Towards Automatic Design of Adaptive Evolutionary Algorithms

Ayman Srour
KU Leuven, CODeS
Aymanih.srour@kuleuven.be

Patrick De Causmaecker
KU Leuven, CODeS
patrick.decausmaecker@kuleuven.be

***Abstract*—Adaptive evolutionary algorithms aim to enhance the performance of traditional evolutionary algorithms by adjusting their parameter in an online manner. The literature is rich with several adaptive evolutionary algorithm approaches. Their success required a good design of adaptive parameter control strategy that suited the problem at hand, which is a non-trivial task due to several alternative designs of adaptation strategies available today. in this paper, a framework for automating the design of adaptive evolutionary algorithm is presented. The framework utilizes the adaptive parameter control model and grammatical evolution to evolve the design of adaptive evolutionary algorithms for different problems.**

Keywords—automatic algorithm design; adaptive evolutionary algorithm; grammatical evolution;

I. INTRODUCTION

In the field of evolutionary computation, several evolutionary algorithms (EAs), such as genetic algorithm, genetic programing, evolution strategy, have been proposed during the last view decades with a notable success when they are applying to a large set of combinatorial optimization problems. Their success is due to several adjustable parameters such as population size, crossover rate and mutation rate. However, most of these algorithms allow control to the user to tune the parameter values when they used for solving different problems to achieve an optimal performance. Furthermore, it has been argued that different parameter values may be optimal at different optimization stages [1-3], which makes the current research more focus on tackling the issue by using adaptive parameter control. Adaptive parameter control (APC) is used to tune the parameter in an online manner and during the algorithm execution by considering a feedback from an EA run such as solutions quality to monitor the algorithm performance and adjust the parameter values for future iterations.

Recently, the EA parameter control has been widely studied and many of APC methods have been proposed in the literature [4, 5]. Most of these methods are a domain and algorithm specific design, and some are limited to one or two parameters. However, the variation in these methods (in terms of its performance and structure) makes the choice of suitable APC methods for a specific problem or instance non-trivial, as there exist no general optimal APC method for EAs. This means that for EAs, the optimal APC method can considerably vary depending on the problem or instance at hand. In addition, some of the previous work such in [6] and [7] tried to combine different APC methods to improve the EA performance. Yet, several alternative combinations of APC methods are still not explored. In the context of EAs, reference [5] has presented an interesting survey about the parameter control methods. The authors concluded that: more investigations about the combination of different control mechanisms could be performed, and developing a generic framework for APC could be helpful.

Reference [4] proposed a generic model of the state-of-the-art APC methods based on a comprehensive review of the literature. The authors distinguished between the optimization process and the control process of the adaptive EA;s. The control process has been further divided into four components, namely, *feedback collection, effect assessment, quality attribution and selection* (*see section II-A*). Each component represents a stage of the parameter control accompanying with several possible adaptive strategies for each stage. However, by considering this model, the architecture of every adaptive EA can be mapped into the four components. Thus, a complete design on any adaptive EA should have at least one strategy for each component, and each strategy can be implemented by using one of many predefined rules or methods. In fact, we can use the model as cornerstone to facilitate the design of any new adaptive EA, but it still time consuming to perform this task manually as it encountered with several non-trivial selection or combination the most suitable variant of strategies (and so its rules) which are associated with each component to address the problem at hand.

In recent years, several automatic designs of the algorithm were presented to overcome this limitation. Reported in literature a range of approaches that used to automate the algorithm design based on selecting or combining different algorithmic components. An example of automatic design, grammatical evolution (GE) has been used to evolve an algorithm such as presented by [8] for evolving data mining algorithms, and [9] used GE for evolving new local search algorithms for the bin-baking problem. In this paper, we propose a framework to evolve the design of adaptive EA. GE is adopted as the design methodology which can simply and automate the design of adaptive EA that works well across different problems.

II. BACKGROUND

Each EA has its predefined set of parameters which are usually fixed and their values should be initialized before an execution. According to the no free of lunch theorem [10], no optimal algorithm configuration exist for solving all problems. Therefore, an algorithm needs to have parameters to be adaptable with respect to the problem at hand. However, the problem of finding optimal parameter configuration for an

algorithm is not trivial and can be seen as an optimization problem which encounters with the difficulty of manual parameters configuration. Therefore, finding a new configuration approach such as automatic parameter tuning is always in demand. Currently, the study of techniques for automatic parameter tuning is an active research area [4, 5, 11, 12]. These techniques can roughly be divided into parameter tuning and parameter control [12]. The former one means the parameter values are assigned before the algorithm execution, whilst the later means the values are not fixed and subject to change during the algorithm execution. According to [12], the parameter control can be further classified into *Deterministic*, *adaptive* and *self-adaptive* parameter control. In short, the deterministic parameter control means the parameter value assigned based on deterministic rules with no knowledge about the search progress. On the other hand, self-adaptive parameter control combines the search of optimal parameter values with the solution search i.e. encode the parameter values in genome to enable them to co-evolve with the solutions. Adaptive parameter control separates the search for optimal parameter values from the solutions search, and monitors an algorithm properties during the optimization process and adjusts the parameter values accordingly.

A. Adaptive EA Design Model

Over the past two decades, a wide variety of Adaptive parameter control has been proposed with several adaptation strategies. A compressive study of the research dictions of parameter control methods has been shown a great momentum of research conducted in the field in the recent years [4, 5, 13, 14]. In the context of EA, more sophisticated literature reviews , such as [4, 15], have focused on studying the design structure of the existing APC methods and tried to decompose the APC process into several elements or components, the components that play an important role, to generalize the design strategies of the most existing APC methods. Corriveau et al. 2016 [15] divides the adaptive parameter control process in EA into four basic elements, namely, parameters involved (the type and states of the parameters involved), feedback indicators (used to evaluate the impact of the current state of the involved parameters), credit assignment scheme (used to convert feedback information into a suitable reward) and parameter selection rule (used to update parameter states).

Aleti et al. 2015 [4] proposed a more sophisticated and comprehensive conceptual model of APC which also consists of four main components. Figure 1 illustrates the general algorithmic flow and components involved in the adaptive EA [4]. The algorithm starts with a population (initial solutions). This population is evolving during the algorithm execution until the stopping criterion is reached. The EA parameters, such as population size, genetic operators and the probabilities of performing of the genetic operators, the number of offspring, etc., can be adjusted by the parameter control methods. At each cycle in the optimization process, the generated solutions are evaluated by using the fitness function(s), which provides valuable information about the algorithm performance as a feedback to guide the parameters control method. These kinds of information can be processed in the feedback collection strategy, which can measure and records a specific property of the algorithm performance during the execution i.e. phenotype/genotype quality and phenotype/genotype diversity etc. [28] measure the algorithm performance using the fitness difference between the created solution(s) and its parent(s). Reference [26] used the weighted sum of the solution improvement and the diversity of the offspring assigned as the Hamming distance. [27] introduced a different diversity measure which estimates the distribution of the solutions in the search space, by calculating the Euclidean distances of the selected solutions to the best solution so far.

The feedback information can be used by effect assessment strategy to assess the cause of a change of the properties of the solutions during the run by measuring the effects of each parameter values on the algorithm performance. The parameter control utilizes the effect assessment information to determine which parameter values will potentially perform well in the future iterations by using a specific rule. The main differences among the existing effect assessment methods is how they evaluate the success of parameter values i.e. the effect assessment method presented in [16] calculates the effect of the parameters using quality differences between the generated solutions compared to their parents, whilst [17] used the overall best solution and [18] used median. A brief description of each APC components and its related strategies, adopted from [4], is presented in Table 2. However, the next component is the quality attribution strategy, which is built from the rules used in the effect assessment strategy. The parameter quality attribution is useful to estimate the quality of parameter values as it can help to choose the next parameter values. Finally, the parameter selection strategy is used to determine which the parameter values will be used for the future iterations.

To illustrate the Adaptive EA model, consider the Integrated-Adaptive Genetic Algorithm (IAGA) [6] as an example. The authors adjust genetic operators and their rates. In the feedback collection strategy, they used phenotype feedback by considering the fitness of the solutions. In the effect assessment, two different effect assessment have been applied, namely, best solution effect and ancestor solution effect. A reinforcement-based rate adaptation model is applied to measure the quality of the parameter values and can be categorized as learned quality attribution. Finally, the author used the proportional selection of reinforcement values of the operator rates to be used in next iterations. From the previous example and other examples presented in table 1. We can see a degree of variation in adapted EA design and find a combination of some existing strategies is also possible. However, classifying the existing adaptive EA based on decomposing it into several components and strategies can help to develop an optimized variant of EAs, especially when an automatic algorithm designed approach is used; then the task can be much easier.

Table 1 Examle of exesting APC stratigyes

Reference	EA Parameter	APC			
		Feedback collection	Effect assessment	Quality attribution	Selection
Hong et al. 2002 [16]	Crossover operator Mutation operator	Phenotype	Ancestor Current	average	Quality proportionate
Barkaoui and Gendreau 2013 [19]	Crossover operator Mutation operator Selection	Phenotype	Current	Immediate	Quality proportionate
Arnone et al. 1994 [20]	Mutation rate Representation	Phenotype Genotype diversity	Best Current	Immediate	Greedy
Eiben et al. 2007 [21]	Population size Selection Crossover rate Mutation rate	Phenotype	Population Current	Learned	Greedy
Smorodkina and Tauritz 2007 [22]	Population size	Computations	Population Best Current	Average	Greedy

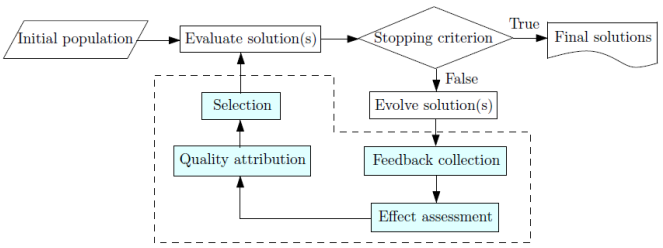

Fig. 1. The main steps of optimization with Evolutionary Algorithms using adaptive parameter control [4]

B. Automatic algorithm design using button-up approach

Recently, several approaches for automating the metaheuristics design were developed, by both researchers and practitioners. Two main approaches for automatic algorithm design have been defined by [23], namely, top-down approach and bottom-up approached. The former one uses a parametrized algorithmic framework to generate an algorithm by starting with a general procedure and integrating different high-level algorithmic components [24]. On the other hand, the bottom-up approach is used by grammar-based genetic programming [8, 9, 23]. In this approach, the design of the algorithm is carried out by defining a context-free grammar, and the design space is represented by a set of production rules. The advantage of this approach is the ability to combine a valid algorithmic component in more fine-grained than top-down approach. Reference [25] used a genetic programming to design a hybrid large neighborhood search algorithms for solving vehicle routing problem by applying crossover and mutilation of a predefined set of terms derived from the grammar. References [8, 9, 23, 26] used a grammatical evolution [27], a variant kind of genetic programing, to generate a complete algorithm by composing a predefined set of algorithmic components. Grammatical evolution (GE) [27] is a grammar-based genetic programming capable of generating a variable length code in any language by using a leaner genuine system representation. The GE uses Backus Naur-form (BNF) specification language to represent a grammar which is used in genotype to phenotype mapping process to generate a complete program from the genotype binary string.

III. FRAMEWORK

As aforementioned, EAs search performance is highly sensitive to its structure and parameters settings. Therefore, the performance of any EAs can be potentially improved by automatic adjustment of its structure/parameter setting. We introduce a framework to automate the design of adaptive EA based on grammatical evolution. The proposed framework can be used to evolve adaptive EAs by adopting different APC strategies from literature [4]. The framework is composed of two main components (see Figure 2): GE optimizer and adaptive EA. The main task of GE optimizer is to evolve individuals that encode effective APC strategies for the adaptive EA. Whereas the Adaptive EA component represents the implantation of many possible adapted EA architecture based on the newly generated APC strategies (GE individuals) from GE optimizer. The component is necessary for evaluating the generated solutions by assigning fitness to each individual by training it on problem instances. Without loss of generality, let's take the traveling salesman problem (TSP) as an example to evolve an adaptive EA for TSP. The generated solutions can be evaluated by assigning each generated APC strategy to a single Adaptive EA, then measure their fitness using TSP problem instances such as *eil76, pr76,.., gr96*.

The GE optimizer adopts the standard architecture of GE, which consists of BNF Grammar, Search engine and Mapper function. The description of each them will be presented in the following subsections:

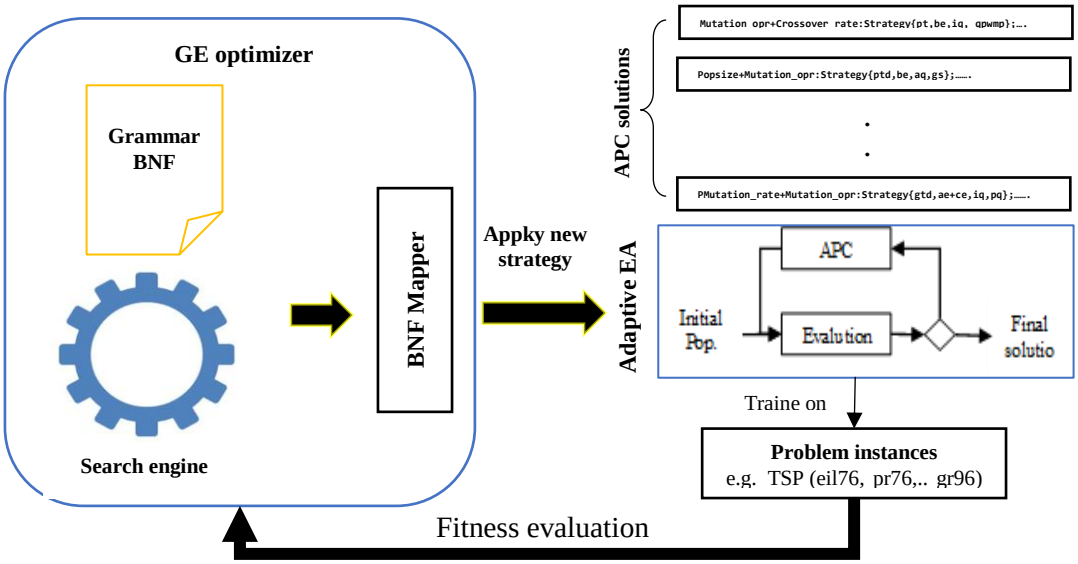

Fig. 2. The proposed framework

Search engine: GE uses a standard genetic algorithm as its search engine [27]. The chromosome (genotype) is represented by variable length binary string, The gene in each chromosome (so-called codon) has 8-bit binary values which are later decoded into an integer (in the range between 0 to 2^8-1). Repeatedly, the integer values produced by a chromosome can be used to convert all terminals to non-terminals symbols via a mapper function. The fitness of each chromosome is then evaluated by executing its corresponding program. In general, the method that searching the best program in GE is usually performed on the basis of evolutionary manner, whilst different search methods can also be applied and tested as optimizer i.e. [23] used *irace* [28] to search the grammar space using the same codon-based representation.

***BNF grammar*:** The BNF grammar should be defined according to the four adaptive parameter control components and their strategies, namely, feedback collection (FC), effect assessment (EFA), quality attribution (QA) and selection (SE). The four components have been selected based on the conceptual model proposed by [4], which are reflecting the actual design of variant APC strategies of adaptive EA in the literature. Each of these components has its own set of adaptive parameter control strategies accompanying with a set of methods/rules for each strategy. Table 2, illustrates the existing APC strategies. However, the BNF grammar can be formulated as a tuple <T, N, S, P>, where T denotes to terminal sets (in our case the high-level APC strategies), N denote to non-terminals set (a set of rules and set of EA parameters), S is the start symbol and finally P denotes to the production rules that maps the elements of N to T (in our case, the production rules that used to generate a variant kinds of APC strategies). Figure 3 shows an example of possible BNF grammar definition which consists of terminal, non-terminal and production rules.

Table 2 Adaptive parameter control strategies description

APC component	Strategy	Description
Feedback collection (FC)	Phenotype (pt)	Refers to the fitness function of solutions.
	Phenotype diversity (ptd)	To measure the diversity in the solutions
	Genotype (gt)	The information about the solution components of a given solution.
	Genotype diversity (gtd)	To measure the diversity in the solution components over a set of solutions.
	Feasibility (fe)	To measure the feasibility of solutions .
	Computations (co)	Refers to the value of time, generations or function evaluations.
Effect assessment (EFA)	Ancestor effect (ae)	Refers to the change in the current solution(s) properties (i.e. fitness) comparing to its (their) parent(s).
	Population effect (pe)	Refers to the change in the current solution(s) properties comparing to the population.
	Best effect (be)	Refers to the change in the current solution(s) properties comparing to the best solution.
	Worst effect (we)	Refers to the change in the current solution(s) properties comparing to the worst solution.
	Median effect (md)	Refers to the change in the current solution(s) properties comparing to the median solution.
	Current effect (ce)	Use the current solution(s) properties to measure the effects of the parameter value
Quality attribution (QA)	Immediate (iq)	refers to the properties of the current solution.
	Average (aq)	Refers to the average change in the properties of all or (a group of) solutions.
	Extreme (eq)	refers the best change in the current solution(s) properties.
	Learned (lq)	Use an AI or machine leering techniques to estimate the quality of parameter values.
Selection (SE)	Quality proportionate (qp)	Use parameter values quality to estimate the selection probability of each parameter value for next iterations.
	Quality proportionate with minimum probability (qpmp)	To avoid missing inferior parameter values, a minimum selection probability is assigned to each parameter value.
	Greedy (gs)	A greedy mechanism is used to estimate the best parameter value.
	Deterministic (ds)	A deterministic rule is used to assign the parameter values based on their quality.

Rule No		No. of choices
(1)	<APC>::=<Param><Strategy>;|<Param><Strategy>;<APC>	2
(2)	<Strategy>::=(<FC>,<EFA>,<QA>,<SE>)|(defult)	2
(3)	<FC>::=<fc>|<fc>+<FC>	2
(4)	<fc>::=<pt>|<ptd>|<gt>|<gtd>|<fe>|<co>	6
(5)	<EFA>::=<efa>|<efa>+<EFA>	2
(6)	<efa>::=<ae>|<pe>|<be>|<we>|<me>|<ce>	6
(7)	<QA>::=<qa>|<qa>+<QA>	2
(8)	<qa>::=<iq>|<aq>|<eq>|<lq>	4
(9)	<SE>::=<se>|<se>+<SE>	2
(10)	<se>::=<qp>|<qpwmp>|<gs>|<ds>	4
(11)	<param>::=<Select_param>:|<Select_param>+<param>	2
(12)	<Select_param>::=Pop_Size|Crossover_opr |Crossover_rate|Mutation_opr |Mutation_rate|Selection	6
.	.	
.	.	
.	.	
(n)	.	

Fig. 3. An example of BNF grammar for adaptive parameter control strategy

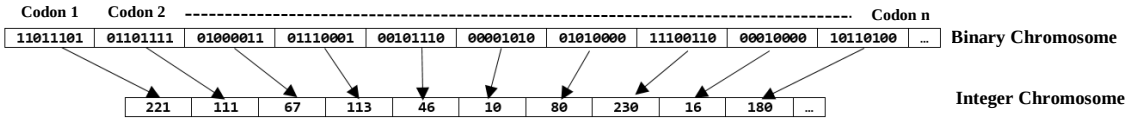

Fig. 4. Example individual shows the integer values generated by converting the 8-bit binary number of each codon into its corresponding integer value.

Dervitation sequence	Mapping rule
<APC>	start
<Parm><Strategy>;<APC>	221 % 2=1
<Select_param>+<param><Strategy>;<APC>	111 % 2=1
Crossover_opr+<param><Strategy>;<APC>	67 % 6 =1
Crossover_opr +<Select_param>+<param><Strategy>;<APC>	113 % 2=1
Crossover_opr+Mutation_opr+<param><Strategy>;<APC>	45 % 6 =3
Crossover_opr+Mutation_opr+<param><Strategy>;<APC>	10 % 2 =0
Crossover_opr +Mutation_opr+Crossover_rate<Strategy>;<APC>	80 % 6 =2
Crossover_opr+Mutation_opr+Crossover_rate:Strategy{<FC>,<EFA>,<QA>,<SE>};<APC>	230 % 1=0
Crossover_opr+Mutation_opr+Crossover_rate:Strategy{<select_fc>,<EFA>,<QA>,<SE>};<APC>	16 % 2 =0
Crossover_opr+Mutation_opr+Crossover_rate:Strategy{pt,<EFA>,<QA>,<SE>};<APC>	180 % 6=0
.	.
.	.
.	.
Crossover_opr + Mutation_opr + Crossover_rate: Strategy {pt,be,iq, qpwmp}; Mutation_rate: Strategy {gtd,ae+current,iq, qp};	stop

Fig. 5. An illustration of the generation of a derivation sequence

***Mapper function*:** the mapper function converts the genotype to phenotype by taking a binary string and BNF grammar as input and map it to the corresponding program. The mapping process can be performed using the following formula:

Rule= (codon integer value) MOD (the number of for the current non-terminal)

To illustrate how the mapping process is performed, consider the individual in figure 4. The integer values will be used to look up figure 4, which describes the BNF grammar example. To perform the divination we should start from the starting symbol <APC> (see figure 5), it can be noted that there are two production rules to be chosen. We can read the first codon "11011101" from the binary chromosome and use it to generate the integer number 221. The number will be used to choose the production rule to use based on the following:

```
(1) <APC>::=<Param><Strategy>;       (0)
           |<Param><Strategy>;<APC> (1)
```

By using the mapping rule defined earlier, 221 MOD 2= 1 means that we must select the production rule no 1 so that <APC> will be replaced by:

```
<Param><Strategy>;<APC>
```

Considering the leftmost divination, we should start with <Param> terminal, which has 2 possible production rules as following:

```
(1) <param>::=<Select_param>:         (0)
             |<Select_param>+<param> (1)
```

The rule choice must be made by reading the next codon number 111 and again using the mapping rule we get 111 MOD 2= 1 i.e. rule (1). Therefore, the production rule <param> must be expanded and replaced by <Select_param>+<param> to obtain:

```
<Select_param>+<param><Strategy>;<APC>
```

New, the leftmost non-terminal <Select_param> will be determined by the next codon number 67, so we have 67 MOD 6=1 which means that the non-terminal <Select_param> will be replaced by Crossover_opr. We have the following:

```
Crossover_opr+<param><Strategy>;<APC>
```

Repeatedly, the mapping process continues until all terminals expanded to non-terminals. The non-terminals can be generated in the form shown in figure 4, as they include two different strategies that assigned to four parameters of EA, namely, `Crossover_opr`, `Mutation_opr`, `Crossover_rate` and `Mutation_rate`. The first three parameters (`Crossover_opr`, `Mutation_opr` and `Crossover_rate`) share the same APC strategy `{pt,be,iq, qpwmp}` which represents the phenotype feedback collection, best effect assessment, immediate quality attribution and quality proportionate with minimum probability, respectively. The phenotype feedback or solutions fitness to provide a major feedback information. The best effect assessment can use the best fitness frequency (bff) method [17] to measure the frequency distribution of the fitness values of the current population. The value of bff indicates either successful (low bff value) of the parameters effect on the search state or stagnation in local optimum (high bff value). However, the immediate effect is used here as quality attribution. The geometric progression method [29] can be used for quality proportionate with minimum probability in the selection component.

The `Mutation_rate` uses different strategy, `{gtd,ae+current,iq, qp}`, to control its parameter values. The genotype diversity is selected for the feedback collection, which can be calculated using i.e. the Euclidean distance of the solutions to the best solution [17]. A combination between ancestor and current effect assessment is used to measure the effects of `Mutation_rate` parameter values on the search state. The quality attribution and selection components use similar strategies that used by `Crossover_opr`, `Mutation_opr` and `Crossover_rate`.

IV. CONCLUSION AND FUTURE WORK

In this paper, a framework for the automatic design of adaptive evolutionary algorithms using grammatical evolution is proposed. The key decision in the development of the framework is the definition the BNF grammar used by GE. As they will decide which APC strategy will be used in the design of adaptive EA. In this paper, we utilize the adaptive parameter control model [4] in defining the BNF grammar. The adaptive parameter control model was built based on a sophisticated review of the current adaptive parameter control strategies. The great momentum of the research in the field has shown a potential to improve evolutionary problem solvers.

Our future work will focus on showing that if the proposed framework can evolve the design of adaptive EA. Thus, to validate our ideas, the future work will be allotted in 1) the framework development, 2) learning and 3) the validation of the applicability of the framework.

The framework development will consider the design of GE optimizer that includes the definition of the BNF grammar for the APC strategies. The implementation of adaptive EA considering bespoke APC strategies and the define of suitable BNF grammar will be also performed in this phase. The learning process will be performed using different problems to test the generality of the framework. The learning process will produce a set of evolved adaptive EA designs for each problem, However, to test the validity of the framework, a set of experiments will be performed to measure the performance of the evolved adaptive EAs.

REFERENCES

[1] T. Bäck, "The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm," in *PPSN*, 1992, pp. 87-96.

[2] J. Hesser and R. Männer, "Towards an optimal mutation probability for genetic algorithms," in *International Conference on Parallel Problem Solving from Nature*, 1990, pp. 23-32.

[3] J. Smith and T. C. Fogarty, "Self adaptation of mutation rates in a steady state genetic algorithm," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, 1996, pp. 318-323.

[4] A. Aleti and I. Moser, "A Systematic Literature Review of Adaptive Parameter Control Methods for Evolutionary Algorithms," *ACM Computing Surveys (CSUR)*, vol. 49, p. 56, 2016.

[5] G. Karafotias, M. Hoogendoorn, and Á. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, pp. 167-187, 2015.

[6] H. Luchian and O. Gheorghieş, "Integrated-adaptive genetic algorithms," in *European Conference on Artificial Life*, 2003, pp. 635-642.
[7] F. Vafaee and P. C. Nelson, "A genetic algorithm that incorporates an adaptive mutation based on an evolutionary model," in *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, 2009, pp. 101-107.
[8] G. L. Pappa and A. A. Freitas, "Automating the Design of Data Mining Algorithms," ed: Springer-Verlag Berlin Heidelberg, 2010.
[9] E. K. Burke, M. R. Hyde, and G. Kendall, "Grammatical evolution of local search heuristics," *IEEE Transactions on Evolutionary Computation*, vol. 16, pp. 406-417, 2012.
[10] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, pp. 67-82, 1997.
[11] K. De Jong, "Parameter setting in EAs: a 30 year perspective," in *Parameter setting in evolutionary algorithms*, ed: Springer, 2007, pp. 1-18.
[12] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith, "Parameter control in evolutionary algorithms," in *Parameter setting in evolutionary algorithms*, ed: Springer, 2007, pp. 19-46.
[13] J. E. Smith and T. C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft computing*, vol. 1, pp. 81-87, 1997.
[14] G. Eiben and M. C. Schut, "New ways to calibrate evolutionary algorithms," in *Advances in metaheuristics for hard optimization*, ed: Springer, 2007, pp. 153-177.
[15] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin, "Bayesian network as an adaptive parameter setting approach for genetic algorithms," *Complex & Intelligent Systems*, vol. 2, pp. 1-22, 2016.
[16] T.-P. Hong, H.-S. Wang, W.-Y. Lin, and W.-Y. Lee, "Evolution of appropriate crossover and mutation operators in a genetic process," *Applied Intelligence*, vol. 16, pp. 7-17, 2002.
[17] M. Giger, D. Keller, and P. Ermanni, "AORCEA–An adaptive operator rate controlled evolutionary algorithm," *Computers & Structures*, vol. 85, pp. 1547-1561, 2007.
[18] B. A. Julstrom, "What have you done for me lately?{A} dapting operator probabilities in a steady-state genetic algorithm," 1995.
[19] M. Barkaoui and M. Gendreau, "An adaptive evolutionary approach for real-time vehicle routing and dispatching," *Computers & Operations Research*, vol. 40, pp. 1766-1776, 2013.
[20] S. Arnone, M. Dell'Orto, and A. Tettamanzi, "Toward a fuzzy government of genetic populations," in *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*, 1994, pp. 585-591.
[21] A. Eiben, M. Horvath, W. Kowalczyk, and M. C. Schut, "Reinforcement learning for online control of evolutionary algorithms," in *International Workshop on Engineering Self-Organising Applications*, 2006, pp. 151-160.
[22] E. Smorodkina and D. Tauritz, "Greedy population sizing for evolutionary algorithms," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 2181-2187.
[23] F. Mascia, M. López-Ibáñez, J. Dubois-Lacoste, and T. Stützle, "Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools," *Computers & operations research*, vol. 51, pp. 190-199, 2014.
[24] M. López-Ibánez and T. Stutzle, "The automatic design of multiobjective ant colony optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 16, pp. 861-875, 2012.
[25] Y. Caseau, G. Silverstein, and F. Laburthe, "Learning hybrid algorithms for vehicle routing problems," *arXiv preprint cs/0405092*, 2004.
[26] J. Tavares and F. B. Pereira, "Automatic design of ant algorithms with grammatical evolution," in *European Conference on Genetic Programming*, 2012, pp. 206-217.
[27] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, pp. 349-358, 2001.
[28] M. López-Ibánez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," Citeseer2011.
[29] T.-P. Hong and H.-S. Wang, "A dynamic mutation genetic algorithm," in *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, 1996, pp. 2000-2005.