Learning Dynamic Models of Compartment Systems by Combining Symbolic Regression with Fuzzy Vector Envisionment

Mehdi Khoury
Computing Science Department
University of Aberdeen
AB243UE, Aberdeen,
Scotland, United Kingdoms
mkhoury@csd.abdn.ac.uk

Frank Guerin
Computing Science Department
University of Aberdeen
AB243UE, Aberdeen,
Scotland, United Kingdoms
fguerin@csd.abdn.ac.uk

George Macleod Coghill
Computing Science Department
University of Aberdeen
AB243UE, Aberdeen,
Scotland, United Kingdoms
gcoghill@csd.abdn.ac.uk

ABSTRACT

This paper is concerned with the learning of dynamic models of compartmental systems visualized as networks of interconnected tanks. This is intended as an intermediary step to learn more complex dynamic biological systems such as metabolic pathways. Our present aim is to learn systems of differential equations from time series data to capture physical models of increasing complexity (u-tube, cascaded tanks, and coupled tanks). To do so, we use Symbolic Regression in Genetic Programming and combine it with a fuzzy representation which has inherent differential capabilities (Fuzzy Vector Envisionment). We use the ECJ[1] framework to implement the learner. Present results show that the system can approximate the target models and that the use of a weighted fitness function seems to accelerate the learning process.

Categories and Subject Descriptors

I.2.1 [**Pattern Recognition**]: Models—*Fuzzy Set*; I.2.6 [**Artificial Intelligence**]: Learning—*Induction, Parameter Learning*

General Terms

Algorithms, Measurement, Performance, Experimentation, Theory

Keywords

Fuzzy Vector Envisionment, Symbolic Regression, Genetic Programming, semi-quantitative modeling, dynamic compartmental model, u-tube, dynamic biological model, SSystems, metabolic pathways

[1]A Java-based Evolutionary Computation Research System available at http://cs.gmu.edu/ eclab/projects/ecj

1. INTRODUCTION

The motivation behind this paper is the desire to learn, in the near future, complex dynamic biological systems from time series data. Compartmental systems can be used as a metaphor to approach metabolic pathways. We therefore focus on learning dynamic models of compartmental systems, visualized as networks of interconnected tanks (see Section 4.1 for a more detailed description). Such a task poses two problems; how to represent this type of system, and how to learn the correct model in the most efficient way. Regarding the first problem, we require a formalism that conciliates differential capabilities and fuzzy representation. We need differential capabilities because, in a dynamic environment, models not only focus on the range of variables (e.g. the concentration of a chemical), but also the rate of change of these variables (e.g. the speed with which the concentration of a chemical increases), and even the curvature of this rate of change. Thus, the ability to manipulate derivatives improves the chance of finding the potential hidden variables that influence the dynamics of the system. Numerous dynamic modeling examples make use of differential representation: gene regulatory networks [4] [10] [20], ecosystem modeling [30] [18] etc. The need for a fuzzy representation arises from the specificities of the domain considered. Biological data sets are generally difficult to work with as they exhibit a certain inherent resilience to pure numerical analysis, are noisy, and are quite often incomplete. They also sometimes distinguish themselves by an overwhelming level of complexity that impairs human readability. In this context, some form of fuzzy representation becomes a necessity [31] [17]. Fuzzy Vector Envisionment (FVE) [6] answers these two requirements (differential capabilities and fuzzy representation) extremely well compared to another fully established fuzzy system such as QSIM[16] (see detailed explanation in section 2.2). It is also worth noticing that FVE is based on fuzzy values rather than symbolic ones. When adding these elements, we obtain a formalism which is quite unique in the area of Fuzzy modeling, and that has not realized its potential yet. Regarding the problem of choosing a learning algorithm; the size of the search space in complex biological systems is often too large for systematic exploration; we therefore plan to use Symbolic Regression in Genetic Programming (GP) as a nonsystematic learning algorithm. As we now enter a post-Darwinian era of AI [27], the combined use of Evolutionary Algorithms (EA) with other techniques in model learning is getting more frequent. So far, most of the closely related work makes use of a different representation. In a purely quantitative context, Koza tried to learn metabolic pathways [15], and others used EA in conjunction with statistical methods [20] [25] or with neural networks [22]. On the opposite side of the scale, Varsek [28] used a purely symbolic qualitative representation to learn such models. Evolutionary fuzzy systems, closer to our type of representation, did show some promising results [9]. Still, these papers have a significantly different and specialized focus: either the design of Fuzzy Logic Controllers [1] [12] or the problematic of Fuzzy Rule Based classifiers [5] [21][11]. One fact remains: at the present moment, evolutionary fuzzy systems have not yet been tested for learning complex dynamic biological models. The "post-genomic era" is fuelling the need to model genetic networks. The exponential growth of data is calling for new types of model learners. In this context, the unique combination of GP and Fuzzy Vector Envisionment is an original approach. We detail in section 2 the semi-quantitative representation. In section 3, we describe the implementation of the model learner. Section 4 contains experiments and results. Section 5 describes the focus of future work: learning metabolic pathways.

2. THE SEMI QUANTITATIVE REPRESENTATION

A model can be broadly defined as a set of constraints describing the relationships linking dependent and independent variables. As we are in the context of a dynamic system, this leads to a system of differential equations. Remembering that our goal is to work with biological data, there is a need to deal with imprecision and ambiguity. This is why, in practice, our model will look like a system of differential equations engineered to support fuzzy representation. Different fuzzy representation techniques are combined to satisfy the specific requirements we have here. We make use of a fuzzy 4-tuple parametric representation to deal with imprecise data and limit the size of the search space when generating constants. We add differential planes with Fuzzy Vector Envisionment[7] (by using vectors composed of multiple fuzzy 4-tuple). We manipulate these "fuzzy vectors" with specific operators originating from the Fusim[26] type of fuzzy arithmetic.

2.1 Fuzzy 4-Tuple Parametric Representation

The Fuzzy 4-Tuple Parametric Representation is a formalism that comes from Fusim[26]. It approximates fuzzy number curves while allowing the concept of degree of membership. This is very useful as it allows us to force down imprecise numerical input data into more manageable fuzzy intervals. This is also invaluable to limit the size of the search space as, when we generate constants, we do not generate any real numbers, but fuzzy values belonging to a specific fuzzy space. This way, instead of having potentially any real number representing the level of water in a tank, we can generate a limited number of constants expressed along a fuzzy scale (for example very low, low , intermediate, high, very high). A 4-tuple is described using four values a, b, α and β which define the fuzzy number "low" as shown graphically in Figure 1 ([3], page 53).

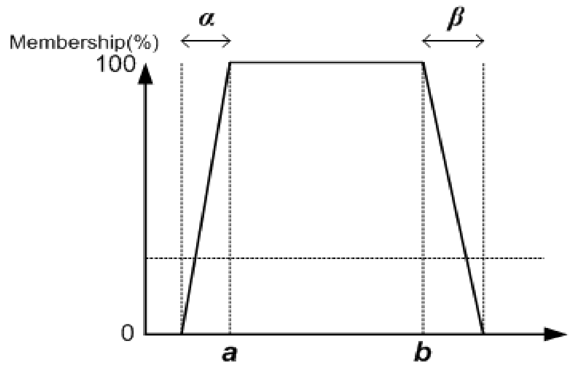


Figure 1: Fuzzy 4-tuple parametric representation.

2.2 Fuzzy Vector Envisionment and Differential Planes

Widely used qualitative model systems such as QSIM are generally limited to representing monotonic functions. The system Morgan [6] introduced multiple derivatives for variables. This allows us to not only observe the rate of change of a variable, but also its curvature. This amount of information would also be available with QSIM [16], but only after explicitly defining which variable is the derivative of another. This is an automated process in Fuzzy Vector Envisionment [6][3] and there is no limitation on the number of degrees of derivation one can examine. If a variable is defined by a vector of length three, it implies that the variable is defined by its magnitude and first and second derivatives. In terms of models of simple physical system, this gives information not only about magnitude, but also about speed and acceleration. The concept of Vector envisionment was extended to the fuzzy domain by Morven[7][8]. There are multiple derivatives per variable, but these are fuzzy values. The combination of Fuzzy Vector Envisionment and Differential Planes gives us the ability to use a powerful Qualitative formalism. We can control the number of derivatives for each variable, and thus simplify equations involving these derivatives. So, in term of symbolic Vector Envisionment, an observable purely qualitative data would be: $(+,+,-)$ where the first variable represents a positive level of water in a tank, the second variable tells us that this level is increasing (positive flow), and the third variable would indicate a decreasing flow. A set of similar data can be used to learn the qualitative dynamic model describing the behaviour of a system like the u-tube (detailed in a more quantitative way in section 4.3). In terms of Fuzzy Vector Envisionment, pure symbolic qualitative data would be replaced by fuzzy 4-tuples. This is to say that one fuzzy data set would consist in a fuzzy tuple representing the level, another one representing the flow and a last one representing the curvature of the flow. Quantitative measurements which could be seen as unreliable would be forced down into different fuzzy 4-tuples.

2.3 Use of Fuzzy Arithmetic

The model itself is a system of differential equations. As such it implies the presence of arithmetic operators. Standard arithmetic operators are applied to fuzzy tuples in the way of Fusim. Table 1 shows an example of basic arithmetic

Table 1: FuSim fuzzy arithmetic - the multiplication operator

Operation	Result	Conditions
$m \times n$	$ac, bd, a\gamma + c\tau - \tau\gamma, b\delta + d\beta + \beta\delta$	$m >_0 0$ and $n >_0 0$
	$ad, bc, d\tau - a\delta + \tau\delta, -b\gamma + c\beta - \beta\gamma$	$m <_0 0$ and $n >_0 0$
	$bc, ad, b\gamma - c\beta + \beta\gamma, -d\tau + a\delta - \tau\delta$	$m >_0 0$ and $n <_0 0$
	$bd, ac, -b\delta - d\beta - \beta\delta, -a\gamma - c\tau + \tau\gamma$	$m <_0 0$ and $n <_0 0$
	$m = \langle a, b, \tau, \beta\rangle$ and $n = \langle c, d, \gamma, \delta\rangle$	

operator[3] applied to Fuzzy 4-tuple parametric representation. Multiplying fuzzy numbers is non-trivial, and is included here for completeness, but it is not a contribution of this work, and the reader may simply view it as a black box for the purpose of understanding this paper. As we use FVE, we need to insert derivation into the standard Fusim arithmetic. One example is the multiplication operator. It is applied to the magnitudes of the variables, but also their derivatives. Therefore, if tuples A and B each have two derivatives such that their vectors are:

$$A = [a, \dot{a}, \ddot{a}]$$

$$\text{and } B = \left[b, \dot{b}, \ddot{b}\right] \text{, then}$$

$$AB = \left[(ab), (\dot{a}b + \dot{b}a), (\ddot{a}b + 2\dot{a}\dot{b} + \ddot{b}a)\right] \quad .$$

2.4 Evaluating the Distance between Two Variable Vectors

As we use Fuzzy Vector Envisionment, we need to calculate the distance between 2 such vectors [26] when evaluating the fitness of a model (see section 3.2). The purpose of this odd bit of fuzzy arithmetic is to measure the distance between a generated fuzzy variable and a target fuzzy variable. For every fuzzy 4-tuple a_i in the Variable Vector A, there is a corresponding fuzzy tuple b_i in the Variable Vector B. If none of them is unknown, we can add the sum of the squares of the difference between the centers of a_i and b_i, and the area of a_i and b_i. Let n be the number of tuples in the Variable Vectors A and B.
If x is a 4-tuple such that:

$$x = [\alpha, \beta, \gamma, \delta]$$

$$x^{center} = \left(\frac{\alpha+\beta}{2} + \frac{\gamma+\delta}{2}\right)/2$$

$$x^{area} = \frac{\gamma+\delta}{2} - \frac{\alpha+\beta}{2}$$

Then, the distance between A and B is:

$$\delta(a_1, ..., a_n; b_1, ..., b_n) = \sum_{i=1}^{n}\left[\sqrt{(a_i^{center} - b_i^{center})^2 + (a_i^{area} - b_i^{area})^2}\right]$$

We also take into consideration the possibility that some of the derivatives in the data sets are unknown. If $i > 1$ and a_i or b_i is unknown, then, the distance between a_i and b_i is considered to be null.

3. SETUP OF THE GENETIC PROGRAMMING MODEL LEARNER

The choice was made not to create the GP learner from scratch, but rather to use an open source package already available. The ECJ platform version 15 using the Java programming language was selected. The main problems encountered were more a matter of choice than programming design: the choice of the function and terminal sets, the fitness function, and the generic parameters.

3.1 Choice of the Function and Terminal Sets

The terminal set contains all the types of leaves of the GP tree and the function set contains all the different types of nodes.

3.1.1 *The Function Set*

There are two types of functions: arithmetic operators and the ADF's function defining branch.

- The arithmetic operators are: $+$, $-$, $\times$, $\div$, and the unary operators inverse and minus.
- The ADF's function defining branch: there is one ADF main function which accepts as many terminal arguments as there are constraints. These constraints can be reused as building blocks to build subsequent and more complex constraints. If there are n constraints, then the ADF function will accept as arguments subtrees named from $X1$ to Xn.

3.1.2 *The Function Set*

The terminal set contains the leaves of the tree. There are three types of terminals in our learner:

- An Ephemeral Random Constant (ERC). It generates constants at random.
- The variables: each variable is in fact a Vector Variable as previously defined by Fuzzy Vector Envisionment, and therefore consists of a set of 4-tuples fuzzy subsets. The first fuzzy tuple represents the qualitative magnitude of the variable, and the next tuples its different orders of derivatives. The number of degrees of derivatives can be set up by the user, but may drastically influence the number of computations.
- The ADF terminals referring to constraints. A model containing three constraints with the following function and terminal sets will be represented as a GP tree similar to the randomly selected sample in Figure 2.

3.2 Choice of the Fitness Function

We have two sets of variables: the positive data used in the learning examples, and the data produced by the newly generated model. We first introduce the input variables from

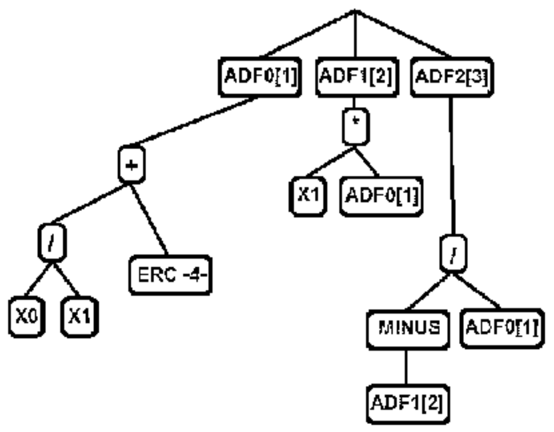


Figure 2: A model made of 3 constraints.

the positive examples in the newly generated model, and then calculate what endogenous variables it creates. We then measure the distance between learning data and generated variables. The distance between fuzzy variables used in fitness evaluation is calculated following the method described in section 2.4. We also introduce fitness calculation with incomplete data by considering that the distance between two variables of unknown magnitude or between a known and unknown magnitude is zero. This works the same way for derivatives. The best fitness is zero, and the worst, infinity. Two different cases of fitness evaluation are examined in this paper:

- The non-weighted fitness evaluation where the overall sum of the distances for all learning examples is used to evaluate the raw fitness.

- The weighted fitness function where the raw fitness is not just the linear sum of distances, but is weighted by the number of correct constraints found in the system. For n examples to learn from, on the first $n-1$ examples, we take away from the base fitness the proportion of constraints satisfied in the generated model. On the last example n, we use the non-weighted fitness evaluation. Not doing so seems to over-score the fitness, meaning that incorrect individuals would be evaluated as perfectly fit. In the end, the learner manages to find a satisfying solution with a gain of performance (see experiment results section 4.2). Our model has a raw fitness α (a positive number), and satisfies δ constraints over μ. We learn over n examples. The weighted fitness ϕ will be:

$$\phi = \alpha - \sum_{i=1}^{n-1} [\alpha \times (\delta/\mu)]$$

3.3 Generic GP Parameters

All experiments were run on one machine (a Pentium 3 GHz with 2GB of RAM). The system first evaluates the individuals using the Koza Standardised Fitness. The results are then displayed using Adjusted Fitness on a scale from 0.0 exclusive (worst) to 1.0 inclusive (best) in order to facilitate the reading and emphasize the differences between the very good individuals. The top part of the individuals are generated using the "full" method while the constraints are generated using the "Ramped half-and-half" method. There is no elitism. The method of selection in use is tournament selection, and the default tournament size is 7. Initially most of the experiments were done without mutation with a population from 20000 to 40000 individuals, these numbers being found at first pragmatically sufficient to solve problems such as our models at every attempt. The maximum number of generations was fixed to 200. Probability of crossover is 90%, and probability of reproduction is 10%. A second set of parameters was used using mutation: probability of crossover 90%, mutation 9%, and reproduction 1%. As it seems more convincing and computationally efficient to obtain results with a small population over lot of generations, we started to use a population of 4000 individuals over 1000 generations. With mutation, better results were obtained with such smaller population sizes.

4. EXPERIMENTS AND RESULTS

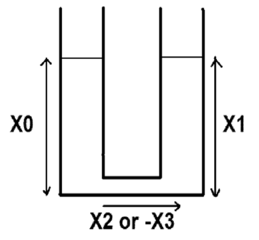
We will first describe the models used in the experiments by order of complexity (u-tube, cascaded tanks and coupled tanks). Then, using easy-to-learn simplified models, we will compare the number of generations needed to learn a model while using a normal and a weighted fitness function. We will also measure the average time per generation as a function of several parameters. Finally, we will analyze the performance of the system when trying to find the three models while varying the population size.

4.1 Description of the models

A model is a set of constraints defining the relationships between measured variables. In the simple physical systems we are examining, these variables (starting with the letter X) are either the levels of water measured, or the flows of water between one place and another place. Generated constants are linked to a level (starting with a letter l), and flow regulators (starting with a letter k) of water. The u-tube model is shown in Figure 3. In this example: $X0$ is the level of water on the left side (our input value), $X1$ is the level of water on the right side, $X2$ is the flow of water from left to right (so, strictly speaking, it is a derivative), $X3$ is the flow of water from right to left (a derivative too), l is a constant expressing the level of water when there is equilibrium, and k is a constant influencing the flow between both sides of the tube (for example the diameter of the pipe). Similarly, the slightly more complex models of cascaded tanks and coupled tanks are shown respectively in Figure 4 and 5.

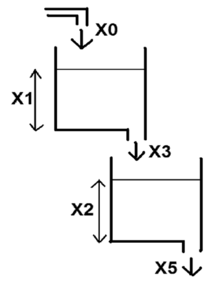
4.2 Comparing Normal Fitness with Weighted Fitness

Models which do not reuse constraints (ADFs are not reused) seem easier to find for the learner as they generally take fewer generations. A possible explanation may be that the models being simpler, it is easier to generate a matching tree. Whatever the reason, these "dummy" models are ideal for testing. The following graph describes the average number of generations needed over five runs to find a dummy model depending on the chosen fitness function. The models have from four to nine constraints. We compare results obtained with a normal fitness function and with a weighted fitness function (see Figure 6). The weighted fitness function seems to enhance performance compared to normal fitness



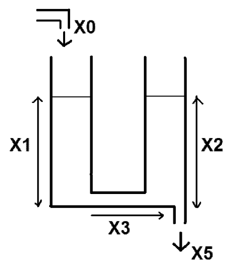
$$\begin{cases} X0 = inputvalue \\ X1 = (2 \times l) - X0 \\ X2 = \frac{dX0}{dt} = k \times (X0 - X1) \\ X3 = \frac{dX1}{dt} = minus(X2) \end{cases}$$

Figure 3: U-tube and Target Model.



$$\begin{cases} X0 = inputvalue1 \\ X1 = inputvalue2 \\ X2 = inputvalue3 \\ X3 = k1 \times (X1) \\ X4 = \frac{dX1}{dt} = X0 - X3 \\ X5 = k2 \times (X2) \\ X6 = \frac{dX2}{dt} = X4 - X5 \end{cases}$$

Figure 4: Cascaded Tanks and Target Model.



$$\begin{cases} X0 = inputvalue1, \\ X1 = inputvalue2, \\ X2 = inputvalue3, \\ X3 = k1 \times (X2 - X1), \\ X4 = \frac{dX1}{dt} = X0 - X3, \\ X5 = k2 \times (X2), \\ X6 = \frac{dX2}{dt} = X4 - X5 \end{cases}$$

Figure 5: Coupled Tanks and Target Model.

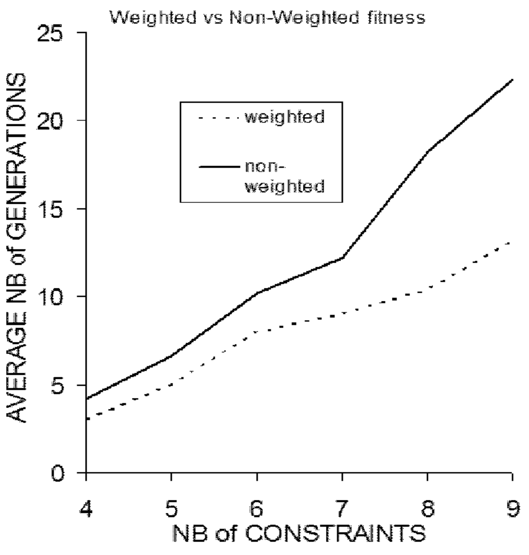


Figure 6: Weighted fitness vs. Normal fitness.

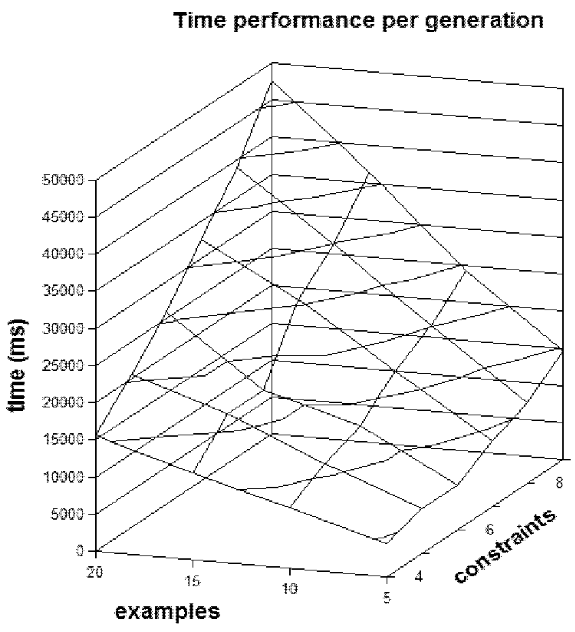


Figure 7: Time in function of two parameters.

as it reduces the number of generations by on average 46% for a 4 to 7-constraints model. From this observation, we advance the hypothesis that if we consider the fitness of a model as the linear sum of the fitnesses of its components (here, the constraints), then the convergence to a solution is slower than if the fitness is the weighted sum of the fitness of its components (detail of weighted fitness is described in Section 3.2).

4.3 Time Analysis

Figure 7 shows the average time per generation it takes to find a dummy model, depending on two parameters: the size of the learning set, and the number of constraints. This might suggest that time is not an exponential but rather a polynomial if not linear function of the size of the learning set, and of the number of constraints present in the target model. This was done without mutation.

4.4 Performances while learning the U-tube, Cascaded Tanks and Coupled Tanks

When not using mutation, depending on the complexity of the target model, the size of the population was sometimes insufficient to find a solution in the given number of generations. This drove us to increase the population size when no solution was found after five runs. The Table 2 describes each model, and gives the population size and number of generations required to converge to a solution.

This result could suggest that the minimum population size required is particularly dependant on the number of variables reused in each system. When using mutation with a population of size 4000, it took on average around 29 generations to find the model of the u-tube, 45 generations for the cascaded tanks, and 80 generations for the coupled tanks. Independently from the use of mutation, during the experiments, the learner did sometimes find the exact target model (constants included), but mostly approximations to it (in the case of the Utube, generally one constraint in four is slightly deviant). Here is one example of a u-tube approximation:

$$\begin{cases} X0 = inputvalue \\ X1 = (2 \times l) - X0 \\ X2 = \frac{dX0}{dt} = X1 - ((2 \times k) - X0)^2 \\ X3 = \frac{dX1}{dt} = minus(X2) \end{cases}$$

We see two possible explanations for this oddity:

- The use of a limited number of examples might enlarge the space of possible solutions. In this experiment, we learnt from a set of only ten fuzzy "data points". The obtained model might be perfectly adapted for these examples, but it also might not work if we add other data.
- There might be a problem with the fitness function which would over-evaluate the fitness of some models. Strangely, if that is the case, it should show less regularity in the results.

4.5 Possible improvements

We have observed how factors such as the number of constraints, the size of the learning set, and the population size parameters could influence the performance of the model learner. We are also driven to look for different ways to optimize the learner. GP may explore a broad search space more efficiently, but it remains a costly method from the point of view of computational resources. Further work is needed to find ways to minimize such costs.

- One interesting direction to investigate would be how to reuse models already found into the learning process by integrating them in the starting population.
- Plurality of models: for one set of data, several models might apply depending on the circumstances. This emphasises the need for a hybrid system type of representation [2] [19] that can express the interaction between

Table 2: Possible factors influencing the population size

Model	Constraints	Constants	Input values	Reused var.	Pop size	Generations
u-tube	4	2	1	4	20000	9
cascaded	7	2	3	<u>6</u>	30000	18
coupled	7	2	3	<u>7</u>	40000	39

the continuous time system and the discrete event system.

- Rule-based representation: for one set of data, one model might work in some circumstances, and another one under different conditions. It is important to identify what triggers this change. The representation has to articulate the links of causality between one state of variables and one particular model. (e.g. by including IF-THEN-ELSE statements).

- Complex mathematical operators: classic operators such as addition, subtraction and multiplication are sometimes too crude to capture the complexity of a relationship between variables. There might be a need for a representation that captures at least periodicity (trigonometric functions) and natural oddities such as exponential and logarithmic relationships.

- Complex behavioral operators: the behavior of a system can for example be repetitive to a certain point. To represent that, recursion and logical operators are necessary [13] [14].

- General viability of generated models can be preserved and improved by applying a strongly typed GP structure allowing only certain types of operators to be the children nodes of others [1].

5. TOWARD LEARNING METABOLIC PATHWAYS

After learning simple dynamic compartmental models, we plan to learn simplified metabolic pathways from time series data. Biochemical Systems Theory (BST) [23] [24][29] provides very promising representation tools for pathway identification tasks based on time series data. With the help of these theoretical tools, a metabolic pathway can be represented as a system of differential equations with a specific structure called an S-System. Every Constraint in the system represents a flow of chemicals. Each flow is the difference between production and degradation of chemicals. Figure 8 illustrates a simple branched pathway that we will try to learn. It has a constant input $X4$ and feedback inhibitions, where the production of $X1$ is controlled by two inhibitory mechanisms that are themselves affected by the synthesis of $X2$ and $X3$. By observing the values of the flows $\dot{X}1$, $\dot{X}2$, $\dot{X}3$, and the different concentrations of chemicals $X1$, $X2$, $X3$ and $X4$ across time, we can induce the following S-System:

$$\begin{cases} \dot{X}1 = \alpha_1 X2^{g_{12}} X3^{g_{13}} X4^{g_{14}} - \beta_1 X1^{h_{11}} \\ \dot{X}2 = \alpha_2 X1^{g_{21}} - \beta_2 X2^{h_{22}} \\ \dot{X}3 = \alpha_3 X1^{g_{31}} - \beta_3 X3^{h_{33}} \\ X4 = constant \end{cases}$$

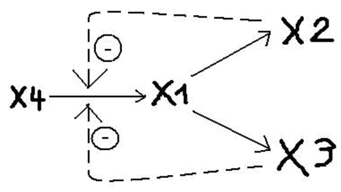


Figure 8: Branched pathway with 2 feedback inhibitions.

Where we hope to induce which chemicals intervene in which reaction, to estimate the kinetic orders g and h, and the constant rates of reactions α and β as fuzzy numbers. This is quite similar to one of Koza's experiments [15] with one major difference: we use fuzzy representation instead of pure quantitative data. He was using a Beowulf cluster of 1000 nodes to find the exact quantities in his model. In our case, the use of fuzzy sets should limit the size of the search space. Questioning a biologist about what were the points of focus of an analysis of a S-System, we discovered that for certain values such as chemical concentration, indications about the magnitude (in 10 fold) were more important than pure quantitative evaluations. This seems to indicate that fuzzy representation might be an appropriate tool. No experiment was started yet, as work is in progress to find plausible fuzzy scales for the values. We plan to first learn basic branching models and then simplified versions of anaerobic fermentation (Saccharomyces cerevisiae) and purine metabolism [29].

6. CONCLUSION

It is worth noticing the positive impact of using a nonconventional "weighted" fitness function during the evaluation. Further research might be done to see if this result can be generalized. At the moment, the learner manages to approximate compartmental models. More work still needs to be done to determine the source of some imprecision that sometimes prevents us from obtaining the exact target models. This is the biggest obstacle so far that separates us from starting to learn biological models. We hope to start learning S-Systems of basic branches soon enough, and incorporate them in a population of simplified models to start learning more complex metabolic pathways.

7. REFERENCES

[1] E. Alba, C. Cotta, and J. M. Troya. Evolutionary design of fuzzy logic controllers using strongly-typed GP. *Mathware & Soft Computing*, 6(1):109–124, 1999.

[2] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, mar 1999.

[3] A. Bruce. *JMorven: A Framework for parallel non-constructive qualitative reasoning and fuzzy interval simulation*. PhD thesis, Department of Computer Science , University of Aberdeen, 2006.

[4] T. Chen, H. He, and G. Church. Modeling gene expression with differential equations. *Pacific Symp. Biocomp. 99*, 4:29–40, 1999.

[5] B.-C. Chien, J. Y. Lin, and T.-P. Hong. Learning discriminant functions with fuzzy attributes for classification using genetic programming. *Expert Systems with Applications*, 23(1):31–37, 2002.

[6] G. M. Coghill. Vector envisionment and compartmental systems. Master's thesis, University of Glasgow, 1992.

[7] G. M. Coghill. *Mycroft: a framework for Constraint Based Fuzzy Qualitative Reasoning*. PhD thesis, Heriott-Watt University, 1996.

[8] G. M. Coghill and M. Chantler. Mycroft: a framework for qualitative reasoning. In *Proceedings of the Second International Conference on Intelligent Systems Engineering*, pages 49–55, Hamburg, Germany, 1994.

[9] O. Cordón, F. A. C. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5–31, 2004.

[10] de Hoon, Imoto, and Miyano. Inferring gene regulatory networks from time-ordered gene expression data using differential equations. In *ICDS: International Conference on Data Discovery, DS*. LNCS, 2002.

[11] M. J. del Jesus, F. Hoffmann, L. J. Navascués, and L. Sánchez. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *IEEE T. Fuzzy Systems*, 12(3):296–308, 2004.

[12] F. Herrera. An evolutionary paradigm for designing fuzzy rule-based systems from examples. Technical report, July 25 2000.

[13] J. R. Koza and D. Andre. Evolution of iteration in genetic programming. In *Evolutionary Programming*, pages 469–478, 1996.

[14] J. R. Koza, D. Andre, F. H. Bennett III, and M. Keane. *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufman, Apr. 1999.

[15] J. R. Koza, W. Mydlowec, G. Lanza, J. Yu, and M. A. Keane. Reverse engineering and automatic synthesis of metabolic pathways from observed data using genetic programming. Technical Report SMI-2000-0851, Stanford Medical Informatics, Nov. 7 2000.

[16] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986.

[17] R. Leitch, Q. Shen, G. Coghill, and M. Chantler. Choosing the right model. In *IEE Procedings - Control Theory and Applications*, pages 435–449, 1999.

[18] A. Nima, B. Stephen, L. Pat, and A. Kevin. Computational revision of ecological process models. In *Proceedings of the Fourth International Workshop on Environmental Applications of Machine Learning*, pages 13–14. Bled, Slovenia., 2004.

[19] S. Pettersson and B. Lennartson. Modelling, analysis synthesis of hybrid systems, July 31 1996.

[20] E. Sakamoto and H. Iba. Inferring a system of differential equations for a gene regulatory network by using genetic programming. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 720–726, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.

[21] L. Sanchez, I. Couso, and J. A. Corrales. Combining GP operators with SA search to evolve fuzzy rule based classifiers. *Inf. Sci*, 136(1-4):175–191, 2001.

[22] L. Sanchez and J. Otero. A fast genetic method for inducting descriptive fuzzy models. *Fuzzy Sets and Systems*, 141(1):33–46, 2004.

[23] M. A. Savageau. Biochemical systems analysis. i. some mathematical properties of the rate law for the component enzymatic reactions. *J Theor Biol*, 25(3):365–369, December 1969.

[24] M. A. Savageau. Biochemical systems analysis. ii. the steady-state solutions for an n-pool system using a power-law approximation. *J Theor Biol*, 25(3):370–379, December 1969.

[25] Y. Shan, R. I. McKay, and D. Paull. Building ecological models using genetic programming. In L. Wang, K. C. Tan, T. Furuhashi, J.-H. Kim, and X. Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02)*, Orchid Country Club, Singapore, 18-22 Nov. 2002.

[26] Q. Shen and R. Leitch. Fuzzy qualitative simulation. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4):1038–1061, July-Aug. 1993.

[27] L. Spector. Evolution of artificial intelligence. *Artificial Intelligence*, Vol. 170, Issue 18:1251–1253, 2006.

[28] A. Varsek. Qualitative model evolution. In *IJCAI*, pages 1311–1316, 1991.

[29] E. O. Voit. *Computational Analysis of Biochemical Systems : A Practical Guide for Biochemists and Molecular Biologists*. Cambridge University Press, September 2000.

[30] C. Walters, V. Christensen, and D. Pauly. Structuring dynamic models of exploited ecosystems from trophic mass-balance assessments. *Reviews in Fish Biology and Fisheries*, Volume 7, Number 2 / June, 1997:139–172, 2004.

[31] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.