Expansion of the GeLog System by Automatic Adaptation of Parameters

GABRIELLA KÓKAI
Department of Programming Languages Friedrich-Alexander University
Martensstr. 3. D-91058 Erlangen, GERMANY

Abstract: - Different techniques have been introduced for the improvement of Evolutionary Algorithms (EA) to increase their performance as well as to achieve better results. Adaptive methods are commonly used. In the scope of this work these extensions are applied to *GeLog*, a system which implements a combination of two different approaches for automatic programming: inductive logic programming and genetic algorithms. [*]

Key-Words: Genetic Programming, Hybrid Intelligent Systems, Inductive Logic Programming

1 Introduction

The *GeLog* system ([6]) is a combination of inductive logic programming ([8],ILP) from the area of machine learning and the genetic algorithms from the area of evolutionary algorithms intended to avoid some problems of existing ILP systems and to join the advantages of both systems. The goal of this paper is to simplify the choice of evolution parameters, which has to be repeated for each new problem, through an automatic adaptation of parameters. As *GeLog* is a very flexible system, which can be employed for various kinds of problems, there are also a big number of possibilities how to use it. Good knowledge of the problem and the system, which the user often lacks, are the precondition of a successful search. Therefore *GeLog* was extended and three new adaptation procedures were added which adapt the possibilities of application of the genetic operators during evolution to the change of fitness or the success rate of operators. In the remainder of this paper first a brief overview about *GeLog* system is given. Section 3 contains the formal description of EA. In Section 4 the automatic adaptation is discussed. The expansion of *GeLog* is described. In Section 6 one example is given to demonstrate the learning ability. Finally in Section 7 a summary and outlook of future work will be made.

2 The GeLog System

GeLog ([6]) is a learning program for the automatic generation of logic programs, which combines the techniques of inductive logic programming (*ILP*, [8]) and the genetic programming (*GP*). Solutions, which can be expressed by Prolog programs, are learned on the basis of background knowledge and training examples. The inductive logic programming is embedded through the use of Prolog programs as phenotypes of the individuals, which, as in other *GP* systems, are depicted on an object graph. The individuals are evaluated by a Prolog process in order to determine the fitness. The process is initiated with the background knowledge and the positive and negative training examples. The *ILP* operators of generalization and specialization are mapped on mutation and recombination. As *GeLog* offers multiple possibilities, the relevant adjustments from a configuration file, which has to be indicated when the program is called, are read in. The goal clause and the possible predicates are also extracted from the configuration file. The reproduction of the examples on the predicates is explained in the background knowledge. The Prolog process is initialized with the basic knowledge and the examples. After generation of the initial generation from these files the real evolution cycle is started. The generations are separately stored in different files and statistics about the completeness and the consistency of the individuals are elaborated. After the end of the program it is thus possible to analyse in great detail the evolution on the basis of the statistics of the populations.

3 Formal description

Just as the evolutionary algorithms have different development streams, there are also different approaches for the formulation of the respective methods. As already explained in [5] we are going to employ a model of description which can mainly be deduced from the theory of evolution strategies but also enables a description of other procedures. First, a population of μ individuals is generated. A selection of the parental generation is carried out in accordance with the fitness of the individuals. Usually, the user has to define the fitness function as well as to determine the selection operator. In contrast to natural sexual reproduction, EAs also permit recombination with more than two or just one partner. The exact number can therefore be indicated explicitly by factor ρ. Mutations, however, always concern just one individual. The exact parameters of the genetic operators (i.e. usually their application probability) are indicated by option vector λ. λ new individuals are produced in

[*] This work is supported by the grants of the Bayerisches Staatsministerium für Wissenschaft

each generation.

$$\left(\mu_0/\rho_0 \overset{+}{,} \lambda_0 \,||\, \Omega_0\right)^{\gamma_0} \tag{1}$$

μ_0:	number of parental individuals (size of population)
ρ_0:	size of recombination pool (usually 2)
λ_0:	number of descendants
γ_0:	number of generations
Ω_0:	parameters of operators employed

There are two more strategies for the creation of a generation of descendants. The comma strategy only accepts new individuals and lets the parents die out. According to the plus strategy, in contrast, parents are also seen as possible candidates.

4 Automatic adaptation of parameters

The user has a lot of possibilities to influence the procedure and thus also the results of the search by using evolutionary algorithms. However, it is not possible to indicate a general and optimal parameterization, as it depends on the nature of the problem and on the possible genetic operators. Especially with the use of flexible programming systems like *GeLog*, which are not designed to specialize on one class of problems and, furthermore, can be extended with regard to the operators that are to be employed, the user already needs good knowledge of system and search area in order to be able to take a useful decision. It would therefore be desirable, to develop an at least partly automated adaptation process. Another problem is however caused by the evolution run itself. At the beginning, operators that lead to large adaptations are preferable, because they can explore the search area more quickly. During the approach towards the searched maximum, however, small adaptations get more important in order to avoid overshooting the mark. Therefore, the use of operators has to be adjusted during run time. In the following, different approaches how to carry out these adaptations are explained, the main emphasis lies on the adaptation of operator probabilities.

4.1 Absolute and empirical updating rules

Basically, it is possible to distinguish between two different methods for adaptation of parameters (cf [1]). For an *absolute adaptation*, a set of statistics of various generations or populations is elaborated. According to the evolution that thereby becomes apparent, the parameters that are subject to change and partly also the size of adaptation are established. The method is absolute, as the necessary adaptations are already known before the real evolution. For such a procedure it is necessary to predict the course of evolution up to a certain degree. This prediction is justified, as long as it actually reflects an existing regularity. In [10], for example, the $1/5$ rule of success is employed, which adapts the global mutation variable according to the success rate of mutation. The assumption especially applies to a search area that can be described by a smooth multimode function, as it is often the case with evolution strategies. Such a rule, however, cannot be applied to other algorithms such as, for example, genetic programming, whose fitness function rises by leaps and bounds.

With *empirical adaptation*, on the contrary, the strategy parameters evolve together with the population. Usually special operators for the evolution of parameters are employed. It would therefore be possible to speak of a parallel evolution of parameters, which, however, will be evaluated in accordance with the fitness of the individuals. Procedures using empirical adaptation are also called *self-adaptive*. As a general rule, adaptation takes place at individual level and is therefore connected to their preservation within a population. Previous knowledge of the present problem is not necessary, which makes the process more adaptable.

4.2 Level of adaptation

Parameters can be adapted on different levels of representation, depending on which parameters are to be adapted. In the following, different approaches are compared. For this means, we will distinguish between adaptation on the population level, on the individual level and on the component level. A more detailed comparison can be found in [1].

4.2.1 Adaptation on the population level

In the easiest case, an evolution run is determined by only a few global parameters that apply to the whole population. As these values are relatively independent of the problem class, such an adaptation can be largely applied. Taking a look at the evolution of a population, you will see that first, the search area is explored into every direction because of the generation of the descendants, and only with the help of the selection operator the search can be focussed. Through adaptation of the global parameters it is possible to focus on promising areas, too. One example is the ARGOT system (cf [13]) In this system the interpretation of individuals is modified dynamically and adapted to the convergence degree of the population. In this case, the adaptation leads to a refined search when evolution becomes stagnant. In contrast to the ones mentioned before, there are also systems that carry out an empirical adaptation on the population level. In [14], for example, the application probability of recombination operators is determined by the number of flags added to the individuals.

4.2.2 Adaptation on individual level

The fact that evolutionary algorithms evolve whole generations parallely to each other does not always lead to a uniform adaptation of all individuals. Especially if the individuals are widely spread in the search area and their direct environments therefore differ very much, a global parameterization is not sufficient. For an individual adaptation, each individual needs additional information on the basis of which the adaptations can be realized. Therefore, more specific values are optimized, for example in order to determine the crossover points. In most processes, evolution is completed with this additional parameter, i.e. they are self-adaptive. However, also in this case there are systems providing absolute adaptation. In the programming system depicted in [11], for example, a fitness function is used in order to evaluate the structure of individuals. If one part of the structure has a greater fitness, no more crossover points may be within its borders for it to remain complete.

4.2.3 Adaptation on component level

Finally, it is possible to take another step down and carry out adaptation of different components of the individuals. The main difference between this approach and the one mentioned before is that here, the kind of manipulation of different components can be specified independently and does not depend on other components. This form of adaptation is used in [2] for excluding further manipulation of certain components. A frozen/unfrozen flag which (like the other strategy parameters) was subject to self-adaptation is thus allocated to each component. In [12], a self-adaptive procedure for ES based on the components is described, which contains a specific variable parameter for each real-valued element of the individuals. It serves to focus the search into the direction, where the predecessors were more successful.

5 Expansion of GeLog

Especially due to the number of possible operators for recombination and mutation as well as the possibilities of adjustment in *GeLog* the user must have familiarized himself with the relevant problem and the effects of the operators for this type of problem. Therefore, one of the aims of this work was to help the user with this choice by introducing an automatic adaptation of the application probabilities of the operators.

An absolute adaptation on the population level was implemented for the *GeLog* system, in order to have a procedure which, on the one hand, causes relatively little or no problems, and on the other hand ensure that the additional effort necessary because of the adaptation is as small as possible. First we will explain the expansion which was introduced in order to save the evolution history of each individual and analyse it. In the following part, the realization of adaptation limits which are used for all adaptation procedures will be explained. During first tests of adaptation, certain defects of the initially implemented fitness-based adaptation became visible. Therefore, a success-based adaptation was added to the system. During the design of the different classes of history we had the idea to consider not only the generation of the predecessors, but also the whole (or also limited as to its length) genealogical tree.

5.1 Expansion of individuals

For adaptation, data about the evolution history of each individual are necessary. In the original *GeLog* system, each operator was used according to its application probability. It was, in particular, possible to use various recombination operators successively, although this did not seem to be very useful, taking into account their potential for adaptation. Through an orientation towards natural evolution it was possible to limit the procedure to just one application of recombination. The recombination operators always operate with two parental individuals, of whom only the fitness as well as the operator applied have to be included into the history. For the mutation operators, however, multiple mutation was maintained and a variable limited to exactly one mutation was permitted. Therefore, an operator vector is listed in the history. To begin with, only the previous generation was regarded. However, it should also be possible to consider various generations for adaptation. The data is therefore stored together with references to the predecessors of the parents. As the adaptation is intended to adapt the application probabilities of each operator, it would be possible to evaluate them separately after each operation. We refrained from applying this method, however, as when it comes to multiple mutation this would require too much work and, furthermore, effects which are only produced when various operators work together could not be taken into account. In addition, the results would be too insignificant. Therefore, each operator participating in the generation of new individuals counts with the total number of adaptations.

5.2 Limits of adaptation

As it is possible that all operators are necessary to reach a global maximum, no operator may disappear because of adaptation. Therefore, a minimal probability can be indicated with the parameter *minProb* which guarantees the maintenance of all operators. In case of multiple mutation, it must, in addition, also always be possible to omit an operator. The upper limit is constituted by the complement $maxProb = 1 - minProb$. In order to make it easier to keep to the minimal limit, it is allocated to each operator in the case of multiple mutation and distributed equally to the operators in case of simple mutation and recombination (cf also [9]). Therefore, the factor $(1 - minProb)$ is added to the real adaptation in order to avoid passing over the limit.

5.3 Fitness-based adaptation

In analogy to the fitness-proportional selection, the application probabilities of operators can be adjusted according to the change of fitness. The version implemented here is mainly based on the description found in

[5]. During adaptation, three factors are taken into consideration for each operator ω: the evaluation of operators of the previous generation $\gamma(\omega, t-1)$, the mean of change of fitness $\Delta\gamma(\omega, t)$ and an expiry rate $\gamma_{decay}(\omega)$. This leads to the following formula:

$$\gamma(\omega, t) = \gamma(\omega, t-1) \cdot \Delta\gamma(\omega, t) \cdot \gamma_{decay}(\omega) \qquad (2)$$

First of all, the change of fitness of all individuals is determined and their mean is calculated. For this means, the mean of the fitness values of the parental individuals is compared to fitness τ of the filial individual for each operator. The standardization is then determined by the number of applications of the respective operator in the present generation. In the following formula for the mean of the change of fitness, $P(\omega, t)$ refers to all individuals onto which the operator ω was applied in the generation t.

$$\Delta\gamma(\omega, t) = |P(\omega, t)|^{-1} \cdot \sum_{s \in P(\omega, t)} \frac{\tau(s)}{\tau(parents(s))} \quad (3)$$

The expiry rate γ_{decay} serves to weaken fitness-neutral operators which neither lead to an improvement nor to a deterioration. The factor is of especial interest, when the operator already counts with a high probability of application. In the case of a simple mutation, the high evaluation of the operator impedes the improvement of other operators, as they are always evaluated relatively to one another. In the case of multiple mutations, however, the expiry rate serves to "punish" the operator for its weakening effect. In general, the value of γ_{decay} should not be much smaller than 1, as the amount of change of fitness is often very small. In order to maintain the new application probability, this $\gamma(\omega, t)$ is evaluated together with the factor *maxProb* and added to the minimal value of operators. The formula for the calculation of application probabilities is:

$$\mathscr{P}_{fitness}(\omega, t) = minProb + maxProb * \gamma(\omega, t) \quad (4)$$

Another possibility of implementation for a fitness-based adaptation is presented in [9]. There, the fitness of the parental node as well as the operator used are stored in pairs for each increase of fitness. p_{all} points are equally distributed to all operators (this corresponds to the above mentioned use of *minProb*). The remaining $p_{left} = 100 - p_{all}$ are distributed to those operators, through whose application the new fitness is closest to the fitness of the parental node. Each operator receives at least one point. If the increase of fitness is under p_{left}, the other points go to the remaining class p_0, which corresponds to the probability that no operator will be used. The use of $minProb$ was adopted from this implementation. Apart from that, the use of percentage points for populations of any possible size does not seem useful for this implementation. Through the selection of operators with the smallest possible change of fitness it is tried to implement adaptation in smaller steps. For this means, the direct application of the previous evaluation as in [5] is more fit and the procedure is more comprehensible when an expiry rate is also introduced.

5.4 Success-based adaptation

One disadvantage of fitness-based adaptation in accordance with [5] is that the value depends directly on the value of the change of fitness. The closer one gets to the maximum, the smaller the positive deviations are and the bigger the probability that application of operators will lead to deterioration. In the case of success-based adaptation, it is tried to loosen the dependence on the value by using operators for the adaptation of the success-rate. The implementation presented here is mainly based on [9]. The success-rate is determined by the relation of successful applications $succ(\omega, t)$ and the total number of application $|P(\omega, t)|$ of the operator:

$$\sigma(\omega, t) = \frac{succ(\omega, t)}{|P(\omega, t)|} \qquad (5)$$

For calculation, only the present generation is taken into account, whereas a history-adaptation as described in the following chapter was not implemented in combination with the success-based adaptation. The formula for the determination of the new probability of application therefore is:

$$\mathscr{P}_{success}(\omega, t) = minProb + maxProb * \sigma(\omega, t) \quad (6)$$

In [9], the successes $succ(\omega, t)$ are additionally squared. This leads to a significantly better evaluation of high success rates than of lower success rates, and this is why it comes to greater deviations among the probabilities of application. as otherwise the total of the success rates would be over 100%. This was however not done here, as subsequent evaluation of a multiple mutation is very labour-intensive.

5.5 History adaptation

In the procedures explained before, only the directly previous generations were taken into consideration for adaptation. It might however be necessary to use various generations for generation of a new best individual, although interim deterioration of fitness can be accepted. Previous methods could not recognize such a longer evolution time and impede or even stop it. For this case, a so-called *history adaptation* was implemented, which, in addition to the normal adaptation also favours those operators, which served for generation of the predecessor. The different items in the history therefore also contain references on the predecessors, so that the whole genealogical tree can be traced back. It does of course not make much sense to include all generations until the beginning of evolution, as on the one hand only small deviations of the adaptation values of the different operators can be expected while on the other hand the additional effort for calculation rises exponentially to the number of generations. As the direct parental generation was, however, already considered in other adaptation procedures, the history-adaptation only applies to the last two generations, i.e. it starts with the "grandparents".

The basic idea for this adaptation was to proceed in accordance with the fitness based adaptation, but distribute the amount of adaptation to the respective genealogical tree of the operators. One half of the sum should be allocated to the operators of the latest/current individual, the other half to those of its procedure. Through recursive application with reduction by half, the influence which is getting weaker with rising length of the distance to the original individual would have been considered at the same time. However, the argument against the realization of this concept was that usually the size of the change of fitness is usually so small, that further division would only have produced insignificant values. Furthermore, the same negative effects that also appeared in the case of fitness-based adaptation because of the direct dependence on the size would have been repeated.

Furthermore, also the assumption that the size of adaptation would be smaller, the bigger the operator depth was proven insufficient. What, after all, is to be reached by adaptation is to favour evolutions which can be successful after various generations. As, however, there is no possibility to determine the exact number of generations needed for an improvement, the reduction of the amount from a certain point on cannot be explained. For this reason, the total amount of history adaptation is not determined by the change of fitness, but is regarded as a constant and adjusted by the maxHistAdapt variable in the configuration file. This amount is again divided and allocated in equal shares to the previous generations by each individual. Finally, the adaptation sum within a generation is allocated to the different operators, so that the amount received by each operator is $add^i(\omega, d)$.

$$add^i(\omega_{op},\ d) = \frac{1}{\lambda} * \frac{adapt_{max}}{d_{max} - 1} * \frac{1}{|\Omega_{op}(d)|} \tag{7}$$

i:	current individual
ω:	operator
d:	current history-low
λ:	number of descendents
op:	recombination or mutation
$\Omega_{op}(d)$:	applied operators of current generation

At an evolution of $\lambda = 100$ descendants, a history-low d = 6 and a value 1 for $adapt_{max}$ each individual counts with an adaptation total of 0.01, i.e. every previous generation has a share of 0.002. With recombination and simple mutation this amount is allocated directly to the operators, with multiple mutation these 0.002 are again allocated to the respective operators. The different sums are finally added, if an individual deteriorated, the sums it contributed are subtracted. Adaptation can also lead to a reduction of probabilities of application. Until now, implementation is only based on a combination with fitness-based adaptation. The history adaptation is simply added to the operator value, the sum being limited to interval $[0; 1]$. The new probability of application is therefore calculated in accordance with the following formula:

$$\mathscr{P}_{hist,fitness}(\omega, t) = minProb +$$
$$+ maxProb * \left(\gamma(\omega, t) + \sum_{i,d} add^i(\omega, d)\right) \tag{8}$$

6 Experimental result

In this chapter, adaptation of parameters is tested in an experiment. The problem is a standard example in the area of machine learning. They were chosen in order to be able to compare them with other learning systems as well as the original system (see [7]). The description of the examples and the data record were taken from [4].

6.1 Chess Endgame (Chess KRK)

The chess problems are challenging problems in the area of ILP. In this paper, the King-Rook-King version is investigated [3], where a given situation on the board always consists of the white king and rook and the black king. Under the precondition that black is on turn, the difference between allowed and disallowed constellations has to be learned. Those positions where white has already won and where black can strike a white figure without subsequently being struck are forbidden. The records in the database are described by the six coordinates of the three figures. A class indicates the optimal number of turns in which white wins the given situation assuming minimax-optimal play. There are 17 different classes for the minimal number of turns from 0 to 16 and the additional class draw are referred to as being positive, while all the others are treated as negative. According to this, 10.1% of the 28056 available examples are positive and 89.9% are negative.

A test run took between 35 and 45 minutes. The evolution was observed over 100 generations with a population size of 100 individuals, the number of descendants was set at $\lambda_0 = 90$. As the target predicate contains three variables, the mutation operators on variable level also came to application in this example. The number of variables was also considered for the evaluation of fitness and was in this case fixed at 1. The value for the positive and negative examples recognized correctly was stipulated to be 100. The selection process was based on the different ranks. The results of training and tests are shown in table 1. The early results without adaptation were relatively bad (83%). Correspondingly, the adaptation procedures lead to significant improvement of the results. Success-based adaptation and simple mutation produce the best results under the plus strategy, but also through the history adaptation in combination with simple mutation the results were significantly improved. This was the case during training as well as dur-

configuration			best error rate in %		midle error ratein %		proper classified examples in %		
	adaption	mult. mutation	indiv.	gener.	indiv.	gener.	all	positive	negative
+	noadapt	–	8,87±0,02	21,33±0,23	12,06±1,43	22,89±0,75	84,40±0,31	98,87±0,27	82,78±0,37
+	noadapt	√	8,87±0,02	32,78±0,51	9,06±0,34	33,46±0,34	84,40±0,31	98,87±0,27	82,78±0,37
+	fitness	–	8,16±1,44	19,21±0,93	9,26±0,41	20,37±0,17	86,03±2,98	98,87±0,27	84,60±3,29
+	fitness	√	8,14±1,46	29,52±0,89	8,94±0,84	30,92±0,60	84,71±2,74	98,94±0,22	84,22±3,05
+	history	–	5,86±2,94	*9,05±0,78*	9,01±0,47	20,06±0,42	90,19±5,47	*99,01±0,41*	89,20±6,06
+	history	√	8,14±1,46	29,52±0,89	8,04±0,84	30,92±0,60	85,71±2,74	98,94±0,22	84,22±3,05
+	**success**	–	***4,10±3,90***	**17,22±3,03**	***7,82±1,48***	**20,91±1,10**	***93,41±7,25***	**98,80±0,17**	***92,81±8,06***
+	success	√	8,87±0,02	17,39±0,68	10,60±1,11	*18,34±0,42*	84,40±0,31	98,87±0,27	82,78±0,37
,	noadapt	–	9,77±1,74	29,37±1,95	13,89±0,97	30,82±3,13	83,09±3,13	99,08±0,28	81,31±3,50
,	noadapt	√	8,87±0,02	40,19±0,52	*8,87±0,02*	41,40±0,38	84,40±0,31	98,87±0,27	82,78±0,37
,	**fitness**	–	***8,17±4,00***	**23,40±2,01**	**12,44±1,21**	**26,12±0,65**	***85,88±7,44***	**98,87±0,27**	***84,43±8,29***
,	fitness	√	8,87±0,02	31,49±0,59	9,07±0,35	32,29±0,33	84,40±0,31	98,89±0,27	82,79±0,37
,	history	–	8,87±0,02	*20,56±1,22*	12,15±1,44	*23,19±0,58*	84,40±0,31	98,87±0,27	82,78±0,37
,	history	√	8,87±0,02	28,33±1,69	9,58±0,65	29,84±0,32	84,40±0,31	98,87±0,27	82,78±0,37
,	success	–	11,48±2,14	26,14±1,82	14,88±2,57	28,28±1,64	79,38±3,88	*99,22±0,27*	77,16±4,33
,	success	√	8,87±0,02	26,30±0,42	11,03±1,47	27,56±0,67	84,40±0,31	98,87±0,27	82,79±0,37

Table 1: Training- and testresults for *chess king-rook-king*

ing application. Under the comma strategy, only fitness-based adaptation with simple mutation reached a better result than the non-adaptive procedure in combination with multiple mutation. With both strategies, the improvements were made possible by better recognition of the negative examples. As they constitute a big part of the data basis, an increase of its recognition capabilities is even more important.

7 Conclusion

In this work, automatic adaptation of the application probabilities of genetic operators was added to the *GeLog* system. We did not only intend to improve the results, but we wanted to simplify configuration of the system for the user by trying to find out the best configurations. With the help of the three adaptation procedures, it was possible to improve the results of the tested example.

Over and above this work, there are of course other possibilities of how to enlarge the *GeLog* System in order to improve its performance or the quality of its results. At the moment, there are, for example, attempts, to include building blocks, where components that probably form parts of the optimal solution are to be discovered and excluded of further variation. These extensions are area of special interest in connection with metaevolution.

References:

[1] P. J. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic System Perspective*, pages 152–161. IEEE Press, 1995.

[2] P. J. Angeline and J. B. Pollack. Evolutionary module acquisition. In W. A. D. B. Fogel, editor, *Proceedings of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society*, 1993.

[3] M. Bain. *Learning Logical Exceptions in Chess*. Dissertation, University of Strathclyde, 1994.

[4] C. Blake and C. Merz. UCI Repository of machine learning databases, Univ. of California, Irvine, Dept. of Information and Computer Sc., 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[5] C. Jacob. *Principia Evolvica – Simulierte Evolution mit Mathematica*. dpunkt.verlag, Heidelberg, Germany, Aug. 1997.

[6] G. Kókai. Gelog - a system combining genetic algorithms with inductive logic programming. In *7th Fuzzy Days LNCS, 2206*, pages 326–345, Dortmund, Germany, October 1-3 2001. Springer.

[7] G. Kókai, Z. Tóth, and S. Zvada. An experimental comparison of monostrategy and multistrategy learning methods. In *submitted GECCO'2002*, New York, 9-13 July 2002.

[8] N. Lavrač and S. Džeroski. *Inductive Logic Programming*. Ellis Horwood, 1994.

[9] J. Niehaus and W. Banzhaf. Adaption of operator probabilities in genetic programming. In *Proceedings of 4th EuroGP Conference*, pages 325–336, Como, Italien, 2001. Springer.

[10] I. Rechenberg. *Evolutionsstrategie '94, Werkstatt Bionik und Evolutionstechnik*. Band 1, Fromman-Holzboog, 1994.

[11] J. P. Rosca. Hierarchical self-organization in genetic programming. In *Proc. of the Eleventh International Conference on Machine Learning*, 1994.

[12] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[13] C. G. Shaefer. The argot system: Adaptive representation genetic optimizing technique. In *Proc. of the 2nd International Conference on Genetic Algorithms*, Hillsdale, New Jersey, 1987.

[14] W. M. Spears. Adapting crossover in evolutionary algorithms. In *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*, Cambridge, Massachusetts, 1995. MIT Press.