Analysis of a Genetic Programming Algorithm for Association Studies*

Robin Nunkesser
Department of Computer Science
TU Dortmund
44221 Dortmund, Germany
Robin.Nunkesser@tu-dortmund.de

ABSTRACT

In this paper a Genetic Programming algorithm for genetic association studies is reconsidered. It is shown, that the application field of the algorithm is not restricted to genetic association studies, but that the algorithm can also be applied to logic minimization problems. In the context of multi-valued logic minimization on incompletely specified truth tables it outperforms existing algorithms. In addition, the facilities of the algorithm in the original application field are complemented by new results and experiments. This includes answers to the open questions of how to automatically choose the best individual in the last population and whether crossover is necessary for the algorithm.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and genetics*; B.6.3 [**Logic Design**]: Design Aids—*Optimization*

General Terms

Algorithms, Experimentation, Performance

Keywords

Genetic Programming, association studies, logic minimization

1. INTRODUCTION

A typical machine learning problem is to understand an unknown procedure e.g. from nature by its input-output behavior. The connected goal is to predict the output of the procedure for further inputs. When using genetic programming (GP) [13] for this purpose, we try to explain the observed data by a simple hypothesis. From the viewpoint of Statistics, this is a classification problem. Many interesting instances of this classification problem originate from biosciences, e.g. problems arising in genome analysis.

Nunkesser et al. [17] have presented a GP algorithm called Genetic Programming for Association Studies (GPAS) which is mainly intended for case-control genetic association studies, i.e. studies where genetic factors that may contribute to a medical condition are identified. They concentrate on data on single nucleotide polymorphisms (SNPs), i.e. genetic variations that occur when different base alternatives exist at a single base pair position. Overviews on approaches for this task are given by Heidema et al. [10] and Hoh and Ott [11]. In the data situations considered in [17] GPAS outperforms the known approaches. The main purpose of this paper is to extend and complement the results presented in [17].

In a first step, we test the qualification of GPAS for a different application. The problems GPAS originally works on are case-control studies on categorical variables. From a computational point of view, this is similar to (multi-valued) logic minimization on incompletely specified truth tables. The major difference is that logic minimization requires to find functions explaining the given truth table completely, while in a case-control study with a complicated underlying process (which is certainly the case, e.g. in genetic association studies) this would inevitably lead to overfitting, i.e. functions that are not able to predict for further inputs. So clearly, standard logic minimization approaches cannot hope to compete with GPAS on genetic association studies. But it is an interesting question, if GPAS is able to compete with logic minimization approaches on problems with an underlying logical process. Hence, we investigate the capability of GPAS for multi-valued logic minimization in this paper. The standard logic minimization tool for this purpose is Espresso MV [20]. For the boolean case of logic minimization on incompletely specified truth tables, there is also a GP algorithm by Droste [8] using ordered binary decision diagrams (OBDDs). Kristensen and Milterson [15] have shown that finding small OBDDs for incompletely specified truth tables is NP-hard. This indicates another reason, why logic minimizers are inapt for genetic association studies. For binary-valued logic minimization on completely specified truth tables and for Boolean concept learning various evolutionary approaches exist (see e.g. [1]).

Apart from this new application of the GP algorithm we consider open questions for the intended application. Considering the problem of overfitting, we introduce automated rules to detect the point, where overfitting begins. This is

*The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, Reduction of complexity in multivariate data structures) is gratefully acknowledged.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

one of the most important tasks in the application of GPAS or any other method that has to cope with the problem of overfitting. Further, we investigate the application to other categorical data and the necessity of crossover and one of the mutation operators for the algorithm.

In Section 2 the considered algorithm and the task it is intended for are presented. Section 3 presents the new results for the application of GPAS. Finally, Section 4 gives a summary and conclusions.

2. METHODS

The Genetic Programming algorithm GPAS [17] works on case-control genetic association studies. From a computational point of view the tackled problem is to understand a procedure that produces output in $B := \{0,1\}$ from inputs from a set $P := \{0,\dots,p\}$. GPAS basically searches for *multiple-valued input, binary-valued output functions* f, which are a mapping

$$f : P^n \to B \ .$$

The search for such functions is not done directly with multi-valued variables but with a mapping to Boolean variables. To our knowledge, there are three approaches to map multi-valued input variables to Boolean variables, leading to a Boolean algebra and an easier interpretation of the functions. The approaches differ in generality and the number of generated literals. The most general approach is by Rudell and Sangiovanni-Vincitelle [20] and defines

$$X^S := \begin{cases} 1, & \text{if } X \in S \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

and its complement

$$\overline{X^S} := \begin{cases} 0, & \text{if } X \in S \\ 1, & \text{otherwise} \end{cases} \ , \tag{2}$$

for a set $S \subseteq P$ of input values leading to 2^p distinct positive literals and 2^p distinct negative literals. Su and Sarris [23] define

$$X^{a,b} := \begin{cases} 1, & \text{if } a \le X \le b \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

and its complement

$$\overline{X^{a,b}} := \begin{cases} 0, & \text{if } a \le X \le b \\ 1, & \text{otherwise} \end{cases} \ , \tag{4}$$

leading to $p(p-1)/2$ distinct positive literals and $p(p-1)/2$ distinct negative literals. Finally, Dussault et al. [9] restrict Su and Sarris definition to literals $X^a = X^{a,b}$ where $a = b$ and their complements, leading to p distinct positive literals and p distinct negative literals.

To restrict the search space of GPAS in size, only functions in disjunctive normal form (DNF) are allowed, i.e. only disjunctions of conjunctions of these literals. An additional benefit of this restriction is that functions in DNF are easier to interpret or may even reveal biological meaningful information in case-control association studies. Depending on the problem at hand, it may also be convenient to allow only a subset of the literals, e.g. if there is a biological reason not to consider certain literals. Here, we only use literals of the form X^a and $\overline{X^a}$.

Definition 1. Let $f_p : P^n \to \{0,1,*\}$ be the *problem specific function* mapping each known input to the corresponding output of the considered problem and all other inputs to $*$. Further, let I be a logic expression in DNF consisting only of literals of the form X^a and $\overline{X^a}$ with $a \in P$ and let $f_I : P^n \to B$ be the corresponding function.

1. The *number of cases predicted correctly* by I is defined by
$$\text{cases}(I) := |\{i \in P^n \mid f_p(i) = f_I(i) = 1\}| \ ,$$
the *number of specified cases* by
$$\text{cases}(f_p) := |\{i \in P^n \mid f_p(i) = 1\}| \ .$$
2. The *number of controls predicted correctly* by I is defined by
$$\text{controls}(I) := |\{i \in P^n \mid f_p(i) = f_I(i) = 0\}| \ ,$$
the *number of specified controls* by
$$\text{controls}(f_p) := |\{i \in P^n \mid f_p(i) = 0\}| \ .$$
3. The *misclassification rate* (MCR) of I is defined by
$$\text{mcr}(I) := 1 - \frac{\text{cases}(I) + \text{controls}(I)}{\text{cases}(f_p) + \text{controls}(f_p)}$$
or the corresponding percentage.
4. The *length* of I is defined by the number of literals I consists of and denoted by $\text{length}(I)$.

The task is to find individuals that have a low MCR as well as small size, i.e. that offer good generalization and prediction. It is therefore convenient to conduct a multi-objective optimization. In this context, an individual *dominates* another individual, if at least one objective has a superior value and none an inferior. An individual is *pareto optimal*, if it is not dominated by another individual. Thus, we seek to find pareto optimal individuals. The GP algorithm proposed for this is described in the following. Note, that the termination criterion is not specified.

Algorithm 1 (GPAS [17]).

1. *Create an initial random population composed of two individuals each of which consists of one randomly selected literal.*
2. *Perform the following steps on the current generation:*
   (a) *Select all individuals in the population for reproduction, and draw seven of the individuals uniformly at random.*
   (b) *Conduct each of the following adaptions to one (mutations) or two (crossover) of the seven randomly selected individuals.*
   - *Perform a crossover: Combine one of the two chosen individuals with one randomly chosen monomial from the other individual.*
   - *Insert a new literal.*
   - *Delete a literal.*
   - *Replace a literal by a new literal.*
   - *Insert a new literal as a new monomial.*

- *Delete a monomial.*

(c) *Evaluate the fitness value of the adapted and reproduced individuals with fitness function f_1 that maps an individual I to a triple:*

$$f_1(I) := (cases(I), controls(I), length(I)) \ . \quad (5)$$

(d) *Select all adapted and reproduced individuals that are not dominated for the next generation.*

3. *If the termination criterion is fulfilled, then output the final population. Otherwise, set the next generation as current and go to step 2.*

3. RESULTS

All experiments are conducted with the R [18] package RFreak [16]. Most experiments are based on 100 runs of each algorithm to obtain significant results. To further substantiate the result statistical testing is frequently applied.

For this task, we choose a Wilcoxon signed rank test when two algorithms are compared and a Friedman rank sum test when more than two algorithms are compared (see e.g. [6] or [12]). Both tests are *nonparametric tests*, i.e. no assumption is made about the distribution of the stochastic optimizer outputs. This is a safe choice for the considered algorithms. The Wilcoxon signed rank test works on the differences of pairs of algorithm outputs, i.e. results different algorithms output for the same data set. The Friedman rank sum test on the ranks of blocks of algorithm outputs for the same data set. These values are ranked again and the rank sum of values supporting the hypothesis is considered. In all conducted experiments, we use one-sided tests. The null hypothesis therefore is that the distributions of the algorithm outcomes differ by a negative (positive) location shift. The alternative hypotheses is often stated in an informal way, implying that one of the outcomes contains better results, i.e. that the distributions of the algorithm outcomes differ by a positive (negative) location shift.

All figures based on experiment runs show boxplots. Boxplots illustrate the minimal and maximal outcomes of the experiment as well as the quartiles and the median of the results. When the boxes have different widths, the width indicates the number of observations building the box, i.e. the width is proportional to the square-roots of the number of observations in the groups.

3.1 Data Simulation

We need data simulation processes for two different purposes. We want to simulate data that is based on a logic circuit and data that is based on a natural process. To simulate both types of data, we use the R package scrime [22], which is intended to mimic a process from nature. The main data sets for this purpose are constructed in the following way. We generate data sets, consisting of m three-valued (0,1,2) inputs on $n \geq 10$ variables $X_1, \ldots, X_n$. The probabilities for a 0, 1, and 2 are 0.5625, 0.375, and 0.0625, respectively. The output y is then randomly drawn from a Bernoulli distribution with mean $\text{Prob}\big(Y = 1\big)$, where

$$\text{logit}\big(\text{Prob}(Y = 1)\big) = -\frac{1}{2} + \frac{3}{2}\big(X_3^0 X_9^0 X_{10}^0\big) + \frac{3}{2}\big(\overline{X_6^0} X_7^0\big) \quad (6)$$

such that the probability for being a case is 0.924 if for an input both logic expressions are true, and is 0.731 if one of them is true. This probability is still 0.378 if neither of the two is true.

To obtain an incompletely specified truth table based on a logic circuit, we set the probabilities for a 0, 1, and 2 equally to 1/3 and determine the output y directly by evaluating the underlying logic expression for each generated line such that the probability for being a case is 1 if one or both of the logic expressions are true, and is 0 if neither of them is true.

For the construction of slightly different data sets, we conduct appropriate modifications.

Table 1: Mean MCR when learning MUX11

	$m = 64$	128	256	512	1024
MCR	28.12	8.59	0.4	0.0	0.0

3.2 Data Sets Based on a Logic Circuit

Here, we consider the application of GPAS to data sets with incompletely specified truth tables based on a logic circuit. This is a new application field for GPAS which has some similarities to the intended application. The main difference is the need to explain the data completely, i.e. to obtain an MCR of 0. An application field that is related even closer to GPAS' domain is concept learning (in this case learning of multiple-valued input, binary-valued output functions). The results from [17] already imply that GPAS is suitable for this task. The results of the following experiments also lead to some further conclusions for this domain.

GPAS searches for single output functions, which complicates finding hard to solve benchmark instances from VLSI design (typically containing multiple outputs, see e.g. [2]). Typical benchmark instances for Boolean concept learners like the multiplexer [13] or the parity function [14] are for multiple reasons also inadequate. Firstly, there is no apparent meaningful variant with multiple-valued input and binary-valued output. Secondly, parity has a DNF of exponential size and GPAS only searches for DNFs. Nevertheless, we also include results on the multiplexer function (GPAS also works on binary-valued input functions), but our main focus is on constructed data sets based on DNFs as described in Section 3.1.

11-multiplexer

Firstly, we consider a multiplexer, more precisely the 11-multiplexer.

Definition 2. The *11-multiplexer* MUX11 : $B^{11} \to B$ is defined as

$$\text{MUX11}(a_2, a_1, a_0, d_7, \ldots, d_0) := d_{a_2 2^2 + a_1 2^1 + a_0 2^0} \ .$$

GPAS needs only a few seconds to find the DNF of MUX11 (which Espresso [2] is unsurprisingly able to compute even faster). This result already hints at the capability of GPAS for logic minimization. To test the ability of GPAS to learn MUX11, we determine the MCRs of runs on sampled training data sets with sizes $\{2^6, \ldots, 2^{10}\}$. We draw 100 samples for each size and report the mean MCR on the complete truth table of MUX11 in Table 1. The results in Table 1 indicate that GPAS is also useful for concept learning.

Table 2: Situations where Espresso MV terminated

	$m = 100$	1000	10000	25000	50000
$n = 10$	yes	yes	yes	yes	yes
20	yes	yes	yes	yes	no
30	yes	yes	yes	no	no
40	no	no	no	no	no
50	no	no	no	no	no

Comparison to Espresso MV (and Standard GP)

As mentioned before, the standard multi-valued logic minimization tool is Espresso MV [20]. To compare the performance of GPAS and Espresso MV, we construct data sets with the process described in Section 3.1 for all 25 combinations of $m \in \{100, 1000, 10000, 25000, 50000\}$ and $n \in \{10, 20, 30, 40, 50\}$. On these data sets we run GPAS and Espresso MV (which is based on the literals defined in (1) and (2)) each for a maximum of 1 hour. When the algorithms terminate within that time, both are able to compute solutions consisting of the minimal possible number of monomials, i.e. 2. But Espresso MV does not compute a solution in the given time span for larger m and n (see Table 2) and for $n \geq 40$ it even did not terminate after 16 hours. GPAS always finds the optimal solution, i.e. the true model, which leads to the conclusion that the application as a concept learner is also successful on these data sets.

It is an interesting question, if the success of our GP algorithm GPAS transfers to Standard GP [13], i.e. if the success is solely based on the evolutionary approach. For this purpose, we run Standard GP with the function set `F = {AND, OR, NOT, IF}` and a terminal set `T` comprising the same literals as used in GPAS on the 25 data sets. The result of the runs is that Standard GP is not able to compute a minimal solution in any of the runs in the given time span. Hence, GPAS drastically outperforms Standard GP on the considered data situations.

These first results already show that GPAS is successful in situations the standard approach cannot cope with.

Choice of the Fitness Function for Logic Minimization

Before we take a look at further situations, we consider the fitness function used. While there are many intuitive reasons to use three pareto objectives and therefore large populations when an underlying natural process is assumed, less objectives may also work when minimizing logical circuits. In addition to fitness function f_1 defined in (5) we consider the following fitness functions for an individual I, a problem specific function f_p, and a length restriction $\ell_{\max}$, i.e. the maximum value allowed for length(I) :

$$f_2(I) := (\text{cases}(I) + \text{controls}(I), \text{length}(I)) \quad (7)$$

$$f_3(I) := \frac{\text{cases}(I)}{\text{cases}(f_p)} + \frac{\text{controls}(I)}{\text{controls}(f_p)} - L \quad (8)$$

Note, that the extra term

$$L := \frac{\text{length}(I)}{\ell_{\max}(\text{cases}(f_p) + \text{controls}(f_p))}$$

prevents an individual of length $\ell+1$ that is not better than an individual of length ℓ in the remaining term of f_3 from being accepted. With these fitness functions, we conduct 100 runs with $n = 50$ and $m = 1000$, i.e. one of the more difficult

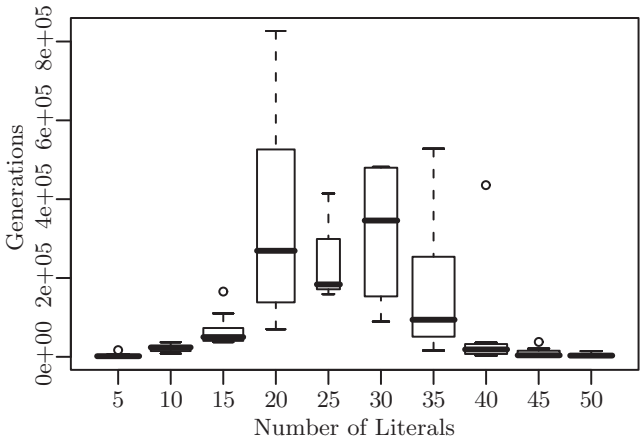

Figure 1: Generations needed to find an optimal solution for underlying DNFs of different sizes. Thinner boxes indicate that not all runs were successful.

situations from Table 2 until the algorithm stagnates for 10000 consecutive generations.

HYPOTHESIS 1. *For the task of logic minimization on the considered data situations, fitness function f_1 outperforms fitness functions f_2 and f_3*

While using f_1 leads to an optimal solution in 100% of the cases, f_2 never delivers an optimal solution and f_3 in only 9% of the cases. A Friedman test on the results of the experiment confirms that the outcomes derive from different distributions with a p-value of less than $2.2 \cdot 10^{-16}$. To clarify, if f_1 really outperforms f_2 and f_3 we conduct multiple Wilcoxon tests with Bonferroni correction. Both a comparison of f_1 with f_2 and a comparison of f_1 with f_3 yield a Bonferroni corrected p-value of less than $7.4 \cdot 10^{-15}$ indicating that the results using f_1 derive from a distribution with a location shift producing better solutions. We conclude, that the pareto optimization with three objectives is the best of the considered alternatives for all considered applications.

Minimization on Larger Functions

The application of logic minimization does of course work well in these examples, because the underlying function has a small DNF (although larger DNFs would not help Espresso MV either). Therefore, we look at the performance of GPAS on larger DNFs in the following. We reconsider the example with $n = 50$ and $m = 1000$ but this time, we generate data sets with more literals than in (6). Strong imbalances in the relation between the number of monomials and the size of the monomials lead to either too much cases or too much controls in the generated data. To cover a reasonable amount of different DNF lengths, we consider DNFs consisting of five monomials. For these DNFs we consider monomial sizes in $\{1, 2, \ldots, 10\}$, again for a maximum running time of one hour per run and report the generation the optimal solution is found.

The result, depicted in Fig. 1, indicates that up to DNF size 25 it gets harder for the algorithm to find optimal solutions and gets easier again after size 25. Note, that the box for size 25 is the thinnest, i.e. for this size many runs did not find an optimal solution in the time given. It is not surprising that it gets easier for the algorithm again for

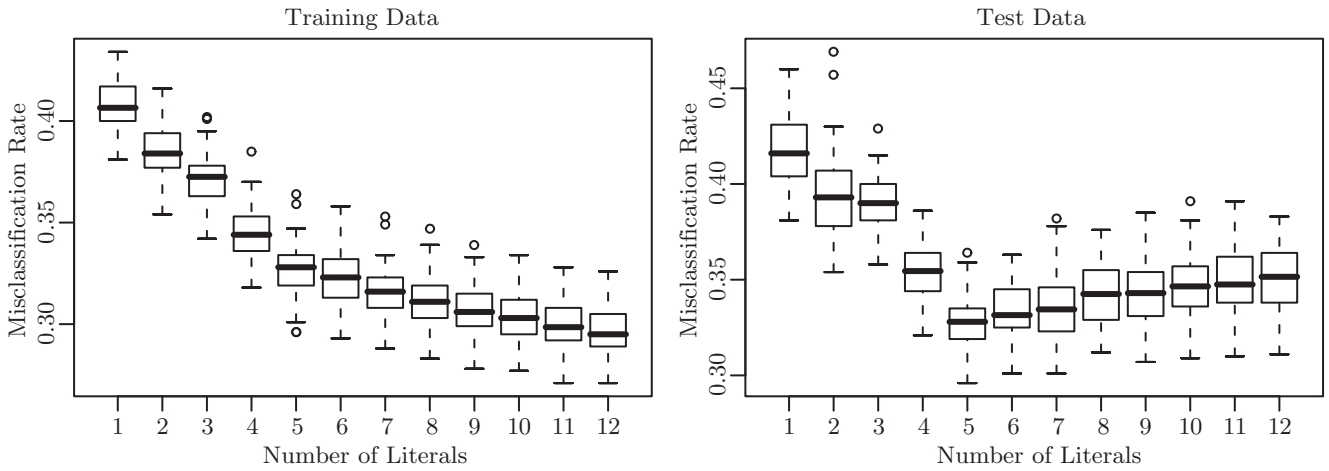

Figure 2: MCRs for the best individuals of different sizes in the 100 training and test runs.

large sizes, because of the values of n and m, i.e. for large sizes of the DNF, the relation between 0s and 1s in the output gets imbalanced and easier functions than originally intended are sufficient to explain the data. When considering $m \in \{100, 1000, 10000, 25000, 50000\}$ output values out of 3^n possible ones this is not surprising and typical for highly inspecified truth tables. Nevertheless, we see that GPAS does well in logic minimization in situations with a high amount of inspecified truth table values.

3.3 Data Sets Based on Natural Processes

As mentioned earlier, the main difference between data sets based on logic circuits and data sets based on natural processes is that explaining all cases and controls is a prerequisite for logic circuits while it would overfit data based on a natural process. Therefore, one of the main tasks when analyzing data sets based on natural processes is to determine the true size of the underlying model.

Automated Rules to Select the Best Individual

The main problem is illustrated in Fig. 2. The figure shows the result of 100 training runs on data sets built according to (6). The best individual, i.e. the individual with the best MCR, for each length between 1 and 12 is then tested against a test data set, which is also built according to (6). We see, that apparently and not surprisingly the longest individuals perform best on the training data. We also see, that the best individuals with the true model size 5 perform best on the test data. Our task here is to find automated rules that conclude from a training run result, that the true model size is indeed 5. Such rules help to evade overfitting of the data, i.e. they avoid choosing a longer individual in the training run with worse generalization and prediction properties.

A first apparent idea is to only consider points on the *convex hull* of the point set S consisting of the points $(\text{length}(I), \text{mcr}(I)) \in \mathbb{N} \times \mathbb{R}$ for all individuals I in a training run.

Definition 3. A subset S of the plane is called *convex*, iff for any pair of points $p, q \in S$ the line segment $\{\tau p + (1-\tau)q \mid 0 \leq \tau \leq 1\}$ is completely contained in S. The *convex hull* $\mathcal{CH}(S)$ of a set S is the smallest convex set that contains S.

When we only consider points on the convex hull, we exclude individuals with a gain in MCR that is relatively too small. For the remaining points we now consider the slope between any two points lying next to each other, i.e. the improvement in MCR per length when considering the longer individual. More precisely, for two points lying next to each other representing individuals I_1 and I_2 with $\text{mcr}(I_1) > \text{mcr}(I_2)$ we consider

$$\frac{\text{mcr}(I_1) - \text{mcr}(I_2)}{\text{length}(I_2) - \text{length}(I_1)} \ . \tag{9}$$

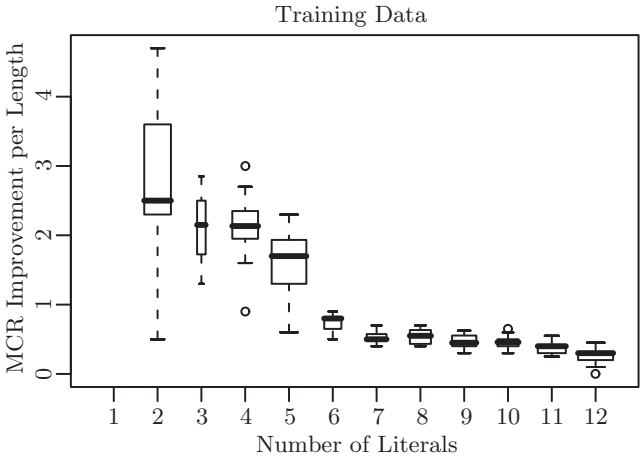

Figure 3: MCR improvement per length for individuals represented by points on the convex hull.

The resulting values are depicted in Fig. 3. The figure illustrates, that the choice of length 5 is a much more apparent choice after these considerations.

On this basis, we try a simple *threshold rule*. We accept the largest individual guaranteeing a higher MCR improvement than 1. We do this on the same simulated data sets as in [17] where a different (not reported) automated rule has been used.

HYPOTHESIS 2. *The threshold rule selects better individuals than the rule originally used in GPAS for data sets simulated according to (6).*

Table 3: Means and standard deviations of the misclassification rates of the applications of several discrimination methods to the simulated data sets.

		Simulation
GPAS with threshold rule	Mean	0.329
	St. Dev.	0.018
GPAS	Mean	0.335
	St. Dev.	0.025
Logic Regression	Mean	0.342
	St. Dev.	0.022
CART	Mean	0.371
	St. Dev.	0.025
Bagging	Mean	0.382
	St. Dev.	0.018
Random Forests	Mean	0.379
	St. Dev.	0.018

With the threshold rule we achieve a mean MCR of 32.92, where the automated rule used in [17] achieved 33.54. This difference may seem small at first sight, but when considering Table 3 it becomes apparent, that it is a significant improvement. In Table 3, the achieved MCR and the MCR achieved in [17] are compared to other discrimination methods. These include logic regression [19], CART [5], bagging [3], and Random Forests [4]. A Wilcoxon signed rank sum test on the outcome derives a p-value of less than 0.005, indicating that the new rule works better on the data. In 92% of the runs, the rule chooses the correct model size.

One may argue, that using a threshold poses too much adaption to the data situation. To investigate this, we try the threshold rule on all sizes between 1 and 12 again with 100 runs per size. The rule still determines the best model size in 70.41% of the runs. In 14.42% it misses the correct size by one and in 15.17% of the runs it misses the correct size by more than one. In conclusion, the automated rule is apt for all considered data situations. For different data situations it may be necessary to replace the constant threshold by a threshold function depending on the length of the individual. Another idea would be to use a gap statistic as proposed by Tibshirani et al. [24] to detect gaps between data clusters.

To restrict the size of GPAS' search space, it is very helpful to incorporate an alleviated version of the automated rule into the algorithm, dynamically changing the maximum size an individual may have. This forestalls overfitting already during the algorithm run and allows a better exploit of the search space in the relevant region. It is easy to store the individual with the best MCR for each length during the algorithm run. Computing a convex hull may be done in time $\mathcal{O}(n \log n)$ for n points in the plane (see e.g. [7] for details). Computation of the values defined by (9) additionally needs time $\mathcal{O}(n)$. The mutation and crossover operations in Algorithm 1 are all possible in amortized constant time due to the use of dynamic arrays, therefore we should not apply the automated rule every generation in order not to slow the algorithm too much.

In contrast to the automated selection rule, the following ideas and experiments also apply for the application as a logic minimizer.

Extension of the Algorithm to General Ordinal Data

So far we only looked at categorical data with three categories which is typical for SNP data. Nevertheless, it is very interesting to incorporate the facility to deal with more categories. Be it for SNPs mixed with, e.g. environmental data, or with other types of genetic data such as microsatellites or haplotypes. To simulate data for this task, we use a pair-decoding [21] approach, i.e. we replace triples of the ternary variables by 27-valued variables. For these data sets, we test, if additional literals $X^{0,b}$ and $X^{a,p}$ ($p = 26$) help and if the algorithm produces reasonable results for a higher number of categories. Note, that $X^{a,b} = X^{a,p}X^{0,b}$ and $\overline{X^{a,b}} = X^{0,a-1} \vee X^{b+1,p}$ and that the probabilities to build e.g. $X^{a,p}X^{0,b}$ and $X^{a,b}$ when using all literals specified in (3) and (4) are similar in GPAS. We do not consider the literals proposed in (1) and (2), because they mostly do not have a meaningful interpretation for the considered data and blow the size of the search space up.

We conduct 100 runs of the algorithm with and without the additional literals.

HYPOTHESIS 3. *Using the specified additional literals for data with more categories leads to better results.*

The MCRs obtained are slightly better when using the additional literals. The Wilcoxon signed rank sum test confirms that the outcomes derive from different distributions with a p-value of $2.98 \cdot 10^{-5}$. The overall results of the runs were comparable to the runs on data sets without pair-decoding. Therefore, GPAS is suited to run on ordinal data other than SNP data especially when incorporating additional literals.

The Role of the Crossover Operation

Finally, we investigate the claim Nunkesser et al. [17] raised, that using the crossover operation specified in Algorithm 1 speeds up computation but does not necessarily lead to better results. We run Algorithm 1 and the same algorithm without crossover 100 times until they create the 10000th individual, i.e. both algorithms create the same number of individuals. Afterwards, we test with a Wilcoxon signed rank sum test, if the best individuals in the last populations of the algorithm with the crossover operation are significantly better than without crossover, i.e. if the crossover operation leads faster to good results.

HYPOTHESIS 4. *GPAS with crossover obtains good results faster than GPAS without crossover on data built according to (6).*

The conducted experiment shows, that the average MCR of the best individuals after 100 runs is indeed better when using crossover. A Wilcoxon signed rank sum test confirms that the outcomes derive from different distributions with a p-value of less than $1.45 \cdot 10^{-8}$.

To analyze if crossover is necessary to attain better solutions, we repeat the experiment until the 1000000th individual is created. This is long enough to reach stagnation in the parts of the final population typically containing the best individual.

HYPOTHESIS 5. *GPAS with crossover leads to better results than GPAS without crossover on data built according to (6).*

The average MCRs are indeed better when using crossover and the Wilcoxon signed rank sum test derives a p-value of less than $8.34 \cdot 10^{-9}$, indicating that crossover not only speeds up computation but is also necessary for the considered data situations to attain good solutions.

Similar to the crossover operation, one might ask, if the mutation replacing a literal by another literal is necessary for the algorithm, because an insertion and deletion mutation could lead to the same individual. To clear this question, we conduct the same experiments as for the crossover operation again for this mutation. The experiment corresponding to the one for Hypothesis 4 derives a p-value of less than 0.002 backing up that this mutation speeds up computation. The corresponding experiment for Hypothesis 5 derives a p-value of less than 0.02 leading to the same conclusion as for crossover. The considered mutation operation speeds up the algorithm and leads to better results for the data situations looked at.

Changes to the Algorithm

To summarize and test the results of Section 3.3 we propose to use the following version of Algorithm 1 which incorporates the dynamic length restriction and the automated selection rule. The crossover and mutation operations are unchanged due to the support of their necessity by our experiments.

ALGORITHM 2.

1. *Initialize* $\ell_{max} := 12$.
2. *Create an initial random population composed of two individuals each of which consists of one randomly selected literal.*
3. *Perform the following steps on the current generation:*
   (a) *Select all individuals in the population for reproduction, and draw seven of the individuals uniformly at random.*
   (b) *Conduct each of the following adaptions to one (mutations) or two (crossover) of the seven randomly selected individuals.*
      - *Perform a crossover: Combine one of the two chosen individuals with one randomly chosen monomial from the other individual.*
      - *Insert a new literal.*
      - *Delete a literal.*
      - *Replace a literal by a new literal.*
      - *Insert a new literal as a new monomial.*
      - *Delete a monomial.*
   (c) *Evaluate the fitness value of the adapted and reproduced individuals with fitness function* f_4*:*
   $$f_4(I) := \begin{cases} f_1(I), & \text{if } length(I) \leq \ell_{max} \\ (0, 0, \ell_{max} + 1) & \text{otherwise} \end{cases}.$$
   (d) *Select all adapted and reproduced individuals that are not dominated for the next generation.*
4. *Let* g *be the number of the current generation. If* $g \bmod (\ell_{max} \log \ell_{max}) \equiv 0$ *then update* ℓ_{max} *by computing the longest individual* I *where the value defined by (9) is larger than 1. Set* $\ell_{max} := length(I) + 2$.
5. *If the termination criterion is fulfilled, then output the final population and the best individual determined by the threshold rule. Otherwise, set the next generation as current and go to step 2.*

To compare the new version of the algorithm to the old one, we conduct 100 short runs (10000 individuals) on data sets built according to (6). The short runs are used to investigate if the dynamic adaption of the maximum allowed individual size derives faster results in the interesting region of the search space. To measure this, we count the number of generated individuals with the optimal size.

HYPOTHESIS 6. *Algorithm 2 obtains individuals with the optimal size faster than Algorithm 1 on data built according to (6).*

Comparing the number of individuals with the optimal size exhibits a larger number for Algorithm 2. A Wilcoxon rank sum tests reinforces this observation by delivering a p-value of less than 0.002. As a second quality indicator, we follow an idea of Zitzler and Thiele [25] and use a *hypervolume indicator*. Again, we consider all individuals $\mathcal{I}$ with the optimal size and measure the hypervolume of the point set $\{(\text{cases}(I), \text{controls}(I)) \mid I \in \mathcal{I}\}$ with reference to the point $(0, 0)$.

HYPOTHESIS 7. *Algorithm 2 obtains individuals with a higher quality measured by the hypervolume quality indicator than Algorithm 1 on data built according to (6).*

The experiment delivers a higher average hypervolume for Algorithm 2. The Wilcoxon rank sum supports this result, giving a p-value of less than 0.004.

Similar to the conducted experiments for the crossover operation, we consider the results of a second experiment with 100 long runs (1000000 individuals) and the corresponding hypotheses.

HYPOTHESIS 8. *Algorithm 2 obtains more individuals exhibiting the optimal size after creating 1000000 individuals than Algorithm 1 on data built according to (6).*

The result of the experiment backs up the hypothesis and the corresponding Wilcoxon test reports a p-value of less than $5 \cdot 10^{-5}$.

HYPOTHESIS 9. *Algorithm 2 obtains individuals with a higher quality measured by the hypervolume quality indicator after creating 1000000 individuals than Algorithm 1 on data built according to (6).*

Again, the hypothesis is backed up by the experiment and the statistical test, this time with a p-value of less than 0.003. All in all, these results demonstrate that Algorithm 2 explores the promising regions of the search space better than Algorithm 1 and therefore has a higher chance to find good generalizing individuals.

4. CONCLUSIONS

In this paper, new complementing results and a new application for a Genetic Programming algorithm for association studies have been presented. The new application as a logic minimizer proved to be successful for the considered data

situations. Considering the main application of the algorithm, important new results have been introduced. The automated rule to choose the best individual in the final population is a big step forward towards solving the problem of overfitting. Further, the extension of the algorithm to other ordinal data has been demonstrated to be reasonable. Experiments on the necessity of crossover for GPAS complement the claims raised in [17].

It remains an interesting task to test the new results on further data and further applications. Many ideas involved are also helpful for other situations than categorical input variables and binary output variables. Apart from the fact, that such situations are interesting in themselves, there are also many applications on such data. Gene expression and epidemiological data are examples where the ideas of the algorithm and the ideas presented here might help.

5. REFERENCES

[1] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[2] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1984.

[3] L. Breiman. Bagging predictors. *Mach. Learn.*, 26:123–140, 1996.

[4] L. Breiman. Random Forests. *Mach. Learn.*, 45:5–32, 2001.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth, Belmont, CA, 1984.

[6] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley and Sons, third edition, 1999.

[7] M. de Berg, M. van Krefeld, M. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*. Springer, 1997.

[8] S. Droste. Efficient genetic programming for finding good generalizing Boolean functions. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzo, H. Iba, and R. L. Riolo, editors, *Proceedings of the Second Annual Conference on Genetic Programming*, pages 82–87, San Francisco CA, 1997. Morgan Kaufmann Publishers, Inc.

[9] J. Dussault, G. Metze, and M. Krieger. A multivalued switching algebra with boolean properties. In *Proceedings of the sixth international symposium on Multiple-valued logic*, pages 68–73, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[10] G. A. Heidema, J. M. A. Boer, N. Nagelkerke, E. C. M. Mariman, D. L. van der A, and E. J. M. Feskens. The challenge for genetic epidemiologists: How to analyze large numbers of SNPs in relation to complex diseases. *BioMed Genet.*, 7(23), 2006.

[11] J. Hoh and J. Ott. Mathematical multi-locus approaches to localizing complex human trait genes. *Nat. Rev. Genet.*, 4:701–709, 2003.

[12] M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. John Wiley and Sons, second edition, 1999.

[13] J. R. Koza. *Genetic Programming*. The MIT Press, Cambridge. Massachusetts, 1992.

[14] J. R. Koza. *Genetic Programming II*. The MIT Press, Cambridge. Massachusetts, 1994.

[15] J. T. Kristensen and P. B. Miltersen. Finding small obdds for incompletely specified truth tables is hard. In *Proceedings of COCOON 2006*, volume 4112 of *Lecture Notes in Computer Science*, pages 489–496. Springer, 2006.

[16] R. Nunkesser. *RFreak: R/FrEAK interface*, 2007. R package version 0.2-0.

[17] R. Nunkesser, T. Bernholt, H. Schwender, K. Ickstadt, and I. Wegener. Detecting high-order interactions of single nucleotide polymorphisms using genetic programming. *Bioinformatics*, 23(24):3280–3288, 2007.

[18] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007.

[19] I. Ruczinski, C. Kooperberg, and M. LeBlanc. Logic regression. *J. Comput. Graph. Stat.*, 12:475–511, 2003.

[20] R. L. Rudell and A. Sangiovanni-Vincentelli. Multiple-valued minimization for pla optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(5):727–750, 1987.

[21] T. Sasao. Multiple-valued decomposition of generalized boolean functions and the complexity of programmable logic arrays. *IEEE Trans. Comput.*, 30(9):635–643, 1981.

[22] H. Schwender and A. Fritsch. *scrime: Analysis of High-Dimensional Categorical Data such as SNP Data*, 2008. R package version 1.0.0.

[23] S. Y. H. Su and A. A. Sarris. The relationship between multivalued switching algebra and boolean algebra under different definitions of complement. *IEEE Trans. Comput.*, 21(5):479–485, 1972.

[24] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[25] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. Evolutionary Computation*, 3(4):257–271, 1999.