Toward Automating EA Configuration: the Parent Selection Stage

Ekaterina Smorodkina and Daniel Tauritz, *Member, IEEE*

***Abstract*— One of the obstacles to Evolutionary Algorithms (EAs) fulfilling their promise as easy to use general-purpose problem solvers, is the difficulty of correctly configuring them for specific problems such as to obtain satisfactory performance. Having a mechanism for automatically configuring parameters and operators of every stage of the evolutionary life-cycle would give EAs a more widely spread popularity in the non-expert community. This paper investigates automatic configuration of one of the stages of the evolutionary life-cycle, the parent selection, via a new concept of semi-autonomous parent selection, where mate selection operators are encoded and evolved as in Genetic Programming. We compare the performance of the EA with semi-autonomous parent selection to that of a manually configured EA on three common test problems to determine the "price" we pay for user-friendliness.**

I. INTRODUCTION

One of the obstacles to Evolutionary Algorithms (EAs) fulfilling their promise as easy to use general-purpose problem solvers, is the difficulty of correctly configuring them for specific problems such as to obtain satisfactory performance [5]. Proper configuration of evolutionary operators and/or parameters requires extensive knowledge about how EAs work; this feature makes EAs unappealing to non-experts. The large-scope goal of this multi-step research project is to create a general-purpose, user friendly, parameterless EA-based solver for problems that are not easily solved with traditional methods. We envision this problem solver requiring minimal EA expertise. The only input that will be needed from the user is the decision on the encoding of candidate solutions[1] and the implementation of the fitness function. It is our hope that such a solver would make EAs more widely used by the non-expert community.

Each stage of the evolutionary cycle: initialization, parent selection, reproduction, competition, and termination requires many decisions on the choice of operators and parameters. In order to create a user-friendly EA-based solver, these decisions need to be automated. We recognize that each stage of the evolutionary cycle is very important to the success of EAs and focus on one stage at a time. In this paper the focus is on the parent selection stage.

The objective of this step of the research project is to eliminate the need for choosing a particular parent selection method to be used by the algorithm and to evaluate the effectiveness of such an EA to determine the "price" of not having to configure a parent selection mechanism manually. Note that at this time we do not completely remove the choice of the parent selection method from the user's task list, but instead reduce the user's responsibility for choosing an appropriate parent selection method.

To automate the choice of the parent selection method we use the concept of self-adaptation and somewhat decentralize the control of parent selection. We use a fixed parent selection method, such as tournament selection, to pick one of the individuals for reproduction, but instead of also using a fixed scheme to choose a mate for the selected individual, each individual has an evolving mate selection function that he uses to select his preferred mate. We use self-adaptation to evolve decentralized mate selection functions and refer to this method as Self-Adapting Semi-Autonomous Parent Selection (SASAPAS). As opposed to traditional parent selection methods, where both parents are centrally selected using the same mechanism, our approach has an evolving mechanism for selecting half of the parents. However, the other half of the parents is still selected by a user-defined method. One way to completely eliminate the need to specify a parent selection method manually is to allow each individual in the population to reproduce.

There are various ways to pair up individuals using their own mate selection functions, we describe two of them. The first, which we call the *democratic* approach, involves having all individuals rate the entire population using their mate selection functions. Based on each individual's preferences, an instance of the stable roommates problem [12] can be created. An instance of the stable roommates problem consists of a set of individuals of even cardinality n. Each individual in the set ranks the remaining $n-1$ individuals in order of his preference. The objective is to find a partition of the set into $n/2$ pairs of roommates such that no two individuals who are not roommates both prefer each other to their actual roommates. We can use a solution to the stable roommates problem created from the preferences of the individuals in an EA population to pair them up. Consequently, if an individual is selected for reproduction, then his preferred mate is automatically selected as well. Unfortunately, some instances of the stable roommates problem have no solutions [12], therefore some compromising would need to be done.

The second pairing method, which we call the *dictatorial* approach, allows each individual who is selected for mating (who becomes the dictator for the moment) to pick whatever mate he wishes. For humans this may not be the most ethical approach, but it is less complex than the democratic approach and may work just as well. In this paper we focus on our proposed Self-Adaptive Semi-Autonomous Dictatorial Parent

E. Smorodkina and D. Tauritz are with the Department of Computer Science, University of Missouri-Rolla, Rolla, MO 65401 USA, email: {eas7d3, tauritzd} at umr.edu.

[1]If non-standard encoding is used, custom recombination and mutation operators will be needed.

1-4244-1340-0/07$25.00 ©2007 IEEE

Selection (SASADIPS)[2] approach and leave the democratic approach for future work.

II. Related Work

The SASADIPS approach uses self-adaptation to configure each individual's mate selection function, decentralizes the control of parent selection, and provides a method for choosing specific mate pairings. In this section we give an overview of related work on 1) self-adaptation, 2) decentralized evolutionary control, and 3) controlling mate pairing.

A. Self-Adaptation

Self-adaptation works by encoding the configuration of the evolutionary operators and/or parameters in the genome along with a trial solution to the problem being solved. A classic example of this is self-adaptation of mutation rates [15]. Although self-adaptation of mutation rates in Genetic Algorithms (GAs) and mutation distribution in Evolutionary Strategies (ESs) sometimes leads to premature convergence to suboptimal solutions [16], [17], self-adaptive EAs are known to be state-of-the-art problem solvers with a high degree of robustness [15]. Traditionally, the research on self-adaptation has focused on adapting the parameters of variation operators (crossover and mutation) [1], [11]. Recently adaptation of population size [2], [8] and selection pressure for parent selection [8] have been shown to be beneficial. In this paper we further explore self-adaptation of parent selection by adapting not the selection pressure but the actual parent selection method.

B. Decentralized evolutionary control

The evolutionary process that drives the vast majority of EAs of all types is almost exclusively centrally controlled. That is, a centrally controlled configuration of evolutionary parameters and operators is applied uniformly to all individuals in the population. The primary exception is self-adaptation of mutation rates which has a long history [15]; this is not a surprise as mutation is the only widely used unary evolutionary operator at the individual level and is, therefore, far easier to autonomously control than multi-ary or population level operators.

Decentralizing the selection of evolutionary operators used on each individual is an alternative approach to centrally controlled evolution. Stańczak [18] created a self-adaptive EA with decentralized selection of variation operators. He did this by evolving a population of individuals each containing a bit string and a vector of probabilities of using each reproduction operator defined for the problem. In his approach the genetic operators applied to the selected individuals were chosen based on the probability vector of that individual. At the end of each generation, the probability vector was updated. When a reproduction operator chosen by an individual required more than one individual as input, another individual who preferred the same reproduction operator was selected. Stańczak tested his approach on a traveling salesman problem and on the problem of scheduling time-dependent jobs on a multiprocessor system. His experiments showed that decentralizing the selection of variation operators was beneficial for his test problems in terms of speed and final solutions. In our paper we go a step further than Stańczak's approach by first decentralizing parent selection and then evolving the actual parent selection method instead of just having individuals select from a predefined set of operators.

C. Centrally controlled mate pairing

Much interesting research has been done on optimizing the pairing of individuals for reproduction using centralized methods without self-adaptation. Our way of optimizing the pairing of individuals for reproduction is different from what has been done in the past because we do not centrally control the selection of all parents participating in reproduction and evolve the mate selection method over time.

Incest prevention in EAs has been studied and was shown to have a positive effect in some cases [6], [9]. Ting and Büning in [19] showed how integrating tabu search and aspiration criteria in multi-parent crossover can improve performance of an EA. Their tabu search checked whether or not an offspring was produced from two or more parents with the same heritage (i.e., it checked for incest) and prevented an offspring from surviving if incest had occurred. Aspiration criteria would override the tabu restriction if the fitness of the offspring produced from "tabu crossover" was superior to the fitness of the best individual in the population. The effectiveness of Ting's and Büning's approach was verified on four common test functions.

Restricted mating, introduced by Booker in [4], is another way of optimizing the pairing of individuals for reproduction. In restricted mating, individuals are only allowed to be paired for reproduction if their genotypes are similar. This method encourages speciation, a process where a single species splits into two or more species [3]. Since often the fitness function has many local maxima, convergence to one local maximum may prevent the algorithm from finding a globally optimal solution. Restricted mating is advantageous for EAs as it prevents the population from converging to only one local maximum and allows finding several local maxima. Restricted mating is a fixed (non-evolving) method for pairing up individuals that is centrally controlled, whereas SASADIPS is decentralized and allows evolution of the way individuals are paired up.

III. SASADIPS Methodology

The purpose of SASADIPS is to eliminate the need to specify the mate selection operator to be used by the EA, without significantly sacrificing performance. SASADIPS is applicable to EAs that use recombination as one of the reproduction operators. In this paper we restrict ourselves to the standard two-parent recombination, but this can be extended to multi-parent recombination. As suggested by its name,

[2]SASADIPS is a subclass of SASAPAS, where our dictatorial approach is used to select the mate for the parent selected by the central mechanism. SASAPAS is an abstract method, where the details of mate selection remain unspecified.

self-adaptive semi-autonomous dictatorial parent selection is not completely autonomous. To produce an offspring the *first parent* is selected by a pre-defined central selection method, such as tournament selection, roulette selection, random selection, etc. The concept of the *first parent* is important here. At this point the first parent becomes the 'dictator' and gets to choose the other parent with whom he will produce offspring. This is the autonomous part of the algorithm, since each dictator has his own mate selection function.

To implement this approach we need a way of representing mate selection functions and a method for evolving them.

A. Mate selection function representation

Each individual in the population consists of a chromosome representing this individual's trial solution to the problem being solved and another chromosome encoding his *mate selection function* in the form of an expression tree as in Genetic Programming (GP). In SASADIPS a mate selection function takes a population of individuals as input and returns one individual from the population as output. The returned individual is the selected parent that is used for reproduction together with the first parent (the dictator).

To construct mate selection functions in the form of expression trees as in GP, it is necessary to supply a set of primitives (terminals and non-terminals) from which the functions are to be constructed [13]. The ability of an individual to select his preferred mate is limited by the primitives used in his mate selection function. Any program, and thus the optimal mate selection function, can be represented with the set of instructions of a programming language. Including all instructions of a programming language would result in an extremely large and complex search space, as this space would contain all possible computer programs. We can greatly reduce the search space by only including some pre-built mate selection components, however such a search space may not contain the optimal mate selection function. In this work we opted for the reduced search space to minimize computational overhead associated with SASADIPS and supplied the individuals with the pre-built selection primitives that we thought might make sense to use for mate selection. The sole terminal primitive was 'Population'. The non-terminal primitives used to construct mate selection functions along with their descriptions are shown in Table I.

Some of these primitives take the entire population as input, while others take two individuals. To ensure type compatibility, we used strongly typed GP. All mate selection functions were randomly initialized. An example of a randomly initialized mate selection function is shown in Figure 1.

B. Mate selection function evolution

When two individuals are selected for reproduction, variation operators (crossover followed by mutation) are applied to those individuals' candidate solution chromosomes

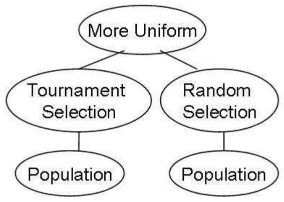

Fig. 1. An example of a randomly initialized mate selection function.

producing c new chromosomes[3]. These new chromosomes become the offspring's candidate solutions to the problem being solved. Next, the offspring need to obtain their own mate selection functions since they may become the 'dictators' in future generations. There are several possibilities for offspring to obtain a mate selection function: (1) randomly generate a new mate selection function (no inheritance), (2) inherit one of the parents' mate selection functions (single parent inheritance), and (3) recombine the parents' mate selection functions (multi-parent inheritance). Since we want to obtain the best possible mate selection function for each offspring, randomly generating a new mate selection function is not a good option. Instead, we want to use the knowledge acquired so far about mate selection functions and pass it along to the offspring.

In a typical EA, if we plot the fitness of the best individual at each generation, the slope of the plot will be steep in the early generations and will decrease in the later generations (sometimes it will be more of an s-curve). An example of such a plot is shown in Figure 2. Increasing the number of generations during which the slope of the maximum fitness plot is steep would be expected to result in a higher maximum fitness which would thus improve the performance of an EA. We will use the slope of the maximum fitness plot to decide how an offspring obtains his mate selection function.

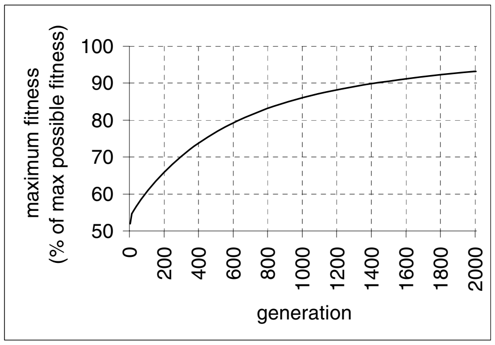

Fig. 2. Example of maximum fitness plot of an EA (taken from experiments on the ONEMAX problem conducted during this research).

[3]In our experiments, c was set to 1.

TABLE I
PRIMITIVES USED IN CONSTRUCTING MATE SELECTION FUNCTIONS.

Primitive name	Description
Tournament Selection	standard deterministic tournament selection with tournament size of 5
Roulette Selection	standard roulette selection
Random Selection	a routine that uniform randomly selects one individual
Parsimony Tournament Selection	lexicographic parsimony pressure tournament selection operator [14]
Max Fitness Individual	a routine that selects the individual with the highest fitness
Min Fitness Individual	a routine that selects the individual with the lowest fitness
Bigger Distance	given two individuals, selects the one with the bigger hamming distance from the first parent
Smaller Distance	given two individuals, selects the one with the smaller hamming distance from the first parent
More Uniform	given two individuals whose candidate solutions are in the set $\{0,1\}^N$, rewrite them as $0^{x_1}1^{x_2}0^{x_3}\cdots 1^{x_i}$ such that i is as small as possible and select the one with the smaller value of i; here exponent indicates concatenation operator, for example 1^3 is 111
Less Uniform	given two individuals whose candidate solutions are in the set $\{0,1\}^N$, rewrite them as $0^{x_1}1^{x_2}0^{x_3}\cdots 1^{x_i}$ such that i is as small as possible and select the one with the larger value of i
Bigger Fitness	given two individuals, selects the one who has a higher fitness
Smaller Fitness	given two individuals, selects the one who has a lower fitness
Subset Fitness GTOE	given a group of individuals, selects a subset of individuals whose fitness is greater than or equal to the fitness of the first parent
Subset Fitness LTOE	given a group of individuals, selects a subset of individuals whose fitness is less than or equal to the fitness of the first parent

The decision on whether an offspring should obtain his mate selection function via single or multi-parent inheritance was made based on the fitness of the offspring's solution to the problem. Recall that each individual in an EA with SASADIPS has two chromosomes: one representing his trial solution to the problem being solved and the other one representing his mate selection function. Let F be the fitness function defined on an individual's chromosome representing his trial solution to the problem being solved. We define improvement of an offspring x whose parents are p_1 and p_2 as:

$$improvement(x) = F(x) - \max\{F(p_1), F(p_2)\}. \quad (1)$$

In an attempt to increase the number of generations during which the slope of the maximum fitness plot is steep, we will keep intact those mate selection functions that result in improvements greater than or equal to the slope of the maximum fitness plot at the previous generation. This is a reasonable approach because if every offspring produced is better than both parents by the same amount (or greater) as the slope of the maximum fitness plot, then the slope will not decrease. The slope of the maximum fitness plot at the previous generation, denoted as $s(g_{i-1})$, is estimated using the difference between maximum fitness at the previous generation, g_{i-1}, and the maximum fitness at generation g_{i-K} divided by K (here K is a constant). Then if *improvement(x)* $\geq s(g_{i-1})$, the child inherits the mate selection function from the first parent. In single parent inheritance we restrict the inheritance to the mate selection function of the first parent only, because the first parent is the one responsible for a particular pairing up of individuals for reproduction. Otherwise, the mate selection function is obtained via recombination of the mate selection functions of both parents. Furthermore, there is a small chance that mate selection functions obtained via recombination will be mutated.

A note on our definition of improvement. In order to have positive improvement, the fitness of an offspring must be greater than the fitnesses of both of his parents. Alternatively, improvement can be defined as the difference between the offspring's fitness and the fitness of the first parent. The latter approach introduces a bias toward selecting the fittest individual in the population. That is, if improvement is defined in such a manner, the majority of the population will be choosing the individual with the highest fitness for mating, which may not be optimal.

Semi-autonomous parent selection requires some additional parameters compared to a standard EA. These parameters are: minimum and maximum tree depth, GP mutation rate, and the number of generations over which the slope of the maximum fitness plot is calculated (previously denoted as K). Optimizing these parameters for a specific problem will increase the performance of the algorithm. Automating the control of these additional parameters will aid in making EAs with SASADIPS easy to use problem solvers and is high on the authors' future work list.

A reader may be concerned about the speed of SASADIPS, as it requires the first parent to select his mate from the entire population. Although this is a valid concern, fitness evaluation is frequently the performance bottleneck in EAs and SASADIPS adds only a small performance overhead.

IV. Experiment Design

To determine the "price" of reducing the user's responsibility for choosing an appropriate parent selection method (i.e., only half of the parents will be chosen by it) we compare our EA with SASADIPS to a regular EA with a fixed central parent selection method. We tested our algorithm on three problems of various fitness landscape complexity: 'counting ones' (ONEMAX) problem, a 4-bit 'bounded deceptive trap' (BOUNDED D-TRAP) problem[4], and instances of the boolean satisfiability problem in conjunctive normal form (SAT). The fitness of a candidate solution to the ONEMAX problem is the number of ones in its bit string. For our experiments, we used bit strings of length 3000, so the maximum fitness of the ONEMAX problem of length 3000 is 3000. In our experiments on the BOUNDED D-TRAP problem, we used a concatenation of 250 copies of a 4-bit trap function with deceptive to optimal ratio of 0.375 [7]; our 4-bit trap function is a function of the number of ones, u, in a 4-bit string defined as:

$$f(u) = \begin{cases} 3-u & \text{if } u \leq 3 \\ 8 & \text{otherwise} \end{cases} \quad (2)$$

The fitness of a trial solution to the BOUNDED D-TRAP problem is the sum of values of each copy of the trap function. In this work we experimented with a BOUNDED D-TRAP problem containing 250 copies of a trap function. The maximum value of each copy of our trap function is 8, so the maximum fitness of our instance of the BOUNDED D-TRAP problem is 2000. Figure 3 shows an example of a trial solution to the BOUNDED D-TRAP instance and the value of each copy of the trap function in this trial solution.

$$\underbrace{0010}_{\text{trap 1}}\underbrace{1101}_{\text{trap 2}}\cdots\underbrace{1111}_{\text{trap N}}$$

Fig. 3. Example trial solution to the instance of the BOUNDED D-TRAP problem used in the experiments. According to the fitness function defined in (2), the fitness of trap 1 is 2, the fitness of trap 2 is 0, and the fitness of trap N is 8.

A standard fitness measure of a candidate solution to a SAT instance is the number of clauses satisfied by the given truth assignment. For these experiments we used five SAT instances, each containing 760 variables and 43780 clauses. These SAT instances were taken from the set of SAT instances that were used in the CSP Solver Competition 2005[5]. Consequently, the fitness of the optimal solution to the SAT instances used in the experiments is 43780.

The performance of the algorithms was measured based on the Mean Best Fitness (MBF) statistic - the average over all the runs of the best fitness found by the particular algorithm on the particular problem.

To ensure fair comparison, both algorithms used the exact same evolutionary operators and parameters, as described in Table II. The parent selection operator used to select the first parent in the EA with SASADIPS was the same as the parent selection operator in the regular EA. The only difference is that in the EA with SASADIPS the fixed parent selection operator was used only to select the first parent; the second parent was selected using the first parent's mate selection function. In the regular EA, the fixed parent selection operator was used to select both parents. We experimentally determined good parameter values for SASADIPS, which were 30% for GP-mutation rate, 2 and 5 for the minimum and maximum tree depth respectively, and 10 for K – the number of generations over which the slope of the maximum fitness plot is computed.

Both algorithms employed a population of fifty individuals for all problems used in the experiments. Fifty offspring were produced at each generation, one from every pair of parents. Preliminary experiments were conducted to determine the number of generations after which the regular EA reaches a plateau or the maximum fitness for each problem. Based on these results we determined that the maximum number of generations for the ONEMAX problem is 4000, for the BOUNDED D-TRAP problem is 2000, and for SAT instances is 300. Both algorithms were terminated when either the maximum fitness was reached or when the maximum number of generations had been evolved.

All experiments were performed using the Open BEAGLE framework[6]. Functionality was added to this framework to enable SASADIPS.

V. Experimental Results

The goal of SASADIPS is to reduce the user's responsibility for choosing an appropriate parent selection method, as one of the steps toward a user-friendly EA. In this section we review the effectiveness of the EA with SASADIPS and determine the price of leaving out mate selection method specification from the user's responsibilities.

In order to statistically validate the performance comparison of the two algorithms, we conducted 50 runs for the ONEMAX and BOUNDED D-TRAP test problems using both EAs. We used five different SAT instances with the same maximum fitness, and ran each EA six times on each SAT instance (for a total of 30 runs on SAT instances).

On two out of three test problems, ONEMAX and SAT, the MBF found by the EA with SASADIPS upon algorithm

[4]These test problems were used in [5].

[5]These SAT instances can be found at `http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/benchmarks.htm` under the link to frb40-19-cnf.tar.gz

[6]This framework and its documentation can be found at `http://beagle.gel.ulaval.ca/`

TABLE II

EVOLUTIONARY OPERATORS AND PARAMETERS USED IN THE EXPERIMENTS.

Operator	Relevant parameters
Random initialization	population size: 50 each bit is initialized to either 0 or 1 with probability 0.5
Tournament parent selection	tournament size: 5
One point crossover	crossover point is uniform randomly selected
Bit flip mutation	each bit is flipped with probability 0.01%
Tournament competition	tournament size: 5 (bit string uniqueness is enforced); elitism size: 1

termination was higher than the MBF found by the regular EA upon algorithm termination. On the BOUNDED D-TRAP problem the regular EA achieved a higher MBF than the EA with SASADIPS upon algorithm termination. The exact results of the experiments are presented in Table III. The two-tailed t-test for two samples assuming unequal variances was used to validate the differences with 5% significance level. From these results we conclude that the price we pay for not specifying parent selection method is: 1) a small computational overhead associated with SASADIPS and 2) a 2.9% lower average performance in the case of the BOUNDED D-TRAP problem.

Aside from looking at the final results, it is also interesting to examine the performance of both algorithms over time and the slope of the maximum fitness plot at each generation, as shown in Figure 4 – Figure 6.

In Figure 4 we see that both algorithms start out with the same steep slope of the maximum fitness plot, but the slope of the maximum fitness plot of the regular EA quickly decreases in the initial generations. The slope of the maximum fitness plot of the EA with SASADIPS also decreases, but not as rapidly. The steeper slope gives the EA with SASADIPS an advantage over the regular EA, resulting in a higher average performance. The situations shown in Figures 5 and 6 are different. In Figure 5 the EA with SASADIPS manages to maintain the slope of the maximum fitness plot steeper than that of the regular EA for the first 300 generations, but after that the slope decreases more rapidly than the slope of the maximum fitness plot of the regular EA. After that point, the regular EA has a steeper slope, crossing the maximum fitness plot of the EA with SASADIPS near generation 500. Finally, in Figure 6 the slope of the maximum fitness plot of the regular EA is steeper than that of the EA with SASADIPS in the first 15 generations. After that point, the EA with SASADIPS has a steeper slope of the maximum fitness curve, crossing the maximum fitness plot of the regular EA near generation 130 and maintaining higher maximum fitness after that.

Large fitness increases in the early generations imposed by SASADIPS may result in premature convergence to suboptimal solutions; however, this did not happen in our experiments (at least not as a result of the SASADIPS approach), except possibly in the BOUNDED D-TRAP problem.

Our experiments show that while our proposed methodology for EAs with SASADIPS is capable of achieving results comparable to those of a regular EA on some problems, it is not guaranteed to do so on all problems. For instance, our method performed worse on the BOUNDED D-TRAP problem. In the worst case the performance of the EA with SASADIPS was 2.9% lower than that of the regular EA, while in the best case it was higher by 1.9%. Experiments on additional common test functions are needed to determine a more accurate and statistically significant bound on the performance gap of the EA with SASADIPS.

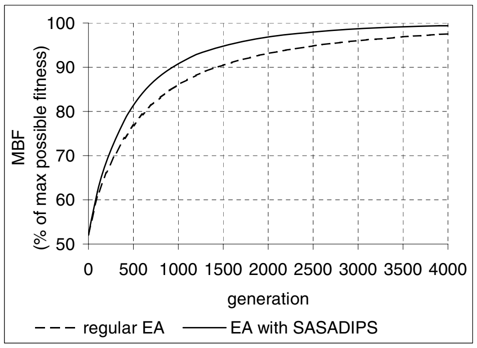

Fig. 4. Average maximum fitness at each generation for the ONEMAX problem.

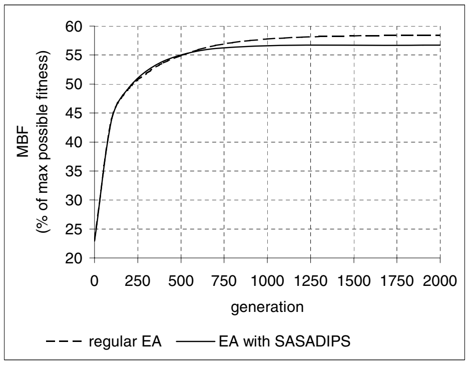

Fig. 5. Average maximum fitness at each generation for the BOUNDED D-TRAP problem.

TABLE III

AVERAGE PERFORMANCE AND STANDARD DEVIATION (IN TERMS OF FITNESS AND PERCENTAGE OF THE MAXIMUM FITNESS) OF THE TWO ALGORITHMS ON THE THREE TEST PROBLEMS

	Regular EA		EA with SASADIPS	
	avg	st.dev.	avg	st.dev.
ONEMAX	2925.74 (97.52%)	6.83 (0.22%)	2982.22 (99.41%)	4.75 (0.15%)
BOUNDED D-TRAP	1168.36 (58.42%)	37.24 (1.86%)	1134.80 (56.74%)	34.31 (1.72%)
SAT	43720.77 (99.86%)	9.48 (0.02%)	43737.87 (99.90%)	9.39 (0.02%)

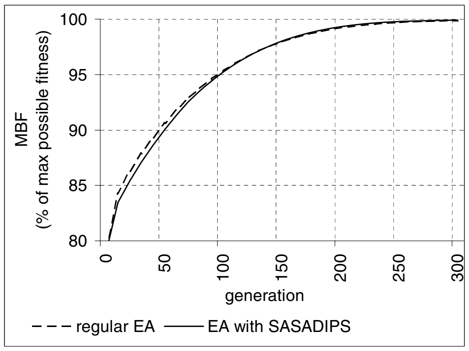

Fig. 6. Average maximum fitness at each generation for SAT instances.

VI. DISCUSSION

In the EA with SASADIPS the evolution of high quality mate selection functions is key to minimizing the performance inferiority of the algorithm, as compared to manually configured EAs. The goal is not to find a problem independent mate selection function, but to optimize parent selection for a specific problem and specific individual at a specific stage of the evolutionary process. In this section we will look at the final mate selection functions of the best individuals from our experiments with the ONEMAX problem and analyze some general trends.

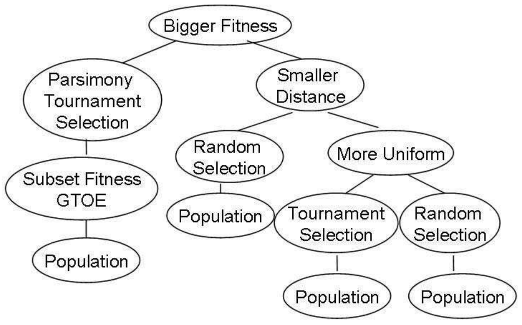

Fig. 7. An example of a mate selection function at the end of the evolution (taken from experiments on the ONEMAX problem).

An example of a highly evolved mate selection function is shown in Figure 7. This example is taken from the ONEMAX experiments. This evolved mate selection function has a high selection pressure, which is enforced by 'Bigger Fitness' and 'Subset Fitness GTOE' primitives. Additionally, this mate selection function has a feature of Assortative Mating Genetic Algorithms [10], a mating strategy where individuals with similar genotypes are more likely to be paired up for recombination. This feature is indicated by the 'Smaller Distance' primitive.

The primitives at the higher level (closer to the root node) of a mate selection function have the most influence on the final selection. The most frequently found primitives (see Table I for a description of each primitive used in the experiments) at the roots of the mate selection functions of the fittest individuals upon the termination of the EA with SASADIPS on the ONEMAX problem were 'More Uniform' (16 out of 50), 'Smaller Distance' (12 out of 50), and 'Bigger Fitness' (11 out of 50).

VII. CONCLUSIONS AND FUTURE WORK

The multitude of parameters and operators needed to properly configure EAs prevents them from gaining wide spread popularity in the non-expert community. Automating the choice of operators and parameters at each stage of the evolutionary cycle without significantly sacrificing performance would make EAs a user friendly, general purpose problem solving tool. In this paper we focused on automating the parent selection stage, leaving the other stages to future work. We introduced semi-autonomous parent selection: a decentralized approach to parent selection in EAs, where each individual has his own mate selection function. Our method uses self-adaptation to evolve mate selection functions for all individuals in the population. We described two approaches to semi-autonomous parent selection on the conceptual level, democratic and dictatorial, focusing on the dictatorial approach. We experimentally compared our EA with SASADIPS to a regular EA on three commonly used test problems: ONEMAX, BOUNDED D-TRAP, and SAT. The results of our experiments showed that in the worst case our EA with SASADIPS achieves performance lower than that of the regular EA by 2.9% on these three problems; this is the price we pay for not configuring mate selection method manually. In the case of the ONEMAX problem, where the fitness landscape is linear, our EA with SASADIPS achieved a performance higher than that of the regular EA by 1.9%, so the price we pay is not always negative.

Experimental evaluation of the EA with SASADIPS on additional commonly used test functions will help determine a more accurate bound on the performance gap of the EA with SASADIPS as compared to the regular EA.

SASADIPS can be easily extended for multi-parent recombination when the number of parents is more than two. One way to do this is to select the first parent by a fixed centrally controlled selection method and then use the first parent's mate selection function k times (with or without replacement), when $k + 1$ parents are needed for recombination. Additionally the multi-parent inheritance of the mate selection function would need to be adjusted accordingly.

In the future, we need to automate the control of additional parameters required for SASADIPS to completely eliminate the need of user's input on the mate selection method. Additional future steps in this research are comparison of the dictatorial parent selection to the democratic approach, investigation of the full automation of the parent selection process, and automation of other stages of the evolutionary life-cycle.

REFERENCES

[1] T. Bäck. *Evolutionary Computation 2*, chapter 21, pages 188–211. Institue of Physics Publishing, 2000.

[2] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart. An empirical study on GAs "without parameters". In *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 315–324, 2000.

[3] D. Beasley, D. Bull, and R. Martin. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181, 1993.

[4] L.B. Booker. Improving the performance of genetic algorithms in classifier systems. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 80–92, 1985.

[5] J. Clune, S. Goings, B. Punch, and E. Goodman. Investigations in meta-GAs: panaceas or pipe dreams? In *GECCO '05: Proceedings of the 2005 Workshops on Genetic and Evolutionary Computation*, pages 235–241, 2005.

[6] R. Craighurst and W. N. Martin. Enhancing GA performance through crossover prohibitions based on ancestry. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 130–135, 1995.

[7] K. Deb and D. Goldberg. Analyzing deception in trap functions. In *Proceedings of the Second Workshop on Foundations of Genetic Algorithms (FOGA 2)*, pages 93–108, 1992.

[8] A. E. Eiben, M. C. Schut, and A. R. de Wilde. Is self-adaptation of selection pressure and population size possible? - A case study. In *PPSN 2006: The 9th International Conference on Parallel Problem Solving from Nature*, pages 900–909, 2006.

[9] C. Fernandes and A. Rosa. A study of non-random mating and varying population size in genetic algorithms using a royal road function. In *CEC 2001: The 2001 Congress on Evolutionary Computation*, volume 1, pages 60–66, 2001.

[10] Carlos Fernandes, Rui Tavares, Cristian Munteanu, and Agostinho Rosa. Using assortative mating in genetic algorithms for vector quantization problems. In *SAC '01: Proceedings of the 2001 ACM Symposium on Applied Computing*, pages 361–365, 2001.

[11] J. Gomez. Self adaptation of operator rates in evolutionary algorithms. In *GECCO 2004: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1162–1173, June 2004.

[12] R.W. Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6:577–595, 1985.

[13] J.R. Koza. *Genetic Programming II: Background on Genetic Algorithms, LISP, and Genetic Programming*. MIT Press, Cambridge, MA, USA, 1994.

[14] S. Luke and L. Panait. Lexicographic parsimony pressure. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 829–836, 2002.

[15] S. Meyer-Nieberg and H.-G. Beyer. Self-Adaptation in Evolutionary Algorithms. In Fernando G. Lobo, Claudio F. Lima, and Zbigniew Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 47–76. Springer, 2007.

[16] G. Rudolph. Self-adaptation and global convergence: a counter-example. In *CEC'99: Proceedings of the 1999 Congress on Evolutionary Computation*, volume 1, pages 646–651, 1999.

[17] G. Rudolph. Self-adaptive mutations may lead to premature convergence. *IEEE Transactions on Evolutionary Computation*, 5(4):410–414, August 2001.

[18] J. Stańczak. Biologically inspired methods for control of evolutionary algorithms. *Control and Cybernetics*, 32(2):411–434, 2003.

[19] C.-K. Ting and H. K. Büning. A mating strategy for multi-parent genetic algorithms by integrating tabu search. In *CEC 2003: The 2003 Congress on Evolutionary Computation*, volume 2, pages 1259–1266, 2003.