Learning Selection Strategies for Evolutionary Algorithms

Nuno Lourenço[1], Francisco B. Pereira[1,2], and Ernesto Costa[1]

[1] CISUC, Department of Informatics Engineering, University of Coimbra, Polo II - Pinhal de Marrocos, 3030 Coimbra, Portugal
[2] Instituto Politécnico de Coimbra, ISEC, DEIS, Rua Pedro Nunes, Quinta da Nora, 3030-199 Coimbra, Portugal
{naml,xico,ernesto}@dei.uc.pt

Abstract. Hyper-Heuristics is a recent area of research concerned with the automatic design of algorithms. In this paper we propose a grammar-based hyper-heuristic to automate the design of an Evolutionary Algorithm component, namely the parent selection mechanism. More precisely, we present a grammar that defines the number of individuals that should be selected, and how they should be chosen in order to adjust the selective pressure. Knapsack Problems are used to assess the capacity to evolve selection strategies. The results obtained show that the proposed approach is able to evolve general selection methods that are competitive with the ones usually described in the literature.

1 Introduction

Evolutionary Algorithms (EAs) are computational methods loosely inspired by the principles of natural selection and genetics, that have been successfully applied over time to complex problems involving optimization, learning or design. EAs work by defining an initial population of candidate solutions to the problem, which are then iteratively improved by means of variation operators. The subset of individuals that undergo the modification process must be selected according to some fitness criteria. The quality of the solutions achieved by the EA depend on the careful adjustment of some its components and/or parameters. The design is usually performed off-line, by hand, and requires the use of expertise knowledge.

Hyper-Heuristics (HH) is a recent area of research, involving the construction of specific, high-level, heuristic problem solvers, by searching the space of possible low-level heuristics for the particular problem one wants to solve [10]. HH can be divided in two major groups [1]: the selection group encompasses HH that search for the best sequence of low-level heuristics, selected from a set of predefined methods usually applied to the problem one intends to solve; the other group includes methods that promote the creation of new heuristics. In the later case, the HH iteratively learns the specific algorithm which is then applied to solve the problem at hand. During this process, the HH are usually guided by feedback obtained through the execution of each candidate solution in instances of the

problem that needs to be solved. Genetic Programming (GP), a branch of EAs, has been increasingly adopted as a HH to search for effective problem solving algorithmic strategies [3][8]. In the recent years, Grammatical Evolution (GE) [7], a form of GP, has been used with success as a HH, since it allows the enforcement of semantic and syntactic restrictions, by means of a grammar.

In this paper we propose and test a GE-based HH framework to evolve a EA particular component. Specifically, we propose a framework to evolve the selection mechanism used by the EA. With this goal in mind, we expect to obtain selection mechanisms that are general, and are able to successfully guide EAs to solve the problem at hand. The selection component is important for the success of the algorithm, since it determines which individuals should be combined to produce new solutions. We describe a set of experiments, where we show that the framework is able to evolve selection algorithms that are competitive with the ones commonly used in EAs. Moreover we investigate the generalization capacity of the evolved algorithms, by applying them to unseen scenarios. The results are statistical validated.

The paper is organized as follows. In section 2 we discuss some previous relevant work on HH for nature-inspired algorithms, and present the grammar used to evolve selection strategies. In section 3 we introduce the experimental setup for the learning phase and present the results. Section 4 deals with the validation and generalization of the learned selection strategies. In section 5 we summarize the results and suggest directions for future work.

2 A Grammatical Evolution Hyper-Heuristic

The proposed HH relies on GE to search for selection methods. GE is a GP branch, more specifically a form of Grammar-based GP, in which the variation operators are applied to solutions encoded as binary strings. A mapping process is then required to decode this information into an executable algorithmic strategy. The mapping is done by means of a grammar and this process decouples the search engine from the evaluation mechanism. For these reasons, a GE system is general and flexible [7].

2.1 Grammar Definition

To apply a GE engine in our HH framework we must define a grammar whose words are specific selection strategies. In this work we propose a grammar with some modifications to the traditional Backus-Naur Form (BNF), inspired by [2]. These modifications aim to overcome some limitations that the BNF imposes, namely the lack of tools to allow repetition of non-terminal symbols and ranges of alternative values. The first extension is the addition of the operator $\sim$ to signal the repetition of non-terminals. The full syntax is as follows: $\sim< a >< NT >$, where $< a >$ is an integer or terminal value, indicating that the non-terminal $< NT >$ should be repeated $< a >$ times. The second extension is the addition of valued range alternatives. A range of numeric alternative values can

be compactly specified, using the operator &. Thus $< int >::= 0\&5$ is equivalent to $< int >::= 0|1|2|3|4|5$. Taking these extensions into account, the grammar used in this work is as follows:

```
<start> ::= <calculate-parents> <selection-strategy>

<selection-strategy> ::= parents = {~number-of-parents<elements>}

<elements> ::= get_rank(<rank>)

<rank> ::= 0 & POP_SIZE

<calculate-parents> ::= number-of-parents = (random01() * POP_SIZE)
```

The $< start >$ symbol represents the grammar axiom. The grammar starts by calculating the number of parents that the strategy should select, according to a percentage of the total individuals available (POP_SIZE). The evolved strategies are targeted for EAs with a crossover operators, thus we enforce an even number of parents in the selection pool. Afterwards, a selection strategy to choose which individuals will appear in the selection pool is generated. The solutions from the current population are ranked by fitness and a selection strategy emerges by defining which ranks should be chosen as parents.

2.2 Related Work

Several efforts have been reported in literature to automatically evolve nature-inspired algorithms. In [11], Tavares et al. adopted GP to evolve a population of mapping functions between the genotype and the phenotype. Experimental results showed that GP finds mapping functions that can obtain results as good as the ones that are designed by hand.

In [3], Keller et al. propose a linear-GP HH to evolve heuristics to Travelling Salesman problem. In their work they propose several small languages to reduce the search space size. They conclude that the proposed HH is able to evolve heuristics that are able to solve the problem at hand, and that they are parsimonious, i.e. the heuristics make a good use of the resources available.

In [12], Tavares et al. proposed a GE framework to evolve Ant Colony Optimization Algorithms (ACO) to the Traveling Salesman Problem. The results showed that the proposed framework is able to evolve ACO algorithms that are competitive with the human designed ACOs.

Lourenço et al. [5] proposed a GE based HH to evolve full-featured EAs. The results showed that the proposed architecture is able to evolve effective algorithms for the problems under consideration.

In [13] Woodward et al. propose an HH to evolve mapping rules that assign fitness values to each individual in the population. These fitness values are then used to select individuals, using a fitness-proportional mechanism. They consider a set of transformations that can be applied to either the rank or the fitness, and then return the new fitness value of each individual. The experiments results conducted showed that the evolved strategies are human competitive.

3 Learning Selection Heuristics

In this section we aim to gain insight into the capacity of the proposed HH to evolve effective selection strategies. The settings adopted by the GE-based HH for all the tests conducted are depicted in Table 1. Individuals evolved by the HH encode potential parent selection strategies. To estimate their relevance, one must access how they help an EA to solve a given problem. Therefore, each HH individual is implanted in a standard EA, which in turn will solve an instance of an optimization problem. The quality of the best solution found by this EA is used to assign fitness to the corresponding evolved selection strategy. Running an EA to assign fitness to each evolved selection strategy is a computational expensive task. To minimize the computational overhead, we rely on the following conditions to assess the quality of the evolved strategies: i) one single instance of moderate size is used to assign fitness; ii) only one run is performed.

We report experiments using three different EA settings as surrogates for the selection strategies. In all of them, the maximum population size (POP_SIZE) is set to 50 and the number of generations is set to 250. Three possible replacement strategies, $R1$, $R2$, and $R3$, are considered (see Table 2). $R1$ corresponds to a standard generational EA, whereas the last two implement a steady-state architecture where descendants compete with existing individuals for survival based on the fitness criterion. Both $R1$ and $R2$ force the evolved selection strategies to select a number of parents that is equal to POP_SIZE, thus the grammar production $< calculate-parents >$ simply becomes $< calculate-parents >::= number-of-parents = POP\_SIZE$. On the contrary, $R3$ allows the selection strategy to choose a number of parents that is lower than POP_SIZE. All three replacement strategies consider uniform crossover with a rate of 0.9 and swap mutation with rate 0.01 as variation operators. Additional combinations of variations operators were tested with similar outcomes to those reported in this paper (detailed results are not shown due to space constraints).

Table 1: Parameter setting for the GE-based Hyper-Heuristic

Parameter	**Value**
One Point Crossover Probability	0.9
Bit Flip Mutation	0.01
Codon Duplication Probability	0.01
Codon Pruning Probability	0.01
Population Size	100
Selection	Tournament with size equal 3
Replacement	Steady State
Codon Size	8
Number of Wraps	3
Codons in the initial population	50-55
Generations	50
Runs	30

Table 2: Replacement strategies used in the surrogate EAs.

Setting	Fixed	Replacement Strategy
R1	Yes	Generational
R2		Steady State
R3	No	Steady State

3.1 The 0-1 Knapsack Problem

The combinatorial optimization *0-1 Knapsack Problem* (KP) was selected as the testbed for our experiments. It can be described as follows: given a set of n items, each of which with some profit p and some weight w, how should a subset of items be selected to maximize the profit while keeping the sum of the weights bounded to a maximum capacity C? In all instances adopted in our study, the knapsack capacity was set to half of the sum of the weights of all items. A standard binary representation is adopted and evaluation considers a linear penalty function to punish invalid solutions [6].

3.2 Results

The KP instance used to evaluate the selection strategies is composed by $n = 100$ items. Table 3 summarizes the results of the off-line learning process. Note that the results are displayed in terms of the normalized root mean squared error. Every cell contains two values: the number of GE runs that discovered selection strategies that helped the EA to discover the optimum ($BestHits$) and the Mean Best Fitness (MBF) together with the corresponding standard deviation. The outcomes reveal that, for all training situations, the HH is able to learn effective selection strategies.

A detailed inspection reveals that the replacement strategies used in the surrogate EA lead to the appearance of selection methods with different selective pressure. The three lines from Fig. 1 (one from each replacement strategy) help to clarify this issue. For every setting we selected the best selection algorithm evolved in each run and created charts displaying the distribution of the appearance of the possible ranks (values displayed are averages of 30 runs). Note that rank 0 corresponds to the best individual and rank 49 to the worst. An inspection of the figure shows that selection strategies evolved inside a generational surrogate ($R1$) have a higher selective pressure than those that evolved in the steady state surrogates. In generational EAs, the whole population is replaced at each generation. The HH acknowledges the risk of losing good quality solutions and promotes the appearance of selection strategies with a high selective pressure, thereby maximizing the likelihood of passing information contained in good quality solutions to the next generations. On the other hand, in steady state surrogate EAs, the ranks are distributed more or less evenly. This results is not unexpected, since in this scenario, the greedy replacement mechanism already ensures selective pressure: an offspring only enters the population if it is better than its parents. Therefore the selection strategy in these EAs can act

more like a diversity preservation mechanism. Finally, in Fig. 2 we exemplify the rank distribution of one of the best evolved strategies, using the $R1$ setting.

Table 3: Hyper-Heuristic learning results (for 30 runs)

	Replacement strategies		
	R1	**R2**	**R3**
Best Hits	30	30	30
MBF	0.000 (±0.000)	0.000 (±0.000)	0.000 (±0.000)

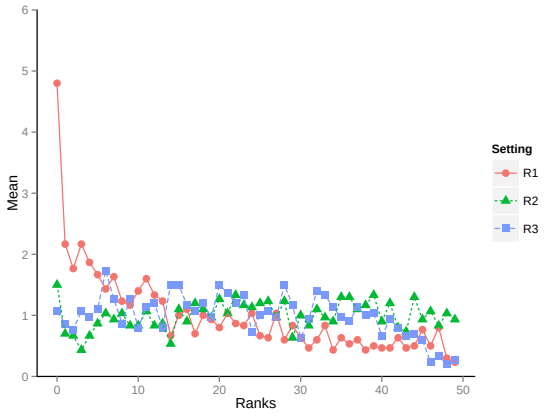


Fig. 1: Rank distribution in the best evolved strategies with the three replacement settings.

4 Validation of the Learned Selection Strategies

The experiments described in this section aim to study how the best strategies discovered by the GE-based HH behave in KP instances that are different from the one used in learning. We selected 4 evolved strategies from each possible replacement strategy, taking into account the following criteria: i) quality of the solution; ii) time taken to reach a solution. In the remainder of this section these selection strategies are identified as $R11$ through $R14$ for methods evolved with the $R1$ replacement strategy, $R21$ through $R24$ for $R2$ replacement strategy, and $R31$ through $R34$ for $R3$ replacement strategy.

This experimental study will help to gain insight into the optimization performance of EAs that have the learned strategies as selection methods. Also, we will verify if the strategies generalize well to unseen instances and are competitive with standard hand-designed selection strategies. Three common selection

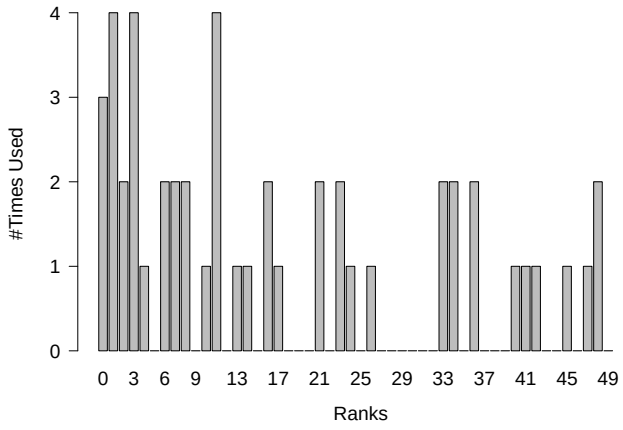


Fig. 2: Example of the rank distribution of a selection strategy evolved with the $R1$ setting.

options (Roulette Wheel, Tournament with size 2, and Tournament with size 3) are considered. We report results obtained with a generational and a steady-state surrogate EAs, both of them relying on uniform crossover with rate 0.9 and binary swap mutation with rate $1/n$ as variation operators. A KP instance with 1000 items and with the knapsack capacity set to half of the sum of the weights of all items was selected for the validation analysis. In every optimization scenario, 30 runs were performed and the best solution found during the execution was recorded. To support our analysis we apply the Friedman's ANOVA test to check for statistical differences in the means. When differences are detected, the *post-hoc* Wilcoxon Signed Rank Test, with Bonferroni correction, is applied to perform the pairwise comparisons. In both tests we used a significance level $\alpha = 0.05$.

Fig. 3 presents the MBF box plot distribution of the 15 selected strategies (12 evolved and 3 hand-designed) for each validation scenario: Panel a) displays the results for the generational surrogate, whereas panel b) presents the results for the steady-state surrogate. Clearly, the performance of the evolved strategies is related to the configuration where they are applied. Strategies $R11-R14$ were evolved with a generational EA surrogate and, as a consequence, they promote a considerable selection pressure. Therefore it is not a surprise that these strategies achieve good results in a validation scenario where a generational surrogate is adopted (see Panel 3a). On the contrary, strategies $R21-R24$ and $R31-R34$ have a low selective pressure and are inadequate for a generational EA environment.

An opposite situation arises in the steady-state validation surrogate (panel 3b). In this scenario, and given the fitness-based replacement strategy adopted, selection methods evolved in a generational environment converge prematurely

to sub-optimal regions of the search space. The remaining 8 evolved strategies were obtained in a scenario similar to the one used in this validation phase. For that reason they contain features that help to maintain diversity and to effectively help the EA to discover the regions of the search space containing the best solutions. The distinction between these two sets of evolved selection methods confirms that the HH framework is able to generate strategies that are suited to the specific features of the training environment.

The results displayed in the two panels of Fig. 3 confirm that the HH is able to evolve selection methods competitive with the hand-designed approaches. The information displayed in Table 4 helps to further clarify the relative performance of learned strategies. Considering the MBFs attained, we performed a full set of pairwise comparisons between evolved strategies and the hand designed algorithms and present a graphical overview: A $+++$ indicates that the algorithm in the row is statistically better than the one in the column, and that the effect size is large ($r \geq 0.5$). As an example, $R11$ clearly outperforms Roulette Wheel in the Generational surrogate. A $++$ sign indicates that there are statistical differences, and that the effect size is medium ($0.3 \leq r < 0.5$). A - signals scenarios where the algorithm in the row is worst than the one in the column. Finally, a $\sim$ indicates that no statistical differences between the algorithms were found. The statistical results confirm that evolved strategies tend to perform better in situations resembling those found during learning. Selection methods $R11 - R14$ excel in the generational scenario and one specific strategy ($R13$) is able to outperform all hand-designed approaches. When the steady-state EA surrogate is adopted, the effectiveness of the $R21 - R24$ and $R31 - R34$ strategies is evident. The performance of methods evolved with the $R2$ setting is particularly impressive, as each one of them outperforms all hand-designed selection mechanisms.

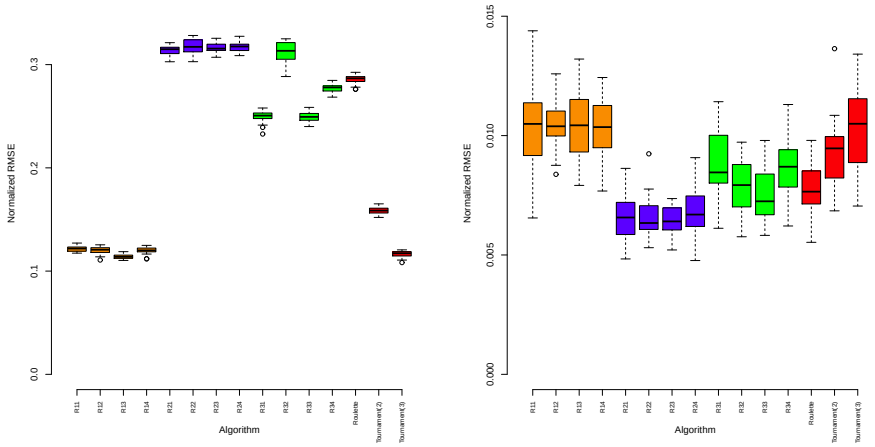
4.1 Generalization

To complete our analysis we briefly investigate if the evolved strategies generalize well to a problem different from that used in the learning step. We maintain our focus on the KP class, but consider the Multiple Knapsack Problem (MKP) variant. The MKP can be described as follows: given two sets of n items and m knapsack constraints (resources), for each item j, a profit p_j is assigned, and for each constraint i, a consumption value r_{ij} is assigned. The goal is to find a subset items that maximizes the profit, without exceeding the given constraint capacities C_i. Note that the KP is a special case of the MKP when $m = 1$. For a formal definition and additional information on the MKP, please refer to [4] or [9]. For our experimental analysis we selected several MKP instances from the *OR-Library*[3]. Due to space constraints we present results obtained with a single MKP instance with $n = 250$ items and $m = 5$ constraints. However, results obtained with other instances follow the same trend.

We maintain the 15 selection strategies adopted in the previous validation analysis and keep all other optimization conditions, including the two same surro-

[3] http://people.brunel.ac.uk/ mastjjb/jeb/orlib/mknapinfo.html

gate EAs. Fig. 4 depicts the MBF box plot distribution of the selection methods, both for the generational (panel a)) and steady-state (panel b)) surrogates. In table 5 we summarize the statistical comparison between the strategies considered in the generalization study. The analysis of the results reveals the exact same trend that was identified in the previous validation. Considering the performance of the evolved selection strategies, there is a clear correlation between the conditions found in the off-line learning step and those of the validation/-generalization experiments. Additionally, optimization results are competitive with those achieved by hand-designed approaches: $R11 - R14$ methods tend to outperform standard selection strategies in generational environments, whereas $R21 - R24$ and $R31 - R34$ excel in steady-state surrogates. These outcomes confirm that the GE-based HH was able to learn strategies that generalize well to different KP variants.

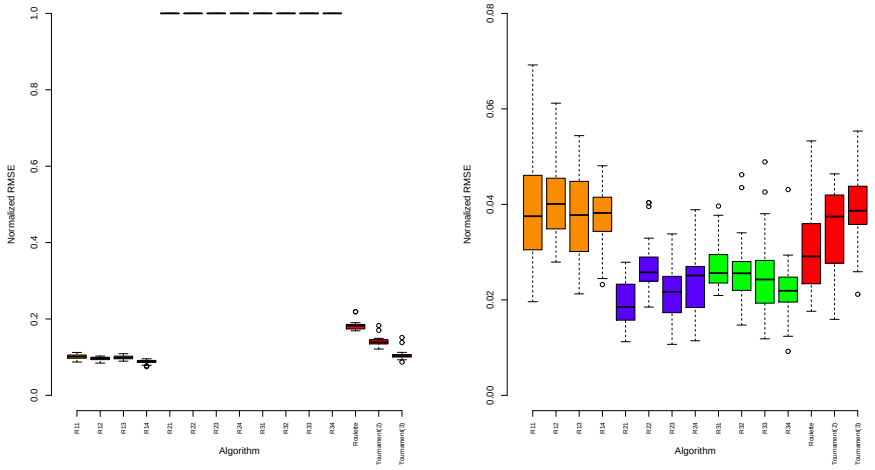


(a) Generational (b) Steady-State

Fig. 3: Optimization results of the 15 selection strategies chosen for the validation study: panels (a), (b), present the results obtained with the generational and steady state EAs, respectively.

5 Conclusions

In this paper we proposed a GE-based HH to discover effective selection strategies for EAs. The proposed grammar is composed by symbols that allow the creation of rank-based selection strategies. We demonstrated the validity of the



(a) Generational (b) Steady-State

Fig. 4: MKP optimization results of the 15 selection strategies chosen for the generalization study: panels (a), (b), present the results obtained with the generational and steady state EAs, respectively.

Table 4: Statistical analysis between the learned strategies and the hand-designed methods in the KP (see text for details on the notation).

	Generational			Steady-State		
	Roulette Wheel	**Tournament(2)**	**Tournament(3)**	**Roulette Wheel**	**Tournament(2)**	**Tournament(3)**
R11	+++	+++	-	-	∼	∼
R12	+++	+++	-	-	-	∼
R13	+++	+++	++	-	-	∼
R14	+++	+++	-	-	-	∼
R21	-	-	-	+++	+++	+++
R22	-	-	-	+++	+++	+++
R23	-	-	-	+++	+++	+++
R24	-	-	-	++	+++	+++
R31	+++	-	-	∼	∼	+++
R32	-	-	-	∼	+++	+++
R33	+++	-	-	∼	+++	+++
R34	+++	-	-	∼	∼	+++

Table 5: Statistical analysis between the learned strategies and the hand-designed methods in the MKP (see text for details on the notation).

	Generational			Steady-State		
	Roulette Wheel	Tournament(2)	Tournament(3)	Roulette Wheel	Tournament(2)	Tournament(3)
R11	+++	+++	∼	∼	∼	∼
R12	+++	+++	+++	-	∼	∼
R13	+++	+++	++	-	∼	∼
R14	+++	+++	+++	-	∼	∼
R21	-	-	-	+++	+++	+++
R22	-	-	-	∼	++	+++
R23	-	-	-	+++	+++	+++
R24	-	-	-	++	+++	+++
R31	-	-	-	∼	++	+++
R32	-	-	-	∼	+++	+++
R33	-	-	-	∼	+++	+++
R34	-	-	-	+++	+++	+++

approach in the domain of different Knapsack Problem variants. Results obtained show that the HH framework adapts the selective pressure of the evolved mechanism, taking into account the specific features of the adopted surrogate. Despite the simplicity of the proposed grammar, the HH was able to learn effective selection strategies, competitive with standard hand-designed mechanisms regularly adopted in the literature. Moreover, the evolved strategies generalize well to different variants of the problem considered in our study.

There are several possible extensions to the work described in this paper. One possibility is to expand this framework to different problems and verify if strategies evolved in one specific optimization situation generalize well to different, possibly related, problems. Another possibility is to consider different learning design options, such as performing multiple runs to evaluate a solution, or the adoption of multiple training instances .

We will also consider several extensions to the grammar, by adding new symbols that take into account different features of the individuals belonging to the population (e.g., age).

6 Acknowledgments

This work was partially supported by Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grant SFRH/BD/79649/2011.

References

1. Burke, E., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.: A classification of hyper-heuristic approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, International Series in Operations Research and Management Science, vol. 146, pp. 449–468. Springer US (2010)
2. Group, N.W.: Augmented BNF for syntax specifications: ABNF (1 2008), `ftp://ftp.rfc-editor.org/in-notes/std/std68.txt`

3. Keller, R., Poli, R.: Linear genetic programming of parsimonious metaheuristics. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. pp. 4508–4515 (2007)
4. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004)
5. Lourenço, N., Pereira, F., Costa, E.: Evolving evolutionary algorithms. In: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion. pp. 51–58. GECCO '12, ACM, New York, NY, USA (2012)
6. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs (3rd ed.). Springer-Verlag, London, UK, UK (1996)
7. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, Norwell, MA, USA (2003)
8. Pappa, G.L., Freitas, A.: Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach. Springer Publishing Company, Incorporated, 1st edn. (2009)
9. Raidl, G.R., Gottlieb, J.: Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. Evolutionary computation 13(4), 441–475 (2005)
10. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: introductory tutorials in optimization and decision support techniques, chap. 17, pp. 529–556. Springer US (2005)
11. Tavares, J., Machado, P., Cardoso, A., Pereira, F., Costa, E.: On the evolution of evolutionary algorithms. In: Keijzer, M., OReilly, U.M., Lucas, S., Costa, E., Soule, T. (eds.) Genetic Programming, Lecture Notes in Computer Science, vol. 3003, pp. 389–398. Springer Berlin / Heidelberg (2004)
12. Tavares, J., Pereira, F.B.: Automatic design of ant algorithms with grammatical evolution. In: Proceedings of the 15th European conference on Genetic programming (2012)
13. Woodward, J., Swan, J.: Automatically designing selection heuristics. In: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation. pp. 583–590. GECCO '11, ACM, New York, NY, USA (2011)