. *Swarm and Evolutionary Computation* **53**, pp. 100640, Elsevier (2020).

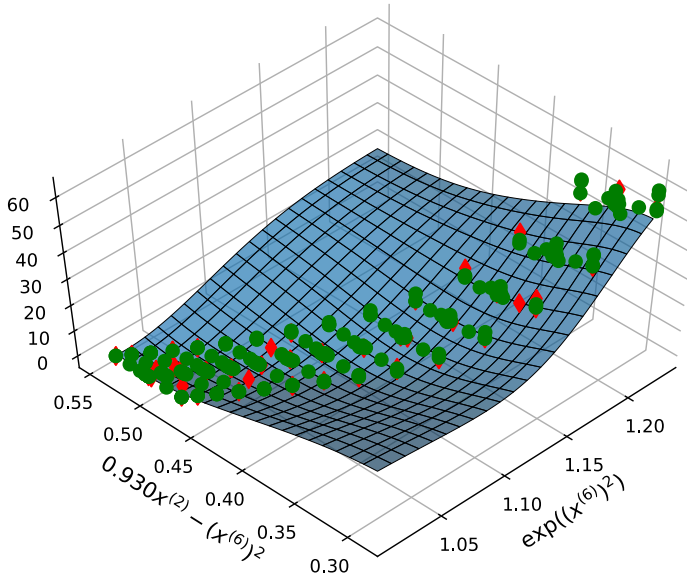

Figure 5.1: Regression surface learned by SVM for the Yacht dataset (in blue), expressed as a 2D function of the two features (on the bottom axes) constructed by our approach. Circles are training samples, diamonds are test samples. The dataset has six features ($x^{(i)}$). Our approach constructs two new features (using GP-GOMEA, see Sec. 5.4.1), which are non-linear transformations of the prismatic coefficient ($x^{(2)}$) and the Froude number ($x^{(6)}$). With only two features the SVM prediction surface can be visualized. Moreover, these new features are understandable. Finally, the modeling quality is actually improved over employing SVM directly on all six features. The coefficient of determination of SVM increased from 85% using the original features to 98% using the two new features.

5.1. Introduction

Feature selection and feature construction are two important steps to improve the performance of any Machine Learning (ML) algorithm [1, 2]. Feature selection is the task of excluding features that are redundant or misleading. Feature construction is the task of transforming (parts of) the original feature space into one that the ML algorithm can better exploit.

A very interesting method to perform feature construction automatically is Genetic Programming (GP) [3, 4]. GP can synthesize functions without many prior assumptions on their form, differently from, e.g., logistic regression or regression splines [5, 6]. Moreover, feature construction not only depends on the data at hand, but also on the way a specific ML algorithm can model that data. Evolutionary methods in general are highly flexible in their use due to the way they perform search (i.e., derivative free). This makes it possible, for example, to evaluate the quality of a feature for a specific ML algorithm by directly measuring what its impact is on the performance of the ML algorithm (i.e., by training and validating the ML algorithm when using that feature).

Explaining what constructed features mean can shed light on the behavior of ML-inferred models that use such features. Reducing the number of features is also important to improve interpretability. If the original feature space is reduced to few constructed features (e.g., up to two for regression and up to three for classification), the function learned by the ML model can be straightforwardly visualized with respect to the new features. In fact, how to make ML models more understandable is a key topic of modern ML research, as many practical, sensitive applications exist, where explaining (part of) the behavior of ML models is essential to trust their use (e.g., in medical applications) [7–10]. Typically, GP for feature construction searches in a subspace of mathematical expressions. Adding to the appeal and potential of GP, these expressions can be human-interpretable if simple enough [8, 11].

Figure 5.1 presents an example of the potential held by such an approach: a multi-dimensional dataset transformed into a 2D one, where both the behavior of the ML algorithm and the meaning of the new features is clear, while the performance of the ML algorithm is not compromised with respect to the use of the original feature set (it is actually improved).

In this chapter we study whether GP can be useful to construct a *low* number of *small* features, to increase the chance of obtaining interpretable ML models, without compromising their accuracy (compared to using the original feature set). To this end, we design a simple, iterative feature construction scheme, and perform a wide set of experiments: we consider four types of feature construction methods (three GP algorithms and random search), five types of machine learning algorithms. We apply their combinations on 21 datasets between classification and regression to determine to what extent they are capable of effectively and efficiently finding crucial and compact features for specific ML algorithms.

The main original scientific contribution of this work is an investigation of whether GP can be used to construct features that are:

- Of small-enough number, to enable visualization of the behavior of the ML model;
- Of small-enough size, to enable interpretability of the features themselves;
- Of sufficient informative power, to retain or even improve the performance of the ML, compared to using the original feature set;

These aspects are assessed under different circumstances:

- We test different search algorithms, including modern model-based GP and random search;
- We test different ML algorithms.

The remainder of this chapter is organized as follows. Related work is reported in Section 5.2. The proposed feature construction scheme is presented in Section 5.3. The search algorithms to construct features, as well as the considered ML algorithms, are presented in Section 5.4. The experimental setup is described in Section 5.5. Results related to performance are reported in Section 5.6 and 5.7, while results concerning interpretability are reported in Section 5.8. Considerations on running time are presented in Section 5.9. Section 5.10 discusses our findings, and Section 5.11 concludes this chapter.

5.2. Related work

In this chapter, we consider GP for feature construction to achieve better explainable ML models. Different forms of GP to obtain explainable ML have been explored in literature, but they do not necessarily leverage feature construction. For example, [12] introduced a form of GP for the automatic synthesis of interpretable classifiers, generated from scratch as self-contained ML models, made of IF-THEN rules. A very different paradigm for explainable ML by GP is considered in [13], where the authors explore the use of GP to recover the behavior of a given unintelligible classifier by evolving interpretable approximation models. Other GP-based approaches and paradigms to synthesize interpretable ML models from scratch, or to approximate the behavior of pre-existing ML models by interpretable expressions, are reported in recent surveys on explainable artificial intelligence such as [8, 9].

Since in this chapter we particularly study what the potential of GP for feature construction is in terms of added value for explaining complex, not directly explainable models learned by various popular ML algorithms, the related work that follows describes GP approaches for feature construction. For readers interested in feature selection, we refer to a recent survey [14].

One of the first approaches of GP for feature construction is presented in [15]. There, each GP solution is a set of K features. The fitness of a set is the cross-validation performance of a decision tree [16] using that set. The results on six classification datasets show that the approach is able to synthesize a feature set that is competitive with the original one, and can also be added to the original set for further improvements. No attention is however given to the interpretability of evolved features.

The work in [17] generates one feature with Standard, tree-based GP (SGP) [3], to be added to the original set. Feature importance metrics of decision trees such as information gain, Gini index and Chi^2 are used as fitness measure. An advantage of using such fitness measures over ML performance is that they can be computed very quickly. However, they are decision tree-specific. Results show that the approach can improve prediction accuracy, and, for a few problems, it is shown that decision trees that are simple enough to be reasonably interpretable, can be found.

Feature construction for high-dimensional datasets is considered in [18], for eight biomedical binary classification problems, with 2,000 to 24,188 features. This approach is different from the typical ones, as the authors propose to use SGP to evolve classifiers rather than features, and extract features from the components (subtrees) of such classifiers. These are then used as new features for an ML algorithm. Results on K-Nearest Neighbors [19], Naive Bayes classifier [20, 21], and decision tree show that a so-found feature set can be competitive or outperform the original one. The authors show an example where a single interpretable feature is constructed that enables linear separation of the classification examples.

Different from the aforementioned works, [22] explores feature construction for regression. A SGP-based approach is designed to tackle regression problems with a large number of features, and is tested on six datasets. Instead of using the constructed features for a different ML algorithm, SGP dynamically incorporates them within an ongoing run, to enrich the terminal set. Every α generations of SGP, the subtrees composing the best solutions become new features by encapsulation into new terminal nodes. The approach

is found to improve the ability of SGP to find accurate solutions. However, the features found by encapsulating subtrees are not interpretable because allowing subsequent encapsulations leads to an exponential growth of solution size.

A recent work that focuses on evolutionary dimensionality reduction and consequent visualization is [23], where a multi-objective, grammar-based SGP approach is employed. K feature transformations are evolved in synergy to enable, at the same time, good classification accuracy, and visualization through dimensionality reduction. The system is thoroughly tested on 42 classification tasks, showing that the algorithm performs well compared to state-of-the-art dimensionality reduction methods, and it enables visualization of the learned space. However, as trees are free to grow up to a height of 50, the constructed features themselves cannot be interpreted.

The most similar works to ours that we found are [24] and [25]. In [24], which is our previous work, the possibility of using a modern model-based GP algorithm (which we also use in our comparisons) for feature construction is explored on four regression datasets. There, focus is put on keeping feature size small, to actively attempt to obtain readable features. These features are iteratively constructed to be added to the original feature set to improve the performance of the ML algorithm, and three ML algorithms are compared (linear regression, support vector machines [26], random forest [27]). Reducing the feature space to enable a better understanding of inferred ML models is not considered.

In [25], different feature construction approaches are compared on gene-expression datasets that have a large number of features (thousands to tens of thousands) to study if evolving class-dependent features, i.e., features that are each targeted at aiding the ML algorithm detect one specific class, can be beneficial. Similarly to us, the authors show visualizations of feature space reduced to up to three constructed features, and an example of three features that are encoded as very small, easy-to-interpret trees. However, such small features are a rare outcome as the trees used to encode features typically had more than 75 nodes. These trees are therefore arguably extremely hard to read and interpret.

Our work is different from previous research in two major aspects. First, none of the previous work principally addresses the conflicting objectives of retaining good performance of an ML algorithm while attempting to explain both its behavior (by dimensionality reduction to allow visualization), and the meaning of the features themselves (by constraining feature complexity). Second, multiple GP algorithms within a same feature construction scheme, on multiple ML algorithms, are not compared in previous work. Most of the times, it is a different feature construction scheme that is tested, using arguably small variations of SGP. Here, we consider random search, two versions of SGP, as well as another modern GP algorithm. Furthermore, we adopt both "weak" ML algorithms such as ordinary least squares linear regression and the naive Bayes classifier, as well as "strong", state-of-the-art ones, which are rarely used in literature for feature construction, such as support vector machine and random forest; on both classification and regression tasks.

5.3. Iterative evolutionary feature construction

We use a remarkably simple scheme to construct features. Our approach constructs $K \in \mathbb{N}^+$ features by iterating K GP runs. The evolution of the k-th feature ($k \in \{1, \ldots, K\}$) uses the previously constructed $k-1$ features.

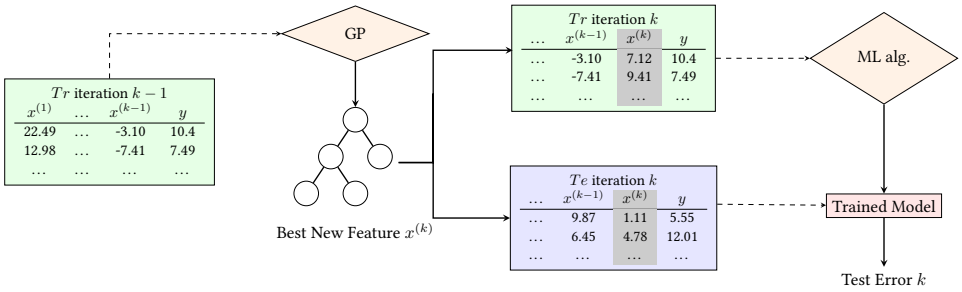

Figure 5.2: Construction of the k-th feature and computation of the k-th test error. Evolved features use the features of the original dataset (not shown) and random constants as terminal nodes. Dashed arrows represent inputs, solid arrows represent outputs.

5.3.1. Feature construction scheme

The dataset D defining the problem at hand is split into two parts: the training Tr and the test Te set. This partition is kept fixed through the whole procedure. Only Tr is used to construct features, while Te is exclusively used for final evaluation to avoid positive bias in the results [28]. We use the notation $x_j^{(i)}$ to refer to the i-th feature value of the j-th example, and y_j for the desired outcome (label for classification or target value for regression) of the j-th example.

The k-th GP run evolves the k-th feature. An example is shown in Figure 5.2. Each solution in the population competes to become the new feature $x^{(k)}$, that represents a transformation of the original feature set. In every run, the population is initialized at random.

We evaluate the fitness of a feature of the k-th run by measuring the performance of the ML algorithm on a dataset that contains that feature and the previously evolved $k-1$ features.

We only use original features (and random constants) as terminals. In particular, the features constructed by previous iterations are not used as terminal nodes in the k-th run. This prevents the generation of nested features, which could harm interpretability.

At the end of the k-th run, the best feature is stored and its values $x_j^{(k)}$ are added to Tr and Te for the next iterations.

5.3.2. Feature fitness

The fitness of a feature is computed by measuring the performance (i.e., error) of the ML algorithm when the new feature is added to Tr. We consider the C-fold cross-validation error rather than the training error to promote generalization and prevent overfitting. The pseudo code of the evaluation function is shown in Algorithm 5.1.

Specifically, the C-fold cross-validation error is computed by partitioning Tr into C splits. For each $c = 1, \ldots, C$ iteration, a different split is used for validation (set V^c), and the remaining $C-1$ splits are used for training (set Tr^c). The mean validation error is the final result.

For classification tasks, in order to take into account both multiple and possibly imbalanced class distributions, the prediction error is computed as 1 minus the *macro* $F1$ score, i.e., 1 minus the mean of the class-specific $F1$ scores:

$$\begin{aligned} 1 - F1 &= 1 - \frac{1}{\#classes} \sum_{\gamma \in classes} F1_\gamma \\ &= 1 - \frac{2}{\#classes} \sum_{\gamma \in classes} \frac{\frac{TP_\gamma}{TP_\gamma + FP_\gamma} \frac{TP_\gamma}{TP_\gamma + FN_\gamma}}{\frac{TP_\gamma}{TP_\gamma + FP_\gamma} + \frac{TP_\gamma}{TP_\gamma + FN_\gamma}}, \end{aligned} \tag{5.1}$$

where $TP_\gamma, FN_\gamma, FP_\gamma$ are the true positive, false negative, and false positive classifications for the class γ, respectively. If the computation of $F1_\gamma$ results in $\frac{0}{0}$, we set $F1_\gamma = 0$.

For regression, the prediction error is computed with the Mean Squared Error (MSE).

Algorithm 5.1 Computation of the fitness of a feature s

```
1 function ComputeFeatureFitness(s)
2     Tr' ←AddFeatureToCurrentTrainingSet(s)
3     error ← 0
4     for c = 1, ..., C do
5         T^c, V^c ←SplitSet(c, C, Tr')
6         M ←TrainMLModel(T^c)
7         error ← error+ComputeError(M, V^c)
8     Return(error/C)
```


5.3.3. PREVENTING UNNECESSARY FITNESS COMPUTATIONS

Computing the fitness of a feature is particularly expensive, as it consists of a C-fold cross-validation of the ML algorithm. This limits the feasibility of, e.g., adopting large population sizes and large numbers of evaluations for the GP algorithms.

We therefore attempt to prevent unnecessary cross-validation calls, by assessing if features meet four criteria. Let n be the number of examples in Tr. The criteria are the following:

1. *The feature is not a constant.* We avoid evaluating constant features as they are likely to be useless for many ML algorithms, which internally already compute an intercept.
2. *The feature does not contain extreme values that may cause numerical errors*, i.e., with absolute value above a lower-bound β_ℓ or above an upper-bound β_u. Here, we set $\beta_\ell = 10^{-10}$, and $\beta_u = 10^{10}$ (none of the datasets considered here have values exceeding these bounds).
3. *The feature is not equivalent to one constructed in the previous $k-1$ iterations.* Equivalence is determined by checking the values available in Tr, i.e., equivalence holds if:
$$\forall j \in Tr, \exists i \in \{1, \ldots, k-1\} : x_j^{(k)} = x_j^{(i)}. \tag{5.2}$$

Note that a constructed feature that is equivalent to a feature of the original feature set can be valid, as long as no other previously constructed feature exists that is already equivalent. Thus, our approach can in principle perform pure feature selection.

4. *The values of the feature in consideration have changed since the last time the feature was evaluated.* GP variation can change the syntax of a feature without necessarily affecting its behavior (e.g., inserting a multiplication by 1 will not change the final values a feature computes). If the values do not change, then the fitness of the feature will not change either (see Sec. 5.3.2). We therefore avoid unnecessary re-computations of feature fitnesses, by caching the feature values prior to GP variation, and checking whether they have changed after variation.

The computational effort for each criterion is $O(n)$ (it is $O((k-1)n)$ for criterion 3, however in our experiments $k \ll n$). The fitness of a feature failing criterion 1, 2, or 3 is set to the maximum possible error value. If criterion 4 fails, the fitness remains the same (although performing cross-validation may lead to slightly different results when using stochastic ML algorithms like random forest).

5.4. Considered search algorithms and machine learning algorithms

We consider SGP, Random Search (RS), and the GP instance of the Gene-pool Optimal Mixing Evolutionary Algorithm (GP-GOMEA) as competing search algorithms to construct features. SGP is widely used in feature construction (see related work in Sec. 5.2). RS is not typically considered, yet we believe it is important to assess whether evolution does bring any benefit over random enumeration within the confines of our study, i.e., when forcing to find small features. GP-GOMEA is a recently introduced GP algorithm that has proven to be particularly proficient in evolving accurate solutions of limited size (Chapters 2 and 3, and [24]).

As ML algorithms, we consider the Naive Bayes classifier (NB), ordinary least-squares Linear Regression (LR), Support Vector Machines (SVM), Random Forest (RF), and eXtreme Gradient Boosting (XGB). NB is used only for classification tasks, LR only for regression tasks, SVM, RF, and XGB for both tasks. We provide more details in the following sections.

5.4.1. Details on the search algorithms

All search algorithms use the fitness evaluation function. A feature s is evaluated by first checking whether the four criteria of Section 5.3.3 are met, and then, if the outcome is positive, by running the ML algorithm over the feature-extended dataset.

For SGP, we use subtree crossover and subtree mutation, picking the depth of subtree roots uniformly randomly as proposed in [29]. The candidate parents for variation are chosen with tournament selection. Since we are interested in constructing small features so as to increase the chances they will be interpretable, we consider two versions of SGP. The first is the classic one where solutions are free to grow to tree heights typically much larger than the one used for tree initialization. In the following, the notation SGP refers to

this first version. The second one uses trees that are not allowed to grow past the initial maximum tree height. We call this version *bounded* SGP, and use the notation SGP_b.

RS is realized by continuously sampling and evaluating new trees, keeping the best [3]. Like for SGP_b, a maximum tree height is fixed during the whole run. If evolution is hypothetically no better than RS, then we expect that SGPb and GP-GOMEA will construct features that are no better than the ones constructed by RS.

GP-GOMEA is a recently introduced GP algorithm that has been found to deliver accurate solutions of small size on benchmark problems (Chapter 2), and to work well when a small size is enforced in symbolic regression (Chapter 3 and [24]). GP-GOMEA uses a tree template fixed by a maximum tree height (which can include intron nodes to allow for unbalanced tree shapes) and performs homologous variation, i.e., mixed tree nodes come from the same positions in the tree. Each generation prior to mixing, a hierarchical model that captures interdependencies (*linkage*) between nodes is built (using mutual information). This model, called Linkage Tree (LT), drives variation by indicating what nodes should be changed *en block* during mixing, to avoid the disruption of patterns with large linkage.

The LT has been shown to enable GP-GOMEA to outperform subtree crossover and subtree mutation of SGP, as well as the use of a randomly-build LT, i.e., the Random Tree (RT), on problems of different nature (Chapters 2 and 3). However, the LT requires sufficiently large population sizes to be accurate and beneficial (e.g., several thousand solutions in GP for symbolic regression), as discussed in Chapter 3. Because in the framework of this chapter fitness evaluations use the cross-validation of a ML algorithm, we cannot afford to use large population sizes. Accordingly, we found the adoption of the LT to not be superior to the adoption of the RT under these circumstances in preliminary experiments. Therefore, for the most part, we adopt GP-GOMEA with the RT ($GP\text{-}GOMEA_{RT}$). This means we effectively compare random hierarchical homologous variation with subtree-based variation. An example of adopting the LT and large population sizes for feature construction is provided in Section 5.10.

5.4.2. Details on the ML algorithms

We now briefly describe the ML algorithms used in this work: NB, LR, SVM, RF, and XGB. NB and LR are less computationally expensive compared to SVM, RF, and XGB. Details on the computational time complexity of these algorithms are reported at: `https://bit.ly/2PG0xse`.

NB is a classifier which assumes independence between features [20, 21]. NB is often used as a baseline, as it is simple and fast to train. We use the *mlpack* implementation of NB [30] and assume the data to be normally distributed (default setting).

Similarly to NB, LR is often used as a baseline as it is simple and fast, for regression tasks. LR assumes that the target variable can be explained by a linear combination of the features [20]. We use the *mlpack* implementation of LR [30].

SVM is a powerful ML algorithm that can be used for non-linear classification and regression [26, 31]. We use the *libsvm* C++ implementation [31]. We consider the Radial Basis Function (RBF) kernel, which works well in practice for many problems, with C-SVM for classification, and $\mathcal{E}$-SVM for regression.

RF is an ensemble ML algorithm which, like SVM, can be used for both classification

Table 5.1: Parameter settings of the GP algorithms.

	SGP(b)	GP-GOMEA$_{\text{RT}}$
Population size	100	100
Initialization method	Ramped Half and Half	Half and Half
Initialization max tree height	2–6 (2 or 4)	2 or 4
Max tree height	17 (2 or 4)	2 or 4
Variation	SX 0.9, SM 0.1	Parameter-less
Selection	Tournament 7, Elitism 1	Parameter-less
Function set	$\{+, \times, -, \div, \cdot^2, \sqrt{\cdot}, \log_p, \exp\}$ for all	
Terminal set	$\{x^{(i)}, \texttt{ERC}\}$ for all	

and regression and can infer non-linear patterns [27]. RF builds an ensemble of (typically deep) decision trees, each trained on a sample of the training set (*bagging*). At prediction time, the mean (or maximum agreement) prediction of the decision trees is returned. We use the *ranger* C++ implementation [32].

XGB is, like RF, an ensemble ML algorithm, typically based on decision trees, and capable of learning non-linear models [33]. XGB works by boosting, i.e., stacking together multiple weak estimators (small decision tress) that fit the data in an incremental fashion. We use the *dmlc* C++ implementation (`https://bit.ly/34fBNeA`).

5.5. Experiments

We perform 30 runs of our Feature Construction Scheme (FCS), with SGP, SGP$_{\text{b}}$, RS, and GP-GOMEA$_{\text{RT}}$, in combination with each ML algorithm (NB only for classification and LR only for regression), on each problem. Each run of the FCS uses a random train-test split of 80%-20%, and considers up to $K = 5$ features construction rounds. We use a population size of 100 for the search algorithms, and assign a maximum budget of $10,000$ function evaluations to each FCS iteration. This results in relatively large running times for complex ML algorithms (see Sec. 5.9). An experiment including larger evolutionary budgets and the use of the LT in GP-GOMEA is presented in the discussion (Sec. 5.10). We use a limit on the total number of evaluations instead of a a limit on the total number of generations because GP-GOMEA$_{\text{RT}}$ performs more evaluations than SGP per generation (see, e.g., Sec. 3.8.1).

For GP-GOMEA$_{\text{RT}}$, SGP$_{\text{b}}$, and RS, we consider two levels of maximum tree height h: 2 and 4. This choice yields a maximum solution size of 7 and 31 respectively (using function nodes with a maximum arity $r = 2$). We choose these two height levels because we found features with $h = 2$ to be arguably easy to read and interpret, whereas features with $h = 4$ can already be very hard to understand. This indication is also reported in Chapter 3 for the evolution of symbolic regression formulas. Note that using a tree height limit over a solution size limit prevents finding deep trees containing the nesting of the arguably more complicated to understand non-linear functions $\cdot^2, \sqrt{\cdot}, \log_p, \exp$. We do not consider bigger tree heights as resulting features may likely be impossible to interpret, defying a key focus of this work.

Other parameter settings used for the GP algorithms are shown in Table 5.1. SGP_b uses the same settings as SGP, except for the maximum tree height (at initialization and along the whole run), which is set to the same of GP-GOMEA$_{RT}$. In GP-GOMEA$_{RT}$ we use the Half and Half (HH) tree initialization method instead of the Ramped Half and Half (RHH) [3] commonly used for SGP. This proved to be beneficial since GOM varies nodes instead of subtrees [11, 24]. For both HH and RHH, syntactical uniqueness of solutions is enforced for up to 100 tries [3]. In GP-GOMEA$_{RT}$ we additionally avoid sampling trees having a terminal node as root by setting the minimum tree height of the grow method to 1. This is not done for SGP and SGP_b, because differently from GP-GOMEA$_{RT}$ where homologous nodes are varied, subtree root nodes for subtree crossover (SX) and subtree mutation (SM) are chosen uniformly randomly. RS samples new trees using the same initialization method as SGP_b, i.e., RHH.

The division operator $\div$ used in the function set is the analytic quotient operator ($a \div b = a/\sqrt{1+b^2}$), which was shown to lead to better generalization performance than protected division [34]. The logarithm is protected $\log_p(\cdot) = \log(|\cdot|)$ and $\log_p(0) = 0$, and so is the square root operator. The terminal set contains the original feature set, and an Ephemeral Random Constant (ERC) [4] with values uniformly sampled between the minimum and maximum values of the features in the original training set, i.e., $[\min x_j^{(i)}, \max x_j^{(i)}], \forall i, j \in Tr$.

The hyperparameter settings for the SVM, RF and XGB are shown in Table 5.2, and are mostly default [27, 31, 32] (for XGB, we referred to `https://bit.ly/2JCM9x4`). NB and LR implementations do not have hyperparameters.

We consider 10 classification and 10 regression benchmark datasets[1] that can be considered traditional, i.e, they have small to moderate dimensionality (number of features). We mostly study this type of dataset because we seek to find small constructed features that can be interpreted. Hence, they can represent a transformation of only a limited number of original features. Details on the datasets are reported in Table 5.3. Rows with missing values are omitted. Most datasets are taken from the UCI Machine Learning repository[2], with exception for Dow Chemical and Tower, which come from GP literature [35, 36].

We further consider a very high-dimensional dataset from UCI[3] to assess whether GP can still be useful to construct features in this type of scenario. The dataset in question concerns the classification of cancer type, given RNA-Seq gene expression levels as features. Five cancer class types are present, and class proportions in the data presents some unbalance: the class frequencies are 0.37, 0.18, 0.18, 0.17, 0.10. A total of 20,531 features are considered, in 801 examples. Since large computational resources are needed to handle this dataset, we consider only NB as ML algorithm for feature construction upon this data.

[1] The datasets are available at `http://goo.gl/9D2z3b`
[2] `http://archive.ics.uci.edu/ml`
[3] `https://bit.ly/334KbgW`

Table 5.2: Salient hyper-parameter settings of SVM, RF, and XGB.

SVM	
Kernel	RBF
Cost	1
Epsilon	0.1
Tolerance	0.001
Gamma	$\frac{1}{k}$
Shrinking	Active
RF	
Number of trees	100
Bagging sampling	with replacement
Classification mtry	$\sqrt{\#\text{features}}$
Regression mtry	$\min(1, \frac{\#\text{features}}{3})$
Min node size	1 classification, 5 regression
Split rule	Gini classification, Variance regression
XGB	
Number of trees	100
Booster	gbtree
Max depth	6
Objective	multiclass softmax, MSE regression
Learning rate	0.3

5.6. Results on traditional datasets

The results of this section aim at assessing whether it is possible to construct few and small features that lead to an equal or better performance than the original set, and whether some search algorithms construct better features than others, for the traditional datasets.

5.6.1. General performance of feature construction

We begin by observing the dataset-wise aggregated performance of FCS for the different GP algorithms and the different ML algorithms, separately for classification and regression.

Classification

Figure 5.3 shows dataset-wise aggregated results obtained for NB, SVM, RF, and XGB, for the 10 traditional classification tasks. Each data point is the mean among the dataset-specific medians of macro F1 from the 30 runs.

In general, the use of only one constructed feature does not perform as good as the use of the original feature set. Constructing more features improves the performance, but with diminishing returns.

Specifically for NB, the use of two constructed features is already preferable to the use of the original feature set. This is likely due to the fact that NB assumes complete independence between the provided features, and this can be implicitly tackled by FCS. SGP (unbounded) is the best performing algorithm as it can evolve arbitrarily complex

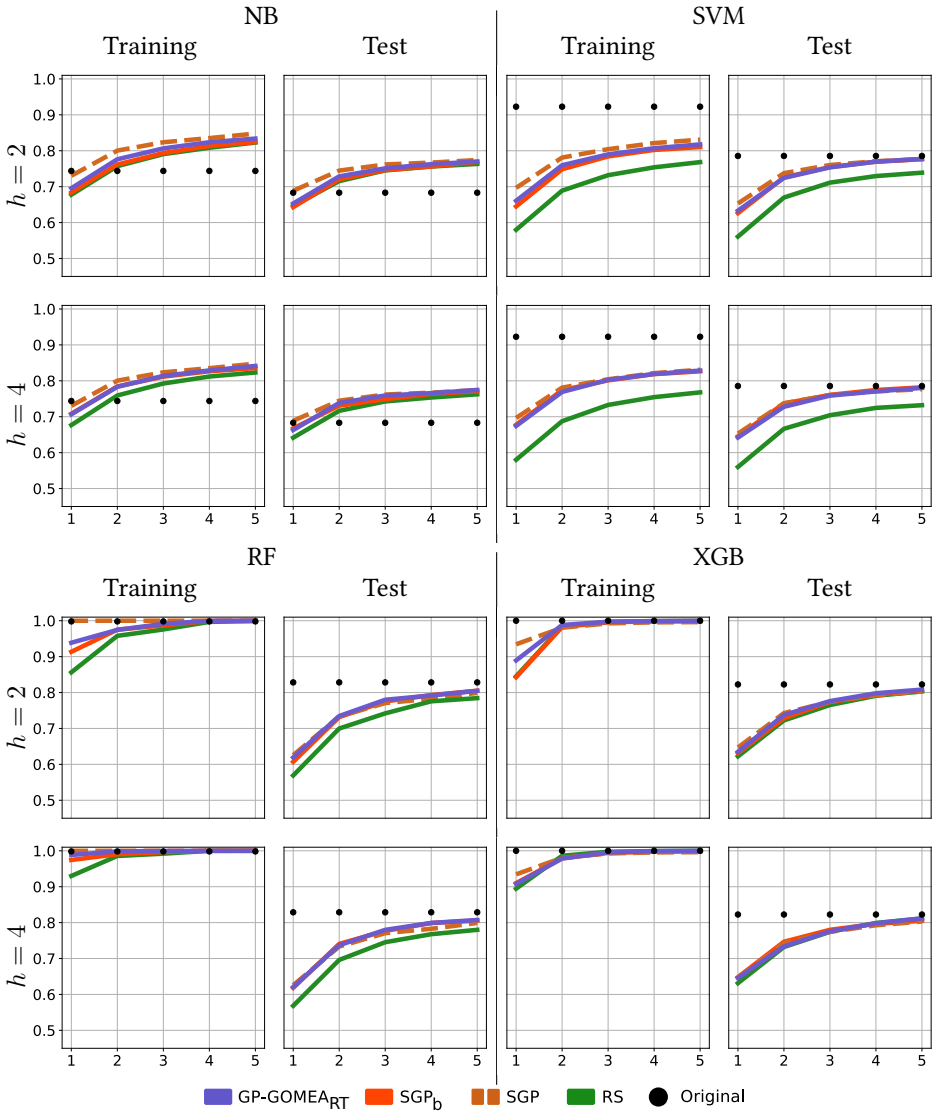

Figure 5.3: Aggregated results on the classification datasets. Horizontal axis: Number of features. Vertical axis: Average of median $F1$ score obtained on 30 runs for each dataset.

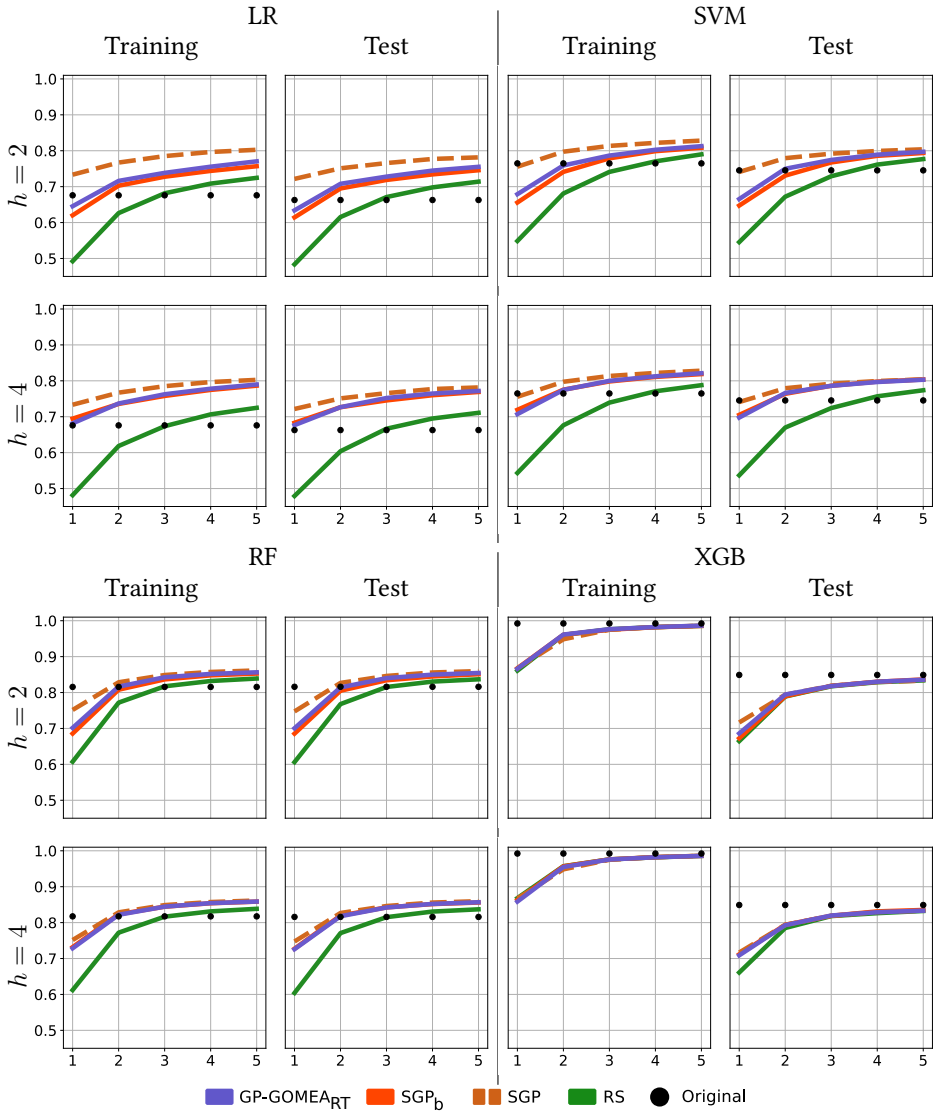

Figure 5.4: Aggregated results on the regression datasets. Horizontal axis: Number of features. Vertical axis: Average of median R^2 score obtained on 30 runs for each dataset.

Table 5.3: Traditional classification and regression datasets.

	Dataset	# Features	# Examples	# Classes
Classification	Cylinder Bands	39	277	2
	Breast Cancer Wisc.	29	569	2
	Ecoli	7	336	8
	Ionosphere	34	351	2
	Iris	4	150	3
	Madelon	500	2600	2
	Image Segmentation	19	2310	7
	Sonar	60	208	2
	Vowel	9	990	11
	Yeast	8	1484	10
Regression	Airfoil	6	1503	–
	Boston Housing	13	506	–
	Concrete	9	1030	–
	Dow Chemical	57	1066	–
	Energy Cooling	9	768	–
	Energy Heating	9	768	–
	Tower	26	4999	–
	Wine Red	12	1599	–
	Wine White	12	4898	–
	Yacht	7	308	–

features, however, the magnitude of improvement of the macro F1 score with respect to GP-GOMEA$_{\text{RT}}$ and SGP$_{\text{b}}$ is limited. For $h = 4$ and $K = 5$, GP-GOMEA$_{\text{RT}}$ reaches the performance of SGP. GP-GOMEA$_{\text{RT}}$ is typically slightly better than SGP$_{\text{b}}$, and RS has worse performance. Training and test F1 scores do not differ much for any feature construction algorithm, meaning that overfitting is not an issue for NB. Rather, compared to the other ML algorithms, NB underfits.

The performance of FCS for SVM has an almost identical pattern to the one observed for NB, except for the fact that the performance is found to be consistently better. However, for SVM it is preferable to use the original feature set rather than few constructed features. This is evident in terms of training performance, but less at test time. In fact, using only 5 constructed features leads to similar test performance compared to using the original set. The GP algorithms compare to each other similarly to when using NB. Compared to NB, it can be seen that SVM exhibits larger gaps between training and test results, suggesting that some overfitting takes place, especially when the original feature set is used.

The way performance improves for RF by constructing features is similar to the one observed for NB and SVM. However, for RF the differences between the search algorithms is particularly small: notice that using RS leads to close performance to the ones obtained by using the other GP algorithms, compared to the SVM case. Moreover, virtually no difference can be seen between GP-GOMEA$_{\text{RT}}$ and SGP$_{\text{b}}$. This suggests that RF already works well with less refined features. Now, the features constructed by SGP are no longer the best performing at test time. This is likely because SGP evolves larger, more complex features than the other algorithms (see Sec. 5.6.1), making RF overfit. In fact, RF exhibits

the largest difference between training and test results compared to NB and SVM, for any feature construction algorithm and h limit. Still, the test results of RF are slightly better than the ones of SVM and markedly better than the ones of NB, meaning that the latter two are underfitting.

The training and test performance obtained when using XGB is similar to the one obtained when using RF, but the differences the between different search algorithms are even less marked than for RF. Some differences can be seen for $K = 1$ on the training set (SGP better than GP-GOMEA$_{\text{RT}}$, and GP-GOMEA$_{\text{RT}}$ better than SGPb and RS), but this difference is much less marked on the test set. When more features are constructed, essentially all search algorithms deliver the same performance. XGB seems to be able to construct non-linear relationships even better than RF. As to potential overfitting, the trend of differences between training and test performance that can be observed for XGB mirrors the one visible for RF.

As to maximum tree height, allowing the constructed features to be bigger ($h = 4$ vs $h = 2$) moderately improves the performance. Interestingly, GP-GOMEA$_{\text{RT}}$ with $h = 4$ reaches competitive performance with SGP on all ML algorithms, despite the latter having no strict limitation on feature size.

Regression

Results on the regression tasks are shown in Figure 5.4, dataset-wise aggregated for LR, SVM, RF, and XGB. We report the results in terms of coefficient of determination, i.e., $R^2(y, \hat{y}) = 1 - MSE(y, \hat{y})/var(y)$. For the four ML algorithms, results overall follow the same pattern. SGP is typically better, especially for LR and SVM, although constructing more features reduces the performance gap with the other GP algorithms. GP-GOMEA$_{\text{RT}}$ is slightly, yet consistently, the best performing within the maximum tree height limitation of 2, while SGP$_{\text{b}}$ is visibly preferable only when a single feature is constructed for LR and SVM, for $h = 4$. Differently from the classification case, two features are typically enough to reach the performance of the original feature set for all ML algorithms except for XGB. Moreover, for LR, SVM, and RF, the performance between training and test is similar, meaning no considerable overfitting is taking place, no matter the feature construction algorithm used nor the limit of h. This however is not the case for XGB, where a large performance gap is encountered. Still, the test performance obtained when using XGB is ultimately slightly better than the obtained for RF.

As for classification, allowing for larger trees results in better performance overall, and reduces the gap between SGP and the other GP algorithms. With XGB, all search algorithms perform similarly.

Feature size

Figure 5.5 shows the aggregated feature size for the different GP algorithms and RS. The aggregated solution size is computed by taking the median solution size per run, then averaging over datasets, and finally averaging over ML algorithms (classification and regression are considered together). The picture shows how, overall, the known SGP tendency to bloat differs compared to the algorithms working with a strict tree height limitation. SGP features are so large that it is nearly impossible to interpret them (see Sec. 5.8.1).

RS finds the smallest features for both height limits $h = 2$ and $h = 4$. Considering that GP-GOMEA$_{\text{RT}}$ and SGP$_{\text{b}}$ generate trees within the same height bounds of RS, we

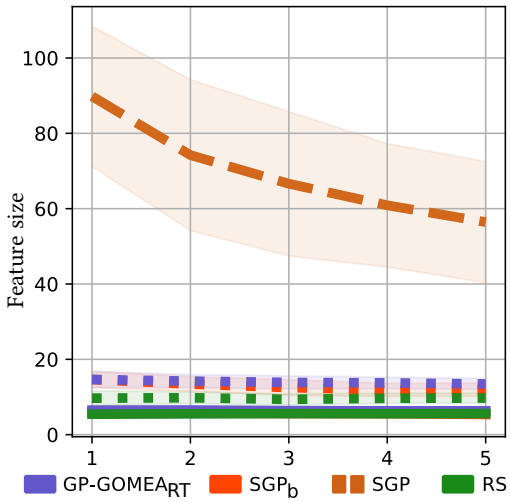

Figure 5.5: Aggregated feature size for $k = 1, \ldots, 5$. Solid (dotted) lines represent solution size for maximum tree height $h = 2$ ($h = 4$). Shaded areas represent standard deviation. SGP is free to grow solutions up to $h = 17$.

conclude that it is the variation operators that allow finding larger trees with improved fitness within the height limit. GP-GOMEA$_{RT}$ seems to construct slightly, yet consistently, larger trees than SGP$_b$.

For SGP, it can be seen that subsequently constructed features are smaller (this is barely visible for GP-GOMEA$_{RT}$ and SGP$_b$ as well). This is interesting because we do not use any mechanism to promote smaller trees. This result is likely linked to the diminishing returns in performance observed in Figure 5.3 and 5.4: constructing new complex and informative features becomes harder with the number of FCS iterations.

5.6.2. Statistical significance: comparing GP algorithms

The aggregated results of Section 5.6.1 show moderate differences between GP-GOMEA$_{RT}$ and SGP$_b$. These are arguably the most interesting algorithms to compare in-depth, as they are able to construct small features that lead to good performance (RS typically constructs less informative features, while SGP constructs very large ones).

We perform statistical significance tests to compare GP-GOMEA$_{RT}$ and SGP$_b$. We consider their median performance on the test set Te, obtained by the FCS, and also compare it with the use of the original feature set, for each ML algorithm and each dataset. In our case, the *treatments* of our significance tests are the two search algorithms (i.e., GP-GOMEA$_{RT}$ and SGP$_b$) and the original feature set, while the *subjects* are the configurations given by pairing ML algorithms and datasets [37].

We first perform a Friedman test to assess whether differences exists among the use of different treatments (GP algorithms and original feature set) upon multiple subjects (ML algorithm-dataset combinations). As post-hoc analysis, we use the pairwise Wilcoxon signed rank tests, paired by subject (ML algorithm-dataset combination), to see how the

treatments compare to each other [37]. We adopt the Holm correction method to prevent reporting false positive results that might have happened due to pure chance [38]. We consider both $h = 2$ and $h = 4$, and focus on $K = 2$, since consideration of only two constructed features makes interpretation easier, and allows human visualization (see Sec. 5.8.1).

Classification

For both $h = 2, 4$, the Friedman test strongly indicates differences between GP-GOMEA$_{\text{RT}}$, SGP$_{\text{b}}$, and the use of the original feature set (p-value $\ll 0.05$).

Figure 5.6 (top) shows the Holm-corrected p-values obtained by the pairwise Wilcoxon tests for classification, where the alternative hypothesis is that the row allows for larger macro F1 scores than the column. No significant differences between GP-GOMEA$_{\text{RT}}$ and SGP$_{\text{b}}$ are found for both $h = 2, 4$. Both the GP algorithms can deliver constructed features that are competitive with the use of the original feature set. The original feature set is not significantly better than using feature construction. Moreover, for GP-GOMEA$_{\text{RT}}$ and $h = 4$, the hypothesis that feature construction is *not* better than the original feature set can be rejected with a corrected p-value below 0.1. The latter result appears to be in contrast with the results from Figure 5.3 for SVM, RF and XGB, where it can be seen that the construction of only two features does, on average, lead to slightly worse test results than using the original feature set. Nonetheless, the opposite is true for NB, and with rather large magnitude. A more in-depth analysis on this is provided in Section 5.6.3.

Regression

As for classification datasets, the Friedman test indicates that differences are presents between the treatments. Figure 5.6 (bottom) shows the Holm-corrected p-values obtained by the pairwise Wilcoxon tests for regression.

The statistical tests confirm the hypothesis that the algorithms are capable of providing constructed features that are more informative than the original feature set, as observed in Figure 5.4 for the regression datasets. Now, GP-GOMEA$_{\text{RT}}$ is significantly better than SGP$_{\text{b}}$ when $h = 2$. For $h = 4$, instead, GP-GOMEA$_{\text{RT}}$ is not found to be significantly better than SGP$_{\text{b}}$.

5.6.3. Statistical significance: two constructed features vs. the original feature set per ML algorithm

Results presented in Section 5.6.1 indicate that our FCS brings most benefit if used with the weak ML algorithms. We now report, for each ML algorithm, on how many datasets 2 features constructed using GP-GOMEA (with $h = 2$ and $h = 4$) lead to statistically significantly (using Holm-corrected pairwise Wilcoxon test, p-value < 0.05) better, equal, or worse results compared to using the original feature set on the test set. This is shown in Table 5.4.

These results confirm what seen in Figures 5.3 and 5.4. Using FCS typically outperforms the use of the original feature set for the weak ML algorithms. For the strong ML algorithms, in most cases, using the original feature set is preferable. However, for some datasets reducing the space to two compact features without compromising performance is still possible.

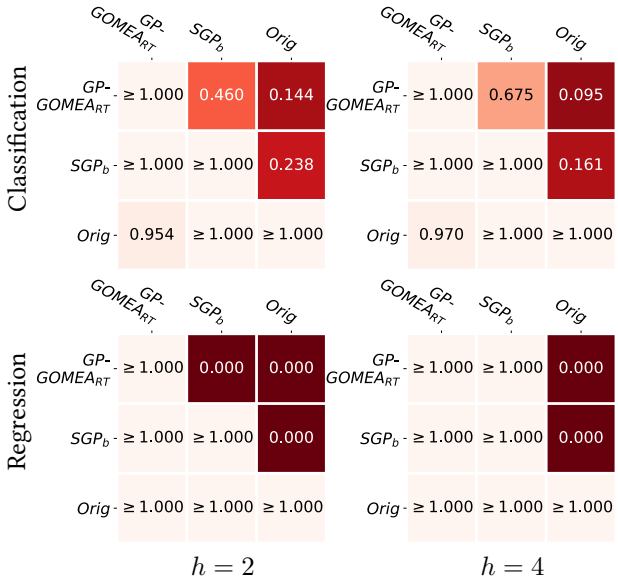

Figure 5.6: Holm-corrected p-values of pairwise Wilcoxon tests on test performance. Rows are tested to be significantly better than columns. *Orig* stands for the original feature set.

The use of the original feature set is generally hardest to beat when adopting RF or XGB. For RF, in the regression case with $h = 4$, FCS brings benefits on the datasets Airfoil, Energy Cooling, Energy Heating, and Yacht; and performs on par with the use of the original feature set on the datasets Boston Housing and Concrete. These datasets are the ones with the smallest number of original features. We find similar results for SVM and for XGB. In the latter case, FCS is, in terms of statistical significance, equal to the original feature set only on Energy Cooling, Energy Heating, and Yacht. It is reasonable to expect that FCS works well when few features can be combined.

In the classification case, findings are different. For RF and $h = 4$, the datasets where using two constructed features bring similar or better results than using the original feature set are Breast Cancer Wisconsin and Iris. The latter does have a small number of original features (4), but the former has more than several other datasets (29). Furthermore, the datasets where FCS helps are different for SVM: FCS performs equally good to the original feature set on Iris and Cylinder Bands (39 features), and better on Madelon (500 features) and Image Segmentation (19 features). Regarding XGB, there is no dataset where FCS is superior to the original feature set, but it is also not worse on almost half of the datasets. For classification datasets, we cannot conclude that a small cardinality of the original feature set is a good indication feature construction will work well. Furthermore, feature construction influences different ML algorithms in different ways.

Table 5.4: Number of datasets where using two features constructed with GP-GOMEA results in significantly better/equal/worse test performance compared to using the original feature set.

	h	NB	SVM	RF	XGB
Clas.	2	8/1/1	2/2/6	1/1/8	0/4/6
	4	8/1/1	2/2/6	1/1/8	0/4/6
	h	LR	SVM	RF	XGB
Regr.	2	5/3/2	5/2/3	4/0/6	0/2/8
	4	7/1/2	5/2/3	4/2/4	0/3/7

5.7. Results on a highly-dimensional dataset

We further consider the RNA-Seq cancer gene expression dataset, comparing FCS by GP-GOMEA$_{\text{RT}}$ with $h = 4$ against the use of the original feature set, when using NB. Figure 5.7 shows that NB with the original feature set overfits: the training performance is maximal, while the test performance reaches an F1 of approximately 0.65. Even though NB is typically considered a weak estimator, the system described by the data is so severely underdetermined (over $20,000$ features vs less than $1,000$ examples) that actual patterns cannot be retrieved. The use of FCS forces NB to use only a small number of constructed features, which, in turn, can contain only a small number of the original features. Essentially, FCS provides both the advantages of feature construction and feature selection. This leads to large F1 scores already when solely two features are constructed.

5.8. Results on interpretability

The results presented in Section 5.6 and 5.7 showed that the original feature set can be already outperformed by two small constructed features in many cases. We now aim at assessing whether constraining features size can enable interpretability of the features themselves, as well as if extra insight can be achieved by plotting and visualizing the behavior of a trained ML model in the new two-dimensional space.

5.8.1. Interpretability of small features

Table 5.5 shows some examples of features constructed by GP-GOMEA$_{\text{RT}}$, for $h = 2$ and $h = 4$. We report the first feature constructed for the $K = 2$ case, with median test performance. We show the first feature as it is typically not smaller than the second (see Fig. 5.5). Analytic quotients and protected logarithms are replaced by their respective definitions. We remark that we do not check whether the meaning of the features is sound (e.g., ensuring a certain unit of measure is returned). Constraining feature meaning is problem-dependent, and outside the scope of this work.

For classification, we choose NB as it is the method which benefits most from feature construction. The dataset considered is Ecoli, where NB achieves the largest median test improvement when $K = 2$: from $F1 = 0.51$ with the original set, to $F1 = 0.63$ for $h = 2$, and to $F1 = 0.66$ for $h = 4$.

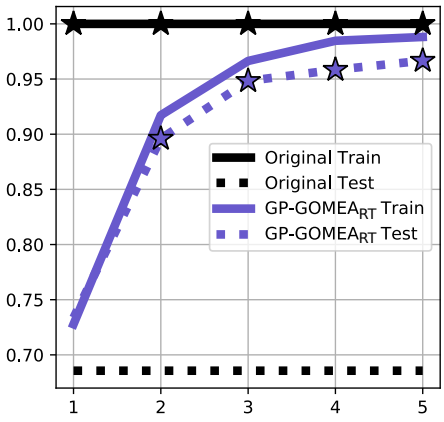

Figure 5.7: Comparison between the use of the original feature set and FCS with GP-GOMEA$_{RT}$ ($h = 4$) on high-dimensional gene expression data. The vertical axis reports the median F1 score, the horizontal axis reports the number of features constructed by FCS. Stars indicate statistical significant superiority (p-value < 0.05) of one method with respect to the other.

For regression, we consider LR on the Concrete dataset, for the same aforementioned reasons. The test R^2 obtained with the original feature set is 0.59, the one with two features constructed by GP-GOMEA$_{RT}$ is 0.76 (0.78) for $h = 2$ ($h = 4$).

For $h = 2$, we argue that constructed features are mostly easy to interpret. For example, the feature shown for LR on Concrete tells us that aging ($x^{(8)}$) has a negative impact on concrete compressive strength, whereas using more water ($x^{(4)}$) than cement ($x^{(1)}$) has a positive effect (both features are in kg/cm^3). The impact of other features is less important (within the data variability of the dataset). For $h = 4$, some features can be harder to read and understand, however many are still accessible. This is mostly because, even though the total solution size reachable with $h = 4$ is 31, constructed features are typically half the size (see Fig. 5.5).

The features constructed for the RNA-Seq gene-expression dataset by GP-GOMEA$_{RT}$ in Section 5.7 are also not excessively complex to be understood. For example, the first two features for the median run are:

$$\begin{aligned}
&\text{1st}: \sqrt{\left(x^{(18382)}\right)^2 + x^{(8014)} + x^{(3885)} + x^{(17316)}} \\
&\text{2nd}: \left(x^{(7491)} + \sqrt{x^{(7296)}} + x^{(19333)}\right) \times \\
&\left(\frac{x^{(5524)} + x^{(18053)}}{\sqrt{1 + \left(x^{(5579)} - x^{(4417)}\right)^2}} + x^{(14153)} + x^{(19751)} - \frac{x^{(13744)}}{\sqrt{1 + \left(x^{(16581)}\right)^2}}\right).
\end{aligned} \tag{5.3}$$

Even though the second feature is somewhat involved, it is arguably still possible to carefully analyze it and obtain a picture of how gene expression levels interact.

Overall, we cannot draw a strict conclusion on whether the features found by our approach are interpretable, as interpretability is a subjective matter and, to date, no clear-

Table 5.5: Examples of features constructed by GP-GOMEA$_{\text{RT}}$ with $h \in \{2, 4\}$, $K = 2$, for NB on Ecoli, and for LR on Concrete.

	h	1st Feature
NB	2	$x^{(3)} + x^{(6)} + x^{(1)} / \sqrt{1 + (x^{(6)})^2}$
	4	$x^{(6)} \left(x^{(7)}\right)^2 x^{(3)} + 0.144 / \sqrt{1 + (\exp(x^{(2)}))^2} - x^{(1)} x^{(2)} x^{(5)}$
LR	2	$x^{(4)} - x^{(1)} + 932.204 / \sqrt{1 + (x^{(8)})^2}$
	4	$\sqrt{19.764 \log \lvert x^{(8)} \rvert} + x^{(2)} + 2x^{(1)} / \sqrt{1 + (x^{(4)})^2}$

$$
\Bigg(\log_p(((((((x^{(4)} + x^{(2)}) \div x^{(4)}) \times (x^{(1)} \div x^{(4)} \div x^{(4)} \times (x^{(1)} \div x^{(4)} x^{(8)} \times 1065.162 \times \\
\log_p((((x^{(4)} x^{(1)} + x^{(1)} \div x^{(8)} \div x^{(8)}) \times (x^{(4)} - (x^{(1)} + ((x^{(2)} \div x^{(4)} + \log_p(x^{(1)})))^2))) + \\
- (x^{(5)} x^{(1)} + x^{(2)} \div x^{(8)} + (((x^{(8)} + x^{(6)} + x^{(2)}) \div ((x^{(1)} \div x^{(4)}) + \sqrt{x^{(4)}} \div x^{(2)})) + \\
\exp((((x^{(8)} + x^{(2)}) \div x^{(8)} + (x^{(4)} + x^{(2)}) \div x^{(4)}) \div x^{(5)} + x^{(6)} \div x^{(4)}))))))))) \div x^{(7)}) \times \\
(x^{(8)} - (441.237 + x^{(2)}))) - x^{(1)}))\Bigg)^{\frac{1}{2}}
$$

Figure 5.8: Example of a relatively "small" feature constructed by SGP, derived from a tree with 96 nodes. Note that the analytic quotient operator ($\div$) and the protected logarithm ($\log_p$) are *not* expanded to their respective definitions to keep the feature contained. This feature is arguably very hard to interpret.

cut metric exists [7, 8] (more on this in Sec. 5.10). Yet, it appears evident that enforcing a restriction on their size is a necessary condition. We generally find that features using 15 or more nodes start to be hard to interpret with respect to our experimental settings, i.e., using our function set. Lastly, features constructed without a strict size limitation (by SGP) are generally very large, and thus extremely hard to understand. For example, Figure 5.8 shows the first of the two features with median test performance constructed by SGP for LR on Concrete (this is smaller than the first feature found by SGP for NB on Ecoli).

5.8.2. Visualizing what the ML algorithm learns

The construction of a small number of interpretable features can enable a better understanding of the problem and of the learned ML models. The case where up to two features are constructed is particularly interesting, since it allows visualization.

We provide one example of classification boundaries and one of a regressed surface, inferred by SVM on a two dimensional feature space obtained with our approach using GP-GOMEA$_{\text{RT}}$. The classification dataset on which we find the best test improvement for $h = 4$ is Image Segmentation, where the $F1$ score of SVM reaches 0.88, against 0.65 using the original feature set (median run). Figure 5.9 shows the classification boundaries learned by SVM. The analytic quotient operator $\div$ and the protected log $\log_p$ are replaced by their definition for readability. The constructed features are rather complex here, yet

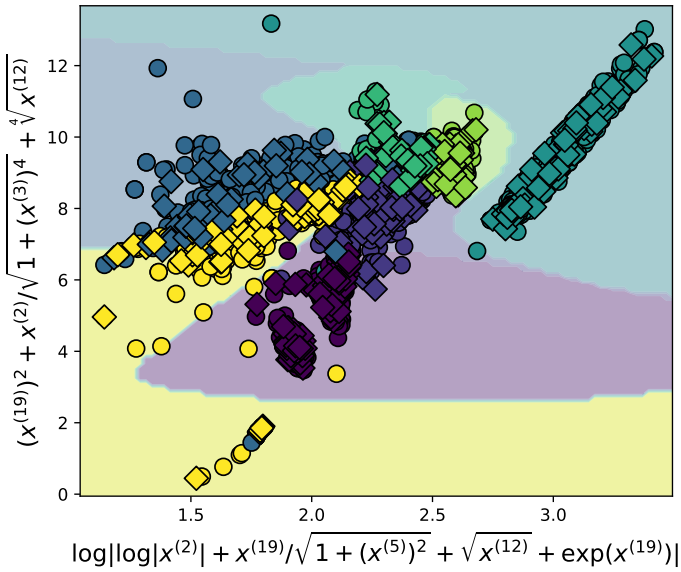

Figure 5.9: Classification boundaries learned by SVM with two features constructed by GP-GOMEA$_{\text{RT}}$ ($h = 4$) on the Image Segmentation dataset. The run with median test performance is shown. Circles are training samples, diamonds are test samples.

readable. At the same time, it can be clearly seen how the training and test examples are distributed in the 2D space, and what classification boundaries SVM learned.

For regression, Figure 5.1 shows the surface learned by SVM on Yacht (median run), where GP-GOMEA$_{\text{RT}}$ with $h = 2$ constructs two features that lead to an R^2 of 0.98, against 0.85 obtained using the original feature set. The features are arguably easy to interpret, while it can be seen that the learned surface accurately models most of the data points.

5.9. Running time

Our results are made possible by evaluating the fitness of constructed features with cross-validation, a procedure which is particularly expensive. Table 5.6 shows the (mean over 30 runs) serial running time to construct five features on the smallest and largest classification and regression datasets, using GP-GOMEA$_{\text{RT}}$ with $h = 4$ and the parameter settings of Section 5.5, on the relatively old AMD Opteron™ Processor 6386 SE[4]. Running time has a large variability, from seconds to dozens of hours, depending on dataset size and ML algorithm. For the traditional datasets and ML algorithms we considered, it can be argued that our approach can be used in practice. However, for very high-dimensional datasets, only fast ML algorithms can be used. The construction of 5 features for the RNA-Seq gene expression dataset took 25 minutes even though NB was used. To use slower ML algorithms would easily require dozens to hundreds of hours.

[4] http://cpuboss.com/cpu/AMD-Opteron-6386-SE

As to memory occupation, it basically mostly depends on the way the chosen ML algorithm handles the dataset. Our runs required at most few hundreds of MBs when dealing with the larger traditional datasets, for SVM and RF. Handling the parallel execution of FCS experiments upon the gene expression dataset required a few GBs.

Table 5.6: Mean serial running time to construct five features using GP-GOMEA$_{\text{RT}}$ ($h = 4$) on the smallest and largest traditional datasets.

	Dataset	Size	NB/LR	SVM	RF	XGB
Clas.	Iris	150×4	7 s	2 m	25 m	42 m
	Madelon	2600×500	4 m	14 h	8 h	10 h
Regr.	Yacht	308×7	8 s	4 m	1 h	1 h
	Tower	4999×26	2 m	34 h	34 h	13 h

5.10. Discussion

We believe this is one of the few works on evolutionary feature construction where the focus is put on both improving the performance of an ML algorithm, and on human interpretability at the same time. The interpretability we aimed for is twofold: understanding the meaning of the features themselves, as well as reducing their number. GP algorithms are key, as they can provide constructed features as interpretable expressions given basic functional components, and a complexity limit (e.g., tree height).

We have run a large set of experiments, totaling more than 150,000 cpu-hours. Our results strongly support the hypothesis that the original feature set can be replaced by few (even solely $K = 2$) features built with our FCS without compromising performance in many cases. In some cases, performance even improved. GP-GOMEA$_{\text{RT}}$ and SGP$_{\text{b}}$ achieve this result while keeping the constructed feature size extremely limited ($h = 2, 4$). SGP leads to slightly better performance than GP-GOMEA$_{\text{RT}}$ and SGP$_{\text{b}}$, but at the cost of constructing five to ten times larger features. RS proved to be less effective than the GP algorithms.

Our FCS is arguably most sensible to use for simpler ML algorithms, such as NB and LR. Constructed features change the space upon which the ML algorithm operates. SVM already includes the kernel trick to change the feature space. Similarly, the trees of RF and XGB effectively embody complex non-linear feature combinations to explain the variance in the data. NB and LR, instead, do not include such mechanisms. Rather, they have particular assumptions on how the features should be combined (NB assumes normality, LR linearity). The features constructed by GP can transform the input the ML algorithm operates upon, to better fit its assumptions.

We found that performance was almost always significantly better than compared to using the original feature set for NB and LR. As running times for these ML algorithms can be in the order of seconds or minutes (Sec. 5.9), feature construction has the potential to be routinely used in data analysis and machine learning practice. Furthermore, FCS (or a modification where the constructed features are added to the original set) can be used as an alternative way to tune simple ML algorithms which have limited or no hyper-parameters.

We have shown that our approach can also be helpful when dealing with high-dimensional data (on the RNA-Seq gene expression dataset), where system underdetermination can cause even simpler ML algorithms to overfit. This is because FCS essentially embodies feature selection, as we only construct a small number of small-sized features.

We remark that we did not adopt high-dimensional datasets concerning image recognition such as MNIST [39], CIFAR [40], or ImageNet [41]. In these datasets, features represent pixels, which have no particular meaning. Constructing features as readable pixel transformations will likely provide no insights on the behavior of a ML model.

Regarding the comparison between the search algorithms, GP-GOMEA$_{\text{RT}}$ was found to be slightly preferable to SGP$_{\text{b}}$ (especially for $h = 2, K = 2$). We believe that significantly better results can be achieved if bigger population sizes and larger evaluations budgets can be employed (we kept the population size limited due to the computational expensiveness of SVM and RF).

Particularly for GP-GOMEA, in Chapter 3 we discussed that having sufficiently large population sizes enables the possibility to exploit linkage estimation and perform better-than-random mixing. To validate this also within the framework of our proposed FCS, we scaled the population size and the budget of fitness evaluations, and compared the use of the LT with the use of the RT, on two traditional classification dataset: Image Segmentation (19 features) and Madelon (500 features), using NB. The outcome is shown in Figure 5.10: the employment of big-enough population sizes (and of sufficient numbers of fitness evaluation) can lead to better performance, *if statistical metrics can be measured reliably*. For Image Segmentation, the number of terminals to be considered in the genotype is relatively small due to the use of 19 features. This allows the LT to estimate node interdependencies reliably, and deliver better-than-random performance. For Madelon, the large number of terminals (500 features) makes it hard for the LT to outperform the RT within a limited computational budget. All in all, we recommend the use of GP-GOMEA as feature constructor since it was not worse on classification and was statistically better for regression. Furthermore, we advice to use the LT if the population size can be of the order of thousands or more (or even better, if exponential population sizing schemes are used as in Chapters 2 and 3). Otherwise, the RT should be preferred.

To assess if small constructed features are interpretable and if it is possible to visualize what the behavior of learned ML models, we showed some examples, providing evidence that both requirements can be reasonably satisfied. However, we did not perform a thorough study on interpretability of the constructed features. Several metrics have been recently proposed to measure some form of interpretability for ML models, that could be used to measure the interpretability of features as well. For example, in [7] two metrics called *simulatability* and *decomposability* are proposed. Simulatability represents the capability of humans to predict the output of a model given an input. Decomposability represents the capacity to intuitively understand the components of a model. Crucially, to measure this type of metrics, user studies need to be conducted. For example, experts of a field should be asked to provide feedback, on features constructed for datasets they are knowledgeable about (e.g., biochemists for data on gene expression, civil engineers for data on concrete strength). Nonetheless, we believe that enforcing features (and GP programs in general) to be small still remains a necessary condition to allow interpretability, although it is often ignored in GP literature (see Sec. 3.9).

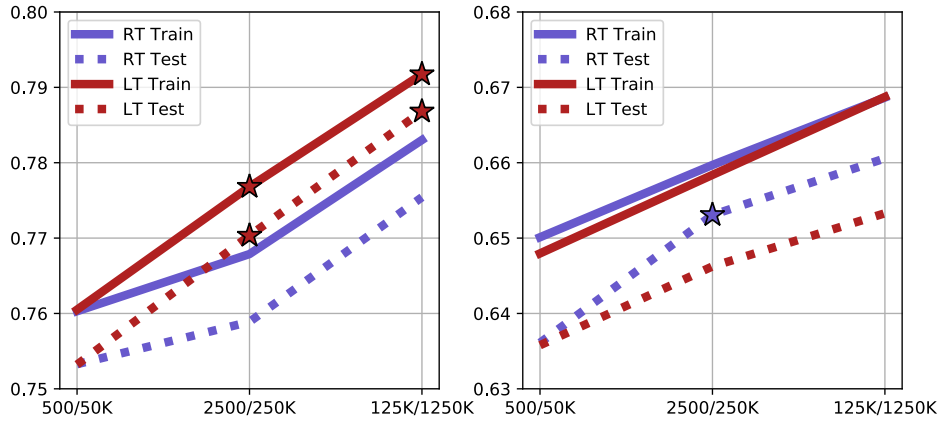

Figure 5.10: Comparison between the use of the RT and of the LT in GP-GOMEA. Vertical axis: median F1 score of 30 runs, obtained by NB on Image Segmentation (left) and on Madelon (right) using the first constructed feature, with $h = 4$ (note the different scale). Horizontal axis: population size / fitness evaluations budget. Stars indicate significant superiority (p-value < 0.05) of one method with respect to the other.

Considering the visualization examples proposed in Section 5.8, it is natural to compare our approach with well-known dimensionality reduction techniques, such as Principal Component Analysis (PCA) [42] or t-Distributed Stochastic Neighbor Embedding (t-SNE) [43]. We remark that those techniques and our FCS have very different objectives. In general, the sole aim of such techniques is to reduce the data dimensionality. PCA does so by detecting components that capture maximal variance. However, it does not attempt to optimize the transformation of the original feature set to improve an ML algorithm's performance. Also, PCA does not focus on the interpretability of the feature transformations. FCS takes the performance of the ML algorithm and interpretability of the features into account, while dimensionality reduction comes from forcing the construction of few features. We compared using 2 features constructed with RS (the worst search algorithm) with maximum $h = 2$, with using the first 2 PCs found by PCA. The use of constructed features over PCs resulted in significantly superior or equal test performance for all ML algorithms and for all problems. We remark, however, that PCA is extremely fast and independent from the ML algorithm.

Our FCS has several limitations. A first limitation regards the performance obtainable by the ML algorithm when the constructed features are used. FCS is iterative, and this can lead to suboptimal performance for a chosen K, compared to attempting to find K features at once. This is because the contributions of multiple features to an ML algorithm are not necessarily perpendicular to each other [25]. FCS could be changed to find, at any given iteration, a synergistic set of K features that is independent from previous iterations. To this end, the search algorithms need to be modified so that they can evolve sets of constructed features (a similar proposal for SGP was done in [15]). Yet, it is reasonable to expect that if K features need to be learned at the same time, larger population sizes may be needed compared to learning the K features iteratively.

Another limitation of this work is that hyper-parameter tuning was not considered. To include hyper-parameter tuning within FCS could bring even higher performance scores, or help prevent overfitting. A possibility could be, for example, to evolve pairs of features and hyper-parameter settings, where every time a feature is evaluated, the optimal hyper-parameters are also searched for. Such a procedure may likely require strong parallelization efforts, as C-fold cross-validation should be carried out for each combination of hyper-parameter values.

Lastly, it would be interesting to extend our approach to other classification and regression settings, e.g., problems with missing data; or to unsupervised tasks, as simple features may lead to better clustering of the examples.

5.11. Conclusion

With a simple evolutionary feature construction framework we have studied the feasibility of constructing few crucial and compact features with Genetic Programming (GP), towards improving the explainability of Machine Learning (ML) models without losing prediction accuracy. Within the proposed framework, we compared standard GP, random search, and the GP adaptation of the Gene-pool Optimal Mixing Evolutionary Algorithm (GP-GOMEA) as feature constructors, and found that GP-GOMEA is overall preferable when strict limitations on feature size are enforced. Despite limitations on feature size, and despite the reduction of problem dimensionality that we imposed by constructing only two features, we obtained equal or better ML prediction performance compared to using the original feature set for more than half the combinations of datasets and ML algorithms. In many cases, humans can understand what the feature means, and it is possible to visualize how trained ML models will behave. All in all, we conclude that feature construction is most useful and sensible for simpler ML algorithms, where more resources can be used for evolution (e.g., larger population sizes), which, in turn, unlock the added benefits of more advanced evolutionary mechanisms (e.g., using linkage learning in GP-GOMEA).

Acknowledgments

The authors acknowledge the Kinderen Kankervrij foundation for financial support (project #187). The majority of the computations for this work were performed on the Lisa Compute Cluster with the support of SURFsara.

REFERENCES

[1] H. Liu and H. Motoda, *Feature extraction, construction and selection: A data mining perspective*, Vol. 453 (Springer Science & Business Media, 1998).

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction* (Springer Science & Business Media, 2009).

[3] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection* (MIT Press, 1992).

[4] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming* (Lulu Enterprises, UK Ltd, 2008).

[5] J. H. Friedman, *Multivariate adaptive regression splines,* Annals of Statistics **19**, 123 (1991).

[6] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, Vol. 398 (John Wiley & Sons, 2013).

[7] Z. C. Lipton, *The mythos of model interpretability,* Queue **16**, 30:31 (2018).

[8] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, *A survey of methods for explaining black box models,* ACM Computing Surveys (CSUR) **51**, 93:1 (2018).

[9] A. Adadi and M. Berrada, *Peeking inside the black-box: A survey on explainable artificial intelligence (XAI),* IEEE Access **6**, 52138 (2018).

[10] B. Goodman and S. Flaxman, *European union regulations on algorithmic decision-making and a "right to explanation",* AI Magazine **38**, 50 (2017).

[11] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, *Improving model-based genetic programming for symbolic regression of small expressions,* (2019), accepted for publication in Evolutionary Computation. ArXiv preprint arXiv:1904.02050.

[12] A. Cano, A. Zafra, and S. Ventura, *An interpretable classification rule mining algorithm,* Information Sciences **240**, 1 (2013).

[13] B. P. Evans, B. Xue, and M. Zhang, *What's inside the black-box?: a genetic programming method for interpreting complex machine learning models,* in *Genetic and Evolutionary Computation Conference (GECCO) 2019* (ACM, 2019) pp. 1012–1020.

[14] B. Xue, M. Zhang, W. N. Browne, and X. Yao, *A survey on evolutionary computation approaches to feature selection,* IEEE Transactions on Evolutionary Computation **20**, 606 (2016).

[15] K. Krawiec, *Genetic programming-based construction of features for machine learning and knowledge discovery tasks,* Genetic Programming and Evolvable Machines **3**, 329 (2002).

[16] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees* (Wadsworth, 1984).

[17] M. Muharram and G. D. Smith, *Evolutionary constructive induction,* IEEE Transactions on Knowledge and Data Engineering **17**, 1518 (2005).

[18] B. Tran, B. Xue, and M. Zhang, *Genetic programming for feature construction and selection in classification on high-dimensional data,* Memetic Computing **8**, 3 (2016).

[19] N. S. Altman, *An introduction to kernel and nearest-neighbor nonparametric regression,* The American Statistician **46**, 175 (1992).

[20] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Pearson Education, 2003).

[21] K. P. Murphy, *Naive Bayes classifiers,* University of British Columbia **18** (2006).

[22] Q. Chen, M. Zhang, and B. Xue, *Genetic programming with embedded feature construction for high-dimensional symbolic regression,* in *Intelligent and Evolutionary Systems* (Springer, 2017) pp. 87–102.

[23] A. Cano, S. Ventura, and K. J. Cios, *Multi-objective genetic programming for feature extraction and data visualization,* Soft Computing **21**, 2069 (2017).

[24] M. Virgolin, T. Alderliesten, A. Bel, C. Witteveen, and P. A. N. Bosman, *Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2018) pp. 1395–1402.

[25] B. Tran, B. Xue, and M. Zhang, *Genetic programming for multiple-feature construction on high-dimensional classification,* Pattern Recognition **93**, 404 (2019).

[26] C. Cortes and V. Vapnik, *Support-vector networks,* Machine Learning **20**, 273 (1995).

[27] L. Breiman, *Random forests,* Machine Learning **45**, 5 (2001).

[28] R. Kohavi and G. H. John, *Wrappers for feature subset selection,* Artificial intelligence **97**, 273 (1997).

[29] T. P. Pawlak, B. Wieloch, and K. Krawiec, *Semantic backpropagation for designing search operators in genetic programming,* IEEE Transactions on Evolutionary Computation **19**, 326 (2015).

[30] R. R. Curtin, J. R. Cline, N. P. Slagle, W. B. March, P. Ram, N. A. Mehta, and A. G. Gray, *MLPACK: A scalable C++ machine learning library,* Journal of Machine Learning Research **14**, 801 (2013).

[31] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines,* ACM Transactions on Intelligent Systems and Technology **2**, 27:1 (2011), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[32] M. N. Wright and A. Ziegler, *Ranger: a fast implementation of random forests for high dimensional data in C++ and R,* (2015), arXiv preprint arXiv:1508.04409.

[33] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system,* in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016) pp. 785–794.

[34] J. Ni, R. H. Drieberg, and P. I. Rockett, *The use of an analytic quotient operator in genetic programming,* IEEE Transactions on Evolutionary Computation **17**, 146 (2013).

[35] D. R. White, J. Mcdermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly, and S. Luke, *Better GP benchmarks: community survey results and proposals,* Genetic Programming and Evolvable Machines **14**, 3 (2013).

[36] J. Albinati, G. L. Pappa, F. E. Otero, and L. O. V. B. Oliveira, *The effect of distinct geometric semantic crossover operators in regression problems,* in *European Conference on Genetic Programming* (Springer, 2015) pp. 3–15.

[37] J. Demšar, *Statistical comparisons of classifiers over multiple data sets,* Journal of Machine Learning Research **7**, 1 (2006).

[38] S. Holm, *A simple sequentially rejective multiple test procedure,* Scandinavian Journal of Statistics **6**, 65 (1979).

[39] Y. LeCun, *The MNIST database of handwritten digits,* (1998), dataset available at `http://yann.lecun.com/exdb/mnist`.

[40] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images,* (2009), technical Report. University of Toronto.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database,* in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2009) pp. 248–255.

[42] S. Wold, K. Esbensen, and P. Geladi, *Principal component analysis,* Chemometrics and Intelligent Laboratory Systems **2**, 37 (1987).

[43] L. v. d. Maaten and G. Hinton, *Visualizing data using t-SNE,* Journal of Machine Learning Research **9**, 2579 (2008).

6

On Automatically Selecting Similar Patients in Highly Individualized Radiotherapy Dose Reconstruction for Pediatric Cancer Survivors

The contents of this chapter are based on the following publication: **M. Virgolin**, I.W.E.M. van Dijk, J. Wiersma, C.M. Ronckers, C. Witteveen, A. Bel, T. Alderliesten, and P.A.N. Bosman. On the feasibility of automatically selecting similar patients in highly individualized radiotherapy dose reconstruction for historic data of pediatric cancer survivors. *Medical Physics* **45** (4), pp. 1504-1517, Wiley (2018).

Purpose: *The aim of this study is to establish the first step towards a novel and highly individualized 3D dose distribution reconstruction method, based on CT scans and organ delineations of recently treated patients. Specifically, the feasibility of automatically selecting the CT scan of a recently treated childhood cancer patient who is similar to a given historically treated child who suffered from Wilms' tumor is assessed.*

Methods: *A cohort of 37 recently treated children between 2 and 6 years old is considered. Five potential notions of ground-truth similarity are proposed, each focusing on different anatomical aspects. These notions are automatically computed from CT scans of the abdomen and 3D organ delineations (liver, spleen, spinal cord, external body contour). The first is based on deformable image registration, the second on the Dice Sørensen coefficient, the third on the Hausdorff distance, the fourth on pairwise organ distances, and the last is computed by means of the overlap volume histogram. The relationship between typically available features of historically treated patients and the proposed ground-truth notions of similarity is studied by adopting state-of-the-art machine learning techniques, including random forest. Also, the feasibility of automatically selecting the most similar patient is assessed by comparing ground-truth rankings of similarity with predicted rankings.*

Results: *Similarities (mainly) based on the external abdomen shape and on the pairwise organ distances are highly correlated (Pearson $r_p \geq 0.70$) and are successfully modeled with random forests based on historically recorded features (pseudo-$R^2 \geq 0.69$). In contrast, similarities based on the shape of internal organs cannot be modeled. For the similarities that random forest can reliably model, an estimation of feature relevance indicates that abdominal diameters and weight are the most important. Experiments on automatically selecting similar patients lead to coarse, yet quite robust results: the most similar patient is retrieved only 22% of the times, however the error in worst-case scenarios is limited, with the fourth most similar patient being retrieved.*

Conclusions: *Results demonstrate that automatically selecting similar patients is feasible when focusing on the shape of the external abdomen and on the position of internal organs. Moreover, whereas the common practice in phantom-based dose reconstruction is to select a representative phantom using age, height, and weight as discriminant factors for any treatment scenario, our analysis on abdominal tumor treatment for children shows that the most relevant features are weight, the anterior-posterior and left-right abdominal diameters.*

6.1. Introduction

Every day, radiation oncologists working on the treatment of childhood cancer patients are faced with the challenge of designing individualized treatment plans which ensure that a sufficiently high dose is delivered to the tumor while the surrounding healthy organs are spared. An excessive exposure of sensitive tissues to radiation may compromise crucial physiological functions and lead to severe health complications. Although the absolute number of young patients undergoing radiation treatment is moderate [1], the presence of malignancy in their developing bodies is likely to impact their entire life both physically and psychologically [2–7]. Moreover, children arguably are the most susceptible to adverse effects of radiation treatment and thus stand to benefit most from improvements in planning under the desired hypothesis of long-term survival, which is currently achieved for the treatment of Wilms' tumor, or nephroblastoma, the most common childhood abdominal malignancy [8, 9].

For adult patients, many follow-up studies exist where the relationship between radiation treatment with specific dose (fractions), and onset of adverse effects is analyzed (see, e.g., the work of QUANTEC [10]). Furthermore, detailed work has been done to understand which specific organ subvolumes are most sensitive to ionizing radiation, by observing fine-grained 3D dose distributions [11–13]. For children, however, the evidence collected so far is limited [14, 15]. Incorporating detailed knowledge on the relationship between 3D dose distributions and late adverse effects may greatly improve the design of treatment plans, ultimately reducing post-treatment complications and improving pediatric cancer survivors' quality of life.

Currently, researchers willing to study possible relationships between detailed 3D dose distributions and the onset of *late* adverse effects in *long-term* pediatric cancer survivors face a major obstacle: the lack of 3D treatment data. In fact, 3D information about the anatomy of historically treated patients could not be acquired before the advent of computed tomography (CT) and 3D treatment planning. The available information consists of patient characteristics recorded in historical patient records, notes on the treatment, and, in some cases, 2D simulator films used for planning at the time. The available data on late adverse effects collected from long-term follow-ups cannot be exploited to its full potential, as it can not be related to fine-grained 3D dose information. Therefore, to enable accurate dose-risk modeling in retrospective studies, a method to accurately reconstruct 3D dose distributions is needed.

The current state-of-the-art method to bridge this gap is the so-called *phantom-based dose reconstruction* [16, 17]. Phantoms are 3D representations of human bodies, constructed according to reference guidelines (e.g., ICRP 89 [18]), stored in libraries in a gender, age, height, and weight dependent fashion. The doses delivered to the organs of a historically treated patient are estimated by simulating the original treatment on a phantom. The dose reconstruction procedure can be summarized in four fundamental steps: 1. *Selection* — a phantom from the library is chosen, which most closely resembles a patient's available features (this is typically done using gender, age, height, and weight); 2. *Adaptation* — the phantom is adapted (i.e., shrunken, stretched) according to other specific features, such as measurements from a 2D simulator film; 3. *Treatment simulation* — the original treatment is simulated, using the phantom's virtual anatomy as a surrogate for the original body; and 4. *Measurement* — parameters about the dose are measured.

Clearly, the accuracy of the estimated dose relies on the completeness of the historical patient record considered, on the representativeness of the phantom library and on the quality of each one of the four aforementioned steps. A poor selection based on irrelevant features, as well as an ineffective adaptation, may compromise the accuracy of dose estimation. For children, growth and development do not follow a standard age-related pattern, thus selecting a representative phantom is especially difficult. Although rich libraries exist with reference phantoms for many height-weight combinations [17], and more and more possibilities to adapt mesh-based models to improve patient individualization are under investigation [19], an inherent limitation of phantom-based dose reconstruction is that it relies on *average* organ shapes and dimensions. Studies have however shown that there can be a great variability of internal organ shape among individuals with a similar body-mass index and that it is practically impossible to establish reference organ anatomy[17, 20].

Recently, an alternative to phantom-based dose reconstruction has been proposed, based on the reconstruction of 3D organs for historically treated patients, using navigator channels and finite element modeling deformable image registration [21–23]. The feasibility of the method has been tested on 3D dose reconstruction for lungs, heart, and breasts of adult Hodgkin lymphoma patients. Relying on CT scans of recently treated patients, a deformation model was built which uses information from 2D simulator films (or digitally reconstructed radiographs) to synthesize the organs of a historically treated patient by deforming the organs of a recently treated patient. A primitive *selection* step is performed to match the historically treated patient to a recent representative patient. This selection is based on 2D thorax measurements and gender, but it is advised that taking a smarter approach, possibly relying on more or different features, may improve the overall outcome. Still, the resulting dose reconstruction method was found to be clinically acceptable (median mean dose difference ≤ 1 Gy and median $V_{5\text{Gy}}$ and $V_{20\text{Gy}}$ differences $\leq 2\%$) [23] and has recently been adopted in practice [24].

In this study, we present a novel approach to select a recently treated patient for whom a CT scan is available to match a historically treated patient, and apply it to the scenario in which the 3D dose distribution of historically treated children with Wilms' tumor needs to be reconstructed. In order to find the features that are important to select a recently treated patient that resembles the 3D anatomy of a historically treated patient, a ground-truth notion of *similarity* among patients is needed. To this end, we propose and analyze five different notions of similarity, each focused on specific anatomical aspects. We choose to focus on anatomical similarity, rather than directly on similarity in 3D dose distributions, because the latter needs a specific treatment to be defined beforehand, whereas the former enables the reconstruction of virtually any treatment on a region of interest. The notions here proposed can be computed in an automatic and reproducible way, starting from CT scans and 3D organ delineations. We compute the five similarities on a cohort of pediatric patients and study correlations among them. We then assess which of the features that are typically available from historical patients' records are the most relevant to explain the similarities. Consequently, a state-of-the-art machine learning model, *random forest*, is trained using the most relevant features and its performance is measured in terms of correctly predicting rankings of similar patients, which ultimately is the goal. Finally, we provide a case of comparison between dose reconstruction based on the most similar

patient according to one of the proposed similarities, and based on the most similar patient according to age, height, and weight.

6.2. Materials & methods

6.2.1. Patient data

The records and CT data of 37 children were included (17 males, 20 females) in the age range of 2 to 6 years. Most of the patients suffered from Wilms' tumor (22); many underwent (partial) nephrectomy (21). All patients received chemotherapy prior to radiation treatment. The patients have been treated at the Academic Medical Center / Emma Children's Hospital in Amsterdam (34) or at the University Medical Center Utrecht / Princess Máxima Center for Pediatric Oncology in Utrecht (3), all after January 2000. A CT scan in supine position of the abdomen, from the top of the 10th thoracic vertebra (T10) to the bottom of 1st sacral vertebra (S1) is available for each patient. The median voxel size is 0.977 mm along left-right (LR) and anterior-posterior (AP) directions, and 2.5 mm along the superior-inferior (SI) direction (slice thickness). Also included are delineations of the liver and the spleen, and of the spinal cord and the outer body contour in the common region of interest from T10 to S1. Delineations of kidneys are not considered due to (partial) nephrectomy (see Sec. 6.2.2), with exception for the right kidney of three patients, used for an example of dose reconstruction (see Sec. 6.2.5).

Features that are typically reported in historical treatment records have been gathered for our cohort, together with measurements from digitally reconstructed radiographs generated from the CT scans, under the reasonable assumption that the old 2D simulator films have been preserved. These features are reported in Table 6.1 and Table 6.2. Details are as follows. Age was recorded at CT acquisition; height, weight, and body-mass index were recorded at intake in the radiotherapy department, which happened up to 3 months before CT acquisition. The distance between iliac crest and spinal cord is defined as the distance between the top point of the left iliac crest and the center of the spinal cord, along the line passing through both iliac crest top points. The LR diameter has been measured at the center of lumbar 2nd. Historically, the AP diameter was measured at the isocenter. Because of the high conformity of the treatment for renal fossa irradiation, such isocenter would typically be located in an SI section within the top of lumbar 1st and the bottom of lumbar 2nd. After inspection of historical treatments, an *average isocenter* has been set at the intersection of an SI line crossing the renal fossa with an LR line crossing the intervertebral disc between lumbar 1st and lumbar 2nd. For our (recently treated) patients, we measured the AP diameter at isocenter on their CT scans. Assuming symmetry of the abdomen, the average isocenter is set either in the left or right renal fossa, according to ease in carrying out the measurement for each patient. We observed on a random sample of 10 patients that the difference between the AP diameter measured on the average isocenter in the left renal fossa and the one in the right renal fossa is below 1 cm. For one patient the height was missing from clinical records, so an age- and gender-matched estimate from the Dutch children growth chart of 2010 has been used.

Table 6.1: Numerical patient features.

Numerical feature (abbreviation)	Unit of measure	Min	Max	Median	Mean	St. Dev.
Age	years	2.21	5.84	3.87	3.94	1.08
Height	cm	89.00	123.00	103.50	104.28	9.58
Weight	kg	10.00	28.00	16.55	16.88	3.75
Body-Mass Index (BMI)	kg/m^2	10.90	18.50	15.36	15.40	1.81
Length Spinal Cord T12-L4 (T12-L4)	cm	7.00	10.90	9.30	9.33	0.89
Distance Iliac Crest-Spinal Cord (IC-SC)	cm	4.30	6.80	5.70	5.58	0.54
LR Diameter at L2 (diam. LR)	cm	16.30	23.50	19.30	19.48	1.28
AP Diameter at Isocenter (diam. AP)	cm	11.30	16.00	13.20	13.37	1.53

Table 6.2: Categorical patient features.

Categorical feature (abbreviation)	Categories (# patients)
Gender	Female (20), Male (17)
Diagnosis	Ependyoma (1), Medulloblastoma (2), Neuroblastoma (9), Rhabdomyosarcoma (3), Wilms' tumor (22)
Tumor Site	Ductus choledochus (1), Fourth ventricle (3), Left kidney (12), Left suprarenal gland (6), Pelvic region (1), Retroperitoneum (1), Right kidney (10), Right lower abdomen (1), Right suprarenal gland (2)
Partial Nephrectomy (part. nephr.)	Left (2), Right (1), None (34)
Radical Nephrectomy (rad. nephr.)	Left (11), Right (10), None (16)

6.2.2. Similarity notions

In the following, we present five different notions of anatomical similarity and how they can be computed from CT scans and organ delineations. We further describe how correlation among similarities is measured.

Deformation-based similarity

This first notion of similarity is based on the amount of deformation that is needed to register one CT to another, via intensity-based deformable image registration. This metric is inspired by previous applications [25].

To compute this similarity, first scans are manually aligned on bony anatomy, using S1, the 5th lumbar vertebra and the iliac crests as reference. Second, for each possible pair of children, deformable image registration is performed to deform the first patient's CT to match the CT of the second patient and vice versa. This two-way registration is performed because of the asymmetry of most practical deformable image registration software. The software *elastix* [26, 27] has been adopted, with mostly standard parameters settings (adaptive stochastic gradient descent optimization, Mattes' mutual information metric, multi-resolution B-spline transformation). For the finest resolution step, a coarse grid size of 28 mm has been chosen following the guidelines for the deformation of large structures as found in the manual of elastix, combined with visual inspection of registration outcomes for several grid sizes. This choice limits the amount of unrealistic deformation on internal anatomy when registering the whole abdominal area (from T10 to S1) at once.

After computing the two registrations, a measure of deformation magnitude can be computed based on the deformations. A deformation is described using a meshed cube C, where each cell is the 3D offset to apply to a specific B-spline control point in order to

register the first image to the second. The deformation magnitude we compute from C is independent from translational components and is normalized on image volume. Specifically, for each cell $c \in C$, the "stretching" or "shrinking" component is computed by summing the absolute differences between the offset of c with the ones of its adjacent cubes, normalized by the number of adjacent cubes. The contribution of all cells is then summed and normalized by the total volume of the image. Formally, the deformation magnitude is thus:

$$D = \sum_{c \in C} \frac{1}{|A_c|} \sum_{a \in A_c} ||\vec{o}_c - \vec{o}_a|| \,, \tag{6.1}$$

where A_c is the sub-cube of cells centered at cell c and $\vec{o}_c$ is the 3D vector of offsets stored in cell c.

The two deformation magnitudes D_1 and D_2 computed from the two registrations are then averaged. Finally, to obtain a measure of similarity, the average magnitude needs to be inverted. We denote this similarity with S_{deform}:

$$S_{\text{deform}} = \left(\frac{D_1 + D_2}{2} \right)^{-1}. \tag{6.2}$$

Organ overlap-based similarity

This similarity notion focuses on internal organ overlap, and is based on the well-known Dice Sørensen coefficient [28, 29] (DSC) that indicates the overlap of two volumes V_1 and V_2.

We denote this measure for a specific organ delineation o by S^o_{DSC}, which is computed after aligning the images on their centers of mass. Thus, $S^o_{\text{DSC}} = 100 \frac{2|V^o_1 \cap V^o_2|}{|V^o_1| + |V^o_2|}$. A measure of similarity S_{DSC} is then computed by combining S^o_{DSC} for the various organs of interest. This is done by taking the Euclidean norm of the vector of all S^o_{DSC} components:

$$S_{\text{DSC}} = \sqrt{\sum_{o \in O} (S^o_{\text{DSC}})^2}. \tag{6.3}$$

For the set of organ delineations O, the liver, the spleen, the part of the spinal cord from T10 to S1, and the section of the external body contour within the field T10-S1 (arms excluded) are considered. Kidneys are not taken into consideration because 21 out of 37 patients of our cohort have been subject to (partial and/or radical) nephrectomy. If a similar patient needs to be found for a treatment that includes a kidney either as target (ispilateral) or as organ at risk (contralateral), then patients who (partially) miss this kidney should be considered completely dissimilar. If the kidney is not interesting for the reconstruction, a patient without (part of) the kidney may still be a good candidate. Therefore, for practical use, the outcome of a matching based on this similarity should be twofold: a similar patient who necessarily shares the kidney configuration with the historical one, and a similar patient who does not. To be able to use the whole cohort in the analysis, we consider the scenario where kidneys are not relevant for the reconstruction.

Organ shape-based similarity

Different from DSC, the Hausdorff distance is another recognized metric used to compare organ shapes which is focused on outlier points [30, 31]. Given two meshed surfaces A and B, the *directed* Hausdorff distance from A to B is defined as the maximum of the minimal Euclidean distances from A to B's vertices, i.e.,

$$h(A,B) = \max_{a\in A}\min_{b\in B}||a-b||. \tag{6.4}$$

The Hausdorff distance between A and B is $H(A,B) = \max\{h(A,B), h(B,A)\}$.

Similar to S^o_{DSC}, the notation $S^o_{\text{Hausdorff}}$ is used to indicate the organ-specific Hausdorff similarity:

$$S^o_{\text{Hausdorff}} = (H^o)^{-1} \times 10^p, \tag{6.5}$$

with p such that all $S^o_{\text{Hausdorff}} \geq 1$ (this ensures that $(S^o_{\text{Hausdorff}})^2 \geq S^o_{\text{Hausdorff}}$). The aggregated $S_{\text{Hausdorff}}$ is then computed as the Euclidean norm of the vector of all $S^o_{\text{Hausdorff}}$ components, where the delineations of liver, spleen, spinal cord (T10-S1), and external body (T10-S1) are considered:

$$S_{\text{Hausdorff}} = \sqrt{\sum_{o\in O}(S^o_{\text{Hausdorff}})^2}. \tag{6.6}$$

Organ constellation-based similarity

This similarity is proposed for the first time here. It is specifically aimed at capturing the variation in organ positions. Specifically, given a patient p, two organ delineations o_i, o_j, and the respective centers of mass $c^{o_i}(p)$, $c^{o_j}(p)$, let $d^{o_i,o_j}(p) = ||c^{o_i}(p) - c^{o_j}(p)||$ be the center-of-mass distance. Again, $o_i \in O = \{$liver, spleen, spinal cord (T10-S1), external body (T10-S1)$\}$. Let E^{o_i} be the set of 4 points that are the projections of o_i's center of mass on the external body along anterior, posterior, left, and right directions. Then the organ constellation-based similarity S_{const} for patients p_1 and p_2 is computed as follows. A first component $D^{\text{org-org}}_{\text{const}}(p_1,p_2)$ is calculated which represents the difference in pairwise organ distances, as:

$$D^{\text{org-org}}_{\text{const}}(p_1,p_2) = \sum_{o_i,o_j\in O, i\neq j}(d^{o_i,o_j}(p_1) - d^{o_i,o_j}(p_2))^2. \tag{6.7}$$

A second component represents the difference in distances between organs and the delineation of the external body:

$$D^{\text{org-ext}}_{\text{const}}(p_1,p_2) = \sum_{o_i\in O}\sum_{e\in E^{o_i}}(d^{o_i,e}(p_1) - d^{o_i,e}(p_2))^2. \tag{6.8}$$

Finally, $S_{\text{const}}(p_1,p_2)$ is:

$$S_{\text{const}}(p_1,p_2) = 1/\sqrt{D^{\text{org-org}}_{\text{const}}(p_1,p_2) + D^{\text{org-ext}}_{\text{const}}(p_1,p_2)}. \tag{6.9}$$

Overlap Volume Histogram-based similarity

The recently introduced Overlap Volume Histogram (OVH) [32, 33] was specifically designed to describe the position and shape of organs at risk near the tumor. In its original formulation, the OVH of an organ is computed by measuring the tumor-organ overlap at each step of a discrete expansion (or shrinkage) of the 3D tumor delineation, centered at the tumor.

Here the OVH is used to describe the shape and displacement of all organs at the same time. To this end, an artificial OVH is adopted, built using a sphere with a starting radius of 1.0 mm that expands from the center of mass of the body contour section within the T10-S1 region of interest. At each iteration, the sphere radius is expanded by 2.5 mm and the overlap with the organ delineation o of interest is computed. We denote with S^o_{OVH} the scaled inverse of the Manhattan distance of two patients' OVH for the organ delineation o. The Manhattan distance of two OVHs is the sum of absolute differences between each pair of histogram bins. A scaling is adopted similarly to what is done for $S^o_{\text{Hausdorff}}$, to ensure that $\left(S^o_{\text{OVH}}\right)^2 \geq S^o_{\text{OVH}}$. Furthermore, we denote with S_{OVH} the aggregated measure considering all organs at once, computed similarly to how it was originally used in the work introducing the OVH [32]:

$$S_{\text{OVH}} = \sqrt{\sum_{o \in O} (S^o_{\text{OVH}})^2}. \tag{6.10}$$

Correlations of similarity notions

The correlation between the similarity measurements is assessed with Pearson r_p and Spearman r_s coefficients. The first assumes a linear relationship between the two variables and is sensitive to outliers, whereas the second focuses on monotonic relationships, by considering only ranks (i.e., from sorting) rather than actual data values.

6.2.3. Regression and feature relevance

The goal now is to reproduce the measurements of similarity among patients using a function of only the features described in Section 6.2.1. However, because a similarity is defined over pairs of patients, individual features cannot be used directly. Instead, pairwise versions of the features are considered. For a numerical feature, the absolute difference of the two individual feature values is taken. Pairwise versions of categorical features are Boolean values, indicating whether the two categories are either the same (1) or different (0). In other words, for the i-th feature f_i of patients 1 and 2, the corresponding pairwise feature is $g_i^{1,2} = |f_i^1 - f_i^2|$ if f_i is numeric (e.g., weight), and it is

$$g_i^{1,2} = \begin{cases} 1 & \text{if } f_i^1 = f_i^2 \\ 0 & \text{if } f_i^1 \neq f_i^2 \end{cases} \tag{6.11}$$

if f_i is categorical (e.g., gender).

A random forest algorithm is used to learn how features can explain similarities, i.e., to learn a function representing similarity, given the features, and to compute feature relevance. Random forest is a widely-adopted machine learning technique which is capable of performing non-linear regression, and is robust in assessing feature relevance thanks

to its intrinsic feature sampling [34, 35]. In particular, the recent *cforest* implementation in R [36] is adopted in this work. Such technique uses conditional inference trees [37] to constitute the forest, providing an unbiased estimation of feature importance in scenarios where features have different scale of measurement or number of categories [38] (e.g., this study).

To understand which features are important for each similarity notion, a separate random forest is trained for each similarity. A split of data into separate training and testing sets is not necessary, since random forest inherently performs *bagging*, i.e., each regression tree in the forest is trained on a random subset of the data, and tested on the remaining. Given the stochastic nature of the method, 10 independent runs (i.e., training a random forest) are performed. The number of trees for the cforest is set to 100, and the number of random features to consider in the splits during tree construction is set to one-third of the total number of features (i.e., $mtry = 1/3$, guideline for regression). Feature relevance is investigated only if a forest out-of-bag pseudo-R^2 (i.e., the fit on the inherent test set, computed as $1 -$ mean squared error/variance of the ground-truth data) is high, since conclusions drawn from models with a low fit are generally false. Feature relevance is computed using the conditional variable importance method of cforest, which adjusts for correlations between predictor variables [39]. Successful forests are further refined by iteratively removing the least important feature, until a statistically significant increase in the out-of-bag mean squared error of the regression is observed. Significance is assessed using the Mann-Whitney-Wilcoxon test with increasing Bonferroni correction at each iteration i, with p-value of $0.05 \times i$. At the end of the procedure, a trained model is obtained for each similarity which is explainable using a subset of salient features.

6.2.4. Prediction and automatic selection of similar patients

For each similarity notion, we investigate the capability of learned random forests to correctly predict rankings of patients, which is the ultimate goal.

For this purpose, recent patients are used instead of historically treated patients. This way, the prediction capability can be tested against the ground-truth similarities. This complete process is performed in a leave-one-out cross validation fashion. Specifically, first, a test patient is removed from the cohort and a random forest is trained over the remaining patients using only the most relevant features. Second, the similarity between the test patient and each of the other patients is predicted by the random forest. This prediction is then sorted to obtain a ranking which is subsequently assigned a performance score. The overall prediction quality is the average of the scores obtained when repeating the steps above for each patient in the cohort. Moreover, runs for individual patients are repeated 10 times to reduce stochastic noise in the forest training phases. The pseudo-code of this procedure is illustrated in Algorithm 6.1.

To score the performance in ranking prediction, the following four indicators are proposed: *head presence*: number of patients in the top k of the predicted ranking who are also in the top k of the ground-truth ranking, i.e., the capability of correctly predicting the most similar patients; *tail presence*: analogous to head presence, but on the bottom k patients of the rankings, i.e., the capability of correctly predicting the most dissimilar patients; *average displacement*: calculated for the patients who are wrongly predicted to be in the top k, the average displacement in positions between the k-th position and the actual

Algorithm 6.1 Function assessing the quality of the automatic selection of similar patients.

```
function scoreAutomaticSelection(cohort, similarityNotion, bestFeatures)
    score ← 0
    for i ∈ {1, ..., 10} do
        for p ∈ cohort do
            others ← cohort \ {p}
            f ← trainForest(others, bestFeatures)
            s ← f.predictSimilarity(p, others)
            s* ← getSimilarity(p, similarityNotion)
            r ← makeRanking(s)
            r* ← makeRanking(s*)
            score ← score + computeScore(r, r*)
    score ← score/(10|cohort|)
    return score
```

position in the ground-truth ranking; and *worst displacement*: similar to average displacement, but calculated only for the worst case, i.e., for the patient who is most dissimilar yet wrongly predicted to be in the top k. Note that for $k = 1$ the average displacement is the same as the worst displacement. All the indicators are reported as percentages. A good prediction is one that reaches high head presence and tail presence, and minimizes average displacement and worst displacement. In the experiments, the parameter k varies in $\{1, 3, 5\}$, where $k = 1$ means that the prediction is assessed only on the most similar patient (dissimilar for tail presence). This corresponds to evaluating an automatic selection which retrieves only one patient. By increasing k, it is possible to see if the prediction is generally good, noisy or consistently poor. An example of the indicators is depicted in Figure 6.1. With $k = 5$, three patients out of five are correctly predicted as similar, i.e., the head presence is 60%, while four are correctly predicted as very dissimilar (tail presence is 80%). Patients 5 and 9 are incorrectly predicted to be within the 5 most similar, whereas in the ground-truth ranking they are respectively 3 and 5 positions away from the head, out of a total of 12 ($= 17 - 5$) dissimilar patients. Thus, the average displacement is 33% and the worst displacement is 42%.

6.2.5. RECONSTRUCTION CASE

Two illustrative dose reconstructions are performed for one patient p: one using a representative who is correctly predicted to be most similar by our model according to S_{deform}, and one using a representative who is the most similar according to age, height, and weight. Specifically, the latter match is performed by taking the patient q with lowest rooted sum of squared age, height, and weight differences (after normalizing all differences to the interval $[0, 1]$):

$$\sqrt{(\text{age}^p - \text{age}^q)^2 + (\text{height}^p - \text{height}^q)^2 + (\text{weight}^p - \text{weight}^q)^2}. \tag{6.12}$$

The reconstruction is performed by applying the treatment plan of patient p to the other two patients, using the treatment planning system Oncentra (version 4.3, Elekta, Stock-

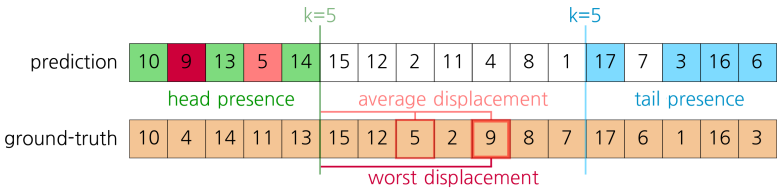

Figure 6.1: Example to illustrate the computation of the four indicators of prediction performance. The prediction and ground-truth rankings contain the IDs of 17 patients instead of 37 for ease of representation. In both rankings, the leftmost IDs are the most similar patients, the rightmost the most dissimilar. Patients correctly predicted in the head (tail) are depicted in green (blue). Patients 5 and 9 are wrongly predicted to be in the head, and determine the average displacement. Patient 9 is the worst to be predicted in the head, being the most dissimilar in the ground-truth ranking, and determines the worst displacement.

holm, Sweden). The treatment plan is a real, clinical plan for left renal fossa irradiation (Wilms' tumor), using an Elekta Linac with a multi-leaf collimator beam limiting device, energy: 6 MV. Under the hypothesis that p is a historically treated patient, a digitally reconstructed radiograph of p is generated displaying the borders of the treatment field. Consequently, radiographs are also generated for the two matched patients, and are used to adjust the field border of the plan to correct for evident discrepancies in the bony anatomy. The monitor units of the original plan are also scaled to keep the dose point in the middle of the field (isocenter) as close as possible to its value before adjustment. This work has been assessed by an experienced pediatric radiation oncologist. Lastly, the treatment is simulated and the following metrics are recorded: mean dose D_{mean}, max dose $D_{2\text{cc}}$, and the dose volume histograms (DVHs), for right kidney, liver, spleen, and spinal cord (T10-S1).

6.3. Results

6.3.1. Correlations of similarity notions

Pairwise Pearson and Spearman correlation coefficients between the similarity measures S_{deform}, S_{DSC}, $S_{\text{Hausdorff}}$, S_{const}, and S_{OVH} are reported in Table III. The two coefficients r_p and r_s show good agreement in general. Whereas S_{deform} and S_{const} are moderately correlated (r_p of 0.64, r_s of 0.55), S_{DSC}, $S_{\text{Hausdorff}}$, and S_{OVH} are more independent. It is crucial to recall that these latter similarities are highly dependent on organ shape. The correlation coefficients of S_{deform} and S_{const} with S^o_{DSC}, $S^o_{\text{Hausdorff}}$, S^o_{OVH}, i.e., the latter similarities separately computed for each organ o, are represented in Figure 6.2 (only Pearson correlation is reported, since Spearman leads to very similar results). These results show that moderate to substantial correlations are present among S_{deform}, S_{const}, $S^{\text{body}}_{\text{DSC}}$, $S^{\text{body}}_{\text{Hausdorff}}$, and $S^{\text{body}}_{\text{OVH}}$. Moreover, $S^{\text{liver}}_{\text{DSC}}$ is highly correlated with $S^{\text{liver}}_{\text{Hausdorff}}$, and $S^{\text{spleen}}_{\text{DSC}}$ is highly correlated with $S^{\text{spleen}}_{\text{Hausdorff}}$. However, these latter similarities are weakly correlated with the former ones, based on deformable image registration, disposition of the internal organs and shape of the abdomen. The fact that $S^{\text{spinal cord}}_{\text{DSC}}$ and $S^{\text{spinal cord}}_{\text{Hausdorff}}$ are not clearly correlated is likely due to the elongated shape and different bending of this organ (see, e.g., Fig. 6.3).

Table 6.3: Pearson r_p and Spearman r_s correlation coefficients for the five (aggregated) similarity notions.

	Pearson r_p					Spearman r_s				
	S_{deform}	S_{DSC}	$S_{\text{Hausdorff}}$	S_{const}	S_{OVH}	S_{deform}	S_{DSC}	$S_{\text{Hausdorff}}$	S_{const}	S_{OVH}
S_{deform}	1.00	0.24	0.32	0.64	0.22	1.00	0.18	0.35	0.55	0.16
S_{DSC}	0.24	1.00	0.39	0.49	0.34	0.18	1.00	0.46	0.50	0.31
$S_{\text{Hausdorff}}$	0.32	0.39	1.00	0.37	0.14	0.35	0.46	1.00	0.48	0.14
S_{const}	0.64	0.49	0.37	1.00	0.52	0.55	0.50	0.48	1.00	0.42
S_{OVH}	0.22	0.34	0.14	0.52	1.00	0.16	0.31	0.14	0.42	1.00

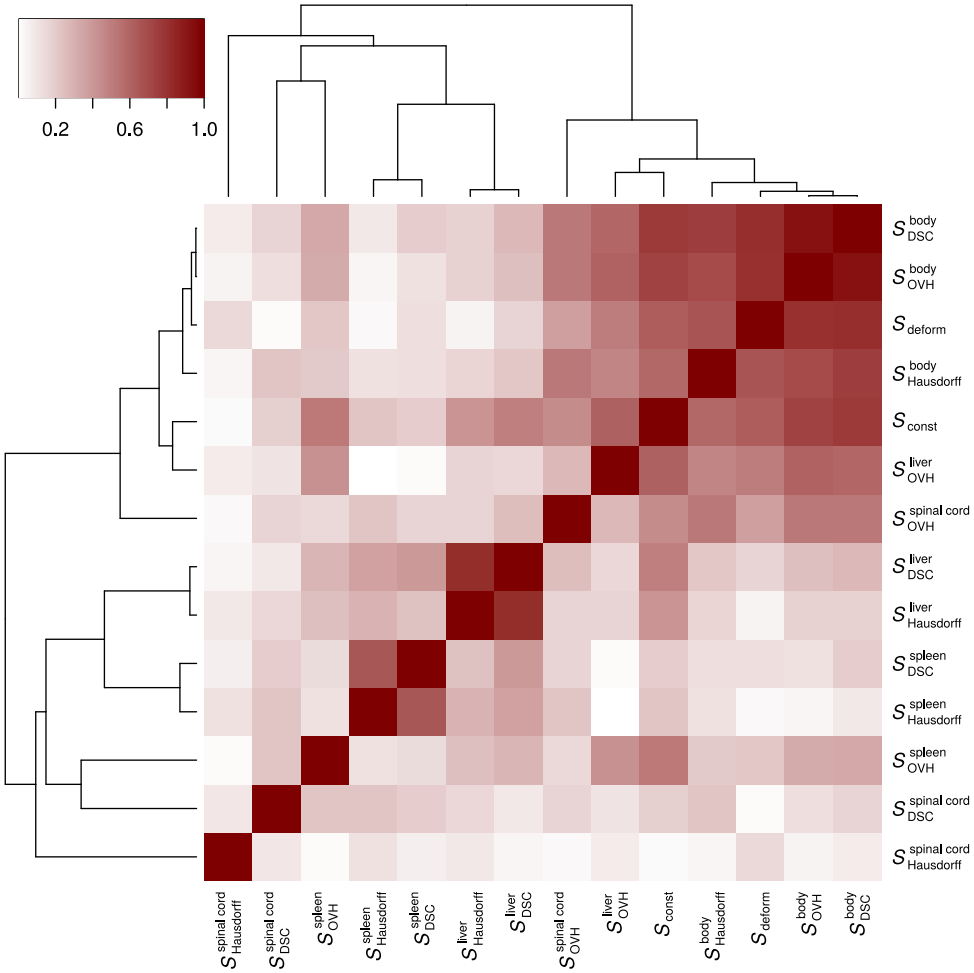

Figure 6.2: Heatmap and hierarchical-clustering dendrogram based on absolute values of Pearson correlation coefficients between S_{deform}, S_{const} and the organ-specific S^o_{DSC}, $S^o_{\text{Hausdorff}}$, and S^o_{OVH}, with $o \in \{$ liver, spleen, spinal cord (T10-S1), body (T10-S1) $\}$.

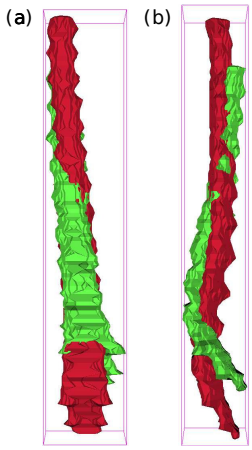

Figure 6.3: Two spinal cords aligned on the center of mass for DSC and Hausdorff distance computation, from AP (a) and LR (b) perspective.

6.3.2. Regression and feature relevance

The magnitude of Pearson correlation coefficients between all pairs of features is depicted in Figure 6.4. The largest correlation coefficients were found for combinations of age and height, LR diameter and weight, and tumor site and radical nephrectomy.

Figure 6.5 shows the pseudo-R^2 of the random forest method (averaged over 10 runs), for each similarity notion. Although with random forest it is possible to learn non-linear interactions among features, only few similarities are modeled with a high pseudo-R^2: S_{deform}, S_{const}, $S_{\text{DSC}}^{\text{body}}$, and $S_{\text{OVH}}^{\text{body}}$ (see Figs. 6.5(a) and 6.5(b)); with, respectively, pseudo-R^2 of 0.70, 0.69, 0.87, and 0.85. In particular, the variation of measurements for the notions aggregating (in a, S_{DSC}, $S_{\text{Hausdorff}}$, S_{OVH}) or specifically focused on internal organ shapes (all similarities in Figs. 6.5(c), 6.5(d), 6.5(e)) can not be modeled well (i.e., low pseudo-R^2). This result clearly shows that the features at hand do not provide enough information to grasp the large variability in the internal anatomy of our young cohort. On the other hand, it is the similarities mainly or specifically focusing on the overall abdomen (S_{deform}, S_{const}, $S_{\text{DSC}}^{\text{body}}$, $S_{\text{OVH}}^{\text{body}}$) that are decently modeled. Not surprisingly, these similarities are found to be correlated among themselves (Sec. 6.3.1).

The feature relevance of the best modeled similarities S_{deform}, S_{const}, $S_{\text{DSC}}^{\text{body}}$ and $S_{\text{OVH}}^{\text{body}}$ is reported in Figure 6.6. The abdominal AP and LR diameters clearly stand out as common relevant features for the four similarities. Although not salient in three out of four cases, weight is always among the predictors with a statistically significant relevance. It is also worth noticing how nephrectomy has a slight, yet relevant influence on the organs' constellation, which may be linked to a possible shift of organs after kidney resection.

6.3.3. Prediction and automatic selection of similar patients

The capability of the models trained using the most important features (obtained in Sec. 6.3.2) to perform automatic selection is now assessed. Table 6.4 shows the quality of the pre-

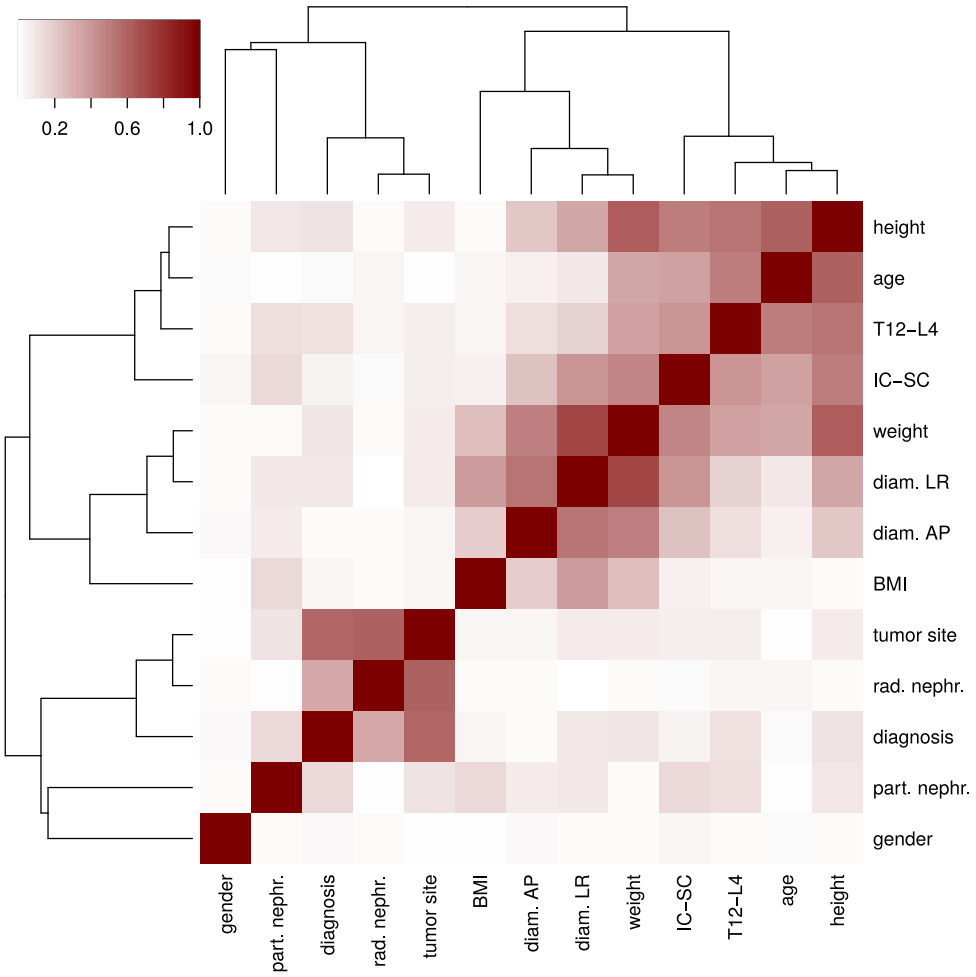

Figure 6.4: Heatmap and hierarchical-clustering dendrogram representing (absolute) Pearson correlation among pairwise features (abbreviations as introduced in Table 6.1 and Table 6.2).

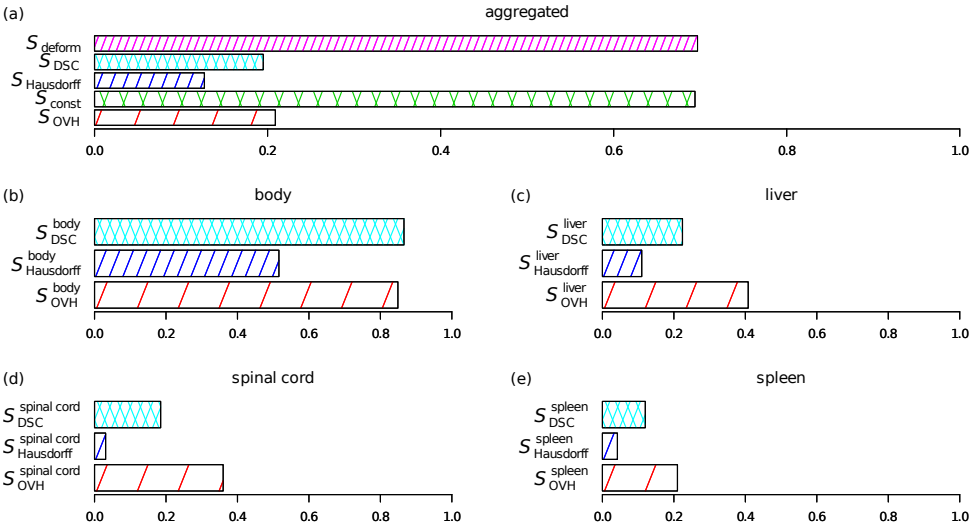

Figure 6.5: Pseudo-R^2 of trained random forests for all the similarities, both aggregated (a), and for individual organs: body (b), liver (c), spinal cord (d), spleen (e).

diction in terms of the four proposed indicators head presence, tail presence, average displacement, and worst displacement, for the similarity notions that could be reliably modeled: S_{deform}, S_{const}, $S^{\text{body}}_{\text{DSC}}$, and $S^{\text{body}}_{\text{OVH}}$. The results are averages over 10 repetitions. When considering only the most similar patient ($k = 1$), the best choice is predicted correctly 22.37% of the times (averaged over all similarity measures, where the worst is S_{deform} with only 16.22%, and the best is S_{const} with 30.00%). When such prediction is wrong, the patient misclassified as most similar is approximately the fifth most similar, i.e., the patient is displaced within the top 11% of the ground-truth ranking (the worst is S_{const} with 13.73%, the best is $S^{\text{body}}_{\text{DSC}}$ with 7.71%). Now, by increasing k to 3 and 5 it can be seen that the head presence increases to roughly 50%, that is, half of the most similar patients become correctly predicted to be in the head of the ranking. A similar behavior can be observed on the tail presence indicator, i.e., the accuracy in predicting the most dissimilar patients. While the head and the tail presence increase considerably with k, the average displacement oscillates slightly. For the best modeled $S^{\text{body}}_{\text{DSC}}$ (pseudo-R^2 of 0.87 with the random forest trained using only the most important features), the worst displacement increase is also particularly limited when increasing k. This trend shows that the models are reliably able to find a coarse notion of similarity, with a quite good capability of identifying which patients constitute a cluster of most similar, and which constitute a cluster of most dissimilar, but with limitations in terms of accurately ordering the most similar patients.

6.3.4. Reconstruction case

Patients with ID 6, 18, and 34 are the ones used to perform an illustrative reconstruction. The right kidney is intact in all three patients. Patient 6 (age: 2.51 y, gender: female, height: 93.0 cm, weight: 14.0 kg) is hypothesized to be a historical patient for whom a

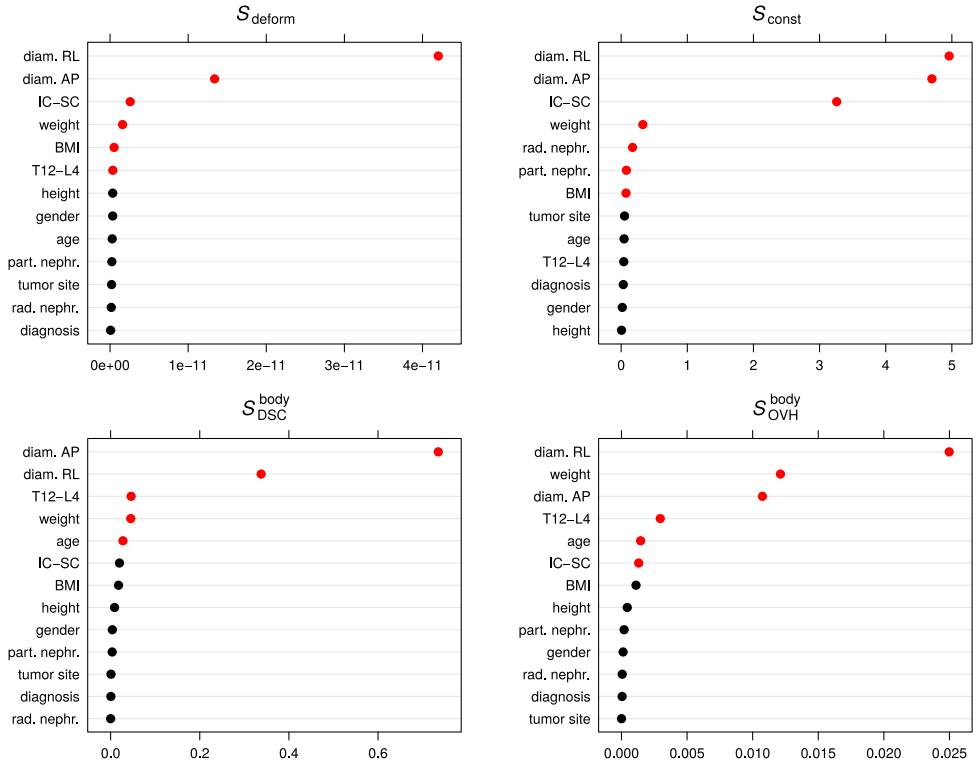

Figure 6.6: Feature relevance from random forest for the similarities that could be modeled with high pseudo-R^2. The relevance represents the beta coefficients in regression models. Dots in red represent the minimal subset of features with which it is possible to obtain a random forest with no significant loss in mean squared error. Note: different scales are used on the x-axes.

Table 6.4: Predicted ranking scores for the explainable similarities, for $k \in \{1, 3, 5\}$. Results in bold are the best scores among similarities for a fixed k.

	head pres.			tail pres.			avg. disp.			worst disp.		
k	1	3	5	1	3	5	1	3	5	1	3	5
S_{deform}	16.22	34.14	44.54	49.73	**82.52**	76.70	13.44	12.60	10.73	13.44	23.78	28.20
$S^{\text{body}}_{\text{DSC}}$	25.95	**52.07**	**65.35**	65.14	75.50	86.00	**7.71**	**5.61**	**5.07**	**7.71**	**12.94**	**18.08**
S_{const}	**30.00**	44.23	50.81	27.57	65.95	78.86	13.73	10.85	11.35	13.73	22.38	32.91
$S^{\text{body}}_{\text{OVH}}$	17.30	41.44	56.43	**83.24**	79.64	**86.92**	10.73	8.11	7.14	10.73	16.91	21.55
mean	22.37	42.79	54.28	56.42	75.90	82.12	11.40	9.29	8.57	11.40	18.75	25.19

Table 6.5: The upper part of the table shows D_{mean} and $D_{2\text{cc}}$ in cGy for right kidney, liver, spleen, and spinal cord (T10-S1) for patients 6, 18, and 34. The lower part reports the relative error of the reconstruction, with lowest errors in bold.

	Right kidney		Liver		Spleen		Spinal cord	
Patient ID	D_{mean}	$D_{2\text{cc}}$	D_{mean}	$D_{2\text{cc}}$	D_{mean}	$D_{2\text{cc}}$	D_{mean}	$D_{2\text{cc}}$
6	1.80	12.29	4.83	13.84	13.96	14.86	11.36	13.80
18	2.95	11.19	4.19	13.63	13.83	14.24	10.52	13.99
34	2.03	11.53	4.44	13.89	13.76	14.14	10.03	13.63
				% relative error from patient 6				
18	63.89	8.95	13.25	1.52	**0.93**	**4.17**	**7.39**	1.38
34	**12.78**	**6.18**	**8.07**	**0.36**	1.43	4.84	11.71	**1.23**

dose reconstruction is needed. Random forest correctly predicts patient 34 (age: 2.21 y, gender: male, height: 90.0 cm, weight: 15.0 kg) to be the closest match to 6, according to S_{deform}. Patient 18 (age: 2.58 y, gender: female, height: 92.0 cm, weight: 13.0 kg) is the most similar to 6 according to age, height, and weight. However, patient 18 is ranked as 16th in terms of S_{deform} similarity to patient 6.

The outcome of the dose reconstruction is presented in terms of D_{mean} and $D_{2\text{cc}}$ in Table 6.5, and qualitatively in terms of DVHs in Figure 6.7. Recurring to patient 34 as reference leads to markedly better dose reconstruction for right kidney and liver, while patient 18 is slightly preferable for spleen and spinal cord (with exception of $D_{2\text{cc}}$ for the latter). In particular, patient 34 excels against patient 18 when comparing D_{mean} of the right kidney, with a relative error of 12.78% for the former, and 63.89% for the latter (51.11% difference). Instead, the case where 18 is mostly preferable is the relative error on D_{mean} of spinal cord, with 7.39% for patient 18 and 11.71% for patient 34 (only 4.32% difference). A qualitative inspection of the DVHs points at a similar conclusion: while patient 18 may seem to be preferable for the reconstruction of spleen and spinal cord (Fig. 6.7(c) and 6.7(d)), patient 34 is markedly better for right kidney and liver (Fig. 6.7(a) and 6.7(b)) reconstruction.

6.4. Discussion

To the best of our knowledge, this study represents a first attempt to understand what are the key anatomical characteristics to represent similarity among childhood cancer patients, and to assess the feasibility of performing an automatic selection of a representative patient for a highly individualized CT-based 3D dose reconstruction method.

To establish a ground-truth notion of similarity between patients, we have proposed and studied five possible measures of similarity. It has been found that the DSC and Hausdorff-based similarities of the same organ are highly correlated for liver and spleen, but not for the spinal cord (likely due to its elongated shape). Furthermore, the two new measures we proposed, S_{deform} and S_{const}, are correlated with the similarities that are based on established shape descriptors (DSC, Hausdorff, and OVH) when using the contour of the external abdomen as shape.

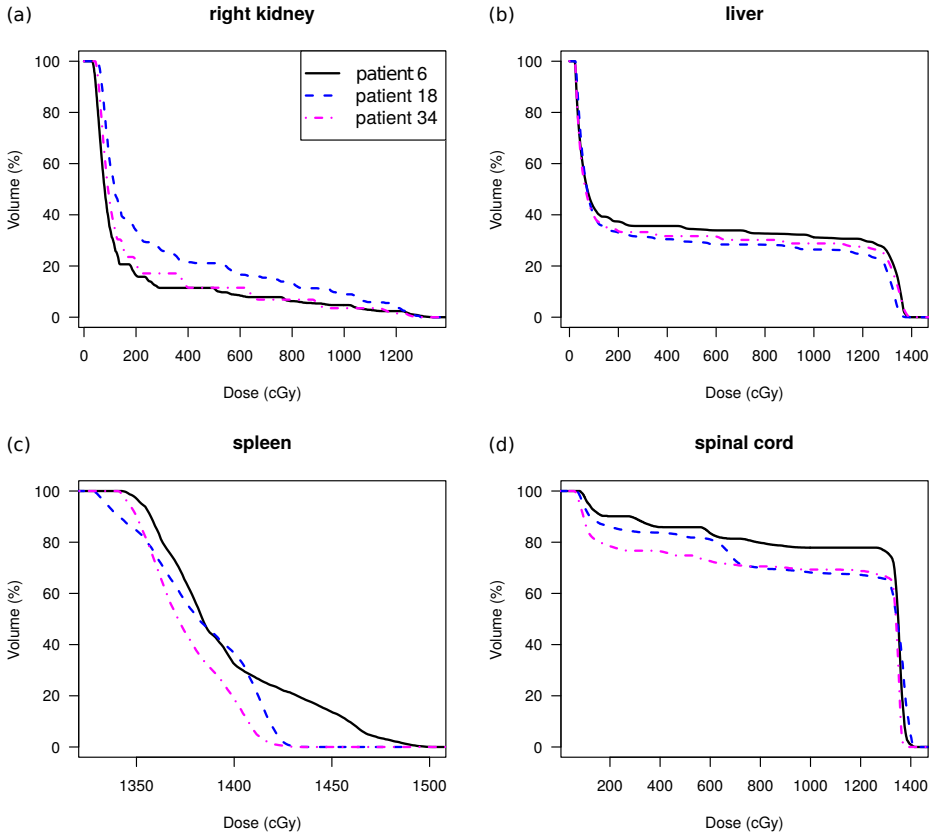

Figure 6.7: DVHs of right kidney (a), liver (b), spleen (c), and spinal cord (T10-S1) (d) of patients 6, 18, and 34. Using patient 34 as reference for the dose reconstruction of patient 6 leads to highly similar DVHs for the right kidney (a) and the liver (b).

Random forest has been adopted to relate (pairwise) historically available features with the ground-truth notions of similarity. This allowed for the modeling of complex, non-linear interactions and the assessment of feature relevance. We found that aggregated similarities focusing on general aspects of the whole abdomen (S_{deform}, S_{const}), as well as the ones focusing specifically on the shape of the external abdomen ($S_{\text{DSC}}^{\text{body}}$, $S_{\text{OVH}}^{\text{body}}$) were decently modeled. However, a relationship between the features at hand and internal organ-specific shape-based similarities could not be learned. Such result is not surprising, given previous literature studies on organ variability (e.g., correlating organ volumes with BMI in adults [20]). Internal organ variability is possibly even further increased by the disease these children suffer from, together with prior drug treatments (resulting in, e.g., possible hepatosplenomegaly). This means that more features are needed to predict the internal anatomy. To this end, we plan to harvest more information from images, i.e., by means of 2D/3D registrations [40], and the usage of navigator channels on 2D (digitally reconstructed) radiographs, which have been proven capable of enabling decent organ shape reconsruction [21, 22] in the adaptation step. Furthermore, the models learned by random forest are complex to interpret. Adopting other machine learning methods may generate equally powerful models of easier interpretation, and hopefully provide more insight on the problem (e.g., genetic programming [41]).

For the well-modeled similarities, results show that the most salient features are the abdominal diameters. This is in contrast with the common practice in phantom-based dose reconstruction of using age, height, and weight (gender is typically not considered for young children) to select a representative phantom for any treatment scenario. We hypothesize that it may be necessary to define a different set of relevant features depending on the specific treatment to perform a very accurate selection. The model-based prediction of similarity rankings are noisy, but roughly coincide with the ground truth. If the first retrieved patient would be taken as the singular best match ($k = 1$), the choice would often be wrong (e.g., three out of four times for $S_{\text{DSC}}^{\text{body}}$), yet the error would be limited (e.g., within the three most similar for $S_{\text{DSC}}^{\text{body}}$). Thus, although sub-optimal, the resulting learned models can be considered robust. When k is increased, the head presence increases much more than the average displacement. Therefore, it may be interesting to assess whether computing the 3D dose distributions for a small number of k patients, and then take the average as the final result, may lead to more accurate reconstructions.

It will be important to understand if and how these similarities can be combined into a single ground-truth. Between highly correlated notions, it would be natural to look for an aggregated compromise that expresses all of them at once. Contrary, highly uncorrelated ones should probably be kept separated. This would mean that actually a multi-objective selection approach is sought after, that is able to retrieve a limited set of different patients who are similar to the historically treated one according to different notions of similarity. Eventually, a physician could decide which CT to use for reconstruction, or, as mentioned above, multiple reconstructions could be performed and the average dose distribution could be taken as final result.

Besides a new perspective on the selection of a reference patient, this work presents some limitations. A first limitation is that we chose to consider the scenario of Wilms' tumor treatment in children, and focused on anatomical similarity of the abdomen. Thus, the results presented in this work are valid within the domain of the chosen region of

interest (abdomen) and the characteristics of the cohort (Caucasian children between approximately 2 to 6 years old). However, the approach presented here is general, and can be applied to other regions of interests and cohorts. Furthermore, note that the choice of trying to machine-learn similar anatomy rather than directly machine-learn similar dose distributions overcomes the limitations of the latter: a specific treatment does not need to be defined and (manually) simulated beforehand on each available CT scan. In fact, our results can be used for any abdominal treatment (e.g., neuroblastoma), within the cohort characteristics (Caucasian children between approximately 2 to 6 years old). Nonetheless, it is well possible to define a similarity based on 3D dose distributions for a specific treatment and learn a model capable of retrieving similar patients in that sense.

A second limitation is the lack of an analysis of the relationship between similarity notions and dose outcomes (e.g., D_{mean}, $D_{2\text{cc}}$, and DVHs). This work showed one exemplary reconstruction, but a validation study involving a statistically relevant number of patients should be performed. Such work may be realized as shown in our illustrative reconstruction, using a number of recently treated patients instead of historically treated ones, as follows. 1. A treatment plan should be simulated on the patient and measurements from the 3D dose distribution should be recorded; 2. Using (historically plausible) features of the patient, a representative patient from a cohort of candidates should be selected according to a similarity notion; 3. Dose measurements should be taken on the representative patient and compared with the ones taken in the first step. The outcome of such a study may tell which similarity notion(s) is preferable, i.e., leads to more accurate dose reconstruction. However, it is important to remark that the selection phase discussed here is but the first step in a dose reconstruction pipeline. In order to comprehensively validate the contribution of this work in dose reconstruction, the reference anatomy retrieved by our method should undergo an adaptation step, to increase its resemblance with the historically-treated patient, and the original treatment should be simulated as close as possible. Consequently, a fair comparison with state-of-the-art phantom-based dose reconstruction methods will be possible. This is however outside the scope of this chapter.

A third limitation is the relatively small size of the cohort examined in this study, and, in general, the availability of data to specific institutes. As generally true with machine learning approaches, we expect that including more data will result in improved models and prediction capabilities. We plan to expand our cohort by including anonymized patient data provided by radiotherapy departments of other institutes.

6.5. Conclusion

This study presents a novel, machine learning-based approach to an important part in the process of 3D dose reconstruction for historically treated patients using recent real patient data rather than phantoms: selecting a good representative recently treated patient.

Similarity measures that consider the overall abdomen and the position of internal organs can be decently modeled (pseudo-$R^2 \geq 0.7$), and automatic selection based on such models reaches a coarse but robust performance. However, it was not possible to find a relationship between features available for historically treated patients and specific organ shapes. All in all, our novel approach shows potential in using CT scans of actual, recent patients directly to perform dose reconstruction, and a number of future research

steps are possible to gain substantial improvements, e.g., extending the data with more patients, exploiting available 2D image data such as simulator films to extend the feature set, and exploring combinations of similarity notions.

Acknowledgments

The authors would like to thank Brian V. Balgobind, Koen F. Crama, Rianne M.A.J. de Jong, Jorrit Visser and Ziyuan Wang (department of Radiation Oncology, Academic Medical Center, Amsterdam, the Netherlands) for their help with the retrieval and pre-processing of data. We further thank Raquel Dávila Fajardo (department of Radiation Oncology, UMC Utrecht Cancer Center, the Netherlands) for sharing the data of 3 patients treated at the UMC Utrecht for inclusion in this study. This work is part of the research project *3D dose reconstruction for children with long-term follow-up — Toward improved decision making in radiation treatment for children with cancer* with project number 187, which is financed by Stichting Kinderen Kankervrij (KiKa). Dr. C. M. Ronckers is supported by the Dutch Cancer Society (KWF), grant number UVA2012-5517.

References

[1] V. Jairam, K. B. Roberts, and B. Y. James, *Historical trends in the use of radiation therapy for pediatric cancers: 1973-2008,* International Journal of Radiation Oncology · Biology · Physics **85**, e151 (2013).

[2] C. A. Sklar and L. S. Constine, *Chronic neuroendocrinological sequelae of radiation therapy,* International Journal of Radiation Oncology · Biology · Physics **31**, 1113 (1995).

[3] R. K. Mulhern, T. E. Merchant, A. Gajjar, W. E. Reddick, and L. E. Kun, *Late neurocognitive sequelae in survivors of brain tumours in childhood,* The Lancet Oncology **5**, 399 (2004).

[4] M. M. Geenen, M. C. Cardous-Ubbink, L. C. M. Kremer, C. van den Bos, H. J. H. van der Pal, R. C. Heinen, M. W. M. Jaspers, C. C. E. Koning, F. Oldenburger, N. E. Langeveld, *et al.*, *Medical assessment of adverse health outcomes in long-term survivors of childhood cancer,* JAMA **297**, 2705 (2007).

[5] Y.-L. Tsai, S.-C. Tsai, S.-H. Yen, K.-L. Huang, P.-F. Mu, H.-C. Liou, T.-T. Wong, I.-C. Lai, P. Liu, H.-L. Lou, *et al.*, *Efficacy of therapeutic play for pediatric brain tumor patients during external beam radiotherapy,* Child's Nervous System **29**, 1123 (2013).

[6] I. W. E. M. van Dijk, F. Oldenburger, M. C. Cardous-Ubbink, M. M. Geenen, R. C. Heinen, J. de Kraker, F. E. van Leeuwen, H. J. van der Pal, H. N. Caron, C. C. Koning, *et al.*, *Evaluation of late adverse events in long-term Wilms' tumor survivors,* International Journal of Radiation Oncology · Biology · Physics **78**, 370 (2010).

[7] S. Thouvenin-Doulet, P. Fayoux, H. Broucqsault, and V. Bernier-Chastagner, *Neurosensory, aesthetic and dental late effects of childhood cancer therapy,* Bulletin du Cancer **102**, 642 (2015).

[8] N. Breslow, A. Olshan, J. B. Beckwith, and D. M. Green, *Epidemiology of Wilms tumor,* Medical and Pediatric Oncology **21**, 172 (1993).

[9] J. S. Dome, N. Graf, J. I. Geller, C. V. Fernandez, E. A. Mullen, F. Spreafico, M. Van den Heuvel-Eibrink, and K. Pritchard-Jones, *Advances in Wilms tumor treatment and biology: progress through international collaboration,* Journal of Clinical Oncology **33**, 2999 (2015).

[10] S. M. Bentzen, L. S. Constine, J. O. Deasy, A. Eisbruch, A. Jackson, L. B. Marks, R. K. Ten Haken, and E. D. Yorke, *Quantitative Analyses of Normal Tissue Effects in the Clinic (QUANTEC): an introduction to the scientific issues,* International Journal of Radiation Oncology · Biology · Physics **76**, S3 (2010).

[11] L. J. Boersma, E. M. F. Damen, R. W. De Boer, S. H. Muller, C. M. Roos, R. A. V. Olmos, N. van Zandwijk, and J. V. Lebesque, *Dose-effect relations for local functional and structural changes of the lung after irradiation for malignant lymphoma,* Radiotherapy and Oncology **32**, 201 (1994).

[12] Y. Cao, C. Pan, J. M. Balter, J. F. Platt, I. R. Francis, J. A. Knol, D. Normolle, E. Ben-Josef, R. K. Ten Haken, and T. S. Lawrence, *Liver function after irradiation based on computed tomographic portal vein perfusion imaging,* International Journal of Radiation Oncology · Biology · Physics **70**, 154 (2008).

[13] F. Buettner, S. L. Gulliford, S. Webb, M. R. Sydes, D. P. Dearnaley, and M. Partridge, *Assessing correlations between the spatial distribution of the dose to the rectal wall and late rectal toxicity after prostate radiotherapy: an analysis of data from the MRC RT01 trial (ISRCTN 47772397),* Physics in Medicine & Biology **54**, 6535 (2009).

[14] L. Constine, D. Hodgson, and S. Bentzen, *MO-D-BRF-01: Pediatric treatment planning II: The PENTEC report on normal tissue complications,* Medical Physics **41**, 419 (2014).

[15] L. S. Constine, C. M. Ronckers, C.-H. Hua, A. Olch, L. C. M. Kremer, A. Jackson, and S. M. Bentzen, *Pediatric Normal Tissue Effects in the Clinic (PENTEC): an international collaboration to analyse normal tissue radiation dose–volume response relationships for paediatric cancer patients,* Clinical Oncology **31**, 199 (2019).

[16] M. Stovall, R. Weathers, C. Kasper, S. A. Smith, L. Travis, E. Ron, and R. Kleinerman, *Dose reconstruction for therapeutic and diagnostic radiation exposures: use in epidemiological studies,* Radiation Research **166**, 141 (2006).

[17] A. M. Geyer, S. O'Reilly, C. Lee, D. J. Long, and W. E. Bolch, *The UF/NCI family of hybrid computational phantoms representing the current US population of male and female children, adolescents, and adults-application to CT dosimetry,* Physics in Medicine & Biology **59**, 5225 (2014).

[18] J. Valentin, *Basic anatomical and physiological data for use in radiological protection: reference values: ICRP publication 89,* Annals of the ICRP **32**, 1 (2002).

[19] S. Lamart, R. Imran, S. L. Simon, K. Doi, L. M. Morton, R. E. Curtis, C. Lee, V. Drozdovitch, R. Maass-Moreno, C. C. Chen, *et al.*, *Prediction of the location and size of the stomach using patient characteristics for retrospective radiation dose estimation following radiotherapy,* Physics in Medicine & Biology **58**, 8739 (2013).

[20] G. L. de la Grandmaison, I. Clairand, and M. Durigon, *Organ weight in 684 adult autopsies: new tables for a Caucasoid population,* Forensic Science International **119**, 149 (2001).

[21] A. Ng, T. Nguyen, J. L. Moseley, D. C. Hodgson, M. B. Sharpe, and K. K. Brock, *Reconstruction of 3D lung models from 2D planning data sets for Hodgkin's lymphoma patients using combined deformable image registration and navigator channels,* Medical Physics **37**, 1017 (2010).

[22] A. Ng, T.-N. Nguyen, J. L. Moseley, D. C. Hodgson, M. B. Sharpe, and K. K. Brock, *Navigator channel adaptation to reconstruct three dimensional heart volumes from two dimensional radiotherapy planning data,* BMC Medical Physics **12**, 1 (2012).

[23] A. Ng, K. K. Brock, M. B. Sharpe, J. L. Moseley, T. Craig, and D. C. Hodgson, *Individualized 3D reconstruction of normal tissue dose for patients with long-term follow-up: a step toward understanding dose risk for late toxicity,* International Journal of Radiation Oncology · Biology · Physics **84**, e557 (2012).

[24] R. Zhou, A. Ng, L. S. Constine, M. Stovall, G. T. Armstrong, J. P. Neglia, D. L. Friedman, K. Kelly, T. J. FitzGerald, and D. C. Hodgson, *A comparative evaluation of normal tissue doses for patients receiving radiation therapy for Hodgkin lymphoma on the Childhood Cancer Survivor Study and recent Children's Oncology Group trials,* International Journal of Radiation Oncology · Biology · Physics **95**, 707 (2016).

[25] S. Klein, M. Loog, F. van der Lijn, T. den Heijer, A. Hammers, M. de Bruijne, A. van der Lugt, R. P. W. Duin, M. M. B. Breteler, and W. J. Niessen, *Early diagnosis of dementia based on intersubject whole-brain dissimilarities,* in *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (IEEE, 2010) pp. 249–252.

[26] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. Pluim, *Elastix: a toolbox for intensity-based medical image registration,* IEEE Transactions on Medical Imaging **29**, 196 (2009).

[27] D. P. Shamonin, E. E. Bron, B. P. F. Lelieveldt, M. Smits, S. Klein, and M. Staring, *Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer's disease,* Frontiers in Neuroinformatics **7**, 50 (2014).

[28] L. R. Dice, *Measures of the amount of ecologic association between species,* Ecology **26**, 297 (1945).

[29] K. H. Zou, S. K. Warfield, A. Bharatha, C. M. C. Tempany, M. R. Kaus, S. J. Haker, W. M. Wells III, F. A. Jolesz, and R. Kikinis, *Statistical validation of image segmentation quality based on a spatial overlap index,* Academic Radiology **11**, 178 (2004).

[30] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, *Comparing images using the Hausdorff distance,* IEEE Transactions on Pattern Analysis and Machine Intelligence **15**, 850 (1993).

[31] Z. Xu, S. A. Panjwani, C. P. Lee, R. P. Burke, R. B. Baucom, B. K. Poulose, R. G. Abramson, and B. A. Landman, *Evaluation of body-wise and organ-wise registrations for abdominal organs,* in *Medical Imaging 2016: Image Processing*, Vol. 9784 (International Society for Optics and Photonics, 2016) p. 97841O.

[32] M. Kazhdan, P. Simari, T. McNutt, B. Wu, R. Jacques, M. Chuang, and R. Taylor, *A shape relationship descriptor for radiation therapy planning,* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2009) pp. 100–108.

[33] B. Wu, F. Ricchetti, G. Sanguineti, M. Kazhdan, P. Simari, M. Chuang, R. Taylor, R. Jacques, and T. McNutt, *Patient geometry-driven information retrieval for IMRT treatment plan quality control,* Medical Physics **36**, 5497 (2009).

[34] L. Breiman, *Random forests,* Machine Learning **45**, 5 (2001).

[35] A. Cutler, D. R. Cutler, and J. R. Stevens, *Tree-based methods,* in *High-Dimensional Data Analysis in Cancer Research* (Springer, 2009) pp. 1–19.

[36] R Core Team, *R: A language and environment for statistical computing,* R Foundation for Statistical Computing (2013).

[37] T. Hothorn, K. Hornik, and A. Zeileis, *Unbiased recursive partitioning: A conditional inference framework,* Journal of Computational and Graphical Statistics **15**, 651 (2006).

[38] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, *Bias in random forest variable importance measures: Illustrations, sources and a solution,* BMC Bioinformatics **8**, 25 (2007).

[39] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, *Conditional variable importance for random forests,* BMC Bioinformatics **9**, 307 (2008).

[40] P. Markelj, D. Tomaževič, B. Likar, and F. Pernuš, *A review of 3D/2D registration methods for image-guided interventions,* Medical Image Analysis **16**, 642 (2012).

[41] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection* (MIT Press, 1992).

7

Machine Learning for Automatic Phantom Construction

Machine Learning (ML) is proving extremely beneficial in many healthcare applications. In pediatric oncology, retrospective studies that investigate the relationship between treatment and late adverse effects still rely on simple heuristics. To assess the effects of radiation therapy, treatment plans are typically simulated on phantoms, i.e., virtual surrogates of patient anatomy. Currently, phantoms are built according to reasonable, yet simple, human-designed criteria. This often results in a lack of individualization. We present a novel approach that combines imaging and ML to build individualized phantoms automatically. Given the features of a patient treated historically (only 2D radiographs available), and a database of 3D Computed Tomography (CT) imaging with organ segmentations and relative patient features, our approach uses ML to predict how to assemble a patient-specific phantom automatically. Experiments on 60 abdominal CTs of pediatric patients show that our approach constructs significantly more representative phantoms than using current phantom building criteria, in terms of location and shape of the abdomen and of two considered organs, the liver and the spleen. Among several ML algorithms considered, the Gene-pool Optimal Mixing Evolutionary Algorithm for Genetic Programming (GP-GOMEA) is found to deliver the best performing models, which are, moreover, transparent and interpretable mathematical expressions.

The contents of this chapter are based on the following preprint: **M. Virgolin**, Z. Wang, T. Alderliesten, and P.A.N. Bosman. Machine learning for automatic construction of pseudo-realistic pediatric abdominal phantoms. *Submitted. Preprint arXiv:1909.03723*, arXiv (2019). The preprint extends the publication: **M. Virgolin**, Z. Wang, T. Alderliesten, and P.A.N. Bosman. Machine learning for automatic construction of pediatric abdominal phantoms. *In Proceedings of SPIE Medical Imaging 2020: Imaging Informatics for Healthcare, Research, and Applications*, International Society for Optics and Photonics (2020) (to appear).

7.1. Introduction

Virtual anthropomorphic phantoms are 3D representations of the human body that are used as surrogates for the anatomy of humans, to estimate the quantity and geometric distribution of radiation dose when having been exposed to radiation, e.g., in radiation treatment for cancer patients [1, 2]. Because anatomy resemblance is one of the key sources of uncertainty in dose estimation [3], the phantom needs to represent the anatomy of the patient for whom estimates are needed with high precision.

Current methods for phantom building have two major limitations. Firstly, traditionally, building phantoms is a manual and time-consuming task. Approximations of human anatomies are produced using simple geometrical shapes [4, 5], or by considering actual organ segmentations from Computed Tomography (CT) scans [6–8]. These anatomies are shaped and/or adapted according to population-based statistics and/or reasonable human-designed criteria [4, 6, 7, 9, 10]. Because the procedure is laborious, a limited number of phantoms is made, each meant to represent a category of patients. The second and perhaps more fundamental limitation is that it is unknown how to best define categorization criteria that best capture resemblance in individual patient anatomy. So far, only simple criteria such as partitioning by combinations of age, gender, percentiles height and weight, have been explored [4, 6–8]. Nevertheless, several studies, including Chapter 6, have indicated that such simple criteria are incapable of capturing the high variance in human internal anatomy [2, 6, 11], and this can ultimately lead to coarse dose estimations [12].

Machine Learning (ML) is becoming more and more a reliable approach to tackle hard and heterogeneous problems in healthcare [13]. This is because ML can infer patterns from data that are hard to spot, and to model for humans. In the context of phantom construction, the use of ML could improve upon the rough categorization methods that are currently being employed. In this chapter, we present a new take on phantom construction that shows that it is possible to use ML to obtain better phantoms. In particular, we propose an automatic phantom-construction pipeline that can be used to generate pseudo-realistic phantoms that are patient-specific. To overcome the need for laborious manual intervention, we propose to re-use 3D patient imaging (CT scans and organ segmentations) collected in a database, to assemble new anatomy combinations. To estimate how to best perform this assembling, i.e., to move beyond the use of too simplistic criteria, we rely on ML. Specifically, we train ML models to learn relationships between patient features and 3D metrics based on their internal anatomy.

We consider a relatively hard scenario where phantoms are needed and patient features are limited: dose reconstruction for historical patients, i.e., patients treated in the pre-3D planning era, when radiation treatment plans were designed using 2D radiographs. As no 3D imaging is available for historical patients to simulate the treatment and estimate the radiation dose distribution, phantoms are necessary to act as surrogate anatomies in order to reconstruct 3D dose distributions [12, 14]. We focus on children between 2 to 6 years, and on dose reconstruction for abdominal radiation treatment, for the following reasons. Firstly, children are typically under-represented in existing phantom libraries, i.e., phantoms are available for few categories [4, 6]. Secondly, the inclusion of radiation treatment has led to high survival rates for several types of pediatric abdominal cancer (e.g., Wilms' tumor, the most common type of kidney cancer), but it is known to cause late adverse effects [15, 16]. Thirdly, it has recently been shown that when CT scans are

selected based on age and gender to serve as a surrogate for pediatric abdominal patients, there is a high risk of obtaining inaccurate dose reconstructions [12]. The ultimate goal is to realize sufficiently accurate dose reconstruction by use of more representative phantoms, which can then be used to better understand how radiation dose contributes to the onset of late adverse effects. Providing this information can support radiation oncologists in the design of better treatment plans with smaller chances of adverse effects for today's abdominal radiation treatment.

7.2. Materials & methods

7.2.1. Data

We built a database using data of 60 pediatric cancer patients, in the age range of 2 to 6 years. The patients were treated after 2002 at the radiation oncology department of the Amsterdam UMC, location AMC, in Amsterdam, or at the University Medical Center Utrecht/Princess Màxima Center for Pediatric Oncology in Utrecht. For each patient, a CT is available that fully includes the lower part of the thorax to the lower part of the abdomen, specifically from the top of the Thoracic 10th vertebra (T10) to the bottom of the Sacral 1st vertebra (S1). The median axial thickness of the CT scan is 2.5 mm, and the median in-plane resolution is $1.0 \text{ mm} \times 1.0 \text{ mm}$.

To simulate the scenario of dose reconstruction for patients treated in the pre-3D planning era, we only consider patient features that were typically recorded at the time. We base our choices on the availability of features for the Emma Children's Hospital/Academic Medical Center (EKZ/AMC) childhood cancer survivor cohort, treated between 1966 and 1996 [16]. One important source of information for the EKZ/AMC cohort is 2D coronal radiographs, which were acquired to plan radiation treatment. The coronal radiographs enable us to perform measurements, along Left-Right (LR) and Inferior-Superior (IS) directions, based on visible anatomical landmarks, i.e., the bony anatomy (see Fig. 7.1). For our cohort of recently treated patients, we simulate historical radiographs using Digitally Reconstructed Radiographs (DRRs), derived from the CTs using in-house developed software. Figure 7.1 (left, middle) shows an example of a historical radiograph and of a DRR.

Table 7.1 lists the features considered in this work. Features involving measurements from DRRs were collected after manual placement of landmarks (using 3D Slicer [17]), exemplified in Figure 7.1 (right). The only source of information on the Anterior-Posterior (AP) direction is the abdominal diameter, that was historically measured using rulers and calipers, at the center of the radiation treatment field (which corresponded with the isocenter for the EKZ/AMC cohort). For our cohort, we measured the abdominal diameter along AP from the CTs, using a typical isocenter position for abdominal flank irradiation, as described in Section 6.2.1. Figure 7.2 shows the Pearson correlation coefficients between the considered features. Most features are moderately correlated, and few are strongly correlated, e.g., height with age and weight. The distance along IS between the top of the right diaphragm and T12 (RDIS) stands out as it is associated with the lowest correlations with any other feature. We measured this feature in an attempt to capture information on the breathing state of the patient, which is known to be correlated with organ position [18, 19]. The low correlation is likely because the particular breathing state of the patient can be reasonably expected to be not correlated with the other features we considered.

Figure 7.1: Left: An example of a 2D coronal radiograph, taken by a radiation treatment simulator used in the pre-3D planning era, including annotations by medical personnel (sensitive information censored). Middle: A digitally reconstructed radiograph built from a CT. Images are acquired in anterior-posterior setting. Liver and spleen are not clearly visible. Right: Example of manually-placed landmarks used to measure features from radiographs. The length of the left and the right diaphragm along the LR direction is derived by fitting a cubic spline to the respective dashes.

We consider two Organs At Risk (OARs), i.e., organs for which exposure to radiation is known to lead to adverse effects: the liver and the spleen. These OARs are particularly interesting because their shape and position is known to vary substantially per individual, and are very hard to predict (Chapter 6). In general, the liver and spleen are not (clearly) visible in historical radiographs (see Fig. 7.1). For each patient in the cohort, 3D segmentations of the OARs and of the external body (delimited along IS between T10 and S1) were firstly automatically generated (with ADMIRE research software, 2.3.0, Elekta AB, Stockholm, Sweden), then manually checked and corrected by experienced radiation treatment technologists (with Velocity software, version 3.2.0, Varian Medical Systems, Inc. Palo Alto, CA, US), and finally approved by a pediatric radiation oncologist.

Table 7.1: Features of our cohort, typically available for patients treated in the pre-3D planning era. Note: gender is categorical, other features are numerical.

Feature name	Abbreviation	Unit	Source	Min	Max	Mean	St.Dev.
Age	AGE	years	records	2.0	6.0	3.8	1.2
Abdominal diameter in AP at typical isocenter	ADAP	mm	records	11.1	16.0	13.3	1.2
Abdominal diameter in LR at middle of L2	ADLR	mm	radiograph	16.3	23.5	19.4	1.4
Distance from top of iliac crest to spinal cord along LR	ICSC	mm	radiograph	4.3	6.8	5.5	0.6
Gender	GEND	–	records	33 females, 27 males			
Heart size along LR	HESZ	mm	radiograph	6.8	9.9	8.5	0.7
Height	HEIG	cm	records	86.0	123.0	103.0	10.7
Left diaphragm length along LR	LDLR	mm	radiograph	6.5	10.7	8.4	0.9
Right diaphragm length along LR	RDLR	mm	radiograph	6.2	10.5	8.3	0.8
Right diaphragm top to T12 distance along IS	RDIS	mm	radiograph	4.0	7.8	5.9	1.0
Spinal cord length along IS from T12 to L4	SPIS	mm	radiograph	7.0	10.9	9.3	0.8
Weight	WEIG	kg	records	10.0	28.0	16.4	3.7

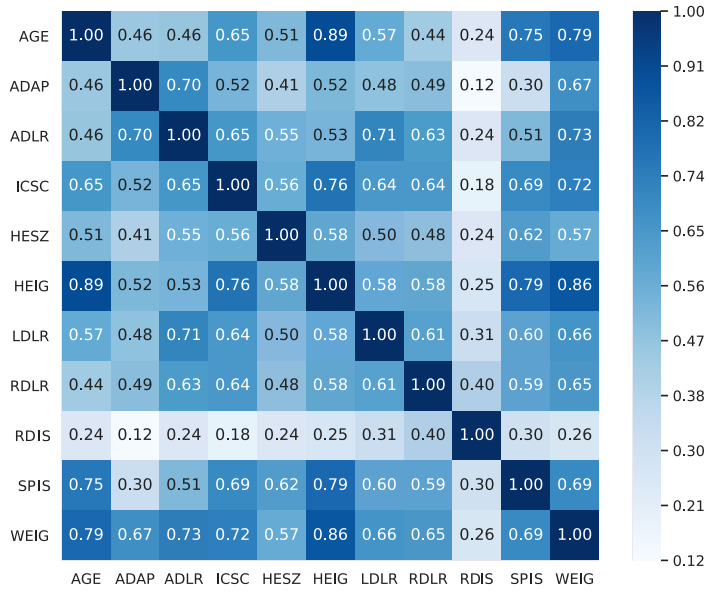

Figure 7.2: Pearson correlation coefficients (number and color-coded) between the considered features. See Table 7.1 for the meaning of abbreviations.

7.2.2. Pipeline for automatic phantom construction

The outcome of our pipeline is a CT-based phantom, built using recent patient imaging data. The goal is to resemble the anatomy of the historical patient as closely as possible.

The pipeline is summarized in Figure 7.3. Given the patient features as input, first an ML model $\mathcal{M}_{\mathrm{S}}^{\mathrm{Body}}$ is used to predict which CT is most resembling in terms of overall body shape (for the abdominal region). We call this CT the "receiver". Then, the OARs of the receiver are "resected". Resection of an OAR is performed by setting the voxels of the receiver that belong to the OAR to Hounsfield Unit (HU) values that represent generic soft abdominal tissues (this is a parameter, we used 78, as done in the phantoms of the University of Florida/National Cancer Institute[6]). Subsequently, for each OAR, its center of mass position is predicted using dedicated models, i.e., one for the LR ($\mathcal{M}_{\mathrm{LR}}^{\mathrm{OAR}}$), one for the AP ($\mathcal{M}_{\mathrm{AP}}^{\mathrm{OAR}}$), and one for the IS position ($\mathcal{M}_{\mathrm{IS}}^{\mathrm{OAR}}$). A fifth model ($\mathcal{M}_{\mathrm{S}}^{\mathrm{OAR}}$) is used to predict which OAR segmentation to retrieve among the ones available based on a shape-focused metric (described in Sec. 7.2.3). We choose to adopt separate models because the features related to one metric may be independent from the ones related to another metric, e.g., we expect features related to the LR direction to be important for the LR coordinate of the center of mass of an OAR, but not (or substantially less) for its AP or IS coordinates and for determining what segmentation has the most promising shape.

We refer to the CT that is chosen by $\mathcal{M}_{\mathrm{S}}^{\mathrm{OAR}}$ to provide the OAR segmentation as a "donor". Next, the segmentation is "transplanted" into the receiver, using the predicted position. Transplantation is achieved similarly to resection: we add the OAR segmentation to the set of segmentations of the receiver, placed in the predicted position, and we set

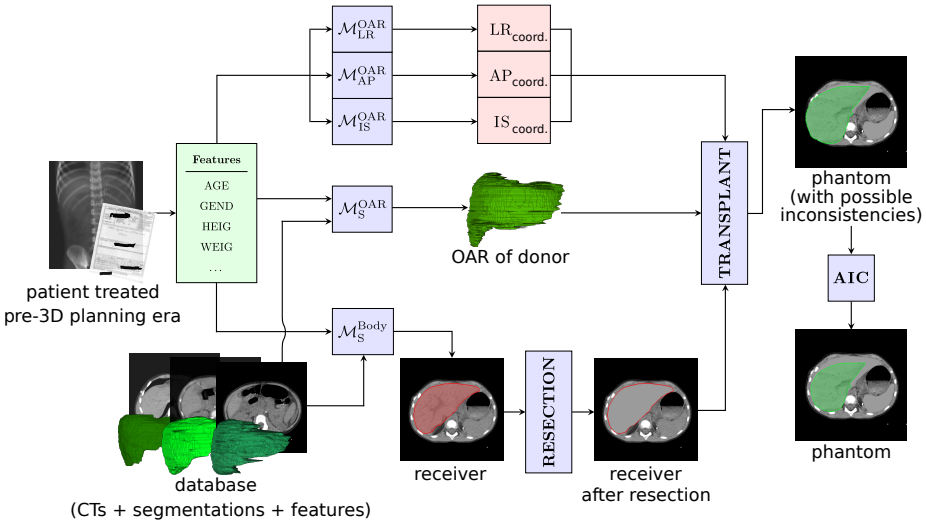

Figure 7.3: Pipeline for automatic phantom building. Pre-trained ML models are used to predict 3D OAR positions as well as what OAR donor segmentation and what receiver CT to retrieve from the database to construct a patient-specific phantom CT (only the liver is considered as OAR in this example). The resected and transplanted OAR is highlighted in red and green, respectively.

the HU values of the voxels in the receiver that belong to the OAR segmentation to the respective HU values from the donor scan.

A final step consists of correcting the phantom for possible anatomical inconsistencies with an Anatomical Inconsistencies Correction (AIC) procedure. This is because the ML models can in principle predict to place segmentations in positions that result in non-realistic anatomy (e.g, OARs overlapping with each other). An optimizer is therefore used to subsequently resize and re-position the OARs to correct for anatomical inconsistencies, with minimal corrections so as to compromise the quality of the ML model predictions as little as possible (more details are given later, in Sec. 7.2.4).

To store CTs and segmentations sets, we relied on the Digital Imaging and Communications in Medicine standard (DICOM). Segmentation sets were stored in DICOM RT-Structure sets. To handle this data format, we used the pydicom package for Python (`https://pydicom.github.io`). We do not provide full details on the implementation here (e.g., which fields of the DICOM files are changed and how), but our source code is available at: `http://github.com/marcovirgolin/APhA`.

7.2.3. Machine learning

This section describes how the ML models in the pipeline are trained. Firstly, we present how datasets for supervised learning are built. Secondly, we present the ML algorithms that we selected, and their hyper-parameters. Finally, we describe the learning strategy, i.e., how we train, tune, and validate the models.

Datasets for training

To train the ML models we prepare separate datasets to learn each ML model independently. As aforementioned, we learn separate models under the assumption that 3D metrics are (largely) independent from each other. Hence, $1+4\times\#\text{OARs}$ datasets are needed: one to learn how to pick the receiver CT, three for each OAR to predict the OAR position, and one more for each OAR to predict which segmentation to retrieve.

We perform numerical feature normalization by z-scoring [20], i.e., each feature is normalized by subtraction of the mean and division by the standard deviation. Since patient gender is categorical, we set it to a binary value: 0 for females and 1 for males. As no correlation coefficient above 0.9 was found between the features (Fig. 7.2), we do not exclude any feature. Z-scoring is also applied to the target variable y.

We construct two types of datasets. One type is used for OAR position modeling along three orthogonal directions (i.e., LR, AP, IS). The other type is for donor and receiver retrieval. The preparation of the datasets for OAR position modeling is straightforward. We define the OAR position relative to a common landmark: the center of the L2 vertebra. In other words, the OAR position in LR, AP, or IS direction is a signed, one-dimensional distance from the center of the L2 vertebra to the center of mass of the OAR's segmentation, measured in mm.

For the datasets concerning the retrieval of a representative OAR segmentation, a measure of segmentation similarity needs to be defined as target variable. Because a measure of similarity is defined between pairs of segmentations, each example needs to be defined over a pair of patients, leading to a total of $\#\text{patients}(\#\text{patients}-1)/2$ training examples. For the features, we use a pairwise version of the features defined as the absolute difference of the features of patients p and q: $x_{p,q} = |x_p - x_q|$, with x_p being the feature x of patient p. For the target variable, several possibilities to address segmentation similarity have been proposed in literature. For example, a possibility is to consider similarity in OAR weight[8]. We do not rely on OAR weight nor volume because they do not account for similarity of shape. Another commonly adopted option to assess OAR similarity is the use of the volumetric Dice-Sørensen Coefficient (DSC) [14, 21, 22]. However, the DSC still has limitations, because it is segmentation-volume dependent. We therefore rely on the recently introduced surface Dice-Sørensen Coefficient (sDSC) [23], which considers only significant millimetric deviations between the surfaces of the segmentations to evaluate similarity. In particular, the sDSC uses a threshold parameter τ that expresses what deviations are acceptable (e.g., as part of inter-observer variability). To choose τ, we consider that the median CT slice thickness in our database is 2.5 mm. Since we deal with inter-patient OAR segmentations, we doubled this value, and adopted $\tau = 5.0$ mm. Because we predict OAR positions separately for each OAR, the sDSC is computed after aligning the segmentations on their center of mass, i.e., to maximally focus on the shape. Furthermore, we consider sDSC values as percentages (0 to 100).

Loss function for learning

As loss function to train and validate the ML algorithms we consider the Mean Absolute Error (MAE), i.e.,

$$\mathrm{MAE}(y, \hat{y}) = \sum_{i=1}^{n} |y_i - \hat{y}_i|, \tag{7.1}$$

where y and $\hat{y}$ are n-dimensional vectors of ground-truth values and ML predictions respectively. Clearly, the MAE needs to be minimized. We choose to use the MAE because it is relatively simple to interpret, as it preserves the unit of measurement of the metric at hand (mm for OAR positions, % for sDSC between OAR segmentations), and weighs all errors equally.

For OAR positions, computing the MAE is straightforward both at training and at test time. Given n cases, it suffices to measure the absolute difference between each ground-truth position and its prediction, and take the average upon all examples.

The sDSC is defined between pairs of patient OAR segmentations. Here, ML algorithms predictions correspond to estimations of sDSC between OAR segmentation pairs, and MAE minimization means to correctly predict the sDSC. At training time, the MAE between predictions and ground-truth sDSC is measured. The measurement of validation MAE is more involved. In particular, when validation is needed for a patient who was not included in the training data, i.e., to obtain a segmentation for a test patient, we firstly create pairwise features (Table 7.1) between that patient and each other patient in the database (as absolute feature differences as described in 7.2.3). Subsequently, the model uses the pairwise features as input, and produces predictions of sDSC between the OAR segmentation of the test patient and the one of each other patient (see the input of $\mathcal{M}_{\mathrm{S}}^{\mathrm{OAR}}$ and $\mathcal{M}_{\mathrm{S}}^{\mathrm{Body}}$ in Fig. 7.3). Now, we do not compute the MAE between these predictions and the ground-truth sDSCs as test error, because the predictions do not correspond to any particular segmentation in the database. Rather, they give an indication of how the OAR segmentation of each patient will fare against the OAR segmentation of the test patient in terms of sDSC. Therefore, we proceed by retrieving the segmentation from the database that has the largest predicted sDSC with the test patient. Finally, we evaluate the actual sDSC between these two segmentations, and report that.

Note that the maximum score of OAR shape similarity achievable is not necessarily 100%, rather, it is the maximum sDSC that can be obtained with the OAR segmentations available in the database. To frame the future comparisons that involve different metrics, i.e., OAR positions and OAR shape, we define $\varepsilon_{\mathrm{sDSC}} = 100 - \mathrm{sDSC}$, which is minimal (0%) when the sDSC is maximal (100%), and is maximal (100%) when the sDSC is minimal (0%). This way, both errors in OAR positioning and in OAR shape retrieval can be considered as a metric to be minimized.

Algorithms

We compare a total of five regression ML algorithms: Least-Angle RegreSsion (LARS) [24], LARS using the Least Absolute Shrinkage and Selection Operator (LASSO) [25], Random Forest (RF) [26], traditional Genetic Programming (GP-Trad) [27, 28], and the Genetic Programming instance of the Gene-pool Optimal Mixing Evolutionary Algorithm

(GP-GOMEA) [29, 30]. These algorithms are interesting because they include hyper-parameters that can be used to prevent overfitting, which is a likely scenario when the data is limited as in many medical applications, including ours.

LARS and LASSO are widely used approaches that work based on the assumption that the features can be combined linearly, and include penalization metrics to avoid overfitting. LASSO also performs feature selection. Both algorithms build models that can be read as mathematical expressions consisting of a linear combination of the features, and are therefore considered interpretable [31]. We optimize the hyper-parameter λ, i.e., the penalization factor, as described in Section 7.2.3. We use the implementation of the package glmnet, in R [32, 33].

RF is also widely used, but it is different from the previous two as it allows non-linear modeling. RF builds a model as an ensemble of decision trees to reduce variance in estimation and thus overfitting. Because the model is an ensemble, it is considered not interpretable [31]. We optimize how many features are randomly chosen when building the nodes that compose each decision tree (*mtry*), and the minimum number of data samples a node should represent (*min. node size*), the same way we optimize λ for LARS and LASSO (see Sec. 7.2.3). We use the R implementation known as ranger [34].

The Genetic Programming algorithms are interesting because, like LARS and LASSO, they deliver models in the form of mathematical expressions, but they can include non-linear feature combinations. These algorithms work by loosely mimicking the concept of Darwinian evolution, i.e., by iterative recombination of candidate mathematical expressions made of atomic functions, and selection of the fittest. Recombination in GP-Trad is highly stochastic, whereas the one of GP-GOMEA includes information theory-based mechanisms to estimate what patterns of atomic functions should be preserved during recombination attempts. We found GP-GOMEA to work particularly well when small, yet accurate expressions are needed (Chapter 3). To reduce the number of hyper-parameters, we use these algorithms within a scheme of interleaved runs of increasing capacity (described in sec 7.2.3). We use the C++ implementation of GP-Trad and GP-GOMEA (`https://github.com/marcovirgolin/GP-GOMEA`).

Hyper-parameter settings

Table 7.2 shows the hyper-parameters used by the ML algorithms. LARS and LASSO use λ to penalize complex models. For RF, we use the default relatively large number of trees (given the datasets at hand) of 500 [35].

For GP-Trad and GP-GOMEA, the function set $\mathcal{F}$ defines which functions to use as model components (tree nodes). The division operator $\div_{\mathrm{A}}$ is the analytic quotient [36], which not only guarantees that the divider can never be null, but also ensures smoothness (in contrast with the protected division operator [28], which can harm generalization [36]). The logarithm operator is protected to avoid infeasible computations [27]. The Ephemeral Random Constants (ERC) are constants for which the value is set by uniform sampling from a defined interval [28]. Mathematical expressions are encoded as parse trees in GP-Trad and GP-GOMEA. We set a small tree height to keep the resulting mathematical expressions short and readable (we found that larger three heights can result in hard to read expressions), and to prevent overfitting.

The number of candidate expressions to evolve, i.e., the population size, is a sensitive parameter for GP algorithms. We run GP-Trad and GP-GOMEA using the Interleaved Mul-

Table 7.2: Hyper-parameters of the ML algorithms. The subscript "tune" means that the hyper-parameter setting is subject to optimization with 5-fold cross-validation grid-search among the listed values.

Algorithm	Hyper-parameter	Settings
LARS and LASSO	λ_{tune}	$10^{-10}, 10^{-9}, \ldots, 10^{10}$
RF	nr. trees	500
	min. node size$_{\text{tune}}$	$5, 10, \ldots, 20, 25$
	mtry$_{\text{tune}}$	$1, 2, \ldots, \frac{\#\text{features}}{2}$
GP-Trad and GP-GOMEA	$\mathcal{F}$	$\{+, -, \times, \div_{\text{A}}, \exp, \log_{\text{P}}\}$
	ERC	$\mathbb{U}[-10, 10]$
	tree height	2
	g_{IMS}	4
	time limit	60s

tistart Scheme (IMS), a method that interleaves multiple runs with increasing population size. We set the number of sub-iterations between runs, g_{IMS}, to 4 as has been reported to work well on benchmark problems (Chapters 2 and 3). Since the IMS can in principle run forever, we set a time limit of 60 seconds. We found this limit to be reasonable because the datasets are small and evaluations are fast, and because the other ML algorithms take only a few seconds to execute. We also preliminarily observed that increasing the time limit (e.g., to 5 or 10 minutes) does not alter the results in a significant way. For further details on GP-Trad, GP-GOMEA, the IMS, and other hyper-parameters, the reader is referred to Chapter 3.

7

Learning strategy

With a limited number of 60 patients available, we perform leave-one-out cross-validation to compute the overall test performance. The leave-one-out cross-validation is performed "patient-wise", i.e., all the examples relative to a particular patient p are removed from the training set, and solely used for testing. This is obvious for the datasets on OAR position, as each row corresponds to exactly one patient. For the datasets on OAR segmentation retrieval, however, each row represents a pair of patients (see Sec. 7.2.3). Therefore, all $\#\text{patients} - 1$ (59) rows where p is considered are removed from the training set and put in the test set. This is necessary to avoid a positive bias in the test results.

Within each iteration of leave-one-out cross-validation, for LARS, LASSO, and RF, we perform grid-search hyper-parameter tuning with 5-fold cross-validation upon the training data, to determine the best hyper-parameter values. We use the R package caret for this purpose[37]. Once the best hyper-parameter settings are found, we train the ML algorithm on the training set using those settings, and test it on the test set. For GP-Trad and GP-GOMEA, we take the best expression found by the interleaved runs started by the IMS.

Since RF, GP-Trad, and GP-GOMEA are stochastic algorithms, we repeat their execution 10 times, and report the mean result.

7.2.4. Anatomical inconsistency correction

The last step of our pipeline repairs possible anatomical inconsistencies present in the assembled phantoms. We automatically compute possible overlaps between the transplanted OARs (liver and spleen), and between the OARs and the spinal cord segmentation of the receiver, which is available for these patients. To have an additional margin, we enlarge the spinal cord segmentation by 10% uniformly in all dimensions. Furthermore, we assess if the transplanted OARs stick out of the segmentation of the body of the receiver. If a larger segmentation of the body exists than the common region of interest between T10 and S1 (used to assess body shape similarity to train ML models), that segmentation is used here. Again, for robustness, the body segmentation is shrunk uniformly in all dimensions, by 2.5%. The values chosen for spinal cord segmentation expansion (10%) and body segmentation shrinking (2.5%) were found to deliver pleasing results by visual inspection.

We use a general purpose, derivative-free real-valued optimization algorithm to modify the transplanted OAR segmentations to eliminate the anatomical inconsistencies, with default parameter settings [38, 39]. The algorithm is configured to act on both liver and spleen at the same time, with modifications (i.e., optimization variables) that can expand or shrink the segmentations (up to 1.25 and 0.25 of the original volume respectively), as well as reposition their center of mass along LR, AP, IS (up to 10 mm for each direction). We set anatomical inconsistencies as hard constraints to satisfy. Modifying the OARs to satisfy the constraints deviates from the predictions of the ML models, and can therefore result in less accurate phantoms. Therefore, we use an objective that conflicts with the constraints, in that it attempts to minimize the effect of OAR modifications: for each OAR being corrected, we use the same objective of the ML algorithms to capture the shape of OARs, i.e., the sDSC (again with threshold of 5 mm), this time compared to its originally-predicted shape and position. The final objective is given by summing the sDSC of both liver and spleen, to be maximized.

7.2.5. Comparing to phantom selection approaches

As we mentioned in the introduction, it is common practice in phantom-based dose reconstruction to select a phantom from a library according to some criteria. As CTs of actual patients can act as phantoms, we consider several approaches to compare with our pipeline: two methods that simulate state of the art human-designed criteria used to build and select from phantom libraries, random selection, and one method to select a single CT from our database based on ML predictions.

Human-designed criteria for phantom selection

The first methods we consider are the criterion used by the University of Texas MD Anderson Cancer Center [4, 5], which we refer to as Human Criterion 1 (HC1), and the criterion used by the University of Florida/National Cancer Institute [6], which we refer to as Human Criterion 2 (HC2). We further consider random selection, to see if the other methods are better than random.

The phantoms on which HC1 has previously been used are virtual cuboid shapes with OARs represented as point clouds [4, 5]. Only the age of the patient is considered as a feature to manipulate the phantom's representativeness by scaling the cuboids according to guidelines on population data. Furthermore, gender is used to exclude/include gender-

specific OARs. To simulate HC1, i.e., age binning, we cluster our database into age bins, by rounding the age to years. This results in 5 bins, with the following distribution: 10 patients of age 2, 18 of age 3, 11 of age 4, 15 of age 5, and 6 of age 6.

For a test patient p, we consider the other patients from the database that share the same age bin with p. Then, for each metric of interest, we report the average error given by comparing the metric value for p with the one for each other patient that has the same age as p. For example, for the assessment of liver segmentation similarity of a patient p that is 3 years old, we compute the sDSC between p and each other patient that is 3 years old, and return the mean. By doing this, we simulate the fact that an average anatomy has been built using the anatomical information of all patients (different from p) that have the same age. We do this because phantoms are built to represent average anatomies.

The phantom library where HC2 is adopted comprises phantoms made by scaling segmentations acquired from actual patient CT scans, thus they are quite realistic [6]. For these phantoms, the features considered to build the library were gender, height, and weight (age was not used). HC2 uses these features to select a representative phantom. We simulate HC2 by clustering our database by gender, and by height and weight, using 5 bins for each of the latter two. We use the bin method of the R package *binr* (`http://jabiru.github.io/binr`) for this purpose, with default settings, which results in 5 bins of 12 patients each. Like for HC1, the error for a metric is given by comparing the metric value for the test patient p with the metric values of each patient that shares the same bin of p, and taking the average.

To assess whether there is any merit in using the human-design criteria, we also consider a control method where the OAR positions and segmentations are retrieved uniformly at random from the set of patients, excluding the test patient p. Because of the stochastic nature of this approach, each iteration is repeated 10 times, and mean results are computed. In the following, we refer to this method as RAND.

As we propose a pipeline to build a phantom, it is interesting to assess how it fares against a simpler approach, e.g., to use ML predictions to select a single, overall most representative CT scan. This approach can be related to literature, where new ways to identify which phantom to pick are studied [14, 40, 41].

While our phantom construction pipeline can predict the different 3D metrics independently (position for each direction, sDSC for each OAR), for a single CT approach, an overall score needs to be defined that expresses how representative a CT scan is. As discussed in Chapter 6, the design of such a score is not trivial. For example, a choice needs to be made on whether 5 mm along the LR direction is more important or less important than an sDSC loss of 5%. For the sake of simplicity, we propose a score measure that considers only OAR positions, with equal importance. We choose to focus on OAR position because we recently found it to be the metric that is most correlated with dose accuracy for pediatric abdominal radiation treatment [42]. In detail, we take as best CT the one that is closest to the predictions of the ML models in terms of squared position differences, i.e.,

$$\text{Best CT} = argmin_{\text{CT}} \sum_{\mathcal{O} \in \text{OARs}} \sum_{\mathcal{D} \in \{\text{LR,AP,IS}\}} \left(\mathcal{D}^{\mathcal{O}}_{\text{ML}} - \mathcal{D}^{\mathcal{O}}_{\text{CT}}\right)^2, \tag{7.2}$$

where $\mathcal{D}^{\mathcal{O}}_{\text{ML}}$ and $\mathcal{D}^{\mathcal{O}}_{\text{CT}}$ are respectively the ML-predicted position and the actually available position in the CT, of the OAR $\mathcal{O}$, along direction $\mathcal{D}$. We denote this approach with sCT.

Table 7.3: Mean training and test MAEs for the ML algorithms on the different OAR-specific regression tasks. Standard deviation is reported in subscript. The MAE for OAR segmentation retrieval is a percentage, the MAE for OAR position estimation is in mm. Results in bold are best in that no other method delivers significantly better ones. The letter "S" stands for segmentation retrieval.

		Body	Liver				Spleen				#Best
		S	LR	AP	IS	S	LR	AP	IS	S	
Training	LARS	16.63 0.08	7.89 0.24	4.27 0.14	7.10 0.33	17.02 0.07	**3.99** 0.21	7.21 0.13	7.71 0.19	15.39 0.10	1
	LASSO	16.64 0.08	7.87 0.36	4.24 0.10	7.28 0.19	17.02 0.07	4.03 0.10	7.24 0.13	7.60 0.13	15.38 0.09	0
	RF	**6.74** 0.33	8.36 0.14	4.87 0.08	8.74 0.09	**12.37** 1.08	5.44 0.07	7.25 0.14	9.33 0.14	**7.21** 0.83	3
	GP-Trad	19.55 0.06	**6.97** 0.11	**3.93** 0.07	6.42 0.10	15.97 0.03	4.01 0.05	6.56 0.13	7.06 0.15	14.08 0.04	2
	GP-GOMEA	19.55 0.06	**6.97** 0.11	**3.93** 0.07	**6.38** 0.09	15.96 0.03	**3.95** 0.05	**6.47** 0.12	**7.05** 0.15	14.07 0.04	6
Test	LARS	23.70 15.31	8.54 6.83	**4.82** 4.57	9.10 5.33	**38.90** 17.84	5.18 3.33	**7.42** 8.06	8.52 7.48	35.12 14.51	3
	LASSO	22.78 15.34	8.87 6.91	**4.86** 4.49	8.98 5.37	**38.98** 19.11	5.02 3.21	**7.37** 8.10	8.68 7.47	36.82 13.59	3
	RF	**21.38** 14.59	8.36 6.55	**4.77** 4.62	7.94 5.73	41.12 16.72	4.98 3.48	7.83 8.00	8.80 7.74	**33.98** 15.47	3
	GP-Trad	27.08 16.67	**7.27** 6.48	**4.68** 4.30	**7.23** 5.89	**40.61** 14.25	5.23 3.40	8.35 8.22	8.43 9.73	35.70 10.57	4
	GP-GOMEA	26.77 16.75	**7.26** 6.48	**4.78** 4.36	7.89 6.03	**39.00** 19.31	**4.56** 3.49	**7.33** 8.32	**8.21** 9.78	35.62 11.43	6

Note that sCT is essentially a hybrid between a human-designed criterion, represented by Equation 7.2, and the use of ML, of which the predictions are used in the equation. Lastly, since we found GP-GOMEA to be the overall best performing ML algorithm (shown later in Sec. 7.3.1), we used the models found by GP-GOMEA to provide the predictions for sCT.

7.2.6. Experimental setup

We divide experiments into two parts. In the first part, we compare the predictions of the ML algorithms, in terms of MAE. In the second part, we compare the prediction of the overall best performing ML algorithm with the phantom selection approaches, in a similar way. In this second comparison, we include the effect of anatomical inconsistency correction, to assess how much it compromises the accuracy of the predictions of the ML models.

We run the ML algorithms on a machine with two Intel® Xeon® CPU E5-2699 v4 @ 2.20 GHz. We assess statistical significance of the results with the Wilcoxon signed-rank test, paired by train-test split (i.e., held-out patient) [43], and using the Bonferroni correction method to prevent type I errors [44]. In particular, since we perform pairwise tests between the algorithms for each metric, we assess whether the test p-value is below a confidence level of 0.05, further reduced by a Bonferroni correction coefficient, to contrast false positive outcomes due to chance.

7.3. Results

7.3.1. Comparison of the machine learning algorithms

The mean (and standard deviation) training and test MAE obtained by the leave-one-out cross-validation of the ML algorithms are reported in Table 7.3. If multiple results are not found to be statistically significantly worse than the best result, they are considered to be equally good. Note that for consistent use of the MAE, to be minimized, the task of segmentation retrieval uses $\varepsilon_{\text{sDSC}}$.

In terms of training performance, GP-GOMEA is overall the best algorithm, as it is not significantly worse than any other for 6 metrics, i.e., OAR position for all directions. RF follows with 3 top performances, in particular for the segmentation retrieval task (S) of all OARs, where it achieves markedly lower errors than all other algorithms. LASSO performs worst, with at least one algorithm significantly outperforming it in each metric.

Regarding the test performance on the patient excluded by the leave-one-out cross-validation, similar to the observed training performance, GP-GOMEA obtains the most significantly best results. GP-GOMEA also generalizes well on predicting which segmentation to select for the liver, but it is inferior to RF when it comes to selecting the segmentations for the body and the spleen. Indeed, for those two segmentations, RF performs better than any other ML algorithm, although the errors at test time are much larger than the ones found at training time. Note that this mismatch between training and test performance can be explained by possible overfitting, as well as by the fact that, at test time, error propagation can happen, since the model prediction is used to retrieve a candidate segmentation from the ones available in the database (this holds for all ML algorithms, see Sec. 7.2.3). GP-Trad scores an extra point compared to RF in terms of number of metrics where it is not outperformed significantly. LARS and LASSO generalize quite well at test time, as they are not inferior to the other ML algorithms on (the same) 3 metrics.

7.3.2. Machine learning vs. phantom selection approaches

Table 7.4 shows results of the ML algorithm found to perform overall best, GP-GOMEA, and of the use of phantom selection approaches. The use of anatomical inconsistency correction upon GP-GOMEA's predictions is also included (named GP-G_{AIC}), to make the pipeline construct pseudo-realistic phantoms. Note that since the use of automatic inconsistency correction does not adapt the body segmentation, there is no difference between GP-GOMEA and GP-G_{AIC} in terms of error for body segmentation retrieval.

Out of the nine metrics, the correction-less predictions obtained with GP-GOMEA are generally best, with the only exceptions being the IS position and segmentation retrieval for the spleen (by relatively small errors on average). The use of anatomical inconsistency correction upon GP-GOMEA's predictions, GP-G_{AIC}, can change shape and position of the OARs in both considerable (e.g., liver AP and liver S) and minor (e.g., liver LR, spleen IS) magnitude. Figure 7.4 shows box plots of the corrections on all phantoms. For our database, the anatomical inconsistency correction was triggered for 31/60 phantoms. The liver is subject to more corrections than the spleen: it is typically shrunk more than the spleen, and its position in AP is subject to large variations. This is not surprising, because the liver is a considerably larger organ than the spleen, and it is more likely to violate the anatomical consistency constraints we imposed.

Overall, correcting for inconsistencies comes at the cost of worsening the accuracy of the ML predictions. GP-G_{AIC} has significantly best performance only on 3 metrics (body S, spleen AP, and spleen S). For spleen AP and spleen S, performing inconsistency correction leads to better test results compared to not applying corrections. However, the correction algorithm solely optimizes for resolving the inconsistencies while attempting to retain maximum prediction fidelity. Moreover, as sCT also obtains good results on spleen AP and S, it can be argued that these metrics are not modeled well by GP-GOMEA to begin with, and thus are more likely to be improved upon by chance.

Table 7.4: Mean test MAEs of the overall best performing ML algorithm GP-GOMEA, also including anatomical inconsistency correction (GP-G_{AIC}), and of the phantom selection approaches. Standard deviation is reported in subscript. The MAE for OAR segmentation retrieval is a percentage, the MAE for OAR position estimation is in mm. Results in bold are best in that no other method delivers significantly better ones. Results underlined are best if GP-GOMEA is excluded from the comparison. The letter "S" stands for segmentation retrieval.

		Body	Liver				Spleen				#Best
		S	LR	AP	IS	S	LR	AP	IS	S	(Underlined)
Test	GP-GOMEA	**26.77** 16.75	**7.26** 6.48	**4.78** 4.36	**7.89** 6.03	**39.00** 19.31	**4.56** 3.49	**7.33** 8.32	8.21 9.78	35.62 11.43	7 (–)
	GP-G_{AIC}	**26.77** 16.75	7.91 6.27	6.21 5.42	**8.07** 6.06	42.20 20.29	5.18 4.04	**7.35** 8.30	8.54 9.67	**34.10** 16.97	4 (7)
	HC1	37.54 10.82	8.88 6.90	**4.83** 4.70	9.10 6.13	**43.19** 8.35	5.65 3.68	7.96 7.75	9.78 8.20	37.58 8.53	2 (2)
	HC2	36.83 18.97	9.84 6.26	5.18 4.74	9.20 6.48	49.67 8.2	5.54 4.21	8.02 7.69	13.91 11.09	34.76 10.37	0 (0)
	RAND	45.10 13.14	11.57 6.19	7.11 4.01	12.38 4.48	44.01 8.96	7.70 3.29	11.14 7.22	13.52 7.05	35.89 8.11	0 (0)
	sCT	37.13 22.00	15.29 10.02	7.44 5.91	11.45 9.2	42.72 18.30	4.85 3.43	**7.40** 8.32	**7.84** 9.37	**32.08** 12.28	3 (5)

Although the use of anatomical inconsistency correction after ML prediction typically leads to larger errors, it remains a valuable approach compared to the other methods. GP-G_{AIC} is overall best when the correction-less GP-GOMEA predictions are excluded from the comparison. The human-designed criteria HC1 and HC2 perform overall worse than GP-G_{AIC}. Despite its simplicity, HC1 performs well for AP and S of the liver. This may be because metrics related to the liver are harder to model by the ML algorithms, and because the use of anatomical inconsistency correction compromises GP-GOMEA's prediction (particularly notable for liver AP). Despite the fact that HC2 is a somewhat more involved criterion compared to HC1 (HC2 considers gender, height, and weight, while HC1 considers only gender and age), it is never found to be competitive on any metric. This could mean that weight and height are not good features when accounting for similarity of internal anatomy in children. This result is in agreement with previous work [42] where lack of correlations between dose reconstruction outcomes were found with respect to height and weight.

Importantly, in some cases, HC1 and HC2 perform particularly bad. HC1 is particularly inaccurate for spleen S compared to the other approaches, and HC2 leads to notably large errors for spleen IS. In the latter case, HC2 is not found to be better than RAND. Furthermore, both criteria perform poorly on body S (as well as sCT). Figure 7.5 shows the reliability function of the considered approaches for body S. This function shows the likelihood of committing errors of a certain magnitude (for segmentation retrieval, in percentage). A curve is better than the others if it is more on the left and if it decreases more rapidly, since this means that the probability of any error magnitude is lower than the ones of the other methods. The figure illustrates that the model learned by GP-GOMEA achieves this behavior. For example, the probability of a GP-GOMEA's model to predict a body segmentation that has an $\varepsilon_{sDSC} > 20\%$ (sDSC $< 80\%$) with the actual body of the patient is 60%. For HC2 and sCT, that magnitude of error happens more frequently, i.e., in almost 80% of the cases. HC1 performs worse, since errors above 20% are almost certain. On the other hand, rarely, (probabilities around 5%), HC2 and sCT can retrieve body segmentations that have errors above 80%.

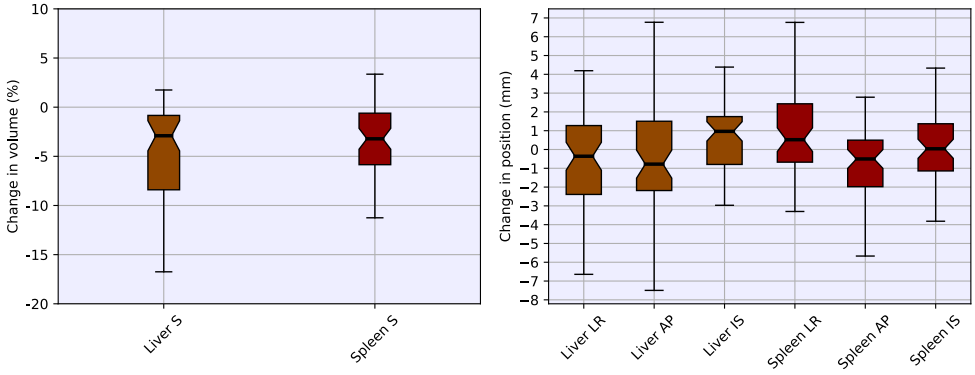

Figure 7.4: Distribution of the effect of the anatomical inconsistency correction on all phantoms (29/60 are not corrected). OAR shape (S) is corrected by volume enlargement (if $> 0\%$) or shrinking (if $< 0\%$) uniformly along the three dimensions. Change of center of mass for AP, LR, and IS is in mm. Phantoms where correction is not needed, contribute to OAR shape modification with 0% (no enlargement nor shrinking), and to OAR position change with 0 mm (no re-positioning). Boxes extend from the 25th to the 75th percentiles, inner bar is the median, and whiskers extend from the 10th to the 90th percentiles.

Regarding sCT, the results indicate that this approach is generally worse than GP-G_{AIC}, but better than the human-designed criteria to predict metrics related to the spleen. sCT performs particularly bad with respect to metrics regarding the liver (note in particular liver LR). This is an interesting result because the CT selected by sCT weighs OAR positions equally for liver and spleen (see Eq. 7.2). We remark that sCT uses the ML models (found by GP-GOMEA) to predict which single CT scan (and accompanying segmentations) to select by means of a hand-designed metric. Essentially, the results confirm our hypothesis that designing a good score to select one CT is not trivial, and that there is added value in constructing a new anatomy by assembling different components into one.

7.3.3. Model interpretability

As we mentioned before, the added value of utilizing the linear ML algorithms (LARS and LASSO), and the GP algorithms (GP-Trad and GP-GOMEA) is that their models are mathematical expressions which, if simple enough, can be interpreted. Interpretability can play a crucial role in determining whether ML can be applied in some clinical settings. For RF, the interpretation of the model is essentially impossible, as it returns an ensemble of (500) decision trees. Similarly, this would not be possible with other popular techniques like deep learning [45], and boosting algorithms [46].

We report all the best-at-test-time models found by GP-GOMEA at `http://bit.ly/2Za4ESy`. Here, we report two of those models, that are remarkably simple. For the prediction of which body segmentation to retrieve, the model found most frequently in 10 repetitions is (assuming the target variable and the features are normalized):

$$0.420 \times (\mathbf{ADAP} + \mathbf{ADLR} + \mathbf{SCIS})\,. \tag{7.3}$$

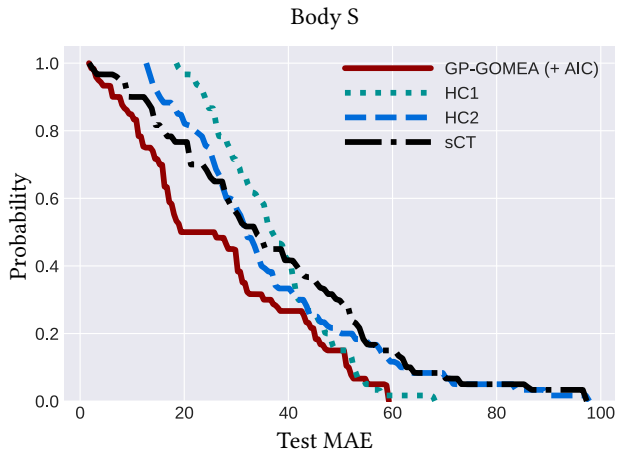

Figure 7.5: Reliability functions for prediction of segmentation retrieval (S) of the external body segmentations. The y value is the probability of committing an error equal or greater than the x value. The test MAE is in $\varepsilon_{\text{sDSC}}$. Note that the anatomical inconsistency correction does not alter the body segmentation. GP-GOMEA leads to much better performance compared to the other approaches.

Notably, this model is just a scaled sum of the abdominal diameters (ADAP and ADLR) and of the spinal cord length (SCIS). Essentially, the model uses an equal contribution (features are normalized) of features capturing information of size relative to the three dimensions LR, AP, and IS, to predict which body segmentation to retrieve. This seems a reliable, simple, and reasonable method to select a representative body segmentation.

A second model we showcase here is the one for the prediction of what spleen segmentation to retrieve:

$$2.718^{\textbf{AGE}} \times 0.057 \times \textbf{SCIS}. \tag{7.4}$$

It is interesting to see that spleen shape is found to be related to age by an exponentiation. This can be considered reasonable for our cohort, because the age of our patients is between 2 to 6 years, where anatomical development is rapid. The length of the spinal cord in IS further weighs the prediction. This feature seems also reasonable to consider, because it captures information relative to the size of the abdomen in IS, and because the spleen is located nearby the spinal cord. Albeit understandable, reasonable, and well-performing, it is arguably unlikely for humans to invent models like these.

7.3.4. Examples of automatically constructed phantoms

Following the quantitative results presented in the previous sections, we present some qualitative ones: examples of constructed phantoms. These qualitative results complement the quantitative ones, as the positioning of OARs and their shapes can be visually evaluated in the context of the receiver CT.

Figure 7.6 shows 5 phantoms generated with our pipeline, where GP-GOMEA was

used to train all models. The images are created by using 3D Slicer [17] with module SlicerRT [47]. These phantoms have been generated for a test patient that was excluded from the ML training process. The predicted liver and spleen segmentations are highlighted in the receiver CT. Note that the segmentations of the external body and of the spinal cord shown in the figure can extend beyond the commonly available region of interest used for training (from T10 to S1). The receiver CT's original liver and spleen can also be identified (in part), as their not-overridden voxels are uniformly set to a specific value (78 HUs as done for the phantoms of the University of Florida/National Cancer Institute [6]) in the resection step. Overlaps of the transplanted OAR can still happen with respect to other organs which are not considered in the training and correction process (e.g., spleen overlapping with the kidneys). Note also the presence of some further remaining limited anatomical inconsistencies, which are not detected by our algorithm (e.g., small overlaps with bony anatomy).

Our current Python implementation takes about 5 minutes to generate a phantom if no anatomical inconsistencies are found, with resection and transplant taking the most time (we expect that parallelizing voxels' HU value overwrites will markedly reduce running time). If anatomical inconsistency correction needs to be performed, the whole pipeline can take from a few minutes to a few hours, depending on how complex the correction is for the optimizer, and on the hardware (the optimizer is highly parallelizable). As mentioned in Section 7.3.2, for our database the anatomical inconsistency correction triggered on half the phantoms. However, in our opinion only roughly half of those cases (so 1/4 of the total number of phantoms) had really noticeable anatomical inconsistencies, e.g., large OAR overlaps, or OARs exceeding the body boundaries, that required corrections of large magnitude. In the other cases, inconsistencies were more subtle (e.g., small OAR overlap), and caused corrections of small magnitude.

Figure 7.7 shows two examples of anatomical inconsistency correction for cases where inconsistency can be considered large, displaying phantoms pre- and post anatomical inconsistency correction, one to correct the liver, and one to correct the spleen.

7.4. Discussion

We have presented a new take on phantom construction: a fully-automatic ML-based pipeline that assembles a patient-specific phantom using a database of delineated 3D CT scans, given the features of a patient. We performed experiments upon data of 60 pediatric patients including imaging of the abdomen, and focused on tailoring the position and shape of the liver and the spleen. Our experimental results strongly suggest that our approach leads to much more representative phantoms than using established human-designed criteria, and than using ML to predict a single best CT scan (according to a reasonable notion of overall anatomical similarity). We compared several ML algorithms to provide accurate models for the pipeline, and found GP-GOMEA to deliver overall best performance and models that can also be interpreted, which may be helpful for researchers and clinicians alike to trust their use in a clinical setting.

One clear limitation of our work is that we could only employ a small database of 60 patients. It can be reasonably expected that, by increasing the database size, both the errors of the ML models, and the onset of large anatomical inconsistencies, will be reduced. To this end, we are currently working on expanding the number of institutes contributing

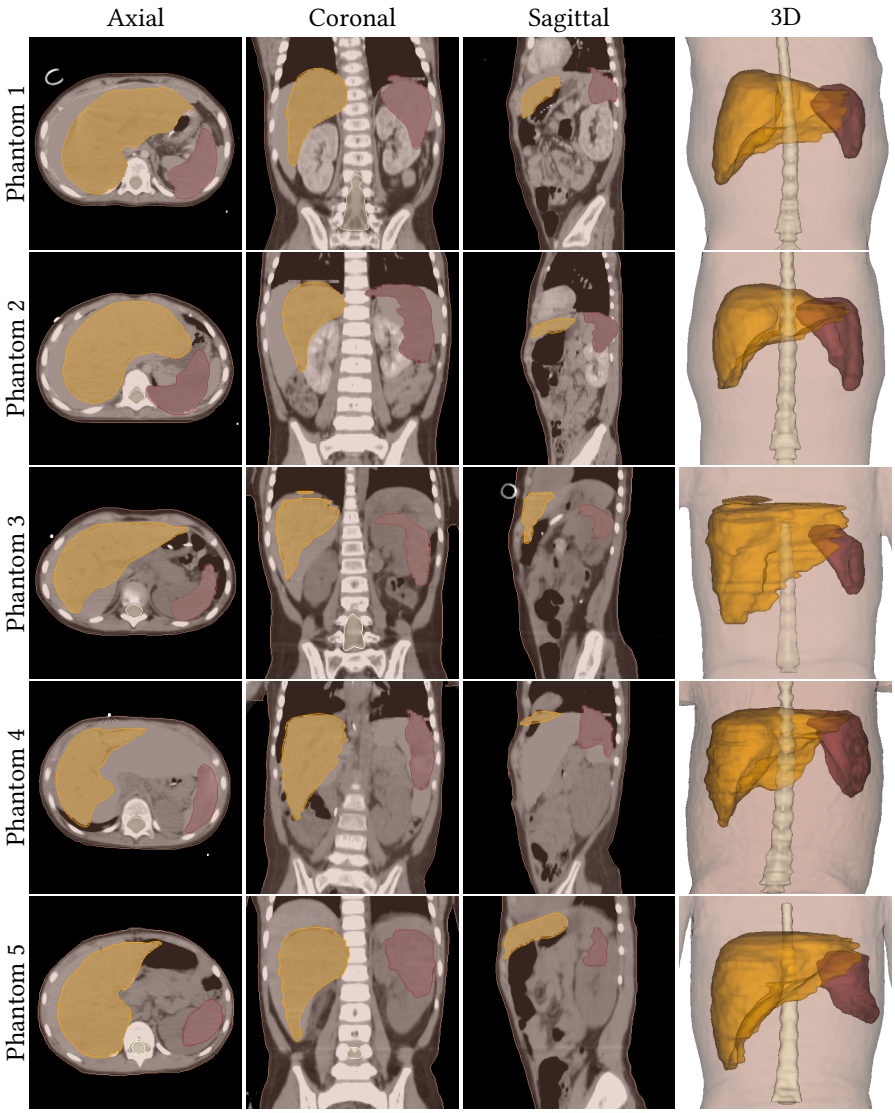

Figure 7.6: Examples of phantoms constructed with our pipeline. CT snapshots are chosen to attempt to display both liver (in ocher) and spleen (in crimson red). Axial views are in IS, coronal views and 3D views are in AP, sagittal views are in LR.

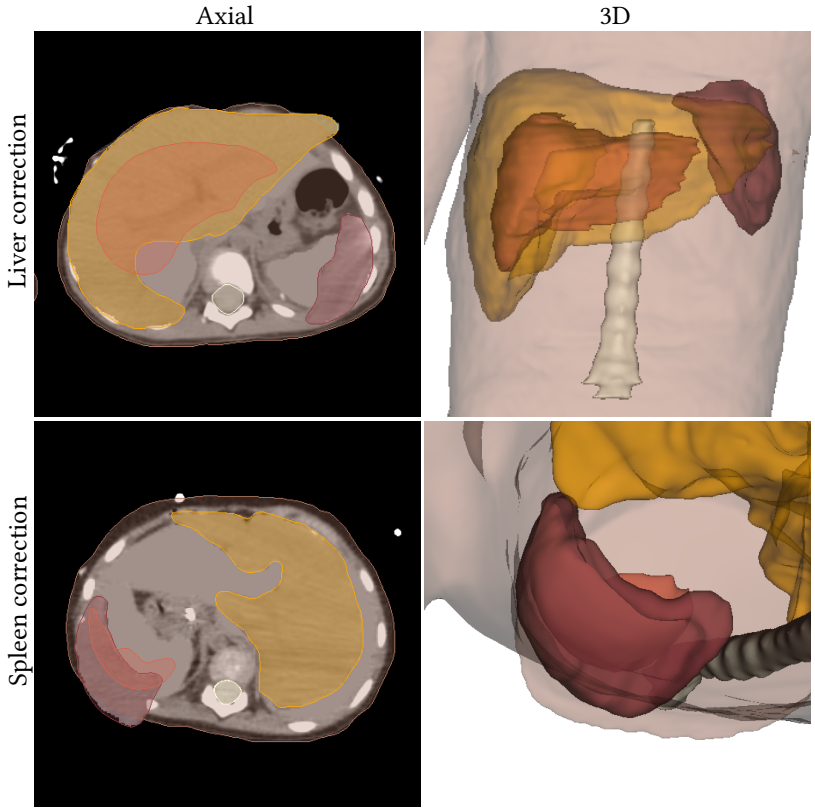

Figure 7.7: Examples of anatomical inconsistency correction tackling OARs being partly positioned outside of the body segmentation. Pre-correction liver is in ocher, post-correction in orange; pre-correction spleen is in crimson red, post-correction is in salmon pink. in Top: Axial (left) and 3D (right) view of large anatomical inconsistency involving the liver. Bottom: Axial (left) and 3D (right) view of large anatomical inconsistency involving the spleen. Note that this axial view is different from the ones in other figures as it is taken in superior-inferior direction (spleen displayed on the left), to be consistent with the 3D visualization. The 3D view shows the back of the patient from head to toes.

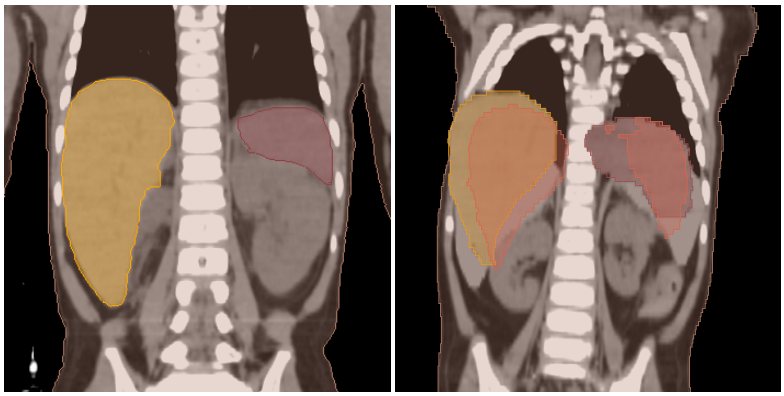

Figure 7.8: Example of the limitations of anatomical inconsistency correction when applied to a coarse prediction of the body segmentation. Colors as in Fig. 7.7. Left: Actual anatomy of the patient. Right: Proposed phantom, including corrections. The receiver CT is quite smaller than the actual CT in IS dimension. The anatomical inconsistency correction shrunk and relocated the liver that was exceeding the body boundaries, and moved the spleen away from the spinal cord. However, liver and spleen are placed too high with respect to the underlying anatomy for it to be realistic.

to the database. Furthermore, we are working on extending the number of OARs to build the phantom (e.g., heart, kidneys) and on extending the region of interest (e.g., abdomen and thorax), for a cohort including older children (2 to 8 years).

The automatic anatomical inconsistency correction method may still warrant further improvement. Although we could always resolve the constraint, often without excessively compromising the predictions of the ML models, anatomical inconsistencies can still be present. This is because the constraints are not comprehensive enough. For example, Figure 7.8 shows a phantom for which our correction does not violate any of the specified constraints (liver and spleen do not overlap with each other nor with the spinal cord, and the organs do not exceed the body contour) yet the anatomy remains unrealistic because the organs are placed too high with respect to the receiver CT. In fact, this is because it is the prediction of the receiver CT that does not work well for this patient, as it retrieves a body which is quite shorter (14 cm smaller SCIS, sDSC of 60%) compared to the actual body of the patient. Fortunately, this is the only phantom out of 60 where such an evident inconsistency was found and likely the probability of this happening only reduces with a growing database size. Still, we intend to study how to further improve our correction method by attempting to craft more constraints, as well as by including more OARs (e.g., the lungs), which then will need to not overlap with each other.

We remark that, in general, phantoms do not need to be anatomically realistic for the purpose of dose reconstruction, especially when doing large dose reconstructions of populations. For example, as aforementioned, the University of Texas/MD Anderson Cancer Center uses virtual cuboid phantoms uniformly made of water-equivalent material, where internal organs are represented by point clouds [4, 5]. However, realistic surrogate anatomies have arguably a larger chance of being considered familiar and trustworthy by

clinicians, whenever e.g., an expert opinion is needed to go beyond simple statistical measurements, and really visualize how the radiation dose distribution may impact healthy tissues.

Another limitation of this study is that we used HC1 and HC2 to simulate the process of phantom selection in state of the art phantom libraries on our own database of CT scans and organ segmentations, rather than using phantom libraries. Indeed, the phantoms in those libraries are built to follow statistics relative to large populations. We did investigate the use of the library of the University of Florida/National Cancer Institute, which is of public access, instead of our database, for HC2. We however found that selecting those phantoms leads to no better results than using our database (the body segmentation was not considered because a region of interest between T10 and S1 is not readily available for those phantoms). In particular, it was found to be significantly better for half of the 3D metrics (LR and S for the liver, LR and IS for the spleen), but worse for the other half. We also found the use of the actual library to be always significantly worse than using GP-GOMEA, with exception for LR of the spleen, where it was equivalent. This may be because the statistics on which those phantoms are based, which come from the United States, are not accurately representing the patients of our cohort, who are Dutch.

In future work we will consider porting our approach to different types of regions of interest (e.g., head for brain tumors) and cohorts (e.g., older patients). Ultimately, we are interested in generating an entire body anatomy for any patient. We believe our approach is very promising because once appropriate features are defined, no modifications in how to train ML algorithms, nor in how the pipeline works, are needed to obtain new phantoms. Moreover, since the availability of different OAR segmentations is currently limited to what is available in the database, and since delineating new OARs requires specialization, experience, and time, ways to use ML to deform an existing OAR template into a patient-specific segmentation could be worth investigating [48].

Finally, for the aim of obtaining 3D dose distributions to relate to the onset of adverse effects, it will be important to validate our pipeline in terms of dose reconstruction accuracy, i.e., by first crafting a highly individualized phantom, and then simulating the radiation treatment plan using such a phantom. This could be performed similarly to our validation analysis, i.e., with cross-validation on recent patients. Dose metrics should be computed on the actual CT, and then compared with the dose metrics computed on the phantom.

7.5. Conclusion

We have presented a new take on phantom construction that leverages machine learning to assemble existing 3D patient imaging into a new anatomy. Contrary to existing approaches, the pipeline we propose requires no manual intervention except for the initial effort of assembling a database of 3D patient imaging (CTs, segmentations, and patient features), and the measurement of few features of the historical patients on their radiographs. With our approach the problem of finding a globally good metric to represent anatomical categorization, typically faced by phantom libraries, is shifted to train machine learning models for parts of phantoms based on specific 3D metrics. Our experimental results on a database of 60 pediatric cancer patients, focused on liver and spleen, showed that this approach can lead to significantly better anatomical resemblance compared to the use of

phantom building criteria that are currently common practice. Positive results were still found after correcting the phantoms for possible anatomical inconsistencies through optimization. Regarding the machine learning algorithm used in the pipeline, we found that GP-GOMEA, a state of the art genetic programming approach, can deliver models that are both accurate and readable. This aspect can be of added value as such models increase the chances of clinicians understanding them better and trusting their use.

Acknowledgments

The authors acknowledge the Kinderen Kankervrij foundation for financial support (project #187), and the Maurits and Anna de Kock foundation for financing a high-performance computing system. We thank dr. Brian V. Balgobind, dr. Irma W.E.M. van Dijk, and dr. Jan Wiersma from the department of radiation oncology of Amsterdam UMC, location AMC, Amsterdam, the Netherlands, and dr. Geert O.R. Janssens and dr. Petra Kroon from the department of radiation oncology of UMC Utrecht Cancer Center, Utrecht, the Netherlands, for providing help in the collection and/or in the assessment of the imaging data used in this work. The authors are grateful to Elekta for providing ADMIRE research software for automatic organ segmentation. We further acknowledge dr. Choonsik Lee from the National Cancer Institute, Division of Cancer Epidemiology & Genetics, Rockville, Maryland, U.S.A., for details on the phantom library of the University of Florida/National Cancer Institute.

REFERENCES

[1] C. Lee, J. W. Jung, C. Pelletier, A. Pyakuryal, S. Lamart, J. O. Kim, and C. Lee, *Reconstruction of organ dose for external radiotherapy patients in retrospective epidemiologic studies,* Physics in Medicine & Biology **60**, 2309 (2015).

[2] X. G. Xu, *An exponential growth of computational phantom research in radiation protection, imaging, and radiotherapy: a review of the fifty-year history,* Physics in Medicine & Biology **59**, R233 (2014).

[3] J. V. Bezin, R. S. Allodji, J.-P. Mège, G. Beldjoudi, F. Saunier, J. Chavaudra, E. Deutsch, F. de Vathaire, V. Bernier, C. Carrie, *et al.*, *A review of uncertainties in radiotherapy dose reconstruction and their impacts on dose–response relationships,* Journal of Radiological Protection **37**, R1 (2017).

[4] M. Stovall, S. S. Donaldson, R. E. Weathers, L. L. Robison, A. C. Mertens, J. F. Winther, J. H. Olsen, and J. D. Boice Jr, *Genetic effects of radiotherapy for childhood cancer: Gonadal dose reconstruction,* International Journal of Radiation Oncology · Biology · Physics **60**, 542 (2004).

[5] R. M. Howell, S. A. Smith, R. E. Weathers, S. F. Kry, and M. Stovall, *Adaptations to a generalized radiation dose reconstruction methodology for use in epidemiologic studies: An update from the MD Anderson late effect group,* Radiation Research **192**, 169 (2019).

[6] A. M. Geyer, S. O'Reilly, C. Lee, D. J. Long, and W. E. Bolch, *The UF/NCI family of hybrid computational phantoms representing the current US population of male and female children, adolescents, and adults-application to CT dosimetry,* Physics in Medicine & Biology **59**, 5225 (2014).

[7] I. Alziar, G. Bonniaud, D. Couanet, J. B. Ruaud, C. Vicente, G. Giordana, O. Ben-Harrath, J. C. Diaz, P. Grandjean, H. Kafrouni, *et al.*, *Individual radiation therapy patient whole-body phantoms for peripheral dose evaluations: method and specific software,* Physics in Medicine & Biology **54**, N375 (2009).

[8] T. Xie, N. Kuster, and H. Zaidi, *Computational hybrid anthropometric paediatric phantom library for internal radiation dosimetry,* Physics in Medicine & Biology **62**, 3263 (2017).

[9] J. Valentin, *Basic anatomical and physiological data for use in radiological protection: reference values: ICRP Publication 89,* Annals of the ICRP **32**, 1 (2002).

[10] R. J. Kuczmarski, K. M. Flegal, S. M. Campbell, and C. L. Johnson, *Increasing prevalence of overweight among us adults: the national health and nutrition examination surveys, 1960 to 1991,* JAMA **272**, 205 (1994).

[11] G. L. de la Grandmaison, I. Clairand, and M. Durigon, *Organ weight in 684 adult autopsies: new tables for a Caucasoid population,* Forensic Science International **119**, 149 (2001).

[12] Z. Wang, I. W. E. M. van Dijk, J. Wiersma, C. M. Ronckers, F. Oldenburger, B. V. Balgobind, P. A. N. Bosman, A. Bel, and T. Alderliesten, *Are age and gender suitable matching criteria in organ dose reconstruction using surrogate childhood cancer patients' CT scans?* Medical Physics **45**, 2628 (2018).

[13] Z. Obermeyer and E. J. Emanuel, *Predicting the future–big data, machine learning, and clinical medicine,* New England Journal of Medicine **375**, 1216 (2016).

[14] M. Virgolin, I. W. E. M. van Dijk, J. Wiersma, C. M. Ronckers, C. Witteveen, A. Bel, T. Alderliesten, and P. A. N. Bosman, *On the feasibility of automatically selecting similar patients in highly individualized radiotherapy dose reconstruction for historic data of pediatric cancer survivors,* Medical Physics **45**, 1504 (2018).

[15] N. Breslow, A. Olshan, J. B. Beckwith, and D. M. Green, *Epidemiology of Wilms tumor,* Medical and Pediatric Oncology **21**, 172 (1993).

[16] I. W. E. M. van Dijk, F. Oldenburger, M. C. Cardous-Ubbink, M. M. Geenen, R. C. Heinen, J. de Kraker, F. E. van Leeuwen, H. J. van der Pal, H. N. Caron, C. C. Koning, *et al.*, *Evaluation of late adverse events in long-term Wilms' tumor survivors,* International Journal of Radiation Oncology · Biology · Physics **78**, 370 (2010).

[17] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, *et al.*, *3D Slicer as an image computing platform for the quantitative imaging network,* Magnetic Resonance Imaging **30**, 1323 (2012).

[18] S. C. Huijskens, I. W. E. M. van Dijk, J. Visser, C. R. N. Rasch, T. Alderliesten, and A. Bel, *Magnitude and variability of respiratory-induced diaphragm motion in children during image-guided radiotherapy,* Radiotherapy and Oncology **123**, 263 (2017).

[19] M. Xi, M. Z. Liu, Q. Q. Li, L. Cai, L. Zhang, and Y. H. Hu, *Analysis of abdominal organ motion using four-dimensional CT,* Chinese Journal of Cancer **28**, 989 (2009).

[20] A. Jain, K. Nandakumar, and A. Ross, *Score normalization in multimodal biometric systems,* Pattern Recognition **38**, 2270 (2005).

[21] L. R. Dice, *Measures of the amount of ecologic association between species,* Ecology **26**, 297 (1945).

[22] K. H. Zou, S. K. Warfield, A. Bharatha, C. M. C. Tempany, M. R. Kaus, S. J. Haker, W. M. Wells III, F. A. Jolesz, and R. Kikinis, *Statistical validation of image segmentation quality based on a spatial overlap index,* Academic Radiology **11**, 178 (2004).

[23] S. Nikolov, S. Blackwell, R. Mendes, J. De Fauw, C. Meyer, C. Hughes, H. Askham, B. Romera-Paredes, A. Karthikesalingam, C. Chu, *et al.*, *Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy,* (2018), arXiv preprint arXiv:1809.04430.

[24] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression,* Annals of Statistics **32**, 407 (2004).

[25] R. Tibshirani, *Regression shrinkage and selection via the Lasso,* Journal of the Royal Statistical Society: Series B (Methodological) **58**, 267 (1996).

[26] L. Breiman, *Random forests,* Machine Learning **45**, 5 (2001).

[27] J. R. Koza, *Genetic programming II: automatic discovery of reusable subprograms,* Cambridge, MA, USA (1994).

[28] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming* (Lulu Enterprises, UK Ltd, 2008).

[29] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, *Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning,* in *Genetic and Evolutionary Computation Conference (GECCO) 2017* (ACM, New York, NY, USA, 2017) pp. 1041–1048.

[30] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, *Improving model-based genetic programming for symbolic regression of small expressions,* (2019), accepted for publication in Evolutionary Computation. ArXiv preprint arXiv:1904.02050.

[31] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, *A survey of methods for explaining black box models,* ACM Computing Surveys (CSUR) **51**, 93:1 (2018).

[32] R Core Team, *R: A language and environment for statistical computing,* R Foundation for Statistical Computing (2013).

[33] J. Friedman, T. Hastie, and R. Tibshirani, *GLMNET: Lasso and elastic-net regularized generalized linear models,* R package version **1** (2009).

[34] M. N. Wright and A. Ziegler, *Ranger: a fast implementation of random forests for high dimensional data in C++ and R,* (2015), arXiv preprint arXiv:1508.04409.

[35] P. Probst and A.-L. Boulesteix, *To tune or not to tune the number of trees in random forest,* Journal of Machine Learning Research **18**, 6673 (2017).

[36] J. Ni, R. H. Drieberg, and P. I. Rockett, *The use of an analytic quotient operator in genetic programming,* IEEE Transactions on Evolutionary Computation **17**, 146 (2013).

[37] M. Kuhn, *The caret package,* Journal of Statistical Software **28**, 1 (2008).

[38] P. A. N. Bosman, J. Grahl, and D. Thierens, *Enhancing the performance of maximum–likelihood Gaussian EDAs using anticipated mean shift,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2008) pp. 133–143.

[39] P. A. N. Bosman, J. Grahl, and D. Thierens, *Benchmarking parameter-free AMaLGaM on functions with and without noise,* Evolutionary Computation **21**, 445 (2013).

[40] A. N. I. Badouna, C. Veres, N. Haddy, F. Bidault, D. Lefkopoulos, J. Chavaudra, A. Bridier, F. de Vathaire, and I. Diallo, *Total heart volume as a function of clinical and anthropometric parameters in a population of external beam radiation therapy patients,* Physics in Medicine & Biology **57**, 473 (2012).

[41] E. J. Stepusin, D. J. Long, E. L. Marshall, and W. E. Bolch, *Assessment of different patient-to-phantom matching criteria applied in Monte Carlo–based computed tomography dosimetry,* Medical Physics **44**, 5498 (2017).

[42] Z. Wang, B. V. Balgobind, M. Virgolin, I. W. E. M. van Dijk, J. Wiersma, C. M. Ronckers, P. A. N. Bosman, A. Bel, and T. Alderliesten, *How do patient characteristics and anatomical features correlate to accuracy of organ dose reconstruction for Wilms' tumor radiation treatment plans when using a surrogate patient's CT scan?* Journal of Radiological Protection **39**, 598 (2019).

[43] J. Demšar, *Statistical comparisons of classifiers over multiple data sets,* Journal of Machine Learning Research **7**, 1 (2006).

[44] J. M. Bland and D. G. Altman, *Multiple significance tests: the Bonferroni method,* BMJ **310**, 170 (1995).

[45] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning,* Nature **521**, 436 (2015).

[46] J. H. Friedman, *Greedy function approximation: a gradient boosting machine,* Annals of Statistics **29**, 1189 (2001).

[47] C. Pinter, A. Lasso, A. Wang, D. Jaffray, and G. Fichtinger, *SlicerRT: radiation therapy research toolkit for 3D Slicer,* Medical Physics **39**, 6332 (2012).

[48] A. Ng, T. Nguyen, J. L. Moseley, D. C. Hodgson, M. B. Sharpe, and K. K. Brock, *Reconstruction of 3D lung models from 2D planning data sets for Hodgkin's lymphoma patients using combined deformable image registration and navigator channels,* Medical Physics **37**, 1017 (2010).

8

Surrogate-free Machine Learning-based Organ Dose Reconstruction

To study radiotherapy-related adverse effects, detailed dose information (3D distribution) is needed for accurate dose-effect modeling. For childhood cancer survivors who underwent radiotherapy in the pre-CT era, only 2D radiographs were acquired, thus 3D dose distributions must be reconstructed from limited information. State-of-the-art methods achieve this by using 3D surrogate anatomies. These can however lack personalization and lead to coarse reconstructions. We present and validate a surrogate-free dose reconstruction method based on Machine Learning (ML). Abdominal planning CTs ($n = 142$) of recently-treated childhood cancer patients were gathered, their organs at risk were segmented, and 300 artificial Wilms' tumor plans were sampled automatically. Each artificial plan was automatically emulated on the 142 CTs, resulting in 42,600 3D dose distributions from which dose-volume metrics were derived. Anatomical features were extracted from digitally reconstructed radiographs simulated from the CTs to resemble historical radiographs. Further, patient and radiotherapy plan features typically available from historical treatment records were collected. An evolutionary ML algorithm was then used to link features to dose-volume metrics. Besides 5-fold cross validation, a further evaluation was done on an independent dataset of five CTs each associated with two clinical plans. Cross-validation resulted in mean absolute errors $\leq$0.6 Gy for organs completely inside or outside the field. For organs positioned at the edge of the field, mean absolute errors $\leq$1.7 Gy for D_{mean}, $\leq$2.9 Gy for D_{2cc}, and $\leq$13% for V_{5Gy} and V_{10Gy}, were obtained, without systematic bias. Similar results were found for the independent dataset. To conclude, we proposed a novel organ dose reconstruction method that uses ML models to predict dose-volume metric values given patient and plan features. Our approach is not only accurate, but also efficient, as the setup of a surrogate is no longer needed.

The contents of this chapter are based on the following preprint: **M. Virgolin**, Z. Wang (shared first co-author), B.V. Balgobind, I.W.E.M. van Dijk, J. Wiersma, P.S. Kroon, G.O. Janssens, M. van Herk, D.C. Hodgson, L. Zadravec Zaletel, C.R.N. Rasch, A. Bel, P.A.N. Bosman, and T. Alderliesten. Surrogate-free machine learning-based organ dose reconstruction for pediatric abdominal radiotherapy. *Submitted. Preprint arXiv:2002.07161*, arXiv (2020).

8.1. Introduction

Patients undergoing radiotherapy (RT) are prone to develop radiation-related Adverse Effects (AEs) [1–3]. To improve the design of future multi-modality treatments, clinicians are interested in better understanding the relationship between radiation dose and onset of AEs. Modern research efforts in this direction delve into dosimetric details, employing dose distribution metrics to a specific organ (or sub-volume) as explanatory variables. Such rich information is obtained by simulating the RT plan on 3D imaging of the patient (i.e., CT scans) with organ segmentations in a Treatment Planning System (TPS) [4–6].

Unfortunately, when so-called *late* AEs (onset can be decades after RT) need to be studied, it is not always possible to straightforwardly obtain detailed information on dose distributions [1]. For patients who underwent RT before the use of planning CTs became commonplace (in the following, *historical patients*), 2D radiographs were used for treatment planning (e.g., this was the case until the 1990s in the Netherlands [2]), meaning no 3D anatomical imaging is available. Consequently, no simulations can be performed in a TPS to estimate 3D dose distributions for these patients [7–9]. The information available for historical patients normally consists of what was reported in treatment records, e.g., features of the patient such as age and gender, and features of the plan such as prescribed dose, geometry of the plan, and the use of blocks. Additionally, 2D radiographs can be available, from which information can be gathered on the internal anatomy (mainly bony anatomy, as internal organs are normally not clearly distinguishable), and on the plan configuration with respect to the patient's anatomy [2, 10].

To improve the understanding of late AEs, recent research is striving to develop increasingly accurate *dose reconstruction* methods, i.e., methods to estimate the 3D dose distribution received by historical patients [7, 9, 11, 12]. State-of-the-art approaches employ *phantoms*, i.e., 3D surrogates of the human anatomy upon which the RT plan can be simulated, to compute the dose distribution. Phantoms exist in different forms: physical or virtual, made by simple geometrical shapes or by adopting and morphing actual CT scans and organ segmentations [7, 11, 12]. Generally, phantoms are built to represent *average* anatomies, for categories of patients (e.g., for a certain age range), and are collected into so-called phantom libraries [13–15]. Whenever dose reconstruction for a historical patient is needed, the phantom that represents the category that the patient belongs to is retrieved from the library and used as surrogate for simulation of the RT plan.

As the largest source of error related to phantom-based dose reconstruction comes from the mismatch between the anatomy of the phantom and the true anatomy of the patient [16], it is important to define the best way to match phantoms to patients. This issue is still under research, and different approaches employ different heuristic matching criteria that are normally hand-crafted and based upon statistics and guidelines drawn from large population studies (e.g., ICRP89, NANTHES) [13–15, 17]. However, the use of heuristic matching criteria has been hypothesized to be too simplistic to capture the high variability of internal human anatomy [11, 15, 18–20]. For example, a popular phantom-based dose reconstruction approach uses solely age and gender for surrogate matching [21]. Our group's recent work focusing on Wilms' tumor (the most common type of kidney cancer for childhood cancer patients) irradiation for pediatric patients showed that utilizing surrogate CTs using age- and gender-based matching can lead to poor dose reconstruction quality in individual cases [22].

To improve the resemblance of a surrogate phantom, there have been efforts to replace the normally hand-crafted heuristic matching criteria with data-driven decisions. For example, statistical models inferred from CTs and 3D organ segmentations of adult patients have been used to drive a deformable image registration procedure that adapted 3D organ segmentations to the 2D anatomy of a specific patient, given features of the latter as measurable from 2D radiographs [9, 23]. Using a state-of-the-art Machine Learning (ML) algorithm, it has been shown that features typically available for historical patients treated for Wilms' tumor can be linked to different 3D anatomy similarity metrics based on organ segmentations and CTs (Chapter 6). Our group recently proposed an automatic pipeline that uses ML to steer the assembling of a new original anatomy based on 3D CTs and organ segmentations of multiple patients using the features of a historical patient (Chapter 7). However, it is important to realize that maximizing some form of overall anatomical resemblance is difficult. Moreover, from the standpoint of optimizing dose reconstruction accuracy, it can be considered sub-optimal for RT dosimetry purposes. This is because in RT dosimetry, what part of anatomy is most meaningful largely depends on the particular RT plan [20].

To the best of our knowledge, although *both* patient anatomy and plan geometry play a key role in determining dose-volume metrics for Organs At Risk (OARs), existing dose reconstruction approaches focused solely on patient anatomy information, to obtain a representative surrogate. Plan information is used only later, to calculate the dose on the surrogate. The purpose of this chapter is to develop and validate an ML approach to predict dose-volume metrics for OARs based on patient anatomy and plan geometry information. Specifically, we propose to use ML to directly learn what dose-volume metrics for an OAR are likely given information on the patient and on the plan, without the need to select or craft any surrogate anatomy. We argue that this is a sensible choice because ML can directly be trained upon what ultimately matters, i.e., dose reconstruction accuracy. In this chapter we present our ML-based organ dose reconstruction approach and its validation, that was performed on a relatively large dataset of artificial plans, as well as on a smaller dataset of clinical plans.

8.2. Materials & methods

We considered pediatric flank RT, and in particular RT for Wilms' tumor, as an application for our dose reconstruction method, in continuity with our previous work. The choice to focus on pediatrics is because children are the most prone to develop late AEs [3], and are typically underrepresented in existing phantom libraries [11]. Moreover, more than 85% of pediatric patients survives Wilms' tumor five years or longer, but considerable chances of the onset of late AEs remain [2].

8.2.1. Patient data

To be able to create a ground-truth to learn dose-volume metrics from, CT scans were needed. Hence, a total of 142 pediatric planning CTs were collected by involving the following institutes (number of CTs in brackets): Amsterdam University Medical Centers / Emma Children's Hospital ($n = 38$), University Medical Center Utrecht / Princess Máxima Center for Pediatric Oncology ($n = 42$), The Christie NHS Foundation Trust ($n = 33$),

Princess Margaret Cancer Centre ($n = 18$), and Institute of Oncology Ljubljana ($n = 11$). Five further CTs were collected from the Amsterdam University Medical Centers and kept aside to be used exclusively for an additional validation step (Sec. 8.2.5).

The inclusion criteria were: patient age at scan acquisition between 1 to 8 years; the CT field of view including a common abdominal region from the tenth thoracic (T10) vertebral body to the first sacral (S1) vertebral body; presence of five lumbar vertebrae (rare cases of patients with six exist); patient scanned in supine position; quality of CT sufficient to perform organ segmentations. The patients underwent RT between 2002 and 2018, mostly but not exclusively for abdominal cancers. The median CT slice resolution was 0.94×0.94 mm, the median slice thickness was 3 mm.

As we focused on Wilms' tumor treatment, four OARs were considered: the liver, the spleen, the contralateral kidney (left or right, depending on the side of the tumor), and the spinal cord (between T10 and S1). To provide accurate and consistent OAR segmentations, we carefully prepared the OAR segmentations in all CTs ($n = 142 + 5$): for 60 CTs, pre-existing clinical segmentations of OARs were manually improved and approved (I.W.E.M. van Dijk, checked by B.V. Balgobind only for difficult cases). To aid the manual segmentation of the OARs for the remaining CTs (n=87), the software ADMIRE (research version 2.3.0) from Elekta (Elekta AB, Stockholm, Sweden) was used to generate multi-atlas based automatic segmentations using the previous 60 CTs as atlas. These segmentations were further manually checked slice-by-slice and possibly adapted (Z. Wang, I.W.E.M. van Dijk), and finally checked and approved (I.W.E.M. van Dijk, checked by B.V. Balgobind only for difficult cases). Some patients did not have both kidneys intact, due to nephrectomy prior to RT. The number of CTs that had only a complete right kidney, only a complete left kidney, and two complete kidneys were 36, 40, and 71, respectively.

8.2.2. Automatic generation of artificial Wilms' tumor plans

A method to automatically generate historical-like abdominal flank irradiation plans (i.e., artificial plans) for Wilms' tumor treatment based on information visible on 2D radiographs was created, in order to obtain large plan variations.

Figures 8.1(a) and 8.1(c) illustrate examples of actual historical plans on respective historical radiographs. As can be observed from the examples, a typical historical flank irradiation field is a rectangular area, with possible shielding blocks, that is located on the right or on the left flank. Flank irradiation is done by beams from anterior-posterior (AP) and posterior-anterior (PA) direction. Along right-left (RL), one field border is located at the edge of the patient's body contour, while the other is located as to include the vertebral column [24]. In some cases, blocks were placed to protect OARs from irradiation (Fig. 8.1(c)). In historical plans the isocenter was positioned in the center of the treatment field that is projected on the coronal plane (Fig. 8.1) and at the middle of the patient's AP abdominal diameter.

To generate artificial plans, two reference digitally reconstructed radiographs (DRRs) were considered, randomly selected from the data. One DRR was derived from a CT of a 5-year old female patient without nephrectomy (ref 1 in Fig. 8.2), and the other was derived from a CT of a 4-year old female patient with nephrectomy of the left kidney (ref 2 in Fig. 8.2). Upon these two DRRs, boundaries defining plan variability were identified by an experienced pediatric radiation oncologist (B. V. Balgobind), to ensure that generated

plans would appear to be reasonable according to historical clinical guidelines. Note that historical clinical guidelines are slightly different from current ones (e.g., currently the iliac crests should be safeguarded, unlike in Fig. 8.1(c)). Figure 8.2 shows two examples of landmark locations identifying possible plan variations, on the two reference DRRs. Specifically, given the boundaries of possible isocenter positions and field borders, plans with a rectangular field were generated by sampling *uniformly* within those boundaries. For each plan generated, an additional version of that plan including one block was generated as well. A block was simulated as the area in the upper lateral corner enclosed by the border of the rectangular field and a line crossing two randomly sampled endpoints. The endpoints were sampled from two regions roughly covering the start and end points of rib 9 and rib 12 on the DRRs (regions indicated by the green boxes in Fig. 8.2). This way, a sampled block covered part of the liver (in right-sided plans) or part of the spleen (in left-sided plans). All plans consist of two opposing and symmetrical beams in AP-PA directions irradiating one side of the abdominal flank. Figures 8.1(b) and 8.1(d) illustrate two examples of sampled artificial plans (without or with a block) on respective DRRs.

A total of 300 artificial plans were generated automatically, of which 150 without a block, and 150 with a block. The random sampling of the plan side led to 142 left-sided plans and 158 right-sided plans (roughly half-half). The same set of plan features used in our previous work was considered to generate plans in DICOM RTPLAN format (e.g., gantry and collimator angles, isocenter location, field sizes) [25].

8.2.3. Generation of the dataset for ML

Figure 8.3 summarizes the pipeline used to generate the dataset for ML. Firstly, we emulated each of the 300 artificial plans on each of the 142 CT scans by the automatic plan emulation method proposed in our previous work [25], leading to a total of 42,600 emulations. The method automatically transfers a plan prepared on one CT to another CT (with quality comparable to human experts), using landmark detection upon the respective DRRs. Secondly, for each of the 42,600 plan emulations, dose-volume metrics of interest (see Sec. 8.2.3) were collected for the different OARs by use of our automatic dose computation pipeline [25]. The pipeline used the collapsed cone dose calculation algorithm of Oncentra TPS (version 4.3, Elekta AB, Stockhom, Sweden). Thirdly, features that are plausible to be available for typical historical cases were collected from the anatomy of the included CTs as visible in the respective DRRs, from the artificial plans, and from the relationship between anatomy and plan geometry.

Response variables: dose-volume metrics

To select dose-volume metrics to use as response variables for ML, we considered metrics typically used to validate state-of-the-art dose reconstruction approaches [9, 12], and typically found to be of clinical relevance in studies of AEs in adults (e.g., QUANTEC) [6, 10, 26]. Studies on dose-volume response relationships for pediatric patients (so-called PENTEC studies [27]) are currently limited.

This reasoning led us to consider mean organ dose (D_{mean}), two levels of percentage of OAR volume receiving at least X Gy (V_{XGy}), and the minimum dose received by the maximally exposed 2 cubic centimeters of an OAR (D_{2cc}), the latter being similar but more robust than the maximum dose to a single point. Typically, V_{5Gy} and V_{20Gy} are considered.

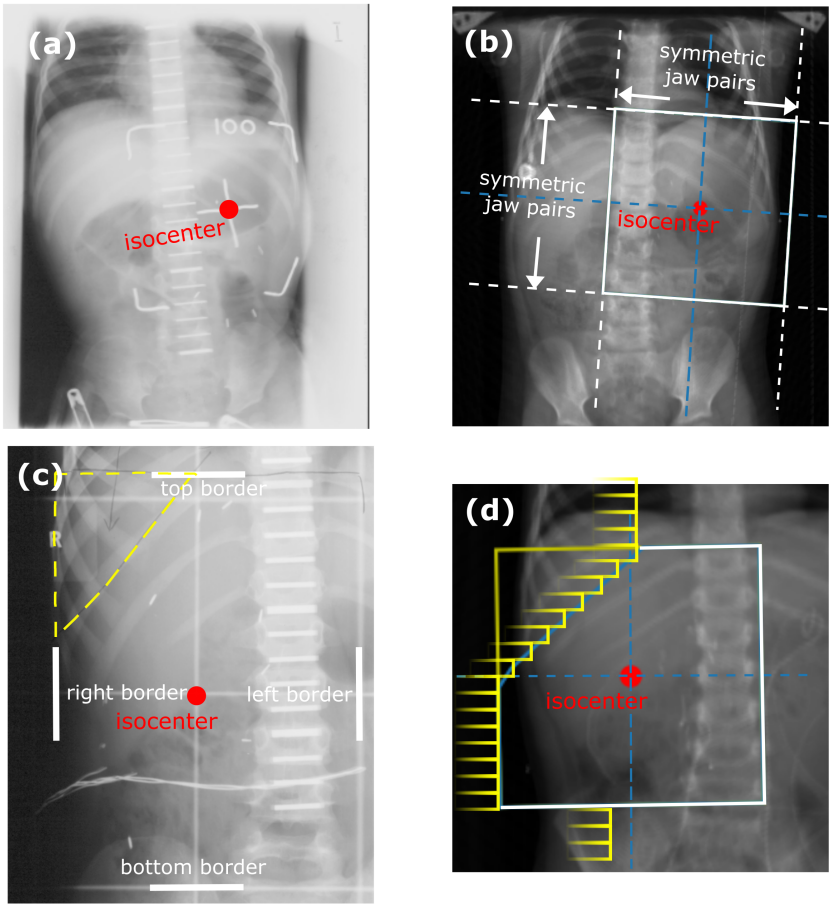

Figure 8.1: (a) An actual hand-drawn plan on a historical radiograph with a rectangular field (indicated by white corners). (b) An artificial plan with a rectangular field (in white lines) plotted on the DRR of a recent patient. (c) An actual hand-drawn plan on a historical radiograph with a rectangular field (in white bars) and an additional block (outlined by dashed yellow lines) to spare part of the liver. (d) An artificial plan plotted on the DRR of a recent patient with a rectangular field (in white bars) and an additional block (obtained by multi-leaf collimators, outlined by yellow lines) to spare part of the liver. For each plot, the isocenter is indicated by a red dot in the middle of the field.

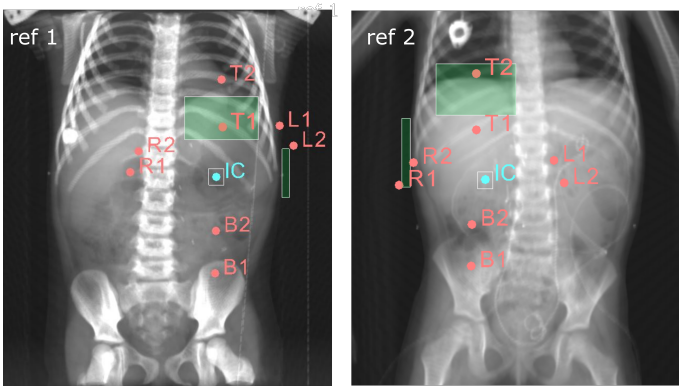

Figure 8.2: Examples of landmark locations to specify geometry variability of two types of artificial plans (left-sided plans in the left figure and right-sided plans in the right figure). Ref 1 is the DRR derived from the reference CT of a 5-year-old female patient and ref 2 is the DRR derived from the reference CT of a 4-year-old female patient. The box around the isocenter (IC) specifies the range of possible isocenter positions. The vertical position of T1/T2 and of B1/B2 specify the lowest/highest position of the upper and lower border of the field, respectively. The horizontal positions of R1/R2 and L1/L2 specify the right-most/leftmost position of the right and left border of the field, respectively. The isocenter and artificial field border positions were sampled uniformly at random within the specified ranges. The green boxes indicate the regions where two endpoints of a line representing a block border can be sampled. This line, together with the upper and left/right field borders, encloses the block.

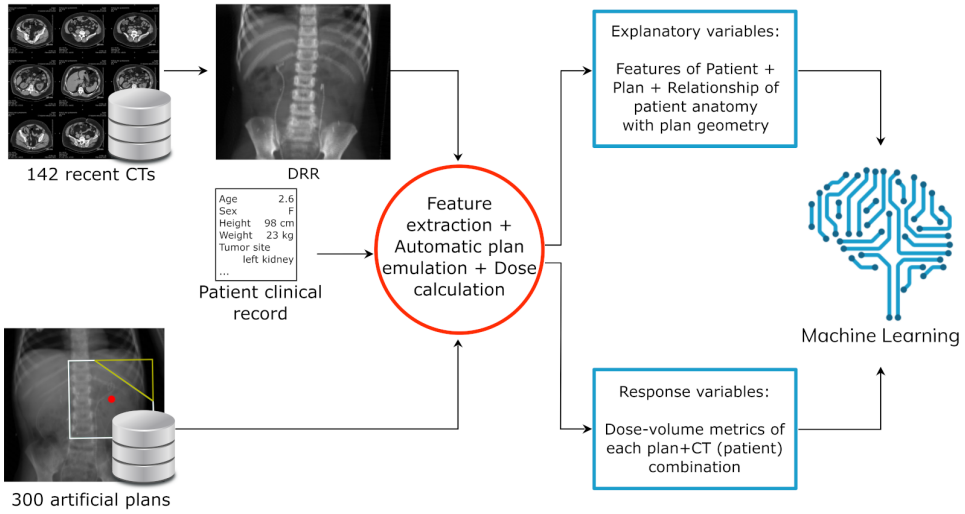

Figure 8.3: Pipeline for data generation. Artificial plans are sampled automatically. The explanatory and response variables are used as input to train the ML model. The explanatory variables include features of the plan (e.g., isocenter location, field size), patient features (e.g., age, nephrectomy), and features on the relationship between the anatomy of the patient and the geometry of the plan (e.g., signed distance between the 2nd lumbar vertebra and the plan isocenter). The response variables are dose-volume metrics for each OAR.

However, instead of $V_{20\text{Gy}}$, we decided to use $V_{10\text{Gy}}$ in this work since our plans have a prescribed dose of 14.4 Gy (thus $V_{20\text{Gy}}$ was always 0 for the OARs). Regarding $D_{2\text{cc}}$, we decided to include this metric because peak dose values to a small OAR portion may be relevant to explain late AEs related to OARs that work in a serial fashion (e.g., the spinal cord).

Explanatory variables: features of patients and plans

To assess what information can be available for historical patients, we considered the Dutch records of the Emma Children's Hospital/Academic Medical Center childhood cancer survivor cohort, who underwent RT between 1966 and 1996 [2]. For this cohort, along with historical patient records and treatment plan details, 2D coronal radiographs were consistently taken, hence providing partial information on the anatomy.

The complete set of features considered in this work is reported in Table 8.1. Note the absence of height and weight (which are used by some phantom-based methods [15]). For 12% of the patients, height and/or weight data were missing and preliminary experiments using automatic imputation methods showed no benefit in including them.

For the features related to anatomical geometry, anatomical landmarks from DRRs were detected automatically using the landmark detection method in our previous work [25]. Note that the landmarks concerned only bony anatomy because other internal anatomy tissues are not reliably visible in historical radiographs. Importantly, we normalized features related to measurements of anatomy and anatomy-plan geometry configuration (e.g., rib-cage width, field sizes, distances between landmarks and the isocenter) by the width and height of the respective DRR they were measured from (after the DRRs were cropped to a same region of interest between T10 and S1). This was done because when plans are emulated, they are scaled based on proportions derived from the landmarks [22, 25]. Since differences in anatomy solely due to overall anatomy scaling do not result in different dose-volume metric values, these differences should not be accounted for by the explanatory variables (confirmed in preliminary experiments).

The abdominal diameter in AP ($Diam^{IC}_{AP}$) is the only anatomical feature not measurable from DRRs generated along AP/PA direction. In historical RT, it was measured using a ruler to determine the isocenter position along the AP axis, and was subsequently reported in the records. For our cohort, $Diam^{IC}_{AP}$ was not reported in the records because a CT scan was used for RT planning. We therefore measured $Diam^{IC}_{AP}$ automatically on the CT scans, by using a pre-determined isocenter position of typical abdominal flank irradiation plans. In particular, the intervertebral disk between the 1$^{\text{st}}$ and the 2$^{\text{nd}}$ lumbar vertebra (L1 and L2) was used to determine the isocenter position along the inferior-superior (IS) axis, and the center of mass of the kidney was used to determine the isocenter position along the RL axis (as the aim of Wilms' tumor plans is to irradiate the renal fossa). For CTs including both kidneys, two $Diam^{IC}_{AP}$ were measured, and the average was taken. Conversely, only one $Diam^{IC}_{AP}$ was measured for CTs including a single kidney.

In our simulations we used for all artificial plans the same fractionation scheme (8×1.8 Gy), beam energy (6 MV), and prescribed dose (14.4 Gy at isocenter). These settings are the most common in historical records, and are still valid in the current Wilms' tumor RT protocol [24]. Moreover, choosing a specific prescribed dose (e.g., 14.4 Gy) does not limit generalizability, since the dose distribution over the entire anatomy depends linearly

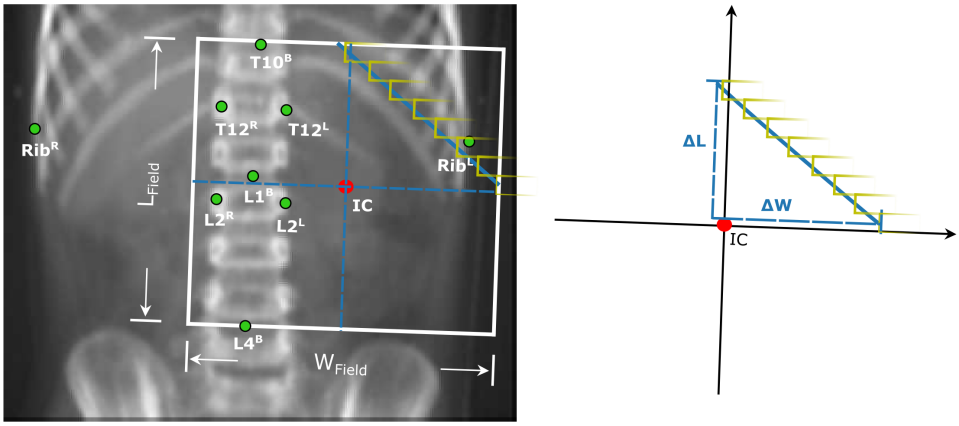

Figure 8.4: An example of the beam's eye view of a plan plotted on a DRR with the landmark locations used to compute the features concerning plan field configuration on top of the patient's anatomy. The plot next to the DRR illustrates how the block is simulated by aligning the center of the leaves with the boundary of the block and how the slope of an MLC-simulated block is calculated.

on the prescribed dose. Thus, if dose reconstruction for a historical case using a different prescription is needed, the dose-volume metrics predicted by the models trained on plans with a 14.4 Gy prescribed dose can be re-scaled.

For both fields with and without a block, the features representing field sizes in RL and IS directions (W_{Field} and L_{Field}) were set by simply considering the full rectangular area (i.e., irrespective of blocking). For fields with a block, the slope of the block (note that the block is formed by Multi-Leaf Collimators (MLCs)) and the ratio between the blocked region and non-blocked region of the field ($Ratio_{Block}$) was computed. In addition, we considered features that relate to how the plan was configured with respect to the patient's anatomy, based on the position of the isocenter and of the bony landmarks. For instance, $\Delta^{IC}_{RL}(T10^B)$ links the bottom of the T10 vertebra to the position of the isocenter in RL direction. Figure 8.4 shows an example of the anatomical landmarks and plan geometrical borders used to calculate features describing plan configuration with respect to the patient's anatomy.

8

Dataset for supervised learning

Features and dose-volume metrics were finally collected in a dataset. The dataset corresponded to a 2D matrix, where the rows represented patient-plan combinations, i.e., examples ($n = 42,600$), and the columns represented features (33) and response variables (4 for each OAR).

8.2.4. Machine learning

In the following sections we describe how ML was performed in terms of training and validation on the artificial plans. We further introduce the ML algorithm adopted, and describe an independent validation on clinical cases.

Table 8.1: Description of the 33 features considered as explanatory variables for ML.

Feature name	Origin	Unit	Description
Age	Records	years	patient age at CT scanning
ArmsUp	DRR	yes/no	whether the patient had arms in a raised position during scanning
$Diam_{AP}^{IC}$	Records	cm	patient AP diameter measured at isocenter
Nephrectomy	Records	yes/no	whether the patient underwent nephrectomy
W_{Rib^R}	DRR	cm	width (in RL) of right-part of the rib cage (from vertebral column to location of right-most rib)
W_{Rib^L}	DRR	cm	width (in RL) of left-part of the rib cage (from vertebral column to location of left-most rib)
W_{VC}	DRR	cm	average vertebral column width
L_{VC}	DRR	cm	length (in IS) of the vertebral column from T11 to L4
W_{Field}	Plan	cm	field width (in RL)
L_{Field}	Plan	cm	field length (in IS)
FieldSide	Plan	right/left	whether the plan concerns left-sided or right-sided flank irradiation
$Intercept_{Block}$	Plan	cm	distance (in RL) between isocenter and block endpoint of the top field border
$Ratio_{Block}$	Plan	%	$Area(Block)/Area(Rectangular\ field)$, 0 for block-free plans
$Slope_{Block}$	Plan	-	$\Delta L/\Delta W$ of the block (see Fig. 8.4); 0 for block-free plans
θ_C	Plan	∘	angle of collimator system with respect to gantry system
$\Delta_{RL}^{IC}(T10^B)$	Plan + DRR	cm	RL distance between bottom of T10 and isocenter
$\Delta_{IS}^{IC}(T10^B)$	Plan + DRR	cm	IS distance between bottom of T10 and isocenter
$\Delta_{RL}^{IC}(T12^R)$	plan + DRR	cm	RL distance between right border of T12 and isocenter
$\Delta_{IS}^{IC}(T12^R)$	Plan + DRR	cm	IS distance between right border of T12 and isocenter
$\Delta_{RL}^{IC}(T12^L)$	Plan + DRR	cm	RL distance between left border of T12 and isocenter
$\Delta_{IS}^{IC}(T12^L)$	Plan + DRR	cm	IS distance between left border of T12 and isocenter
$\Delta_{RL}^{IC}(L1^B)$	Plan + DRR	cm	RL distance between bottom of L1 and isocenter
$\Delta_{IS}^{IC}(L1^B)$	Plan + DRR	cm	IS distance between bottom of L1 and isocenter
$\Delta_{RL}^{IC}(L2^R)$	Plan + DRR	cm	RL distance between right border of L2 and isocenter
$\Delta_{IS}^{IC}(L2^R)$	Plan + DRR	cm	IS distance between right border of L2 and isocenter
$\Delta_{RL}^{IC}(L2^L)$	Plan + DRR	cm	RL distance between left border of L2 and isocenter
$\Delta_{IS}^{IC}(L2^L)$	Plan + DRR	cm	IS distance between left border of L2 and isocenter
$\Delta_{RL}^{IC}(L4^B)$	Plan + DRR	cm	RL distance between bottom of L4 and isocenter
$\Delta_{IS}^{IC}(L4^B)$	Plan + DRR	cm	IS distance between bottom of L4 and isocenter
$\Delta_{RL}^{IC}(Rib^R)$	Plan + DRR	cm	RL distance between location of right-most rib and isocenter
$\Delta_{IS}^{IC}(Rib^R)$	Plan + DRR	cm	IS distance between location of right-most rib and isocenter
$\Delta_{RL}^{IC}(Rib^L)$	Plan + DRR	cm	RL distance between location of left-most rib and isocenter
$\Delta_{IS}^{IC}(Rib^L)$	Plan + DRR	cm	IS distance between location of left-most rib and isocenter

Abbreviations: R (in superscript): right, L (in superscript): left, RL: right-left, AP: anterior-posterior, IS: inferior-superior, IC: isocenter, VC: vertebral column, W: width, L in L_{VC} and L_{Field} : length.

Training and evaluation of ML models

Since dose metrics are scalars, we treated the learning problem as a regression problem. We trained a separate ML model for each combination of dose-volume metric and OAR.

Preliminary analysis showed that right-sided plans and left-sided plans led to markedly different distributions of possible dose-volume metric values for all OARs except for the spinal cord. Thus, ML models were set to be composed of two sub-models, each to be trained independently on a particular sub-set of the data based on plan side (right or left).

The quality of the models was estimated with a 5-fold cross-validation. This means that a random partition of 1/5th of the total number of patients and plans was held out (test set), and training was performed on the remaining data. Then, the prediction error was measured on the test set. This process was repeated five times, each time considering a different data partition for the test set. No patient nor plan that was in the test set was included in the data at training time.

Each training step included hyper-parameter tuning by grid-search with internal 5-fold cross-validation (upon the training set), as well as feature selection (which resulted in eight features being systematically discarded, see Sec. 8.2.4). For each dose-volume metric $k \in \{D_{\text{mean}}, D_{\text{2cc}}, V_{\text{5Gy}}, V_{\text{10Gy}}\}$, the Root Mean Square Error (RMSE) loss was used, i.e.,

$$RMSE(Y^k, \hat{Y}^k) = \sqrt{\frac{1}{\nu}\sum_{i=1}^{\nu}\left(Y_i^k - \hat{Y}_i^k\right)^2}, \tag{8.1}$$

where Y^k are the ground truth values and $\hat{Y}^k$ are the model predictions for the dose-volume metric k, and ν is the total number of rows in the training set. The RMSE was chosen to regularize ML, i.e., to penalize larger errors more [28].

To account for the stochastic nature of the ML algorithm employed (see Sec. 8.2.4) and for the random partitioning of the data, the 5-fold cross-validation was repeated ten times. The averages and standard deviations over the 5×10 validation results (five folds repeated ten times) were considered.

To put the results of ML into perspective, for each cross-validation, a baseline prediction was considered that simply used the average dose-volume metric observed in the training data. The Wilcoxon signed-rank test was used to assess whether the results obtained by ML and by this baseline are significantly different (p-value < 0.05).

Feature selection

An automatic feature selection step was performed before training the ML models, as follows.

1. Compute the absolute Pearson correlation coefficient $|\rho|$ between all pairs of features based on the values of the training set.

2. If $|\rho| > 0.95$ (highly positive or negative correlation) between two features, discard the second feature.

3. Repeat (2) until no two features that have $|\rho| > 0.95$ remain.

Note that, at (2), one could discard a feature at random. We systematically discarded the second feature to obtain a deterministic outcome.

As aforementioned, a model for the prediction of the dose-volume metric of an organ was composed of two sub-models, trained independently based on the side of the plan. We found that partitioning the dataset by plan side does not influence feature selection significantly, i.e., the pairs of features that have $|\rho| > 0.95$ remain the same. Figure 8.5 shows the value $|\rho|$ between features across the entire dataset (averaged between left- and right-sided plans). The following eight features were systematically discarded in our experiments:

$\Delta^{IC}_{RL}(T12^R)$ – highly correlated with $\Delta^{IC}_{RL}(T10^B)$ (and others);
$\Delta^{IC}_{IS}(T12^R)$ – highly correlated with $\Delta^{IC}_{IS}(T10^B)$ (and others);
$\Delta^{IC}_{RL}(L2^R)$ – highly correlated with $\Delta^{IC}_{RL}(L1^B)$ (and others);
$\Delta^{IC}_{IS}(L2^R)$ – highly correlated with $\Delta^{IC}_{IS}(L1^B)$ (and others);
$\Delta^{IC}_{RL}(L2^L)$ – highly correlated with $\Delta^{IC}_{RL}(L1^B)$ (and others);
$\Delta^{IC}_{IS}(L2^L)$ – highly correlated with $\Delta^{IC}_{IS}(L1^B)$ (and others);
$\Delta^{IC}_{RL}(Rib^L)$ – highly correlated with $\Delta^{IC}_{RL}(L1^B)$ (and others);
$\Delta^{IC}_{IS}(Rib^L)$ – highly correlated with $\Delta^{IC}_{IS}(L1^B)$ (and others).

Machine learning algorithm

The recently introduced Genetic Programming version of the Gene-pool Optimal Mixing Evolutionary Algorithm (GP-GOMEA) was considered as ML algorithm, as it was found to achieve competitive performance on a variety of benchmark problems (Chapters 2 and 3), as well as in previous work concerning radiotherapy (Chapter 7 and [29]).

In addition, GP-GOMEA can perform symbolic regression, i.e., it can generate a regression ML model in the form of a (symbolic) mathematical expression. GP-GOMEA incorporates information theory methods that enable it to synthesize fairly accurate expressions of particularly compact size, an aspect that makes these expressions lightweight and fast to execute, and that can aid human-interpretability.

The salient hyper-parameters of GP-GOMEA are reported in Table 8.2. Other hyper-parameter settings that are not reported in the table were set to the ones used for benchmarking GP-GOMEA in Chapter 3 (Sec. 3.8). A short description of the hyper-parameters in Table 8.2 follows.

Tree height. GP-GOMEA encodes machine learning models with symbolic trees. In our case, trees were binary. The maximal tree height of a tree limits how complex the encoded machine learning model can be. For instance, a tree with height 2 can contain up to 7 nodes, a tree with height 4 can contain up to 31 nodes. Larger trees can encode relatively complex models (but risk overfitting), smaller trees can encode relatively simple formulas (but risk underfitting).

Evaluations limit. The evaluations limit is used to terminate GP-GOMEA. An evaluation is the computation of the loss function of a model. Terminating early/late can be useful to prevent overfitting/underfitting.

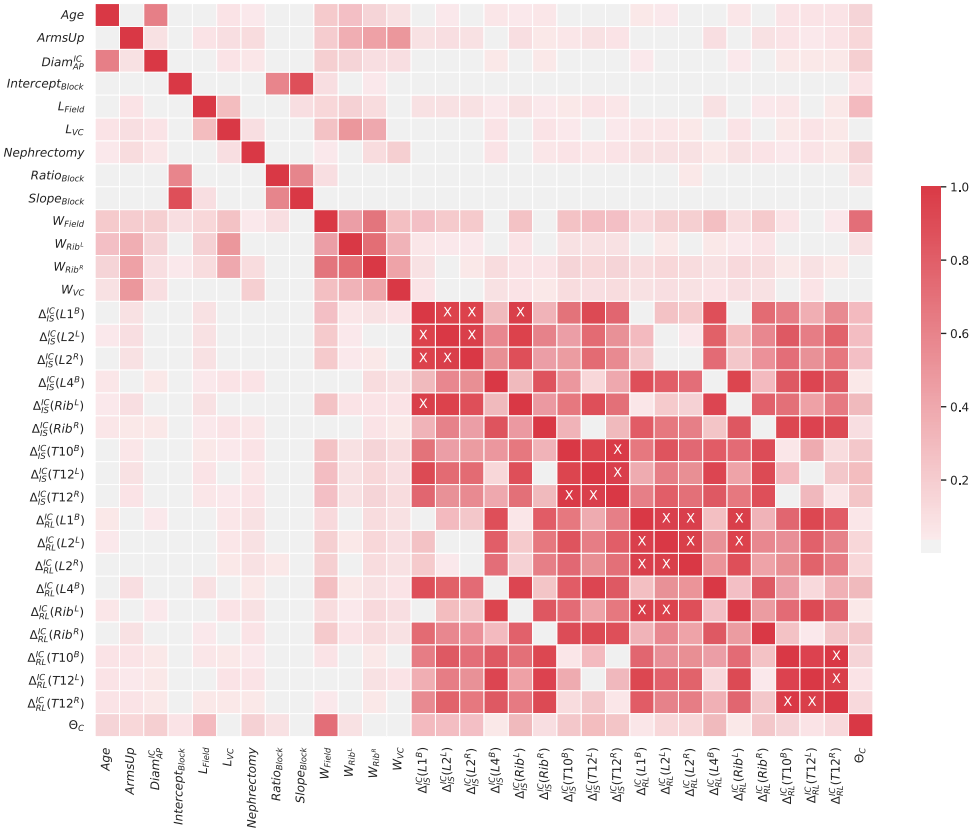

Figure 8.5: Absolute Pearson correlation coefficient ($|\rho|$) between features (average between left- and right-sided plans). Cells marked by a cross have $|\rho| > 0.95$ (excluding cells in the diagonal).

Hyper-parameter	Setting(s)
Tree height*	$\{2, 3, 4\}$
Evaluations limit*	$\{10^4, 10^5, 10^6\}$
Function set*	$\{[+, -, \times, \div_p], [+, -, \times, \div_p, \cdot^2, \sqrt{\lvert\cdot\rvert}, \sin, \cos, \exp]\}$
Interleaving generations g	6
Starting population size	500

Table 8.2: Salient hyper-parameters of GP-GOMEA and their settings. Starred hyper-parameters are subject to grid-search tuning, using the settings reported within curly braces.

Function set. The atomic components that can be instantiated as nodes can be features (of Table 8.1), random constants, or functions from a pre-defined function set. Random constants are sampled uniformly at random between $\{-\rho, +\rho\}$, where $\rho := 5 \times \max(\mathbf{x})$, and $\mathbf{x}$ is the 2D matrix containing all numerical feature values for all patients and plans (in the training set). We considered a simpler function set of algebraic functions ($[+, -, \times, \div_p]$), and a more complex one including trigonometric and transcendental functions ($[+, -, \times, \div_p, \cdot^2, \sqrt{\lvert\cdot\rvert}, \sin, \cos, \exp]\}$). More complex functions can help fit the data better, at the risk of overfitting.

Interleaving generations & starting population size. Whereas typical GP algorithms perform a single evolutionary run, GP-GOMEA uses a scheme of multiple runs of increasing evolutionary budget. Larger-budget runs are started later in the scheme, and executed in an interleaved fashion such that smaller-budget runs perform more iterations (called *generations*) than larger-budget runs. The first run r_1 uses the starting population size and executes g generations before the next run r_2 is started. The run r_2 has a population size that is double of that of r_1 (1000), and performs 1 generation every g generations of r_1. The first time r_2 has executed g generations, r_3 is started, with double the population of r_2 (2000). Subsequently, r_3 will execute 1 generation every time r_2 has executed g generations. If a run converges to all identical machine learning models, then it is terminated. A run is also terminated if a larger-budget run exists that has found a better machine learning model.

As shown in Table 8.2, we used a starting population size of 500 and a number of generations for interleaving $g = 6$. These choices are based on what is reported in Chapter 3, except for the starting population size (set to 50 in the chapter), as the chapter shows that runs with population sizes above a thousand are typically started anyway within the first couple of minutes (Fig. 3.7). Hence, computations performed by runs with smaller population sizes are essentially wasted. We set $g = 6$ because it is the intermediate value (between 4 and 8) considered in Chapter 3, and because GP-GOMEA is fairly robust to the choice of g, i.e., the results for $g = 4$ are similar to the ones of $g = 6$ and $g = 8$.

8.2.5. INDEPENDENT EVALUATION ON CLINICAL PLANS

As aforementioned, the 300 plans used to cross-validate our approach were generated with an automatic sampling procedure (Sec. 8.2.2). To assess whether our results on artificial

plans can be valid for clinically-used plans, we further evaluated our approach on an independent dataset for which clinical plans were crafted manually.

For this validation, we trained ML models (as a reminder, one model per OAR - dose-volume metric combination) on the dataset using the 142 CTs and the 300 artificial plans, and evaluated their prediction accuracy on a separate set of five CTs each associated with two clinical plans. We gathered five clinical plans (three right-sided, two left-sided) for these five CTs. Under the supervision of an experienced pediatric radiation oncologist (B.V. Balgobind), two adapted versions of each plan were manually created that both had the isocenter in the middle of the fields. In one plan no block was used and in the other plan a block was introduced to protect part of the liver or spleen, depending on the plan side. Training (including 5-fold cross-validation to determine the best hyper-parameter settings) and validation were repeated ten times to account for the stochastic nature of GP-GOMEA. Averages and standard deviations were computed over these ten repetitions.

8.3. Results

8.3.1. Dose-volume metric data distribution

Among the 300 artificial plans, plan side and OAR type was found to influence the distribution of a dose-volume metric considerably. To illustrate the effect of OAR type and plan side on the dose, Figure 8.6 shows the distributions found for D_{mean} and $D_{2\text{cc}}$ for the liver and the spleen, in case of left- and right-sided plans. For D_{mean} for the liver, distributions approximately resembling the normal distribution were obtained (in case of right-sided plans with particular high variance and long left tail). The distribution in case of right-sided plans had a mean of 9.5 Gy (typically a major part of the liver was in-field), the distribution in case of left-sided plans had a mean of 3.4 Gy (typically a minor part of the liver was in-field). In terms of $D_{2\text{cc}}$, for the liver we observed values close to the prescribed dose (14.4 Gy) both in case of left- and right-sided plans. The distributions of D_{mean} and $D_{2\text{cc}}$ for the spleen associated with the different plan sides had more marked differences than the ones for the liver. In case of right-sided plans, values close to 0 Gy were obtained for both metrics (typically the spleen was outside the field). For left-sided plans, large values of D_{mean} were found to be much more frequent than low values. The distribution of $D_{2\text{cc}}$ exhibited a peak around the prescribed dose. For the contralateral kidney and for the spinal cord the distributions are similar for both plan sides, as the contralateral kidney should be outside the field and the spinal cord should be included within the field (according to protocol).

For all OARs, distributions obtained for $V_{5\text{Gy}}$ and $V_{10\text{Gy}}$ largely resembled the ones obtained for D_{mean}. In fact, Pearson correlation coefficients above 98% were found when comparing D_{mean} with $V_{5\text{Gy}}$ and $V_{10\text{Gy}}$ for almost all OARs. Smaller (yet still large) correlation coefficients were found between D_{mean} and $V_{10\text{Gy}}$ for the left and right kidney, with values of 96% and 91% respectively.

8.3.2. Validation on artificial plans

For each considered OAR, the Mean Absolute Errors (MAEs) (and standard deviation) at validation time for D_{mean}, $D_{2\text{cc}}$, $V_{5\text{Gy}}$, and $V_{10\text{Gy}}$ from the ten repetitions of the 5-fold cross-validation procedure using the artificial plans are reported in Table 8.3. We further

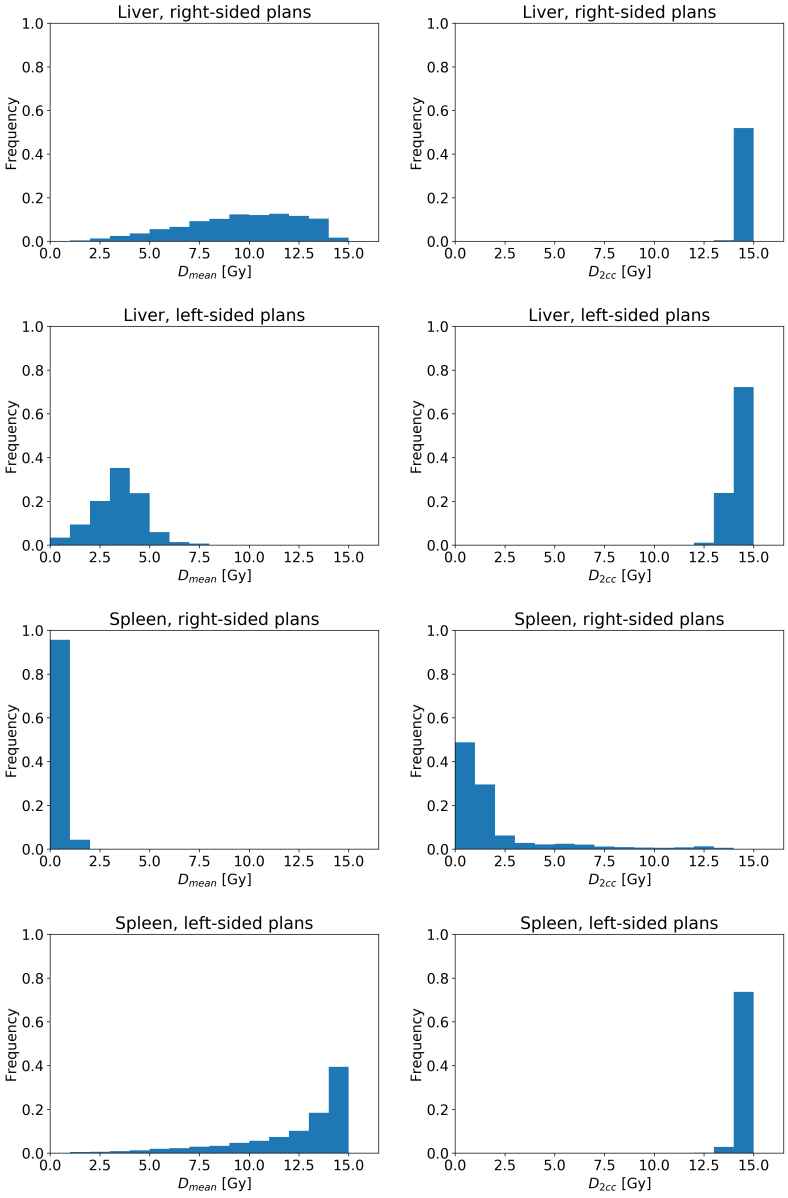

Figure 8.6: Distributions for the liver and the spleen of D_{mean} and $D_{2\text{cc}}$ obtained by the automatic plan sampling procedure used to generate artificial plans and by applying the plans to the CT scans.

present the average decrease in MAE when using ML compared to when using the baseline (i.e., the effect size), as well as the outcome of statistical significance tests (Wilcoxon signed-rank) comparing ML to the baseline.

The errors for D_{mean} and $D_{2\text{cc}}$ were generally below 2 Gy, which corresponds to approximately 14% of the prescribed dose of 14.4 Gy. For all OARs but for the spinal cord, the plan side had considerable impact on the magnitude of the errors. As the spinal cord in RL direction was in-field no matter the plan side, the MAEs of dose-volume metrics predictions were found to be small: < 1 Gy for D_{mean} and $D_{2\text{cc}}$, $< 4\%$ for $V_{5\text{Gy}}$ and $V_{10\text{Gy}}$. For OARs that were almost out-of-field, e.g., the spleen in case of right-sided plans, small MAEs of D_{mean} were found (< 0.1 Gy), as very low values were obtained across all patient-plan combinations (see Fig. 8.6). Note that in this case (and also for the $D_{2\text{cc}}$ for the liver in both left- and right-sided plans), ML performs significantly but not substantially better than the baseline.

For the liver in case of right-sided plans, and for the spleen in case of left-sided plans, larger MAEs were found (liver: 1.7 Gy for D_{mean}, 12.1% for $V_{5\text{Gy}}$, 12.6% for $V_{10\text{Gy}}$; spleen: 1.5 Gy for D_{mean}, 9.3% for $V_{5\text{Gy}}$, 10.7% for $V_{10\text{Gy}}$). These errors can be attributed to the particular configuration of the position of these OARs and the field of the plans.

Among the dose-volume metrics, $D_{2\text{cc}}$ for the (partly) in-field OARs had low variability, with a $D_{2\text{cc}}$ close to the prescribed dose (14.4 Gy). For example, small errors were obtained for the $D_{2\text{cc}}$ for the liver (< 0.4 Gy), as we consistently obtained a large $D_{2\text{cc}}$ value for both left- and right-sided plans (see Fig. 8.6). In contrast, $D_{2\text{cc}}$ was harder to predict when the OAR was contralateral to the plan side. The MAEs obtained for $D_{2\text{cc}}$ for the spleen in case of right-sided plans was 1.6 Gy. This was 2.9 Gy for the left kidney, and 1.4 Gy for the right kidney.

The largest average error was found for $D_{2\text{cc}}$ for the left kidney, amounting to 20% of the prescribed dose. For all dose-volume metrics for the right kidney, and for $D_{2\text{cc}}$ for the spleen, we found that ML predictions were slightly worse compared to using the baseline (note the negative effect sizes), but not significantly so. Lastly regarding the kidneys, although errors in $D_{2\text{cc}}$ were relatively large, errors in $V_{10\text{Gy}}$ were relatively small (compared with $V_{10\text{Gy}}$ for the other OARs). In fact, only a small percentage of the contralateral kidney, from 0 to less than 3% typically received at least 10 Gy.

Although not reported in Table 8.3, we remark that the errors were found to be unbiased: no systematic over- nor under-estimations of dose-volume metrics were found on average, with the mean (non-absolute) error being close to zero for all metrics.

8.3.3. Independent validation on clinical plans

Figure 8.7 and Figure 8.8 show, for each clinical case, the ground truth dose-volume metric values and the predictions obtained by the ML models (trained on the artificial plans). Results for D_{mean} and $D_{2\text{cc}}$ are presented in Figure 8.7, and results for $V_{5\text{Gy}}$ and $V_{10\text{Gy}}$ are presented in Figure 8.8.

The errors in D_{mean} between predictions and ground truth values were generally low, totaling an average of 1.0 Gy (with a range of 0.0-4.9 Gy) across all OARs. Compared to the results obtained in the cross-validation using the artificial plans (Table 8.3), for the liver in case of right-sided plans, the average error on the clinical plans was found to be smaller (1.2 Gy vs. 1.7 Gy). Similar average errors in D_{mean} were found for the kidneys (0.8 Gy vs.

Table 8.3: Mean test MAE ± standard deviation and effect size (in small font) against the baseline (MAE of baseline - MAE of ML), of ten repetitions of 5-fold cross-validation for each OAR and dose-volume metric on the artificial plans. Bold results are significantly better than the baseline (the opposite is never found).

Side	OAR	D_{mean} [Gy]	$D_{2\text{cc}}$ [Gy]	$V_{5\text{Gy}}$ [%]	$V_{10\text{Gy}}$ [%]
Right (22436 plans)	Liver	**1.7 ± 0.2** 0.7	**0.2 ± 0.0** 0.0	**12.1 ± 1.5** 5.1	**12.6 ± 1.8** 5.1
	Spleen	**0.1 ± 0.0** 0.0	1.6 ± 0.5 −0.1	0.9 ± 0.2 0.0	0.4 ± 0.1 0.0
	Left kidney	0.6 ± 0.5 0.1	**2.9 ± 0.2** 0.5	4.4 ± 0.5 0.1	2.5 ± 0.4 0.0
	Spinal cord	**0.4 ± 0.1** 0.7	0.2 ± 0.0 0.0	**3.2 ± 0.4** 5.6	**3.3 ± 0.4** 5.8
Left (20164 plans)	Liver	**0.8 ± 0.0** 0.2	**0.4 ± 0.1** 0.0	**5.8 ± 0.4** 1.1	**5.5 ± 0.3** 1.4
	Spleen	**1.5 ± 0.2** 0.7	0.3 ± 0.0 0.0	**9.3 ± 1.1** 5.2	**10.7 ± 1.4** 6.0
	Right kidney	0.6 ± 1.0 −0.3	1.4 ± 0.4 −0.1	2.3 ± 1.4 −0.4	0.8 ± 0.7 −0.1
	Spinal cord	**0.4 ± 0.0** 0.9	0.2 ± 0.0 0.0	**3.1 ± 0.3** 7.5	**2.9 ± 0.3** 7.4

0.6 Gy for the left kidney and 0.5 Gy vs. 0.6 Gy for the right kidney), the liver in case of right-sided plans (1.0 Gy vs. 1.7 Gy), and the spinal cord (0.4 Gy vs. 0.4 Gy). Larger average errors in D_{mean} were found for the spleen (0.5 Gy vs. 0.1 Gy in case of the right-sided plans and 3.6 Gy vs. 1.5 Gy in case of left-sided plans). The largest error of 4.9 Gy was found for the spleen of case P_{L2B} (1.9 Gy error found for the spleen of case P_{L2}), indicating that the impact of the block on the plan was not well modeled. Furthermore, the error in spleen D_{mean} of P_{L1} and P_{L1B} was large (2.8 Gy and 4.7 Gy, respectively), indicating both field types (without and with a block, respectively) were not well modeled for this case (see the discussion in Sec. 8.4).

Regarding $D_{2\text{cc}}$, similar results to the ones obtained on artificial plans were found for the spinal cord, where an average error of 0.1 Gy was obtained in $D_{2\text{cc}}$ (with a range of 0.0-0.4 Gy). For the spleen of cases P_{R1} and P_{R1B}, large errors in $D_{2\text{cc}}$ were found (12.3 Gy on average), as ML predictions essentially wrongly represented the spleen to be out-of-field. Whereas this was the case for P_{R2} and P_{R2B}, and for P_{R3} and P_{R3B}, where small errors were obtained (1.1 Gy on average). Similarly, large errors in $D_{2\text{cc}}$ were found in some cases for the contralateral kidneys (e.g., the left kidney: 5.1 Gy on average, with a range of 0.5-7.6 Gy).

Results for $V_{5\text{Gy}}$ (average error 8% with a range of 0-35%) and $V_{10\text{Gy}}$ (average error 7% with a range of 0-49%) mostly followed the trend of the errors for D_{mean}, as these metrics were found to be correlated for most OARs.

8.4. Discussion

In this chapter we presented a new and different paradigm in organ dose reconstruction. By leveraging the modeling power of ML, we showed how patient and plan features can be used to predict organ dose-volume metrics directly, without the need of adopting a surrogate anatomy. Once the ML models are trained, they can readily be used to compute dose-volume metric predictions for a new historical patient and plan, by using their features as input.

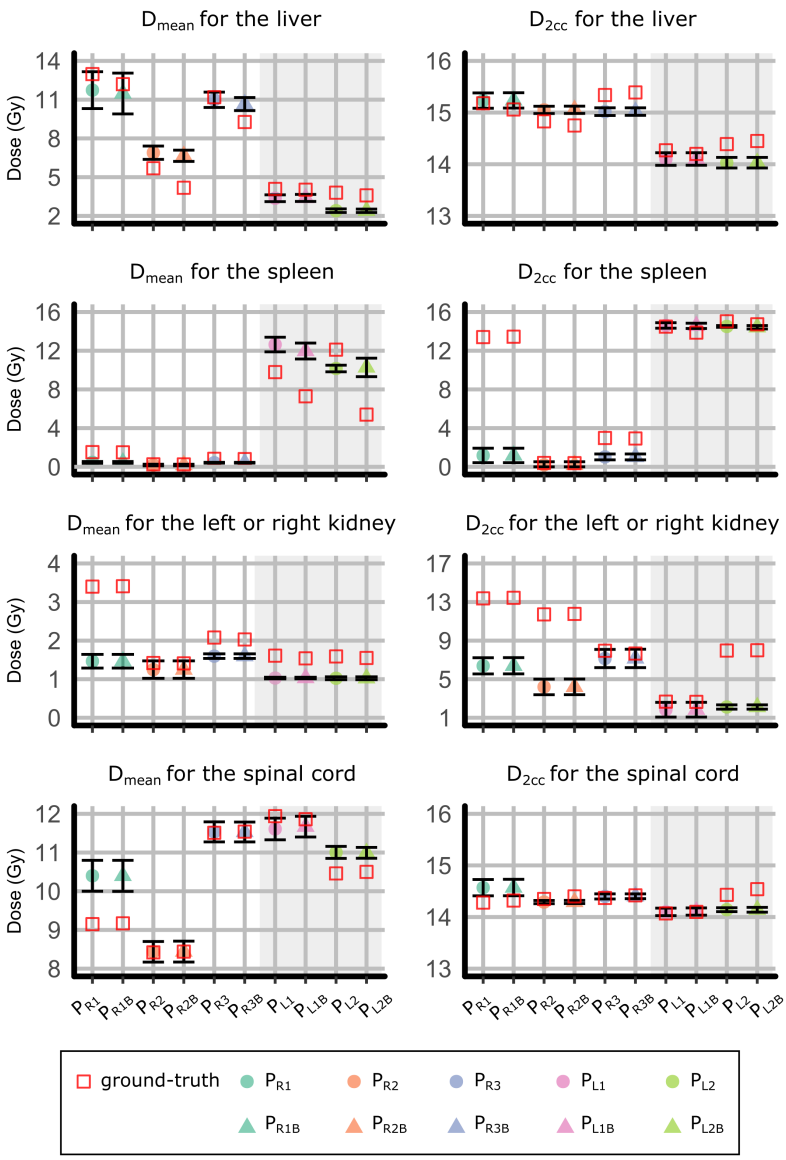

Figure 8.7: Mean and standard deviation of predictions across ten repetitions for the dose-volume metrics D_{mean} and $D_{2\text{cc}}$ of the ten clinical plans. A plan-patient combination is encoded by color, and presence or absence of blocking is encoded by marker shape. Note that different plots have different scales of the vertical axis. For each case, the ground-truth dose-volume metric is indicated by a red square. In each plot, the first six cases (white background, plan subscripts starting with 'R') are right-sided plans, the last four cases (gray background, plan subscripts starting with 'L') are left-sided plans.

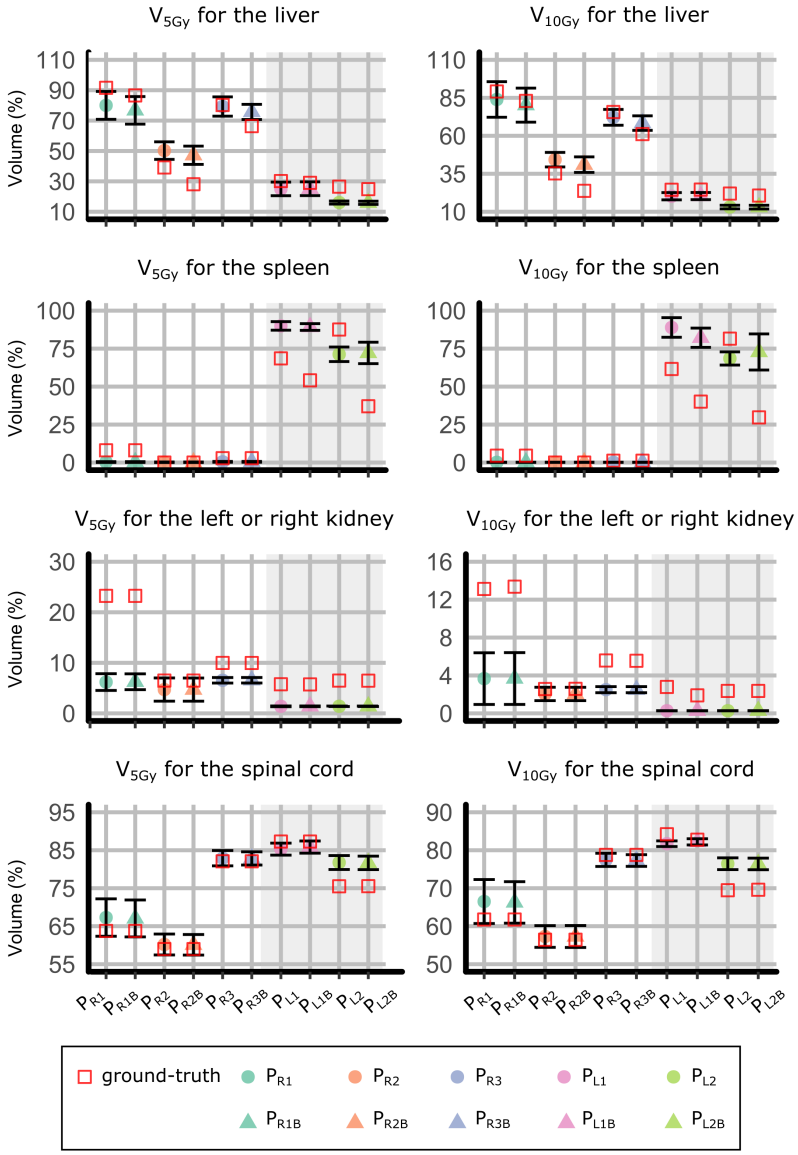

Figure 8.8: Mean and standard deviation of predictions across ten repetitions for the dose-volume metrics $V_{5\text{Gy}}$ and $V_{10\text{Gy}}$ of the ten clinical plans. A plan-patient combination is encoded by color, and presence or absence of blocking is encoded by marker shape. Note that different plots have different scales of the vertical axis. For each case, the ground-truth dose-volume metric is indicated by a red square. In each plot, the first six cases (white background, plan subscripts starting with 'R') are right-sided plans, the last four cases (gray background, plan subscripts starting with 'L') are left-sided plans.

Key to obtaining a decent amount of data to perform ML were the collaboration of five international institutes to gather pediatric patient CTs (147), the development of a new automatic sampling procedure yielding artificial Wilms' tumor RT plans, and the creation of an automatic dose reconstruction pipeline to calculate the dose for all patient and plan combinations. We validated our approach on 300 automatically generated artificial plans, and on ten manually created clinical plans, to assess whether the results of the validation on the artificial plans generalize well in practice. Our approach showed promising levels of accuracy in dose reconstruction in both settings.

Errors were found to be overall similar between the validation on the ten clinical plans and the validation on the artificial plans. However, for some metrics, errors were larger for the ten clinical plans. This may be due to chance, because ten is a small number to validate upon. Another possibility is that the artificial plan generation method needs to be improved. Artificial plans were generated by sampling geometry properties *uniformly* within predefined boundaries on two reference DRRs. Uniform sampling might not be representative of the distribution clinical plans have. Moreover, we consulted a single radiation oncologist to define clinically acceptable boundaries to use in the sampling of artificial plans. Consulting multiple experts and allowing for a larger variation might better help covering the extent of variation that is present in historical plans (Sec. 8.2.2). For example, the isocenter locations of artificial left-sided plans were never sampled below the 1st lumbar vertebra (see Fig. 8.2) and approximately half of the D_{mean} values for the spleen in case of the artificial left-sided plans were close to the prescribed dose (14.4 Gy, see Fig. 8.6), which means that the spleen was often almost completely in-field in our artificially generated set of left-sided plans. When a block was applied, only a small part of the spleen was spared. However, in clinical practice, isocenter locations can be lower, and a larger part of the spleen might actually be outside the field (see Fig. 8.9). This might explain the relatively large errors observed in Figure 8.7 for P_{L1B} and P_{L2B} where the isocenter location is lower than the sampled range. Ultimately, effort should be done to improve the sampling of artificial plans.

In the validation performed upon artificial plans as well as in the one performed upon clinical plans, a main result that emerges is that dose-volume metrics for an organ are hard to predict when, due to the field setup, it is unclear whether the OAR is (partially) included in the field or not. For example, consider the $D_{2\text{cc}}$ as opposed to the D_{mean} for the spleen for P_{R1} and P_{R1B} in Figure 8.7 a tiny part of the spleen being inside the field causes a large $D_{2\text{cc}}$ (wrongly predicted to be small), and small D_{mean} (correctly predicted to be small). As experimentally observed in Chapters 6 and 7, 2D bony anatomy provides only coarse information on OAR shape and position even for ML algorithms (e.g., an MAE of 6.4 mm for the prediction of the liver position along the IS axis was reported in Chapter 7). Yet, because bony anatomy is the only structure that is reliably visible in historical radiographs, most of the anatomical features rely on it. Patients with similar anatomical features derived from bony anatomy may have different OAR shape and position, and thus different dose-volume metrics. Furthermore, impreciseness in feature values due to e.g., uncertainties in landmark detection and plan emulation, aggravate the situation.

Compared to conventional dose reconstruction methods (that use surrogate anatomies and heuristics to decide what surrogate to use), we considered a relatively large number of features: 33. Phantom-based methods consider, e.g., only age and gender [7, 21], or gender

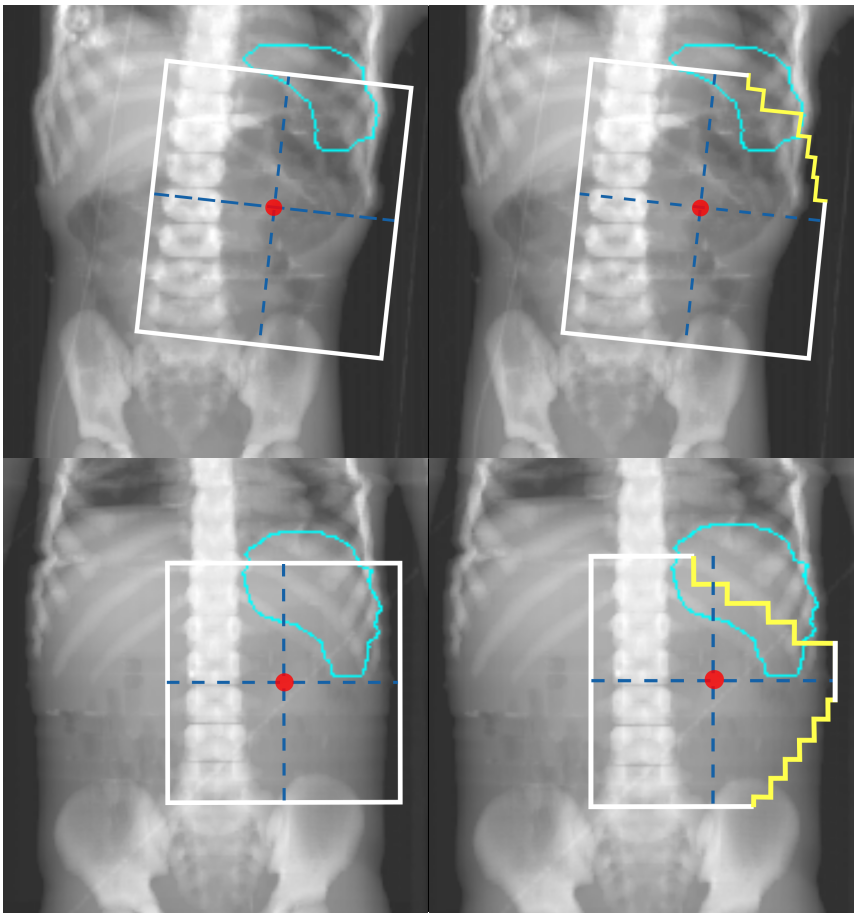

Figure 8.9: Effective field shape of plans P_{L1} (on the upper left), P_{L1B} (on the upper right), P_{L2} (on the lower left) and P_{L2B} (on the lower right) plotted on top of the associated DRR. The fields are placed lower than most of the sampled artificial plans (see Fig. 8.2) and consequently a (large) part of the spleen (indicated by the light blue contours) is outside the field (for P_{L1B} and P_{L2B} an even larger part of the spleen was blocked).

together with height and weight percentiles [15]. However, if a 2D radiograph is available, the added value of this information should be exploited. In our work, the majority of the features we considered, i.e., 23 out of 33 (minus eight due to automatic feature selection), regarded patient anatomy as visible on a 2D radiograph, which we simulated with DRRs. Our DRRs were generated in a conformal fashion, e.g., the abdomen was always fully included in RL direction. The automatic landmark detection that was used to generate features expects this conformity to achieve precise detection [25]. When dealing with actual historical radiographs, however, several challenges need to be taken into account. For example, our automatic landmark detection method requires further development to account for noise in the radiograph (e.g., the presence of hand-writing on the radiograph). Moreover, educated guesses of landmark locations may be needed in some cases, as some historical radiographs do not include the entire abdomen (see Fig. 8.1(c)). Nevertheless, as long as the features are somehow collected (e.g., manually), they can be used as input for the ML models to get respective dose-volume metric predictions.

There are disadvantages of our approach compared to conventional dose reconstruction methods that use surrogate anatomies beyond the need for patient radiographs, which are not always available in retrospective data. In particular, a key limitation is that ML models do not predict the entire 3D dose distribution an organ receives, but only the metrics they were trained for. Potentially useful information to link to AEs may be contained in 3D dose distributions. To predict 3D dose distributions, the ML models would need to be trained to predict a 3D output. Surrogate-based methods do allow to obtain the entire 3D dose distribution to an organ, since the distribution can be visualized on the organ of the surrogate anatomy, after plan simulation. However, considering the magnitude and variations of the errors of organ mean dose obtained by conventional approaches [22], it is questionable whether the full 3D distribution will be sufficiently reliable. Our approach, as currently proposed, can straightforwardly be extended to predict any (scalar) dose-volume metric that is suspected to be useful to study AEs (it suffices to train ML on that metric).

Another limitation of our approach is that it does not take into consideration uncertainties related to OAR motion. For validation, we aimed at reconstructing the dose based on the particular snapshot of anatomy at the moment the CT (ground-truth) / respective DRR (to simulate historical radiographs) was taken. Yet, OAR motion plays a key role in the uncertainty of organ positioning at the edge of the field, which can lead to a discrepancy between the planned dose and the actual delivered dose. In RT practice, radiation delivery is performed over a number of days, with fractionation schemes. The OAR position can therefore vary (i.e., inter-fractional position variation). Intra-fractional organ motion due to, e.g., respiration variation, contributes to the difference between planned dose and delivered dose as well [30].

Lastly, a main limitation of our approach is that the ML models we generated are specific to pediatric patients (1 to 8 years of age) and Wilms' tumor RT plans: they can only predict reliable dose-volume metrics of specific OARs they were explicitly trained for. The RT plans we have sampled were also restricted to a standard AP-PA setup without considering wedges, boost fields, or other radiation sources such as Cobalt-60. Moreover, the predictions of the ML models (as well as the validation performed in this study) are based on the dose calculation algorithm we adopted when preparing training data, which has inaccuracies. Specifically, we used a collapsed cone algorithm available in Oncentra

TPS. Though good accuracy was reported in the in-field and near-field region ($<$ 5 cm from the field borders, achieves an error of 1-2% of the prescribed dose), in low dose regions (10-15 cm from the field border) an underestimation of 10% of the dose in the region was reported [31]. We remark that the OARs we considered in this study were mostly within 5 cm near the field border (except for the spleen in case of the right-sided plans). To make the method more general for OARs far from field borders, more advanced Monte Carlo dose calculation algorithms should be applied in future implementations. However, we believe that the core ideas of our work can be replicated for other cohorts and other types of plans. Essentially, as long as a sufficient number of anatomies and plans are collected or generated, and a large number of dose reconstructions are performed to be used as examples, new ML models can be trained to predict how the dose-volume metrics are linked to anatomy-plan configurations. As was the case in our study, the collection and preparation of sufficient data for ML is likely to be the largest required effort.

Our proposed approach presents several advantages compared to traditional dose reconstruction methods. First of all, we found our validation results to compare favorably with respect to our recent work considering dose reconstruction for a similar childhood cancer cohort [22]. The work considered 31 patients aged 2 to 6, 12 Wilms' tumor clinical plans, and a total of 50 dose reconstruction combinations, which were performed by matching a surrogate CT based on age and gender. The work reported an MAE for the D_{mean} for the liver of 1.6 Gy (average across both left- and right-sided plans), and an MAE for the D_{mean} for the spleen of 2.6 Gy. For the liver, we obtained an MAE of 1.3 Gy when validating on artificial plans, and of 1.1 Gy when validating on ten clinical plans. For the spleen, we obtained an MAE of 0.8 Gy on artificial plans, and of 1.7 Gy for the clinical plans. Furthermore, our ML-based predictions resulted in much smaller variations. The inter-quantile range (25th to 75th percentile) of the (non-absolute) prediction error of our previous work was 3.6 Gy for the liver, and 4.7 Gy for the spleen [22]. On the artificial plans, we obtained a range of 2.0 Gy for the liver, and of 1.2 Gy for the spleen. On the clinical plans, the range for the liver was 1.9 Gy, and the one for the spleen was 2.2 Gy. We remark that since the dose reconstruction accuracy is largely influenced by the particular plans considered, these values may not be a fair comparison. We are currently working on a multi-institute study to compare our approach with two state-of-the-art, phantom-based dose reconstruction approaches [12, 21]. In that study, a same set of patients and plans will be used for validation.

Finally, a benefit of having ML models is that, once features are collected, they can be used as inputs for the model to obtain the prediction of a dose-volume metric immediately. Running a model on a computer simply means to follow the steps encoded by the formula the model represents, which takes a few milliseconds. Conversely, in a surrogate-based approach, the features are used to craft or select a surrogate anatomy. Then, effort and time must be put to emulate the plan on the surrogate anatomy, calculate the dose, and obtain the dose-volume metrics [12, 21, 25].

8.5. Conclusion

We presented the first surrogate-free organ dose reconstruction method based on ML. Our method was enabled by the collection of large amounts of patient and CT data, and the automatic generation of artificial plans and of dose distribution data. We assembled

a dataset of dose-volume metrics corresponding to features of patient anatomy and plan geometry, and subsequently trained ML models to predict how features of patient anatomy and of treatment plans influence dose-volume metrics. The predictions were validated upon both artificial and clinical RT plans, and achieved good accuracy in both cases.

Acknowledgments

The authors acknowledge Stichting Kinderen Kankervrij (KiKa) for financial support (project #187), and the Maurits and Anna de Kock foundation for financing a high performance computing system. Elekta is acknowledged for providing the research software ADMIRE for automatic segmentation. The authors thank Abigail Bryce-Atkinson for her help in data preparation and feature extraction, and Dr. Cécile M. Ronckers for sharing her expertise.

References

[1] H. Birgisson, L. Pahlman, U. Gunnarsson, and B. Glimelius, *Adverse effects of preoperative radiation therapy for rectal cancer: long-term follow-up of the Swedish rectal cancer trial,* Journal of Clinical Oncology **23**, 8697 (2005).

[2] I. W. E. M. van Dijk, F. Oldenburger, M. C. Cardous-Ubbink, M. M. Geenen, R. C. Heinen, J. de Kraker, F. E. van Leeuwen, H. J. van der Pal, H. N. Caron, C. C. Koning, *et al.*, *Evaluation of late adverse events in long-term Wilms' tumor survivors,* International Journal of Radiation Oncology · Biology · Physics **78**, 370 (2010).

[3] Y. T. Cheung, T. M. Brinkman, C. Li, Y. Mzayek, D. Srivastava, K. K. Ness, S. K. Patel, R. M. Howell, K. C. Oeffinger, L. L. Robison, *et al.*, *Chronic health conditions and neurocognitive function in aging survivors of childhood cancer: A report from the childhood cancer survivor study,* JNCI: Journal of the National Cancer Institute **110**, 411 (2017).

[4] E. Donovan, N. Bleakley, E. Denholm, P. Evans, L. Gothard, J. Hanson, C. Peckitt, S. Reise, G. Ross, G. Sharp, *et al.*, *Randomised trial of standard 2D radiotherapy (RT) versus intensity modulated radiotherapy (IMRT) in patients prescribed breast radiotherapy,* Radiotherapy and Oncology **82**, 254 (2007).

[5] F. Y. Feng, H. M. Kim, T. H. Lyden, M. J. Haxer, M. Feng, F. P. Worden, D. B. Chepeha, and A. Eisbruch, *Intensity-modulated radiotherapy of head and neck cancer aiming to reduce dysphagia: early dose–effect relationships for the swallowing structures,* International Journal of Radiation Oncology · Biology · Physics **68**, 1289 (2007).

[6] T. Bölling, I. Ernst, H. Pape, C. Martini, C. Rübe, B. Timmermann, K. Fischedick, R.-D. Kortmann, and N. Willich, *Dose–volume analysis of radiation nephropathy in children: Preliminary report of the risk consortium,* International Journal of Radiation Oncology · Biology · Physics **80**, 840 (2011).

[7] M. Stovall, R. Weathers, C. Kasper, S. A. Smith, L. Travis, E. Ron, and R. Kleinerman, *Dose reconstruction for therapeutic and diagnostic radiation exposures: use in epidemiological studies,* Radiation Research **166**, 141 (2006).

[8] D. Verellen, M. De Ridder, and G. Storme, *A (short) history of image-guided radiotherapy,* Radiotherapy and Oncology **86**, 4 (2008).

[9] A. Ng, K. K. Brock, M. B. Sharpe, J. L. Moseley, T. Craig, and D. C. Hodgson, *Individualized 3D reconstruction of normal tissue dose for patients with long-term follow-up: a step toward understanding dose risk for late toxicity,* International Journal of Radiation Oncology · Biology · Physics **84**, e557 (2012).

[10] W. M. Leisenring, A. C. Mertens, G. T. Armstrong, M. A. Stovall, J. P. Neglia, J. Q. Lanctot, J. D. Boice Jr, J. A. Whitton, and Y. Yasui, *Pediatric cancer survivorship research: experience of the childhood cancer survivor study,* Journal of Clinical Oncology **27**, 2319 (2009).

[11] X. G. Xu, *An exponential growth of computational phantom research in radiation protection, imaging, and radiotherapy: a review of the fifty-year history,* Physics in Medicine & Biology **59**, R233 (2014).

[12] C. Lee, J. W. Jung, C. Pelletier, A. Pyakuryal, S. Lamart, J. O. Kim, and C. Lee, *Reconstruction of organ dose for external radiotherapy patients in retrospective epidemiologic studies,* Physics in Medicine & Biology **60**, 2309 (2015).

[13] V. F. Cassola, F. M. Milian, R. Kramer, C. A. B. de Oliveira L O V B, and H. J. Khoury, *Standing adult human phantoms based on 10th, 50th and 90th mass and height percentiles of male and female Caucasian populations,* Physics in Medicine & Biology **56**, 3749 (2011).

[14] W. P. Segars, J. Bond, J. Frush, S. Hon, C. Eckersley, C. H. Williams, J. Feng, D. J. Tward, J. T. Ratnanather, M. I. Miller, *et al.*, *Population of anatomically variable 4D XCAT adult phantoms for imaging research and optimization,* Medical Physics **40**, 043701 (2013).

[15] A. M. Geyer, S. O'Reilly, C. Lee, D. J. Long, and W. E. Bolch, *The UF/NCI family of hybrid computational phantoms representing the current US population of male and female children, adolescents, and adults-application to CT dosimetry,* Physics in Medicine & Biology **59**, 5225 (2014).

[16] J. V. Bezin, R. S. Allodji, J.-P. Mège, G. Beldjoudi, F. Saunier, J. Chavaudra, E. Deutsch, F. de Vathaire, V. Bernier, C. Carrie, *et al.*, *A review of uncertainties in radiotherapy dose reconstruction and their impacts on dose–response relationships,* Journal of Radiological Protection **37**, R1 (2017).

[17] J. Valentin, *Basic anatomical and physiological data for use in radiological protection: reference values: ICRP publication 89,* Annals of the ICRP **32**, 1 (2002).

[18] G. L. de la Grandmaison, I. Clairand, and M. Durigon, *Organ weight in 684 adult autopsies: new tables for a Caucasoid population,* Forensic Science International **119**, 149 (2001).

[19] M. Virgolin, I. W. E. M. van Dijk, J. Wiersma, C. M. Ronckers, C. Witteveen, A. Bel, T. Alderliesten, and P. A. N. Bosman, *On the feasibility of automatically selecting similar patients in highly individualized radiotherapy dose reconstruction for historic data of pediatric cancer survivors,* Medical Physics **45**, 1504 (2018).

[20] Z. Wang, B. V. Balgobind, M. Virgolin, I. W. E. M. van Dijk, J. Wiersma, C. M. Ronckers, P. A. N. Bosman, A. Bel, and T. Alderliesten, *How do patient characteristics and anatomical features correlate to accuracy of organ dose reconstruction for Wilms' tumor radiation treatment plans when using a surrogate patient's CT scan?* Journal of Radiological Protection **39**, 598 (2019).

[21] R. M. Howell, S. A. Smith, R. E. Weathers, S. F. Kry, and M. Stovall, *Adaptations to a generalized radiation dose reconstruction methodology for use in epidemiologic studies: An update from the MD Anderson late effect group,* Radiation Research **192**, 169 (2019).

[22] Z. Wang, I. W. E. M. van Dijk, J. Wiersma, C. M. Ronckers, F. Oldenburger, B. V. Balgobind, P. A. N. Bosman, A. Bel, and T. Alderliesten, *Are age and gender suitable matching criteria in organ dose reconstruction using surrogate childhood cancer patients' CT scans?* Medical Physics **45**, 2628 (2018).

[23] P. Mishra, R. Li, S. S. James, R. H. Mak, C. L. Williams, Y. Yue, R. I. Berbeco, and J. H. Lewis, *Evaluation of 3D fluoroscopic image generation from a single planar treatment image on patient data with a modified XCAT phantom,* Physics in Medicine & Biology **58**, 841 (2013).

[24] M. M. Van Den Heuvel-eibrink, J. A. Hol, K. Pritchard-Jones, H. Van Tinteren, R. Furtwängler, A. C. Verschuur, G. M. Vujanic, I. Leuschner, J. Brok, C. Rübe, *et al.*, *Position paper: rationale for the treatment of Wilms tumour in the UMBRELLA SIOP–RTSG 2016 protocol,* Nature Reviews Urology **14**, 743 (2017).

[25] Z. Wang, M. Virgolin, P. A. N. Bosman, K. F. Crama, B. V. Balgobind, A. Bel, and T. Alderliesten, *Automatic generation of three-dimensional dose reconstruction data for two-dimensional radiotherapy plans for historically treated patients,* Journal of Medical Imaging **7**, 1 (2020).

[26] B. Emami, J. Lyman, A. Brown, L. Cola, M. Goitein, J. E. Munzenrider, B. Shank, L. J. Solin, and M. Wesson, *Tolerance of normal tissue to therapeutic irradiation,* International Journal of Radiation Oncology · Biology · Physics **21**, 109 (1991).

[27] L. S. Constine, C. M. Ronckers, C.-H. Hua, A. Olch, L. C. M. Kremer, A. Jackson, and S. M. Bentzen, *Pediatric Normal Tissue Effects in the Clinic (PENTEC): an international collaboration to analyse normal tissue radiation dose–volume response relationships for paediatric cancer patients,* Clinical Oncology **31**, 199 (2019).

[28] C. M. Bishop, *Pattern recognition and machine learning* (Springer, 2006).

[29] M. Virgolin, T. Alderliesten, A. Bel, C. Witteveen, and P. A. N. Bosman, *Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2018) pp. 1395–1402.

[30] S. C. Huijskens, I. W. E. M. van Dijk, R. de Jong, J. Visser, R. D. Fajardo, C. M. Ronckers, G. O. Janssens, J. H. Maduro, C. R. Rasch, T. Alderliesten, *et al.*, *Quantification of renal and diaphragmatic interfractional motion in pediatric image-guided radiation therapy: a multicenter study,* Radiotherapy and Oncology **117**, 425 (2015).

[31] T. Krieger and O. A. Sauer, *Monte Carlo-versus pencil-beam-/collapsed-cone-dose calculation in a heterogeneous multi-layer phantom,* Physics in Medicine & Biology **50**, 859 (2005).

9

Concluding Discussion

With our society increasingly leveraging big data and machine learning to drive informed decisions, it is imperative to design scalable machine learning algorithms, preferably such that the outcomes of these algorithms can be understood by humans. The work in this thesis provides steps to improve Genetic Programming (GP) in this direction, and presents practical applications of the resulting algorithms. Ideas from modern Optimal Mixing Evolutionary Algorithms (OMEAs) for discrete optimization have been harnessed, ported to GP, and improved, to perform better-than-random variation, leading to GP-GOMEA. The findings show that not only can GP-GOMEA solve a variety of synthetic benchmark problems in a scalable fashion, it can also deal competitively with the real-world machine learning task of symbolic regression when small expressions are desired. Moreover, GP-GOMEA has been found to be particularly useful for real-world applications in the health domain. In this final chapter, merits and limitations of the findings, possibilities for future work, and also potential implications for society are discussed with respect to the research questions posed in the introduction.

Structure of the discussion

The discussion of this thesis is structured by first providing answers to the research questions posed in Section 1.5. These answers will highlight the merits obtained and the major limitations encountered. The discussion proceeds by presenting general ramifications of the results of this thesis with respect to its main goal, i.e., *improving GP to be more efficient and effective in synthesizing programs/machine learning models*, with the focus of *obtaining relatively small models, to increase the chances of them being interpretable*. These ramifications will include limitations and future work directions for the scientific community, as well as implications for society.

9.1. Answers to the research questions

Research question 1. How can an OMEA be designed for GP, and how does it fare against state-of-the-art GP algorithms?

In Chapter 2 we showed that the concepts behind OMEAs can indeed be extended to GP. We created *GP-GOMEA*, a GP version of the GOMEA family of algorithms. To realize GP-GOMEA, we enforced program structure to adhere to a fixed, maximal template, so that linkage learning could be performed. The use of a fixed template was not found to necessarily be a strict limitation, because it can be expanded over time when runs are started by the Interleaved Multistart Scheme (IMS).

Experimental results on well-known benchmark problems showed that GP-GOMEA can achieve at least competitive scalability when compared with state-of-the-art GP algorithms. In particular, GP-GOMEA scaled better than other algorithms when the problem had an inherent structure that could be exploited if identified. Moreover, optimally-performing programs obtained with GP-GOMEA were found to normally be one order of magnitude smaller (in number of tree nodes that encode the program) than the optimal programs obtained with the other tested GP algorithms. GP-GOMEA can thus be seen as a promising approach with respect to the goal of this thesis, i.e., improving the efficiency and effectiveness of GP, with a focus on synthesizing relatively small programs (to increase interpretability chances).

While the work presented in Chapter 2 provides evidence that answers positively to the research question, a gap still exists with respect to reaching the main goal of the thesis. This is due to two major limitations:

- The benchmark problems that were considered in Chapter 2 require an *optimal* program to be found (see Sec. 2.4.1). In this context, a program is optimal if it embodies a particular structure (for the so-called artificial benchmark problems), or if it provides the correct output to *all possible* input cases (for the so-called Boolean benchmark problems). For supervised learning problems such as symbolic regression on real-world data, no conditions for program optimality (or, equivalently in this context, for machine learning models) are known. More specifically, no particular program structure is required, nor are all conceivable input/output pairs available (the data is limited) to test whether the program (or model) perfectly captures the intrinsic relationships in the data. Moreover, for realistic data, there are no guarantees that the optimal problem structure exists in the program encoding. Hence, a main limitation of our answer to this research question is that validating the performance

of GP-GOMEA on the considered benchmark problems may not be sufficiently representative of GP-GOMEA's performance on real-world symbolic regression (and supervised learning in general). However, since GP-GOMEA was shown to be able to solve benchmark problems efficiently and to find smaller programs than the other GP algorithms, it can be expected that GP-GOMEA will have a better performance when searching for a small program that represents a model that fits some real world data (investigated in the next research question).

- The programs found by GP-GOMEA were typically one order of magnitude smaller than the programs found by the other algorithms, but they still contained a relatively large number of instructions, i.e., over one hundred for the largest problem instances. For a Boolean problem, reading a program containing one hundred `AND`, `OR`, `NAND`, and `NOR` instructions can be considered unreasonable, and will likely not help in obtaining a full understanding of its workings (to prevent, e.g., unexpected outcomes). Still, it is reasonable to hypothesize that GP-GOMEA's capability of consistently finding relatively small optimal programs will help the algorithm deal proficiently with scenarios where the programs are *forced* to be small, and in particular small enough to make readability and interpretability much more likely.

Research question 2. Does GP-GOMEA work well on realistic cases of symbolic regression?

An answer to this research question is provided in Chapter 3. GP-GOMEA was used to synthesize programs that represent machine learning models for realistic symbolic regression datasets (of moderate dimensionality), in the form of mathematical formulas composed of arithmetic operations. To consider realistic datasets means to overcome the first of the two major limitations identified for the answer to the first research question, i.e., that benchmark problems might not well represent real-world supervised learning tasks.

To overcome the second major limitation of our answer to the first research question, we enforced relatively strict limits on the maximum number of instructions allowed in a program. We performed experiments with limits of 15, 31, and 63 instructions (perfect binary trees of height 3, 4, and 5). This was done to increase the chances of interpretability. In our opinion, symbolic regression programs with around 15 to 25 instructions (as arithmetic functions) can already be found to be borderline interpretable when re-written as mathematical formulas, depending on what instructions are used and how they are combined. Larger programs are arguably very hard to impossible to interpret.

We provided methodological improvements to GP-GOMEA to make it better suited to deal with supervised learning tasks. Subsequent experimental results on ten realistic datasets showed that GP-GOMEA was overall preferable to the algorithms it was compared with in terms of model accuracy. This leads us to give a positive answer to research question 2, and to substantially strengthen the hypothesis that, indeed, efficiency and effectiveness of GP can be improved for applications of real-world interest, by using concepts of OMEAs within GP.

The major limitations of our answers and results that can be identified are:

- GP-GOMEA uses a maximal tree template. A template was configured to be a *perfect binary tree* capable of containing at most a predefined maximum number of nodes

(e.g., for a limit of 31 nodes a template tree with height of 4 was used). In comparison to classic GP, with the latter forced to use the same maximal program structure enforced by the template used by GP-GOMEA, GP-GOMEA was found to be markedly better than classic GP. This could be expected because using a same maximal template means to set same search space conditions for the two algorithms, and differences in performance must be attributed to competence in searching. However, when classic GP was set to evolve tree structures freely (e.g., unbalanced maximum-sized trees), then GP-GOMEA was found to be only moderately better than classic GP. This means that being able to evolve other maximal tree shapes than a specific one (i.e., a perfect binary tree) can bring an advantage to fit the data even if variation is inefficient. Therefore, it can be concluded that the need to use a specific maximal template in GP-GOMEA (to be able to perform linkage learning and optimal mixing) is an important limitation.

- In our experimental evaluation, we also considered regression decision trees [1], as an alternative approach to obtain interpretable machine learning models [2]. Albeit efficient compared to other GP algorithms, GP-GOMEA still remains a relatively expensive approach. Since a pre-established termination condition based on quality is typically unknown in realistic problems (no optimum is defined), GP-GOMEA requires a different type of termination criterion. In Chapter 3, we used a time limit of ten minutes, which can be considered a reasonable investment of time. Conversely, building a decision tree takes a well defined amount of computations (time complexity of $O(m \log n)$ for m features and n samples), which amounted to a few seconds for the datasets considered (including hyper-parameter tuning by grid search). We remark, however, that GP-GOMEA (with base parameter settings) was found to be better than (tuned) decision trees in terms of the accuracy of the models.

Research question 3. Does geometric semantic variation work well on realistic cases of symbolic regression?

In Chapter 4, the Random Desired Operator (RDO), one of the most famous approximate geometric semantic variation operators, was found *not* to be capable of dealing with symbolic regression for real-world data. Consequently, we provided enhancements to RDO that actually made it become markedly more efficient and effective, and capable of dealing with realistic cases of symbolic regression in a competitive manner.

Still, we did not address one of the main limitations of (approximate) geometric semantic variation, which is that this type of variation induces programs to grow large in the number of instructions. Actually, in Chapter 3 we showed that RDO does not work well when forced to evolve small programs. This limitation provides a further motivation to explore and use algorithms like GP-GOMEA, that can be considered to be complementary in that they are better suited to deal with scenarios where particularly compact programs are desired, e.g., to enhance the chances of interpretability.

Research question 4. Can the capability of GP-GOMEA to find small and accurate programs be used to make a machine learning algorithm generate transparent models?
We showed in Chapter 5 that GP-GOMEA can be used to perform feature construction in such a way that: (i) Constructed features can be particularly small, to allow interpretation of their meaning; (ii) Constructed features can be small in number, to allow the behavior (input-output mapping) of a model learned by a machine learning algorithm to be visualized in terms of these features; and (iii) The performance of a machine learning model that is trained on such constructed features can be on par with the performance of a model that is trained on the original feature set (actually, it can even be better). These aspects can help explaining how the model learned by a machine learning algorithm behaves and why it behaves that way.

The use of feature construction was experimentally found to be particularly beneficial for "weak" machine learning algorithms, such as the naive Bayes classifier and ordinary linear regression, which are relatively fast to train but make particular assumptions on properties of the features (e.g., linear contribution to the target). The feature construction process can forge features that themselves can overcome the limitations of the assumptions made so as to enable better performance. GP-GOMEA was found to be moderately preferable in performing feature construction when compared to other search algorithms.

The major limitations encountered in Chapter 5 are reported below.

- We could only use a limited population size (100) and a small budget of allowed evaluations (10,000) for the feature construction algorithms, due to the expensiveness of carrying out the evaluation of feature construction quality, and the large number of experiments considered in the study. These settings were found insufficient to allow a proficient use of linkage learning by GP-GOMEA. In fact, we found that using linkage learning (by means of the linkage tree) with such limited resources was not helpful, or even harmful, compared to using random variation (by means of the random tree). In the discussion of Chapter 5, however, we provided experimental evidence of our expectation that, if the dataset under consideration has moderate dimensionality, and larger population sizes and number of evaluations are used, then linkage learning can be advantageous.

- Similarly to Chapter 3, we enforced the constructed features to be small to enhance the chances of them being interpretable. Still, we had no means to guarantee interpretability. We provided examples of constructed features encoded as mathematical formulas that are arguably easy to read and understand (corresponding to programs with at most 7 instructions), and of some that are borderline interpretable (corresponding to programs with at most 31 instructions).

Research question 5. Can GP-GOMEA find models that are accurate and likely to be interpretable for a clinical problem in pediatric radiation oncology?
The answer to this research question spans Chapter 7 and Chapter 8. Chapter 6 is part of the application being considered in that it introduces the problem of radiation dose reconstruction for childhood cancer survivors, and in that it shows the feasibility of using machine learning to reconstruct 3D anatomical information from patient features available

in historical records. However, Chapter 6 does not involve GP-GOMEA, hence it is not discussed further here.

For Chapter 7 and 8, actually, it can be argued that the particular choice of using GP-GOMEA is not instrumental to the application. We believe it is fair to say that the major scientific contribution that we provided for this application consisted in showing that the dose reconstruction problem (or a part of it, such as the phantom construction step) can be cast as a supervised learning task (regression), which can be tackled using machine learning. Still, particularly for Chapter 7 where comparisons with other machine learning algorithms were made, we did show that GP-GOMEA finds more accurate models than the other methods. For Chapters 7 and 8, moreover, we provided examples showcasing that we safeguarded the possibility of these models being interpretable. Enabling clinicians to understand how a model operates and why it operates that way can be important for them to trust its use.

The largest limitation of this application is that the available data was very limited. For Chapter 7, only 60 pediatric CT scans where available. For Chapter 8, we managed to increase the number of pediatric CT scans to 147, by involving five hospitals world-wide. These numbers are relatively small in the field of machine learning, because overfitting becomes easier the smaller the number of samples available [3]. Yet, GP-GOMEA was found to be particularly proficient in this application. To avoid overfitting one essentially expects (among other aspects) that programs will need to be small-sized, as that prevents convoluted function compositions to take place. GP-GOMEA works particularly well when programs (and their corresponding genotypes) are small, as linkage learning needs to deal with a limited number of possibilities (genotype symbols). Accurate linkage estimation leads to efficiency and effectiveness, as was shown in benchmark problems and datasets. Moreover, forcing programs to be small means that the corresponding mathematical formulas will be small as well, which, in turn, increases interpretability chances.

9.2. Ramifications, limitations, future work, and societal impact

9.2.1. General ramifications

The research presented in this thesis shows that, by harnessing the statistical information on interdependency within program structure that emerges during the search process, it is in fact possible to improve the efficiency and effectiveness of GP. Compared to other recent and de-randomized variation methods, the improvements brought by GP-GOMEA in terms of efficiency and effectiveness need not come at the cost of large increases in program size (limitation of geometric semantic variation), and can scale up to practically interesting problems such as symbolic regression for real-world data (limitation of Estimation of Distribution Algorithms (EDAs) for GP). Very interesting results in GP research may be enabled by designing new methods inspired by the idea that statistical information should be harnessed and exploited during the search.

Still, it is important to realize that certain conditions need to hold for such methods to be proficient. The crucial issue at the core of attempting to leverage any statistical information is that this information needs to be modeled with sufficient accuracy to be helpful. In Chapters 3 and 4, experimental evidence led us to argue that the number of samples (in

GP, the population size) needs to be large with respect to the number of random variables (the maximum number of instructions allowed) and their sample space (the number of possible instructions) for linkage learning to be advantageous.

For discrete optimization, the problem of optimal population sizing has been studied for traditional genetic algorithms as well as for OMEAs [4–7], typically using assumptions on the degree of interdependencies occurring within and among building blocks. For GP, these studies are yet to be done, and can be considered particularly complicated to be performed because the search in GP is defined by several hyper-parameters such as, e.g., how large programs can grow, what primitives are used, what task the programs must do (which can be data-dependent), and, perhaps most importantly, what the optimal dependency structure is. Hence, optimal population sizing remains an open problem in GP. Nonetheless, we showed that a simple incremental population sizing-scheme (i.e., the IMS) can suffice for GP-GOMEA to be effectively capable of achieving good results within reasonable computational resources even for real-world problems (of moderate dimensionality). We remark that, to the best of our knowledge, this has not been shown for other model-based GP approaches (i.e., essentially for EDA-based GP). This might be because of what is already known for binary optimization: EDAs typically need much larger population sizes than OMEAs to work well.

All in all, for practical applications, it is important to estimate the resources at hand, to have a coarse assessment of whether GP-GOMEA can achieve its full potential. As of now, it is still unknown what specific properties a problem needs to have for methods like GP-GOMEA to work particularly well on that problem.

In an attempt to make the discovered programs/machine learning models more transparent and interpretable, constraints on program compactness were enforced in this thesis, particularly in terms of a maximum number of allowed instructions. However, forcing programs to be small is only a necessary condition to allow a human to read and understand what a program means. The field of GP can, in general, be very useful to obtain machine learning models with a good chance of being interpretable for many people. Surveys on explainable artificial intelligence include GP among the methods to explain other models or directly generate explainable models [2, 8]. To improve upon the state-of-the-art, GP experts need to communicate with experts on explainable artificial intelligence to design algorithms that can have better chances at providing interpretable outcomes. Under the assumption that mechanisms to steer the search towards more interpretable programs can work alongside the methods to de-randomize the search proposed in this thesis, GP-GOMEA represents a good candidate to build upon, because it works particularly well when programs need to be small to begin with.

A last important general point is that although GP-GOMEA can be more efficient compared to other GP algorithms, GP-GOMEA remains a GP algorithm, and is therefore computationally expensive when compared to several other types of machine learning algorithms. Moreover, if a large model is needed anyway to capture patterns among a large number of dimensions, e.g., as in image recognition, GP-GOMEA (and GP in general) is probably not a good choice compared to, e.g., convolutional neural networks [9].

9.2.2. Main limitations and future work directions

In the following, major technical limitations are presented alongside future work directions. The first two points are in regard to how to improve efficiency and effectiveness in GP, beyond the results achieved in this thesis. The last point concerns interpretable machine learning, and the role of GP within it.

Beyond syntactic linkage learning

Given a program synthesis task, for a target performance metric, GP-GOMEA normally finds a much smaller program than other GP algorithms (Chapter 2). Given the same search space enforced by a relatively strict program size limitation, GP-GOMEA normally outperforms other GP algorithms (Chapter 3). Key to enable the discovery of very small yet very proficient programs (as trees), is the fact that the Gene-pool Optimal Mixing (GOM) variation operator can modify, in one go, nodes that are unconnected (differently from, e.g., subtree crossover). A very large number of possibilities exists if any number of unconnected nodes can be changed. The number of possible changes has factorial growth with respect to the number of nodes. From this perspective, the use of a family of subsets such as the linkage tree can be considered a necessary factor, to guide GOM to attempt a limited number of variation steps (linear with respect to program size) that are most promising. In particular, the variation steps based on linkage estimation are promising under the sensible hypothesis that hidden interdependencies between program instructions inherently exist and should be exploited. Still, the linkage tree is but one possible way of driving the variation GOM performs, and is not necessarily the optimal one. Different methods for the construction of a family of subsets have been (and are currently) studied for discrete optimization [10]. Research on OMEAs for GP might benefit from the exploration of the ideas from discrete optimization. Vice versa, new ideas developed in GP might be beneficial for the development of OMEAs in discrete optimization.

With a large branch of modern variation approaches showing the promise of harnessing program semantics, i.e., the (intermediate) outputs of (sub-)programs, it is natural to wonder whether the concepts of OMEAs can be applied to program semantics as well. Building (probabilistic) models (that drive variation, intended as in Sec. 1.4) in the semantic space instead of in the syntax space could reduce the population size needed for sufficiently accurate model building, and thus improve efficiency. This is because the mapping between semantic space and fitness space is typically less redundant than the mapping between syntax space and semantic space. Consequently, variation could be set to compose different semantics based on their interdependencies. These semantics might be realized by sub-programs which are collected in a library (as typically done with RDO and other approximately semantic variation operators [11]).

In practice, however, the number of possible semantics that programs can have for a certain problem is generally large. The lower bound for the number of possible semantics depends on the instructions and on their properties (e.g., commutativity, existence of neutral operands). Given a fixed template that programs can have, the upper bound is the total number of possible (syntactically valid) instruction permutations with repetitions within that template. Yet, if the fitness function is such that neighborhoods in the semantic space lead to neighborhoods in fitness space (e.g., the mean squared error), then similarity in semantics means similarity in fitness values. Therefore, semantic-based model building methods could be set to work upon a tractable number of clusters that group similar

semantics together, rather than on the total number of possible semantics. This could drastically reduce the quantity of possibilities that a model needs to capture. Such clusters could be defined by means of locality sensitive hashing techniques [12], and could be dynamically updated during the search, as to refine the granularity of the clustering the more the search converges.

Another interesting research direction is the design of variation methods that take into account the flow of computations within program architectures. A famous example is the automatic search for neural network architectures [13]. For the sake of simplicity, consider the need to design a feed-forward neural network such as a multilayer perceptron, where a permutation of some hidden layers should be chosen to create a best neural network architecture. A finite set of possible hidden layers can be conceived, where hidden layers are considered to be different based on what activation functions they use, and/or based on how many neurons they contain (layer size). Now, it can be reasonably assumed that some layers will work better than others for a certain problem, and that what layer works best in some position depends on the configuration of the other layers. In particular, one may expect that the possible outcomes of an intermediate layer l is conditional on what computations are possible in the layers that precede it, $l-1, l-2, \ldots, 1$.

In genetic algorithms literature, the *leading ones* benchmark problem can be considered an example representative of (a very simplified version of) the aforementioned situation:

$$\arg\max_{x \in \{0,1\}^l} \left(\sum_{i=1}^{l} \prod_{j=1}^{i} x_j \right).$$

Essentially, given a binary string, the number of consecutive 1s appearing from left to right should be maximized. The 1s that appear after a 0 are not considered. This problem can be thought of as an extremely simplistic neural architecture search for multilayer perceptrons where only two layer types are employed: 1s represent performant layers, and 0s represent faulty layers. The presence of a "0-layer" undermines the performance of subsequent "1-layers". For example, a 0-layer may be one that uses too few neurons to propagate sufficient information to improve the network performance. For a multi-objective version of the leading ones problem (leading ones-trailing zeros), it has been showed that OMEAs using linkage learning (in particular, the LT) are not capable of solving this problem efficiently. The key reason is that current linkage learning methods do not account for the order in which variation should be performed [14].

Studying how mixing order impacts search efficiency might be crucial to enable metaheuristic approaches such as GP to search more competently in the space of highly-complex machine learning model architectures (e.g., neural architectures). A mixing order can be hard-coded based on prior-knowledge on how computations flow within the possible architectures that a machine learning model can have. Alternatively, methods could be investigated to automatically detect best mixing orders based on information that emerges during the search. This kind of new variation operators might lead to more scalable automated discovery of machine learning model architectures.

Dealing with fixed-size representations

An important limitation we encountered when comparing GP-GOMEA to other GP algorithms is that GP-GOMEA currently needs a maximal tree template to represent programs.

For some problems, classic GP can still outperform GP-GOMEA when using a same limit on the maximal number of instructions allowed, simply because it has the freedom to craft program-encoding trees of arbitrary shape.

The use of a fixed-sized maximal template is not, in itself, a problem. The problem is to identify what a best template shape should be, when a maximum program size is fixed. For some problems, setting a particular shape for the maximal template can allow the discovery of particularly effective programs. For example, for programs encoded by a perfect binary tree, only so many functions can be nested in compositions. An unbalanced binary tree allows deep nesting of function compositions, an aspect which might be desirable for certain problems.

In Chapter 2, the IMS was used to increase the height of the maximal tree template (perfect binary) every other run. Nonetheless, simply scaling the maximal height can be inefficient, because increasing the tree height has an exponential effect on the increase of tree nodes (program instructions). Future work might focus on ways to automatically determine promising shapes for a maximal tree template (or for an equivalent representation), without necessarily increasing the total number of nodes.

A different way to overcome limitations caused by the use of a maximal template consists of augmenting the expressiveness that primitives can have. For instance, this can be done by encapsulating sub-routines into new node types. In Chapter 2, we proposed to encapsulate sub-routines that exhibited certain regularities in their output into new function and terminal nodes (with the input-space entropy-based building-block learning method). This proved to be very beneficial to solve a highly-modular Boolean problem (even parity) and, notably, required no changes to be made to the workings of GP-GOMEA. Other similar methods have explored similar ideas to dynamically augment the expressiveness of GP for different applications [15, 16]. To date, however, a general method to discover and re-use salient sub-routines that is problem-agnostic, still needs to be discovered.

Lastly, indirect representations can be leveraged to obtain, from fixed-length encodings in general (not only maximal templates), programs/trees of any length and shape. As mentioned in Section 1.4, EDAs have been set to work on grammar-based GP, where often the encoding is binary, and represents the chain of rules that will be used to sample program instructions. As mentioned in Section 1.4.2 and Section 3.2, we are not aware of any EDA for (grammar-based) GP capable of delivering state-of-the-art results on supervised learning problems with at least dozens of features. Moreover, it is unclear whether the use of indirect representations can actually hinder (probabilistic) model building approaches, because to use an indirect representation means to introduce further redundancy between syntax (encoding) and fitness [17, 18].

Recently, an adaptation of GOMEA has been proposed to be used for GP using indirect representations as in grammatical evolution, called *GOMGE* [19]. The results indicated that GOMGE can work particularly well when a pre-specified linkage model is used (called the *natural family of subsets*), that is designed to account for how the mapping operates on the encoding. At the same time, the results indicated that automatic linkage learning (as done in this thesis, i.e., by means of the linkage tree family of subsets) was not performing particularly better than the use of a pre-specified linkage model, nor than using random noise as linkage. This result might be attributed to possible redundancy introduced by the use of the indirect representation. However, another important reason why linkage

learning did not excel might be attributed to the use of a relatively small population size (500). Future research should be carried out to assess whether OMEAs can work better than EDAs on GP with indirect representations.

On interpretability

In this thesis, GP programs were constrained in size in an attempt to improve the chances of interpretability of the corresponding mathematical formulas. Clearly, this approach has its limitations, especially considering that interpretability is a subjective matter. Users may be more, or less, familiar with encountering certain types of functions/program instructions, and certain types of function/instruction compositions.

In modern practical applications of GP, end-users for whom a model is needed are typically consulted beforehand, and heuristics are crafted to penalize programs that contain functions that are less intuitive (see, e.g., [20]). Penalization terms are then incorporated in the fitness function, or used as secondary objective in multi-objective GP search [21]. In the latter case, fronts of programs that range from more accurate but opaque, to less accurate but more transparent, can be obtained. Yet, crafting penalization terms and metrics is often highly arbitrary, and might not be sufficiently personalized.

In the field of explainable artificial intelligence in general, much research work has been put in proposing objective metrics to be used as surrogate for interpretability [22, 23]. However, the limitations of attempting to establish an objective metric is that subjectivity cannot be captured. Therefore, research directions to improve on this should perhaps take inspiration from the personalized strategies adopted in modern content recommendation systems. Every day, systems like YouTube, Netflix, and Amazon tailor their suggestions to the user by employing machine learning algorithms that attempt to automatically learn "*what the user wants*". In GP, since search is an iterative process, it may well be possible to conceive of a mechanism to allow users to steer the ongoing search, by providing online feedback on the programs evolved so far regarding their level of understanding. This feedback could itself be modeled, based on features of the programs (e.g., number of dimensions considered, type of function compositions). A model of program interpretability could therefore be refined during the search, to provide guidance on what the user prefers, and to steer the search in that direction by penalizing programs that do not meet the user's preferences.

An important direction to explore to make GP programs both capable to handle high-dimensional problems and relatively easy to understand, is modularity. One of the most famous contributions to GP in enforcing program modularity is represented by the *automatically defined functions*, which essentially are modules (or sub-programs) that can be instantiated multiple times within the overall program, and are evolved along with how they should be used in a GP solution [24]. In this thesis, a way to identify modules (called building blocks) was proposed to tackle high-dimensional Boolean problems (Sec. 2.3). Methods for the identification of modules have been shown to be greatly helpful to enhance the effectiveness of GP whenever the problem at hand is highly-dimensional, but decomposable. Modularity can enable understanding by *decomposability*, i.e., the ability to understand how a program (or machine learning model) works by understanding how its modules work, and how these modules are combined [23]. Future work in making GP programs more interpretable could focus on promoting the evolution of programs com-

posed of repeating modules. Attempts should be made to make these modules, and their composition, as transparent as possible.

A last important problem that needs to be addressed for the use of GP to obtain explainable models rises from the stochastic nature of GP. In particular, if multiple runs deliver different (explainable) models, it is not clear what model represents the best explanation of the patterns in the data. Picking a model that best generalizes to a validation set is not guaranteed to be the best explanation to the ground-truth phenomenon of which the data was collected. For example, this might be due to having small a validation set which results in small generalization error by chance. Regarding GP-GOMEA, we observed that the use of the linkage tree instead of the random tree as family of subsets reduces the variance in what model is found (almost one fourth of 100 runs found equivalent models for the experiment of Sec. 3.5.4). Our expectation is that the larger the population size is set, the less variance can be expected in the best found models.

This said, a possibility to choose one model among many is to take the mode of multiple repetitions as the final answer, potentially including corrections for small differences in constants (e.g., the mean can be taken) if all other model components are equal; or to observe if different models are actually using highly-correlated features interchangeably. Another possibility is for experts to inspect the models, and recommend what models seem most trustworthy (e.g., considering the features at play).

Even though the stochastic nature of GP can be considered unappealing with respect to the search for explainable models, we remark that GP is not the only source of randomness that is involved in practice. In fact, when using machine learning in practice, further stochastic processes are often employed that can ultimately influence what model is generated, even if the machine learning algorithm is deterministic. For example, it is common to improve model fitting by hyper-parameter tuning methods that repeatedly use different random data partitions. Different data partitions can lead to different hyper-parameter settings being selected by the tuning process. In turn, different hyper-parameter settings can influence the way a machine learning algorithm synthesizes a model. Therefore, research should be done to assess to what extent stochastic processes involved in machine learning practice influence explainability. Principled ways should be explored to reliably find a unique interpretable model when the data samples (from a same underlying distribution) change.

9.2.3. Implications for society

Parts of this thesis presented pragmatic uses of GP-GOMEA: symbolic regression on realistic datasets (Chapter 3), feature construction to obtain more transparent machine learning models (Chapter 5), and symbolic regression synthesis of likely readable models for healthcare data (Chapters 7 and 8). In its current state, GP-GOMEA can be used to discover small and fairly accurate models that have the potential to be interpretable, for a variety of supervised learning applications, as long as structured data is available. If training a black-box machine learning model is preferable to the direct use of GP-GOMEA, e.g., because the former is capable of providing better accuracy, then GP-GOMEA can still be used to construct small and few features that can enable to train the black-box model in a feature space where its predictions can be visualized.

Society can greatly benefit from algorithms that deliver machine learning models which workings can be understood. By using GP-GOMEA for radiation dose reconstruction for childhood cancer survivors, we have provided examples of accurate machine learning models that are intelligible. Currently, we are exploring the use of GP-GOMEA to discover prediction models that, on the one hand, discriminate between cancer cell lines that are sensitive to drugs, and, on the other hand, are sufficiently compact to enable the inference of what the reasons may be for some cancer cell lines to be more or less drug sensitive than others.

Still, GP-GOMEA can also be used on problems of a very different nature. Our algorithms could therefore be very useful for the efficient discovery of effective and transparent solutions for, e.g., design problems of different engineering disciplines. In Chapter 2, the algorithms developed in this thesis were used to discover solutions for several benchmark problems. Solutions need not even necessarily be programs, e.g., for the artificial problems of Chapter 2, trees with particular inner structures are sought. Rather, once a fitness function is designed and suitable primitives are chosen, GP-GOMEA can be used. Ultimately, in comparison to other GP approaches, GP-GOMEA has the potential to discover a compact (hopefully interpretable) solution within limited computational effort.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees* (Wadsworth, 1984).

[2] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, *A survey of methods for explaining black box models,* ACM Computing Surveys (CSUR) **51**, 93:1 (2018).

[3] C. M. Bishop, *Pattern recognition and machine learning* (Springer, 2006).

[4] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, *The gambler's ruin problem, genetic algorithms, and the sizing of populations,* Evolutionary Computation **7**, 231 (1999).

[5] D. E. Goldberg, K. Sastry, and T. Latoza, *On the supply of building blocks,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (Morgan Kaufmann Publishers Inc., 2001) pp. 336–342.

[6] D. Thierens and P. A. N. Bosman, *Optimal mixing evolutionary algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2011) pp. 617–624.

[7] Y.-Y. Liao, H.-W. Hsu, Y.-L. Juang, and T.-L. Yu, *On the investigation of population sizing of genetic algorithms using optimal mixing,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2019) pp. 820–828.

[8] A. Adadi and M. Berrada, *Peeking inside the black-box: A survey on explainable artificial intelligence (XAI),* IEEE Access **6**, 52138 (2018).

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).

[10] S.-H. Hsu and T.-L. Yu, *Optimization by pairwise linkage detection, incremental linkage set, and restricted/back mixing: DSMGA-II,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2015) pp. 519–526.

[11] T. P. Pawlak, B. Wieloch, and K. Krawiec, *Semantic backpropagation for designing search operators in genetic programming,* IEEE Transactions on Evolutionary Computation **19**, 326 (2015).

[12] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, *Locality-sensitive hashing scheme based on p-stable distributions,* in *Proceedings of the 20th Annual Symposium on Computational Geometry* (ACM, 2004) pp. 253–262.

[13] T. Elsken, J. H. Metzen, and F. Hutter, *Neural architecture search: A survey,* (2018), arXiv preprint arXiv:1808.05377.

[14] N. H. Luong, H. La Poutré, and P. A. N. Bosman, *Multi-objective gene-pool optimal mixing evolutionary algorithms,* in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2014) pp. 357–364.

[15] S. Roberts, D. Howard, and J. Koza, *Evolving modules in genetic programming by subtree encapsulation,* in *Genetic Programming* (Springer, 2001) pp. 160–175.

[16] Q. Chen, M. Zhang, and B. Xue, *Genetic programming with embedded feature construction for high-dimensional symbolic regression,* in *Intelligent and Evolutionary Systems* (Springer, 2017) pp. 87–102.

[17] A. Thorhauer, *On the non-uniform redundancy in grammatical evolution,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2016) pp. 292–302.

[18] E. Medvet, *A comparative analysis of dynamic locality and redundancy in grammatical evolution,* in *European Conference on Genetic Programming* (Springer, 2017) pp. 326–342.

[19] E. Medvet, A. Bartoli, A. De Lorenzo, and F. Tarlao, *GOMGE: Gene-pool optimal mixing on grammatical evolution,* in *International Conference on Parallel Problem Solving from Nature (PPSN)* (Springer, 2018) pp. 223–235.

[20] D. Hein, S. Udluft, and T. A. Runkler, *Interpretable policies for reinforcement learning by genetic programming,* Engineering Applications of Artificial Intelligence **76**, 158 (2018).

[21] E. J. Vladislavleva, G. F. Smits, and D. den Hertog, *Order of nonlinearity as a complexity measure for models generated by symbolic regression via Pareto genetic programming,* IEEE Transactions on Evolutionary Computation **13**, 333 (2009).

[22] F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Vaughan, and H. Wallach, *Manipulating and measuring model interpretability,* (2018), arXiv preprint arXiv:1802.07810.

[23] Z. C. Lipton, *The mythos of model interpretability,* Queue **16**, 30:31 (2018).

[24] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection* (MIT Press, 1992).

Acknowledgements

First of all, I would like thank my supervisory team: Cees, Peter, and Tanja. Needless to say, this thesis would not have been what it is without their invaluable efforts, which spanned beyond sole supervision. I am also grateful for the contagious passion and love for research they transmitted to me.

I wish to further spend some words to better acknowledge Peter. Peter is more than a supervisor to me, he is a friend. I remember, whenever I was in doubt of what of n options to explore in my research, he always advised me to *do both* (also for $n > 2$). At the same time, he himself ended up *doing both*: Peter did not only help me grow as a scientist–which is to be expected; he helped me grow as a man–which, I think, is truly exceptional. I am very grateful for this.

I would like to thank my project partner Ziyuan, without whose effort many results on the clinical application (Chapters 6, 7 and 8) would not have been possible. For their contribution, I further thank the collaborators and co-authors from the Department of Radiation Oncology of the Amsterdam University Medical Centers, location AMC.

On a more personal note, I thank my colleagues and friends from Centrum Wiskunde & Informatica (CWI), and my friends from after-work activities, such as the Italian crew, and the climbing buddies. From the union of these sets, Alessandro, Chang-Han, Enrico, Erni, Hoang, Macs, and Tommaso, have a special place in my heart. I also wish to thank the friends from the Machine Learning Lab of Trieste, Andrea and Eric, for the nice scientific side-work we did together. Last but not least, I particularly thank my (almost-)paranymphs (due to COVID-19 impeding their presence), Anton and Stef, who have been good friends and reliable comrades in this journey.

I cannot thank enough my family, who always encouraged me and loved me. I wish my parents Roberto and Gemma to know any achievement I reach is as much as mine as it is theirs, as they made me who I am today. I wish to thank my sister Veronica for having been the most courageous sister a brother can wish for. I am so proud of you.

Lastly, I wish to thank my partner, Dafni. Julius Caesar said "*Veni; Vidi; Vici*" ("*I came; I saw; I conquered*"). Dafni came to the Netherlands, saw me, and conquered my heart. By being by my side every day, Dafni supported me more than anyone else. Thank you amore.

Curriculum Vitæ

Marco was born in Monfalcone (Italy) on July 30th 1989. He completed his high school studies at the Albert Einstein school of Cervignano del Friuli (Italy) in 2008, bringing to the final exam an essay on artificial intelligence. He obtained a bachelor degree in Information Engineering in 2012 from the University of Trieste (Italy). His final bachelor project consisted of developing a smartphone application for the front-end and a web service for the back-end, to visualize geospatial information on pollutant emissions. He studied Computer Engineering at the same university, and obtained his Master of Science degree *cum laude* in 2015. During these studies, between 2013 and 2014, he worked part-time as web developer at Promoscience s.r.l. in Padriciano (Italy). He served as an intern at the Machine Learning Lab of the University of Trieste in the summer of 2014, studying evolutionary computation for cybersecurity. His Master of Science thesis concerned the design of an evolutionary method for the automatic detection of natural language text on gene-protein interactions. The same year in November he started his doctoral research in Amsterdam, the Netherlands, at Centrum Wiskunde & Informatica, in the Life Sciences and Health Group.

His main research interests are two *E*s of machine learning: *Explainable* and *Evolutionary*. His main expertise is genetic programming, with one focus on the design of methods and techniques that enable the scalable generation of transparent machine learning models, and another on developing synergies with other machine learning techniques. He is also interested in medical applications of evolutionary optimization and machine learning. Marco has experience with pediatric radiation therapy, a topic on which he worked during his time at Centrum Wiskunde & Informatica in collaboration with the Department of Radiation Oncology of the Amsterdam University Medical Centers, location AMC.

More details are available at: https://marcovirgolin.github.io.

List of Publications

16. **M. Virgolin**, A. de Lorenzo, E. Medvet, and F. Randone. Learning a formula of interpretability to learn interpretable formulas. *Submitted. Preprint arXiv:2004.11170*, arXiv (2020).

15. T. den Ottelander, A. Dushatskiy, **M. Virgolin**, and P.A.N. Bosman. Local search is a remarkably strong baseline for neural architecture search. *Submitted. Preprint arXiv:2004.08996*, arXiv (2020).

14. **M. Virgolin**, Z. Wang (shared first co-author), B.V. Balgobind, I.W.E.M. van Dijk, J. Wiersma, P.S. Kroon, G.O. Janssens, M. van Herk, D.C. Hodgson, L. Zadravec Zaletel, C.R.N. Rasch, A. Bel, P.A.N. Bosman, and T. Alderliesten. Surrogate-free machine learning-based organ dose reconstruction for pediatric abdominal radiotherapy. *Submitted. Preprint arXiv:2002.07161*, arXiv (2020).

13. **M. Virgolin**, Z. Wang (shared first co-author), B.V. Balgobind, I.W.E.M. van Dijk, J. Wiersma, D.C. Hodgson, A. Bryce-Atkinson, M. van Herk, C.R.N. Rasch, L. Zadravec Zaletel, P.S. Kroon, G.O. Janssens, A. Bel, P.A.N. Bosman, and T. Alderliesten. Highly-individualized dose reconstruction for pediatric abdominal radiotherapy with machine learning. *Accepted for oral presentation at ESTRO 2020, Radiotherapy and Oncology* (2020) (to appear).

12. **M. Virgolin**, T. Alderliesten, and P.A.N. Bosman. On explaining machine learning models by evolving crucial and compact features. *Swarm and Evolutionary Computation* **53**, pp. 100640, Elsevier (2020).

11. **M. Virgolin**, Z. Wang, T. Alderliesten, and P.A.N. Bosman. Machine learning for automatic construction of pediatric abdominal phantoms. *In Proceedings of SPIE Medical Imaging 2020: Imaging Informatics for Healthcare, Research, and Applications*, International Society for Optics and Photonics (2020) (to appear).

10. **M. Virgolin**, Z. Wang, T. Alderliesten, and P.A.N. Bosman. Machine learning for automatic construction of pseudo-realistic pediatric abdominal phantoms. *Submitted. Preprint arXiv:1909.03723*, arXiv (2019).

9. **M. Virgolin**, T. Alderliesten, and P.A.N. Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*, pp. 1084-1092, ACM (2019).

8. Z. Wang, B.V. Balgobind, **M. Virgolin**, I.W.E.M. van Dijk, J. Wiersma, C.M. Ronckers, P.A.N. Bosman, A. Bel, and T. Alderliesten. How do patient characteristics and anatomical features correlate to accuracy of organ dose reconstruction for Wilms' tumor radiation treatment plans when using a surrogate patient's CT scan? *Journal of Radiological Protection* **39** (2), pp. 598-619, IOP Publishing (2019).

7. **M. Virgolin**, T. Alderliesten, C. Witteveen, and P.A.N. Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Accepted for publication in Evolutionary Computation. Preprint arXiv:1904.02050*, arXiv (2019).

6. Z. Wang, **M. Virgolin**, P.A.N. Bosman, B.V. Balgobind, A. Bel, and T. Alderliesten. Automatic radiotherapy plan emulation for 3D dose reconstruction to enable big data analysis for historically treated patients. *In Proceedings of SPIE Medical Imaging 2019: Imaging Informatics for Healthcare, Research, and Applications* **10954**, pp. 203-211, International Society for Optics and Photonics (2019).

5. E. Medvet, **M. Virgolin**, M. Castelli, P.A.N. Bosman, I. Gonçalves, and T. Tŭsar. Unveiling evolutionary algorithm representation with DU maps. *Genetic Programming and Evolvable Machines* **19** (3), pp. 351-389, Springer (2018).

4. **M. Virgolin**, T. Alderliesten, A. Bel, C. Witteveen, and P.A.N. Bosman (2018). Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*, pp. 1395-1402, ACM (2018).

3. **M. Virgolin**, I.W.E.M. van Dijk, J. Wiersma, C.M. Ronckers, C. Witteveen, A. Bel, T. Alderliesten, and P.A.N. Bosman. On the feasibility of automatically selecting similar patients in highly individualized radiotherapy dose reconstruction for historic data of pediatric cancer survivors. *Medical Physics* **45** (4), pp. 1504-1517, Wiley (2018).

2. **M. Virgolin**, T. Alderliesten, C. Witteveen, and P.A.N. Bosman. Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*, pp. 1041-1048, ACM (2017).

1. A. Bartoli, A. De Lorenzo, E. Medvet, F. Tarlao, and **M. Virgolin**. Evolutionary learning of syntax patterns for genic interaction extraction. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '15)*, pp. 1183-1190, ACM (2015).

SIKS Dissertation Series

2011 01 Botond Cseke (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models
02 Nick Tinnemeier (UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language
03 Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems
04 Hado van Hasselt (UU), Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference
05 Bas van der Raadt (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.
06 Yiwen Wang (TUE), Semantically-Enhanced Recommendations in Cultural Heritage
07 Yujia Cao (UT), Multimodal Information Presentation for High Load Human Computer Interaction
08 Nieske Vergunst (UU), BDI-based Generation of Robust Task-Oriented Dialogues
09 Tim de Jong (OU), Contextualised Mobile Media for Learning
10 Bart Bogaert (UvT), Cloud Content Contention
11 Dhaval Vyas (UT), Designing for Awareness: An Experience-focused HCI Perspective
12 Carmen Bratosin (TUE), Grid Architecture for Distributed Process Mining
13 Xiaoyu Mao (UvT), Airport under Control. Multiagent Scheduling for Airport Ground Handling
14 Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets
15 Marijn Koolen (UvA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval
16 Maarten Schadd (UM), Selective Search in Games of Different Complexity
17 Jiyin He (UVA), Exploring Topic Structure: Coherence, Diversity and Relatedness
18 Mark Ponsen (UM), Strategic Decision-Making in complex games
19 Ellen Rusman (OU), The Mind's Eye on Personal Profiles
20 Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach
21 Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems
22 Junte Zhang (UVA), System Evaluation of Archival Description and Access
23 Wouter Weerkamp (UVA), Finding People and their Utterances in Social Media
24 Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

25 Syed Waqar ul Qounain Jaffry (VU), Analysis and Validation of Models for Trust Dynamics
26 Matthijs Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots
27 Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns
28 Rianne Kaptein (UVA), Effective Focused Retrieval by Exploiting Query Context and Document Structure
29 Faisal Kamiran (TUE), Discrimination-aware Classification
30 Egon van den Broek (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions
31 Ludo Waltman (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
32 Nees-Jan van Eck (EUR), Methodological Advances in Bibliometric Mapping of Science
33 Tom van der Weide (UU), Arguing to Motivate Decisions
34 Paolo Turrini (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
35 Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training
36 Erik van der Spek (UU), Experiments in serious game design: a cognitive approach
37 Adriana Burlutiu (RUN), Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
38 Nyree Lemmens (UM), Bee-inspired Distributed Optimization
39 Joost Westra (UU), Organizing Adaptation using Agents in Serious Games
40 Viktor Clerc (VU), Architectural Knowledge Management in Global Software Development
41 Luan Ibraimi (UT), Cryptographically Enforced Distributed Data Access Control
42 Michal Sindlar (UU), Explaining Behavior through Mental State Attribution
43 Henk van der Schuur (UU), Process Improvement through Software Operation Knowledge
44 Boris Reuderink (UT), Robust Brain-Computer Interfaces
45 Herman Stehouwer (UvT), Statistical Language Models for Alternative Sequence Selection
46 Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
47 Azizi Bin Ab Aziz (VU), Exploring Computational Models for Intelligent Support of Persons with Depression
48 Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
49 Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012 01 Terry Kakeeto (UvT), Relationship Marketing for SMEs in Uganda
02 Muhammad Umair (VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

03 Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories
04 Jurriaan Souer (UU), Development of Content Management System-based Web Applications
05 Marijn Plomp (UU), Maturing Interorganisational Information Systems
06 Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks
07 Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
08 Gerben de Vries (UVA), Kernel Methods for Vessel Trajectories
09 Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms
10 David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment
11 J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
12 Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems
13 Suleman Shahid (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
14 Evgeny Knutov (TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems
15 Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.
16 Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
17 Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
18 Eltjo Poort (VU), Improving Solution Architecting Practices
19 Helen Schonenberg (TUE), What's Next? Operational Support for Business Process Execution
20 Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
21 Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval
22 Thijs Vis (UvT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
23 Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
24 Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval
25 Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
26 Emile de Maat (UVA), Making Sense of Legal Text
27 Hayrettin Gurkok (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
28 Nancy Pascall (UvT), Engendering Technology Empowering Women
29 Almer Tigelaar (UT), Peer-to-Peer Information Retrieval

30 Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making
31 Emily Bagarukayo (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
32 Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning
33 Rory Sie (OUN), Coalitions in Cooperation Networks (COCOON)
34 Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications
35 Evert Haasdijk (VU), Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics
36 Denis Ssebugwawo (RUN), Analysis and Evaluation of Collaborative Modeling Processes
37 Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation
38 Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms
39 Hassan Fatemi (UT), Risk-aware design of value and coordination networks
40 Agus Gunawan (UvT), Information Access for SMEs in Indonesia
41 Sebastian Kelle (OU), Game Design Patterns for Learning
42 Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning
43 Withdrawn
44 Anna Tordai (VU), On Combining Alignment Techniques
45 Benedikt Kratz (UvT), A Model and Language for Business-aware Transactions
46 Simon Carter (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
47 Manos Tsagkias (UVA), Mining Social Media: Tracking Content and Predicting Behavior
48 Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data
49 Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
50 Steven van Kervel (TUD), Ontologogy driven Enterprise Information Systems Engineering
51 Jeroen de Jong (TUD), Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

2013 01 Viorel Milea (EUR), News Analytics for Financial Decision Support
02 Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
03 Szymon Klarman (VU), Reasoning with Contexts in Description Logics
04 Chetan Yadati (TUD), Coordinating autonomous planning and scheduling
05 Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns
06 Romulo Goncalves (CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
07 Giel van Lankveld (UvT), Quantifying Individual Player Differences
08 Robbert-Jan Merk (VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
09 Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications

10 Jeewanie Jayasinghe Arachchige (UvT), A Unified Modeling Framework for Service Design.
11 Evangelos Pournaras (TUD), Multi-level Reconfigurable Self-organization in Overlay Services
12 Marian Razavian (VU), Knowledge-driven Migration to Services
13 Mohammad Safiri (UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
14 Jafar Tanha (UVA), Ensemble Approaches to Semi-Supervised Learning Learning
15 Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications
16 Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation
17 Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid
18 Jeroen Janssens (UvT), Outlier Selection and One-Class Classification
19 Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling
20 Katja Hofmann (UvA), Fast and Reliable Online Learning to Rank for Information Retrieval
21 Sander Wubben (UvT), Text-to-text generation by monolingual machine translation
22 Tom Claassen (RUN), Causal Discovery and Logic
23 Patricio de Alencar Silva (UvT), Value Activity Monitoring
24 Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning
25 Agnieszka Anna Latoszek-Berendsen (UM), Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System
26 Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning
27 Mohammad Huq (UT), Inference-based Framework Managing Data Provenance
28 Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience
29 Iwan de Kok (UT), Listening Heads
30 Joyce Nakatumba (TUE), Resource-Aware Business Process Management: Analysis and Support
31 Dinh Khoa Nguyen (UvT), Blueprint Model and Language for Engineering Cloud Applications
32 Kamakshi Rajagopal (OUN), Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development
33 Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere
34 Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search
35 Abdallah El Ali (UvA), Minimal Mobile Human Computer Interaction
36 Than Lam Hoang (TUe), Pattern Mining in Data Streams
37 Dirk Börner (OUN), Ambient Learning Displays
38 Eelco den Heijer (VU), Autonomous Evolutionary Art
39 Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems
40 Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games

41 Jochem Liem (UVA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
42 Léon Planken (TUD), Algorithms for Simple Temporal Reasoning
43 Marc Bron (UVA), Exploration and Contextualization through Interaction and Concepts

2014 01 Nicola Barile (UU), Studies in Learning Monotone Models from Data
02 Fiona Tuliyano (RUN), Combining System Dynamics with a Domain Modeling Method
03 Sergio Raul Duarte Torres (UT), Information Retrieval for Children: Search Behavior and Solutions
04 Hanna Jochmann-Mannak (UT), Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation
05 Jurriaan van Reijsen (UU), Knowledge Perspectives on Advancing Dynamic Capability
06 Damian Tamburri (VU), Supporting Networked Software Development
07 Arya Adriansyah (TUE), Aligning Observed and Modeled Behavior
08 Samur Araujo (TUD), Data Integration over Distributed and Heterogeneous Data Endpoints
09 Philip Jackson (UvT), Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language
10 Ivan Salvador Razo Zapata (VU), Service Value Networks
11 Janneke van der Zwaan (TUD), An Empathic Virtual Buddy for Social Support
12 Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control
13 Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains
14 Yangyang Shi (TUD), Language Models With Meta-information
15 Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare
16 Krystyna Milian (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria
17 Kathrin Dentler (VU), Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability
18 Mattijs Ghijsen (UVA), Methods and Models for the Design and Study of Dynamic Agent Organizations
19 Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support
20 Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link
21 Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments
22 Marieke Peeters (UU), Personalized Educational Games - Developing agent-supported scenario-based training
23 Eleftherios Sidirourgos (UvA/CWI), Space Efficient Indexes for the Big Data Era
24 Davide Ceolin (VU), Trusting Semi-structured Web Data

25 Martijn Lappenschaar (RUN), New network models for the analysis of disease interaction
26 Tim Baarslag (TUD), What to Bid and When to Stop
27 Rui Jorge Almeida (EUR), Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty
28 Anna Chmielowiec (VU), Decentralized k-Clique Matching
29 Jaap Kabbedijk (UU), Variability in Multi-Tenant Enterprise Software
30 Peter de Cock (UvT), Anticipating Criminal Behaviour
31 Leo van Moergestel (UU), Agent Technology in Agile Multiparallel Manufacturing and Product Support
32 Naser Ayat (UvA), On Entity Resolution in Probabilistic Data
33 Tesfa Tegegne (RUN), Service Discovery in eHealth
34 Christina Manteli (VU), The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.
35 Joost van Ooijen (UU), Cognitive Agents in Virtual Worlds: A Middleware Design Approach
36 Joos Buijs (TUE), Flexible Evolutionary Algorithms for Mining Structured Process Models
37 Maral Dadvar (UT), Experts and Machines United Against Cyberbullying
38 Danny Plass-Oude Bos (UT), Making brain-computer interfaces better: improving usability through post-processing.
39 Jasmina Maric (UvT), Web Communities, Immigration, and Social Capital
40 Walter Omona (RUN), A Framework for Knowledge Management Using ICT in Higher Education
41 Frederic Hogenboom (EUR), Automated Detection of Financial Events in News Text
42 Carsten Eijckhof (CWI/TUD), Contextual Multidimensional Relevance Models
43 Kevin Vlaanderen (UU), Supporting Process Improvement using Method Increments
44 Paulien Meesters (UvT), Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.
45 Birgit Schmitz (OUN), Mobile Games for Learning: A Pattern-Based Approach
46 Ke Tao (TUD), Social Web Data Analytics: Relevance, Redundancy, Diversity
47 Shangsong Liang (UVA), Fusion and Diversification in Information Retrieval

2015 01 Niels Netten (UvA), Machine Learning for Relevance of Information in Crisis Response
02 Faiza Bukhsh (UvT), Smart auditing: Innovative Compliance Checking in Customs Controls
03 Twan van Laarhoven (RUN), Machine learning for network data
04 Howard Spoelstra (OUN), Collaborations in Open Learning Environments
05 Christoph Bösch (UT), Cryptographically Enforced Search Pattern Hiding
06 Farideh Heidari (TUD), Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes
07 Maria-Hendrike Peetz (UvA), Time-Aware Online Reputation Analysis
08 Jie Jiang (TUD), Organizational Compliance: An agent-based model for designing and evaluating organizational interactions
09 Randy Klaassen (UT), HCI Perspectives on Behavior Change Support Systems

10 Henry Hermans (OUN), OpenU: design of an integrated system to support lifelong learning
11 Yongming Luo (TUE), Designing algorithms for big graph datasets: A study of computing bisimulation and joins
12 Julie M. Birkholz (VU), Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks
13 Giuseppe Procaccianti (VU), Energy-Efficient Software
14 Bart van Straalen (UT), A cognitive approach to modeling bad news conversations
15 Klaas Andries de Graaf (VU), Ontology-based Software Architecture Documentation
16 Changyun Wei (UT), Cognitive Coordination for Cooperative Multi-Robot Teamwork
17 André van Cleeff (UT), Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs
18 Holger Pirk (CWI), Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories
19 Bernardo Tabuenca (OUN), Ubiquitous Technology for Lifelong Learners
20 Lois Vanhée (UU), Using Culture and Values to Support Flexible Coordination
21 Sibren Fetter (OUN), Using Peer-Support to Expand and Stabilize Online Learning
22 Zhemin Zhu (UT), Co-occurrence Rate Networks
23 Luit Gazendam (VU), Cataloguer Support in Cultural Heritage
24 Richard Berendsen (UVA), Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation
25 Steven Woudenberg (UU), Bayesian Tools for Early Disease Detection
26 Alexander Hogenboom (EUR), Sentiment Analysis of Text Guided by Semantics and Structure
27 Sándor Héman (CWI), Updating compressed colomn stores
28 Janet Bagorogoza (TiU), Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO
29 Hendrik Baier (UM), Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains
30 Kiavash Bahreini (OU), Real-time Multimodal Emotion Recognition in E-Learning
31 Yakup Koç (TUD), On the robustness of Power Grids
32 Jerome Gard (UL), Corporate Venture Management in SMEs
33 Frederik Schadd (TUD), Ontology Mapping with Auxiliary Resources
34 Victor de Graaf (UT), Gesocial Recommender Systems
35 Jungxao Xu (TUD), Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction

2016 01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VU), Publishing and Consuming Linked Data

05 Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
09 Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UVA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VU), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VU), Refining Statistical Data on the Web
19 Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UVA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UVA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (UvT), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example

34 Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UVA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned
49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UVA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VU) , Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TUE), Robust Process Mining with Guarantees

13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (UvT), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UVA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VU), Logics for causal inference under uncertainty
23 David Graus (UVA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
28 John Klein (VU), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (UvT), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (UvT), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VU), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle

42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TUE), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UVA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems
09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UVA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TUE), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (UVT), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TUE), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OUN), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araujo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology

27 Maikel Leemans (TUE), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (UVT), Emotion Recognition from Mandarin Speech
30 Wouter Beek, The "K" in "semantic web" stands for "knowledge": scaling semantics to the web

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TUE), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TUE), Process Mining with Streaming Data
06 Chris Dijkshoorn (VU), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
09 Fahimeh Alizadeh Moghaddam (UVA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VU), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TUE), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts
18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents
20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
21 Cong Liu (TUE), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VU),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VU), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing

25 Emiel van Miltenburg (VU), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OUN), The Gamification Design Process applied to (Massive) Open Online Courses
28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VU), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VU), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TUE), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TUE), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
35 Lisa Facey-Shaw (OUN), Gamification with digital badges in learning programming
36 Kevin Ackermans (OUN), Designing Video-Enhanced Rubrics to Master Complex Skills
37 Jian Fang (TUD), Database Acceleration on FPGAs
38 Akos Kadar (OUN), Learning visually grounded and multilingual representations

2020 01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
04 Maarten van Gompel (RUN), Context as Linguistic Bridges
05 Yulong Pei (TUE), On local and global structure mining
06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
07 Wim van der Vegt (OUN), Towards a software architecture for reusable game components
08 Ali Mirsoleimani (UL), Structured Parallel Programming for Monte Carlo Tree Search
09 Myriam Traub (UU), Measuring Tool Bias & Improving Data Quality for Digital Humanities Research
10 Alifah Syamsiyah (TUE), In-database Preprocessing for Process Mining
11 Sepideh Mesbah Semantic-Enhanced Training Data Augmentation Methods for Long-Tail Entity Recognition Models
12 Ward van Breda (VU), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment