Evolving Fitness Functions for Mating Selection

Penousal Machado and António Leitão

CISUC, Department of Informatics Engineering, University of Coimbra
Polo II of the University of Coimbra, 3030 Coimbra, Portugal
machado@dei.uc.pt, apleitao@student.dei.uc.pt

Abstract. The tailoring of an evolutionary algorithm to a specific problem is typically a time-consuming and complex process. Over the years, several approaches have been proposed for the automatic adaptation of parameters and components of evolutionary algorithms. We focus on the evolution of mating selection fitness functions and use as case study the Circle Packing in Squares problem. Each individual encodes a potential solution for the circle packing problem and a fitness function, which is used to assess the suitability of its potential mating partners. The experimental results show that by evolving mating selection functions it is possible to surpass the results attained with hardcoded fitness functions. Moreover, they also indicate that genetic programming was able to discover mating selection functions that: use the information regarding potential mates in novel and unforeseen ways; outperform the class of mating functions considered by the authors.

1 Introduction

The choice of an adequate representation, genetic operators and parameters is critical for the performance of an Evolutionary Algorithm (EA). To attain competitive results it is usually necessary to choose or develop problem specific representations, operators and fitness functions, and fine-tune parameters. This can be a complex and time-consuming process. The use of self-adaptive EAs – e.g., EAs that automatically adjust their parameters or components in order to improve the performance in a specific problem or set of problems – has the potential to overcome this problem.

The fitness function has a deep effect on the selection and survival process. In optimization problems the choice of a fitness function may appear to be straightforward. Nevertheless, in the context of sexual reproduction, individuals may have an advantage in choosing mating partners in accordance to criteria other than fitness. For instance, an individual that chooses its mates in accordance to their genetic compatibility may gain an evolutionary advantage.

We focus on the evolution of fitness functions for mating selection, using as test problem the Circle Packing in Squares (CPS). Each individual is composed by two chromosomes: (i) a candidate solution for the CPS problem encoded as a vector (ii) a mating fitness function which is used to evaluate the potential mates of the individual. We test two different approaches for the representation

S. Silva et al. (Eds.): EuroGP 2011, LNCS 6621, pp. 227–238, 2011.

of the evolving mating fitness functions. In the first mating fitness functions are weighted sums, and the weights are under evolutionary pressure. In the second, the mating fitness function is conventional Genetic Programming (GP) tree [1].

We begin by making a brief overview of the state of the art on adaptive EAs focusing on their classification. Next, we introduce the CPS problem and its equivalence with the point scattering problem. In the fourth section, we describe our approach to the evolution of mating fitness functions. The experimental setup is described in the fifth section, while the experimental results are presented and analyzed in the sixth. Finally, we present overall conclusions and future research.

2 State of the Art

EA researchers often face the challenge of designing adequate EAs for specific problems. To automate the task of finding a good design, and set of parameters, one may adapt these components and variables along the evolutionary process. Over the years, several approaches have been proposed for the evolution of EAs. Angeline [2] has presented a classification of these approaches, which was later expanded by Hinterding [3]. He proposes two axis of classification: *adaptation type* and on the *adaptation level*.

Adaptation Type. Evolving EAs may be *static* or *dynamic*. In *static adaptation* the tuning of parameters is made between runs. This is usually accomplished by an external program (or human) that performs a set of test runs and attempts to find a link between settings and performance. The work of De Jong [4] constitutes an example of such an approach. He focused on single-point crossover and bit mutation probabilities in GA, performing a wide number of tests, and empirically establishing recommended mutation and crossover probabilities for several problems.

Dynamic adaptation relies on mechanisms that modify EA parameters during the evolutionary run without external control. They may use, or not, some form of feedback from the evolutionary process to control adaptation. This approach can be further divided into *deterministic*, *adaptive* and *self-adaptive*.

Deterministic dynamic adaptation uses no feedback from the EA. Instead, it uses a set of deterministic rules that alter the strategy parameters in a predetermined way when a given event takes place. For instance, Hinterding [3] describes an approach where the mutation rate of a Genetic Algorithm (GA) decreases as the number of generations increases.

Adaptive dynamic adaptation approaches use feedback information regarding the progress of the algorithm to determine the direction and magnitude of the changes made to strategy parameters. E.g., Bean and Hadj-Alouane [5] adjust a penalty component for constraints violation on a GA by periodically stopping the run and evaluating the feasibility of the best individuals of the last generations.

Self-adaptive dynamic adaptation relies on evolution to modify EA parameters. The control parameters are encoded into each individual in the population and undergo recombination and mutation as part of the genotype. Eiben et al.

[6] studied the self-adaptation of the tournament size. Each genotype has an extra parameters that encodes the tournament size. On each selection step tournament size is determined by a voting system. Spears [7] proposes the use of an extra gene to encode the type of crossover, two-point or uniform, that should be use when recombining the genetic code of the corresponding individual.

Adaptation Levels. The level within the EA where adaptation takes place is another axis of classification. Adaptation can be defined in four levels: *environment*, *population*, *individual* and *component*.

Environment level adaptation takes place when the response of the environment to the individual is changed. Mostly this affects, somehow, the fitness function, either by changing the penalties or weights within it or by changing the fitness of an individual in response to niching. In [8], Angeline argues that competitive fitness functions have many advantages over typical approaches, specially when there is little knowledge about the environment.

Population level adaptation consists of adapting parameters that are shared and used by the entire population. The work of Fogarty [9] where the mutation rate of the entire population evolves through time is an example of such an approach.

Individual level adaptation evolves strategy parameters that affect particular individuals. Often these components are encoded in each individual, e.g., Braught [10] uses a self-adaptive scheme where an extra gene that codifies the mutation rate of each individual.

Component level adaptation acts on strategy parameters that affect specific components or genes of an individual. Fogel's study [11] on the self-adaptation of finite state machines is an example of this approach, exploring the association of a mutability parameter to each component of the finite state machines.

2.1 Other Approaches

Grefenstette [12] suggests conducting a search over a parameterized space of GAs in order to find efficient GAs for a function optimization task (this work was later applied to GP as well). The search uses two levels: a *macro level* representing a population of EAs that undergo evolution; a *micro level* where each represented EA acts on a population of candidate solutions for a given problem. The use of linear variants of GP to evolve EAs has also been an active field of research, consider for instance the work of Oltean on Multi Expression Programming (MEP) [13] and Linear Genetic Programming (LGP) [14]. Spector and Robinsons [15] explore the use of Push – a stack-based programming language able to represent GP systems – to evolve EAs, including the co-evolution of mate selection.

2.2 Evolving the Fitness Function

Evolving the fitness function implies changes in the way the environment responds when evaluating individuals of a population. Darwen and Yao [16] have synthesized some previous work done on *fitness sharing* to tackle a multi optima

problem. The fitness sharing technique modifies the search landscape during the evolutionary run by reducing the payoff in niches that become overcrowded. Hillis [17] explores the co-evolution of two populations – candidate solutions and test cases – for the evolution of sorting networks. The fitness of the candidate solutions is determined accordingly to their ability to sort the test cases of the second population; the fitness of a test case is determined by the amount of sorting errors it induces.

3 Circle Packing in Squares

We focus on CPS, a surprisingly challenging geometrical optimization problem relevant for real life storage and transportation issues. It consists in finding a spacial arrangement for a given number circles of unitary radius that minimizes the area of the minimal square that contains them. The circles may not overlap. In Fig. 1 we present the optimal solutions for the packing of 7, 15, and 24 circles into a square. The optimal solutions for sets from 2 to 24 circles are known.

The CPS problem is closely related to that of scattering n points in a unit square such that the minimal distance between any of them is maximized [18]. In order to transform from this model to that of a square containing fixed size circles one must apply the transformation presented in Fig. 2 where r is the radius of the circles, d is the greatest minimal distance between points and S is the side of the resulting square.

We conducted a wide set of experiments using GA and GP approaches [19], which show that, in an evolutionary context, it is advantageous to see the CPS problem as a scattering problem: it avoids the overheads caused by invalid candidate solutions (e.g. where circles overlap) leading to better results. As such, we will adopt this representation in the experiments described in this paper.

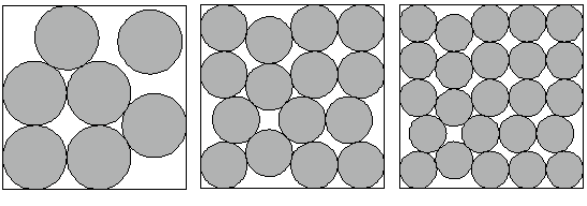

Fig. 1. Optimal solutions for 7, 15, and 24 circle packing in a square

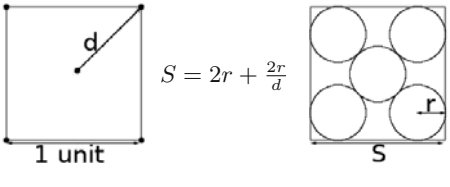

Fig. 2. Calculation of the side of the square for the 5 circles optimal solution

4 Mating Selection

Without loss of generality, considering tournament based selection, the mating selection procedure can be described as follows: (i) a parent is selected from the population using tournament selection based on fitness; (ii) a set of t mating candidates is randomly selected from the population; (iii) the candidate that, according to the parent, is fittest for mating purposes is selected for sexual reproduction with this parent; The process is repeated until sufficient offsprings are generated.

4.1 Mating Selection in the Circle Packing in Squares Problem

In order to better understand the means by which the mating fitness can be evolved, there's a characteristic of the CPS problem that should be mentioned. Usually, optimal solutions can be decomposed in subsets that are in turn optimal (or near optimal) solutions for a different instance of the problem. More formally, the optimal solution for the packing of n circles in a square may also encode optimal solutions for the packing of $n-i$ circles for i in $[1, n-2]$. As Fig. 3 shows the optimal representation for 9 circles also encodes the optimal solution for 5 circles, 4 circles and 2 circles. One may be inclined to think a good performance in a $n-i$ problem promotes a good performance in the n packing problem. However, this is not always the case. For instance, to pack 7 circles optimally one must use an inefficient packing of 4 and 6 circles, and an optimal packing of 4 (see Fig. 1) . In other words, the optimal packing of 7 circles can be created by adding 3 circles to the optimal packing of 4 circles; adding circles to the optimal packing of 5 or 6 circles leads to an inefficient packing of 7 circles. Thus, for evolving solutions for the 7 circle instance it makes sense to value the ability of the candidates to efficiently pack 4 circles and penalize candidates that pack 5 or 6 circles optimally. These observations made us consider the following hypotheses: the performance of an individual on a $n-i$ packing problem may provide information regarding its performance on the target n packing problem; this information may be valuable for mating selection purposes.

As such, when evaluating a mating candidate, the individual has access to information regarding the fitness achieved by the candidate in the packing of $n-i$ circles for all i in $[1, n-2]$. Our expectation is to evolve mating fitness functions that use this information wisely, leading to a better performance in the n circles instance.

To determine the fitness of an individual packing n circles in all instances of the CPS problem of lower dimensionality one would need to consider a total of $C_2^n + C_3^n + ... + C_{n-2}^n + C_{n-1}^n$ packings of circles.

This becomes computationally intensive. In order to reduce computation and increase the speed of the tests executed we perform the following simplification: For determining the fitness an individual in the packing of k circles (with $k < n$) we only consider the first k first circles represented in the chromosome, instead of C_k^n possible combinations of k circles. With this simplification the computational overhead becomes minimal and negligible.

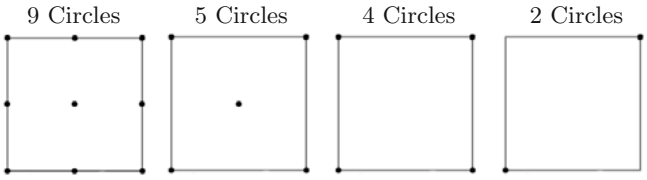

Fig. 3. Composition of optimal solutions by close to optimal subsets

```
evaluate_mating_candidates(mating_candidates,parent) {
  for i = 1 to #(mating_candidates) {
   candidate_mating_fitnesses_i ←eval(mating_candidates_i,parent)
  }
  return candidate_mating_fitnesses
}
```

Fig. 4. Evaluation of the mating candidates [20]

Each parent picks its mate among the candidates and so is bound to evaluate them. In order to do this each individual encodes a mating fitness function that it uses to evaluate its mating candidates. The *evaluate_mating_candidates* step, described in Fig. 4, has been designed to receive the set of mating candidates and the parent which is evaluating them, returning the candidates' mating fitnesses. The *eval* step uses the mating fitness function encoded in the parent to evaluate each candidate.

Each individual is composed of two chromosomes. The first encodes a candidate solution for the CPS problem through a vector of n x, y coordinates. The second encodes a mating fitness function. We explore two different approaches for the encoding of the mating fitness function: one using GA and the other using GP.

In the GA approach the mating fitness functions are weighted sums and the chromosome encodes a set of weights. The $MatingFitness$ of candidate mc according to parent p is given by the following formula:

$$MatingFitness^{mc} = \sum_{k=2}^{n} w_k^p F_k^{mc} \ ,$$

where w_k^p is the weight given by parent p to the fitness of the candidates in the packing of k circles and F_k^{mc} is the fitness of the candidate mc in the packing of k circles. In the GP approach the chromosome encodes a standard GP tree that defines the mating fitness function. The terminal set includes the variables F_k^{mc}, which allows the individual to use this information to access the mating candidates.

5 Experimental Setup

In order to understand the effect of mating selection and of evolving of the mating fitness function we need to have a basis for comparison. We performed several experiments using a standard GA approach to solve the CPS problem [19]. We considered a wide set of parameters for tournament size, mutation rate and type of mutation, and performed static adaptation over this set of parameters. The parameters that resulted in best overall performance are presented in table 1.

Table 1. Parameter set for the standard GA approach

Representation	$x_1, y_1, ..., x_n, y_n$
Initialization	x_n, y_n in $[0, 1]$
Parents Selection	$Tournament\ size = 5$
Crossover	$1 - point$
Crossover probability	90%
Mutation	Gaussian mutation; $mean = 0.0$; $stdev = 0.08$
Mutation probability	2%
Elitism	1 individual

Table 2. Additional settings used in GA-based mating selection evolution

Representation	$x_1, y_1, ..., x_n, y_n, w_2, ..., w_n$
Initialization	x_n, y_n in $[0, 1]$; $w_{s(n)}$ in $[-1, 1]$
Crossover	$1 - point$
Crossover probability	90%
Mating Candidates Selection	$Tournament\ size = 5$
Mutation	Gaussian mutation; $mean = 0.0$; $stdev = 0.08$
Mutation probability	5%
Evolving Mating Fitness	$MatingFitness^{mc} = \sum_{k=2}^{n} w_k^p F_k^{mc}$

To evaluate the effects of evolving the mating fitness function we conducted experiments using the GA and GP based approaches for mating selection.

In GA-based mating fitness evolution the mating fitness function has been designed by the authors to be a weighted sum of the fitness values of the candidate in smaller instance of the CPS problem. An additional chromosome is added to the standard GA representation to encode the weights. The genetic operators are applied independently to each chromosome. For the chromosome encoding the candidate solutions we use the previously established set of parameters presented in table 1. For the chromosome encoding the mating fitness function we use the parameters presented in table 2. As previously this set of parameters was established empirically using static adaptation (See section 2).

Likewise, in GP-based mating selection evolution we have two chromosomes: a linear chromosome encoding the candidate solution; a mating fitness function

Table 3. Additional settings used in GP-based mating selection evolution

Representation	$x_1, y_1, ..., x_n, y_n, GPtree$
GP terminals	$F^{mc}_{s(2)}, .., F^{mc}_{s(n)}, 0, 1, 2$
GP functions	$+, -, *, /$
GA Initialization	x_n, y_n in $[0, 1]$
GP Initialization	half-half
Mating Candidates Selection	$Tournament\ size = 5$;
Evolving Fitness Function	Output from the execution of the tree

represented by a GP tree. For the candidate solution chromosome we use the parameters presented in table 1, for the GP chromosome the parameters and operators are those employed by Koza [1]. The additional settings for this approach can be found in table 3.

The fitness of an individual in the CPS problem is used for parent selection in all approaches, and it is given by the size of the minimum square that contains the circles (see Fig. 2). As such this is a minimization problem.

6 Experimental Results

For a given number of n circles, a given set of parameters and a chosen approach, 30 runs are executed. In each run 100 individuals evolve along 5000 generations and the fitness in the CPS problem of the best individual and population mean is saved at each generation. We performed tests for all n in $[2, 24]$. Table 4 shows the results obtained in this study. Column N indicates the instance of the CPS problem being tackled, while column *optimal* presents the optimal values for that instance. Column *Standard* shows the results obtained by the standard GA approach, i.e. both parents chosen by tournament selection based on CPS fitness. Column *Random* present the results attained by choosing the first parent using tournament selection based on CPS fitness and its mating partner randomly from the current population. Column *GA* presents the results attained by using GA-based mating fitness selection – i.e. the first parent is chosen using using tournament selection based on CPS fitness and its mating partner is selected by tournament selection based on the mating fitness of the candidates accordingly to the mating fitness function encoded by the first parent. Likewise, column *GP* presents the results achieved by GP-based mating fitness selection.

For each approach, the *best* column presents the CPS fitness of the best individual found in the 30 runs, while the *avg* column presents the average fitness of the best individuals over the 30 runs. Underlined values indicate the result is better than the one attained using the Standard approach. Bold values indicate that the value is better and that a statistically significant difference exists (obviously, this only applies to average values).

A comparison of the results attained by the Standard and Random mating selection approaches reveals that for small instances of the CPS it is often advantageous to select mates randomly. It is important to notice that this is not

Table 4. Comparison of the results attained by the different approaches. Results are averages of 30 runs. Lower values indicate better performance. Underlined values signal results better than the ones attained using the Standard approach. Bold values indicate that a statistically significant difference exists. Statistically significance determined through the Wilcoxon–Mann–Whitney test. Confidence level of 0.95.

		Static Mating Selection Function				Evolved Mating Selection Function			
		Standard		Random		GA		GP	
N	optimal	best	avg	best	avg	best	avg	best	avg
2	3.4142	3.4142	3.4142	3.4142	3.4142	3.4142	3.4142	3.4142	3.4142
3	3.9319	3.9319	3.9320	3.9319	3.9320	3.9319	3.9319	3.9319	3.9319
4	4.0000	4.0000	4.0266	4.0000	4.0001	4.0000	4.0255	4.0000	4.0001
5	4.8284	4.8288	5.0056	4.8287	4.9911	4.8285	**4.9250**	4.8286	**4.9475**
6	5.3282	5.3296	5.3669	5.3299	5.3674	5.3306	5.3685	5.3303	5.3804
7	5.7321	5.7426	5.8227	5.7379	5.8081	5.7353	5.8296	5.7348	5.8098
8	5.8637	5.8665	6.0212	5.8714	5.9615	5.8693	5.9913	5.8643	5.9898
9	6.0000	6.0072	6.5184	6.0086	6.4907	6.0042	6.5401	6.0018	6.5154
10	6.7474	6.7564	6.8936	6.7804	6.8854	6.7642	6.9110	6.7581	**6.8536**
11	7.0225	7.0323	7.1619	7.0822	7.1764	7.0600	7.2232	7.0418	7.1564
12	7.1450	7.1540	7.3966	7.2416	7.3565	7.1966	7.4809	7.1682	7.3438
13	7.4630	7.4977	7.8088	7.6036	7.8167	7.5663	7.8355	7.4816	**7.7147**
14	7.7305	7.8059	8.0705	7.8859	8.0950	7.9190	8.1509	7.8498	**8.0048**
15	7.8637	8.0332	8.3324	8.1102	8.4173	8.0296	8.4345	7.9677	**8.2581**
16	8.0000	8.4015	8.7014	8.4542	8.8632	8.3030	8.8153	8.3980	**8.6012**
17	8.5327	8.6688	8.8765	9.0022	9.2345	8.7143	9.0836	8.7065	8.8665
18	8.6564	8.8566	9.0996	9.1902	9.4966	8.9189	9.2724	8.8582	9.0984
19	8.9075	9.1482	9.4442	9.4789	9.9422	9.2049	9.6036	9.0178	**9.3511**
20	8.9781	9.3889	9.7212	9.9433	10.2839	9.2951	9.7641	9.1795	**9.6030**
21	9.3580	9.6980	9.9788	10.2998	10.7402	9.7305	10.1307	9.6730	9.9425
22	9.4638	9.9210	10.2610	10.6887	11.0512	9.9546	10.3705	9.9969	10.2693
23	9.7274	10.0625	10.5201	10.9262	11.5476	10.0631	10.6498	10.0943	10.5892
24	9.8637	10.3198	10.7725	11.2717	11.8382	10.5232	10.8163	10.4678	10.8034

equivalent to performing random search, the first parent is chosen by tournament selection using CPS fitness, only its mate is selected randomly. By choosing mating partners randomly one lowers the selection pressure. Furthermore, it may promote population diversity. The combination of these factors may avoid premature convergence, explaining the best performance of the Random mating selection approach in small instances. The analysis of the evolution of CPS fitness throughout the evolutionary runs supports this hypothesis. For larger instances the lack of selection pressure penalizes the Random mating selection approaches.

The performance of the GA-based mating selection approach is disappointing. Although it was able to find better solutions than the standard approach for six instances of the problem, the average of the best solutions is only better than the one attained by the Standard approach in four instances, and the difference is only statistically significant in two of them. This suggests that the overhead inflicted on the evolution process by the necessity of evolving adequate weights

is not rewarding. The approach should be able to reproduce the original fitness function, and by doing so it would be expectable for it to achieve closer results to those of the Standard approach. An analysis of the runs indicates that the approach is unstable, in the sense that some runs produce significantly better results than others, which partially explains the worst average fitness. Despite this, comparing the results with the ones attained by the Random mating selection approach, revels that better results are attained for large instances of the CPS problem. This indicates that the individuals are able to evolve mating fitness functions that take advantage of the information provided to them, although the reward is not enough to compete with the Standard approach.

Considering these results, the performance of the GP-based mating selection approach is surprising. It attains better averages than the Standard approach in eighteen instances and the differences are statistically significant for thirteen of them. Moreover, the best solutions surpass those found using the Standard approach in ten of the instances. For small instances of the CPS problem, where Random mating performed better than the Standard approach, GP-based mating selection is also competitive, the averages are comparable to those attained by Random mating, and the best individuals found tend to be better. Overall, these results indicate that: the GP-based approach is able to evolve mating selection functions that take advantage of the information regarding the performance of the candidate mates in smaller instances of the CPS problem in a useful way, leading to better overall performance in spite of the overheads caused by the need to evolve the mating selection functions.

The results attained by GP-based mating selection contrast with the ones of the GA-based approach. This was a totally unexpected result, that demands further analysis.

In GA-based mating, the mating fitness function was designed by the system developers in a way that appeared adequate to the problem at hand, and that is focused on the selection of the mating candidate that seem most fit for the role. Moreover, the evolution process only evolves the weights used in the mating fitness function. In GP-based mating selection the system must evolve the entire mating fitness function. Thus, the search space of GP-based contains all the mating fitness functions that can be evolved by the GA-based and a vast number of functions of a different kind. It is, therefore, significantly larger. By these reasons, the task of the GP-based approach appeared to be significantly harder than the one of the GA-based approach. We find two, non-exclusive, hypotheses for the better performance of the GP-based approach: (i) The "ideal" mating fitness function varies during the course of the run – e.g., the best way to choose mating partners in the beginning of the run may be useless in later generations – and GP is able to adapt faster to these changing requirements than GA. (ii) GP was able to evolve fitness selection functions that outperform the ones constructible using the weighted sum template designed by us, and based on our knowledge on the regularities and irregularities of the CPS problem.

Although, the experimental results are insufficient to draw definitive conclusions, the analysis of the evolution of the weights during individual runs appears to

support the first hypothesis. The analysis of the fitness mating functions evolved by the GP-based approach is extremely difficult and the bottom line is that we are unable to understand exactly how mating candidates are being evaluated. Nevertheless, it is clear that what the fitness functions evolved by this approach are doing is radically different from a weighted sum, which supports the second hypothesis. Thus, the analysis points towards a combination of both factors.

Our inability to explain how the GP-based approach is determining mating fitness makes it impossible to fully understand the results. It does, however, also indicate that the GP-based approach is using fitness mating functions that we would be unlikely to design by hand, and surpassing the performance attained by the ones we designed.

7 Conclusions

We focus on the self-adaptive evolution of mating fitness functions using the CPS problem for test purposes. We perform an overview of previous work in the area, and introduce the CPS problem, describing its regularities and irregularities, and proposing ways to explore them for the purpose of developing mating fitness functions that improve the performance of the EA. Two approaches, based on GA and GP, are presented for the evolution of mating fitness functions.

The experimental results in the packing of two to 24 circles are presented and discussed. These results reveal that the GA-based mating fitness evolution approach – which evolves parameters for a mating fitness function designed by the authors based on their knowledge of the CPS problem – is unable to surpass the performance of a conventional approach. Contrastingly, the GP-based mating fitness evolution approach, which explores a significantly larger search space of mating fitness functions, is able to outperform all of the considered approaches. Although the nature of the mating fitness evolved by the GP-based approach was not fully understood, it is safe to say that they are very different from the ones designed by the authors. This indicates that the GP-based approach is not only able to evolve fitness mating functions that outperform hand coded ones, but also to discover mating fitness function designs that would be unlikely to be created by hand.

Acknowledgment

This work has been partially supported by the project PTDC/EIA-EIA/102212/-2008, High-Performance Computing over the Large-Scale Internet.

References

1. Koza, J.R., Poli, R.: Genetic programming. In: Search Methodologies, pp. 127–164. Springer, Heidelberg (2005)
2. Angeline, P.J.: Adaptive and self-adaptive evolutionary computations. In: Computational Intelligence: A Dynamic Systems Perspective, pp. 152–163. IEEE Press, Los Alamitos (1995)

3. Hinterding, R., Michalewicz, Z., Eiben, A.E.: Adaptation in evolutionary computation: A survey. In: Proc. of the 4th International Conference on Evolutionary Computation, pp. 65–69 (1997)
4. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor, MI, USA (1975)
5. Bean, J., Hadj-Alouane, A.: A dual genetic algorithm for bounded integer programs. Technical Report 92-53, University of Michigan (1993)
6. Eiben, A., Schut, M., de Wilde, A.: Boosting genetic algorithms with self-adaptive selection. In: IEEE Congress on Evolutionary Computation, pp. 477–482 (2006)
7. Spears, W.M.: Adapting crossover in a genetic algorithm. In: Proc. of 4th Annual Conference on Evolutionary Programming, pp. 367–384 (1995)
8. Angeline, P.J., Pollack, J.B.: Competitive environments evolve better solutions for complex tasks. In: Proc. 5th International Conference on GAs, pp. 264–270 (1994)
9. Fogarty, T.C.: Varying the probability of mutation in the genetic algorithm. In: Proc. of the 3rd International Conference on Genetic Algorithms, pp. 104–109 (1989)
10. Braught, G.: Evolving evolvability: Evolving both representations and operators. In: Adaptive and Natural Computing Algorithms, pp. 185–188. Springer, Heidelberg (2005)
11. Fogel, L., Angeline, P., Fogel, D.: An evolutionary programming approach to self-adaptation on finite state machines. In: Evolutionary Programming, pp. 355–365 (1995)
12. Grefenstette, J.: Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 16(1), 122–128 (1986)
13. Oltean, M.: Evolving evolutionary algorithms with patterns. Soft Computing - A Fusion of Foundations, Methodologies and Applications 11, 503–518 (2007)
14. Oltean, M.: Evolving evolutionary algorithms using linear genetic programming. Evolutionary Computation 13, 387–410 (2005)
15. Spector, L., Robinson, A.: Genetic programming and autoconstructive evolution with the push programming language. Genetic Programming and Evolvable Machines 3, 7–40 (2002)
16. Darwen, P., Yao, X.: Every niching method has its niche. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 398–407. Springer, Heidelberg (1996)
17. Hillis, W.D.: Co-evolving parasites improve simulated evolution as an optimization procedure. In: Emergent Computation, pp. 228–234. MIT Press, Cambridge (1991)
18. Hifi, M., M'Hallah, R.: A literature review on circle and sphere packing problems: Models and methodologies. Advances in Operations Research (2009)
19. Leitão, A.: Evolving components of evolutionary algorithms. MSc Thesis, Faculty of Science and Technology, University of Coimbra (2010)
20. Tavares, J., Machado, P., Cardoso, A., Pereira, F.B., Costa, E.: On the evolution of evolutionary algorithms. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 389–398. Springer, Heidelberg (2004)