PARAMETER CONTROL IN EVOLUTIONARY ALGORITHMS BY DOMAIN-SPECIFIC SCRIPTING LANGUAGE PPCEA

Shih-Hsi Liu
Department of Computer and Information Sciences
University of Alabama at Birmingham, USA
liush@cis.uab.edu

Marjan Mernik
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
marjan.mernik@uni-mb.si

Barrett R. Bryant
Department of Computer and Information Sciences
University of Alabama at Birmingham, USA
bryant@cis.uab.edu

Abstract Programmable Parameter Control for Evolutionary Algorithms (PPCEA), a domain-specific scripting language, solves the problems of control parameter settings in a programmable fashion. It keeps the evolutionary algorithm simple and lifts the problems of control parameter settings into a higher abstraction layer by using metaprogramming. From our experiments, PPCEA outperforms the trial-and-error approach and performs the adaptable, reusable and controllable solutions of control parameter settings for evolutionary algorithms in parameter tuning, deterministic, and adaptive aspects.

Keywords: Evolutionary algorithms, Parameter tuning, Parameter control, Domain-specific scripting language

1. Introduction

It has long been acknowledged that the parameters that control an Evolutionary Algorithm (EA) such as the population size, the probability of crossover, etc., have significant impacts on EA's performance. Therefore, the designer of an EA has a problem with deciding what control parameter settings are likely to produce the best results. Choosing the right parameter values is a time-consuming and tedious task. Researchers can use experience from previous similar application scenarios or follow guidelines described in [10, 16] for particular numeric optimization problems. Most often, for new problems previous experience is not available nor are the above mentioned guidelines applicable. This leads researchers to choose the trial-and-error approach, spending a considerable amount of effort on this task. To make the problem even harder, different values of parameters might be optimal at different stages of the evolutionary process. For example, in the early stages the larger population is needed than in the later stages when fine tuning of sub-optimal solutions is done.

More recently, researchers recognized that specific problems require specific values of control parameters and that a general near-optimal control parameter setting is not appropriate. The problem is well known to the researchers working on EAs [5] and has not been solved sufficiently yet. Various EAs which differentiate substantially from standard EAs have been proposed (e.g., GAVaPS - genetic algorithm with varying population size [1], 1/5 success rule [2] in evolution strategies) to partially solve this problem. All these new ideas and approaches may be incorporated in one's own EA. However, in this case an EA becomes extraordinarily complex and needs to be adapted whenever we decide to implement different strategies for control parameter settings (e.g., from deterministic to adaptive scheme) or whenever a new method is incorporated into the existing algorithm.

In this paper, a novel solution to this problem is presented. We keep EA simple as before and lift the problem of control parameter settings into a higher abstraction layer. A domain-specific scripting language (DSSL) called Programmable Parameter Control for Evolutionary Algorithms (PPC_{EA}) has been implemented to address these needs, where control parameters are set in a programmable fashion with small programs which interact with the original EA.

The paper is organized as follows. Section 2 describes the related work. In Sect. 3, we introduce PPC_{EA}. Section 4 shows the examples and experimental results. Finally, Sect. 5 addresses the conclusion.

2. Related Work

There have been a variety of studies [5, 6, 7] on determining the best control parameter values to use for EAs. The main problem is to find control parameters,

which optimally balance exploration and exploitation. Recommendations on control parameters for a particular set of problems can be found in [10, 16].

In [5] an overview of this problem has been given, where the authors distinguish between parameter tuning and parameter control. In parameter tuning, the values of control parameters are set before the run of the EA (e.g., Matlab approach [4]), and in parameter control, values are changed during the run. It was argued also that parameter tuning is less appropriate and parameter control is preferred. One of the problems in parameter tuning of the trial-and-error approach is that one control parameter is usually tuned at a time. This leads to sub-optimal values since control parameters interact in a complex way. Furthermore, methods for parameter control have been classified into deterministic, adaptive, and self-adaptive categories [5]: the deterministic category adjusts parameters by deterministic rules; the adaptive category utilizes the feedback of the evolutionary process to control the direction and magnitude of parameters; the self-adaptive category encodes parameters into chromosomes and undergoes mutation and recombination.

Another approach is to consider the search for the best values of control parameters of an EA as an optimization problem itself which can be solved by another EA, leading to a meta-evolutionary approach. Several meta-evolutionary approaches have already been proposed. In [7] the meta-evolutionary approach is used to determine population size, crossover probability, mutation rate, generation gap, scaling window, and selection strategy. In [6] the meta-evolutionary approach is used to determine a large set of 20 components of genetic algorithms for the traveling-salesman problem (TSP). In [12] the meta-evolutionary approach in searching for the best combination of crossover operators for TSP is also described. One of the major shortcomings of the meta-evolutionary approach is too large processing time.

Interesting work is presented in [8] where a parameter-less genetic algorithm is presented. The objective of their work is to provide robust and simple genetic algorithms (GAs) where users are relieved from control parameter settings despite the fact that peak performance can not be achieved. In [8] only selection rate, crossover probability and population size are taken into account. This is achieved by setting the selection rate (s) and crossover probability (p_c) to the predefined values ($s = 4$, $p_c = 0.5$) to avoid very high or very low selection pressure. In some sense, this approach is similar to the De Jong work [10] except that settings are not based on experiments, yet on sound theoretical work on schema theorem. The population size parameter is simply eliminated by iteratively running the GAs with different population size, doubling the population size each time the population converges or by establishing a race among populations of various sizes. However, the authors leave integration of mutation probability into their framework as future work. Another problem with

their approach is that specific problems most often require specific parameter settings.

3. PPCEA

In order to solve control parameter settings of EAs in programmable fashion, we have constructed an interpreter for PPCEA following the techniques of [13]. Figure 1 shows the framework of PPCEA. JLex [3] is a lexical analyzer generator for Java, and CUP (Constructor of Useful Parsers) [9] is the parser generator for Java. The cooperation of both JLex and CUP defines the grammar of PPCEA syntactically and semantically. This grammar consists of both EAs' specialized statements and common linguistic elements of programming languages. In addition, CUP helps to define how PPCEA's interpreter executes/interpretes the source code of PPCEA at runtime. Initializing from the bottom of the parse tree of PPCEA's source code, CUP traces up to check the syntax and execute the semantics. As CUP meets EAs' statements (e.g., call_EA), the interpreter processes the EA operations. As CUP encounters linguistic elements common to all programming languages (e.g., if-then-else statement), the interpreter executes the behaviors of these elements. Consequently, the users can acquire the results of control parameter settings of an EA by writing a PPCEA source code regarding desired control parameters. The typical scenario for choosing control parameters is as follows. From parameter tuning, intervals where control parameters produce best results are identified. In the selected intervals, control parameters are then adjusted using appropriate deterministic or adaptive rules expressed in a programmable fashion. Each run of the PPCEA program produces also diagrams that assist users in control parameter settings.

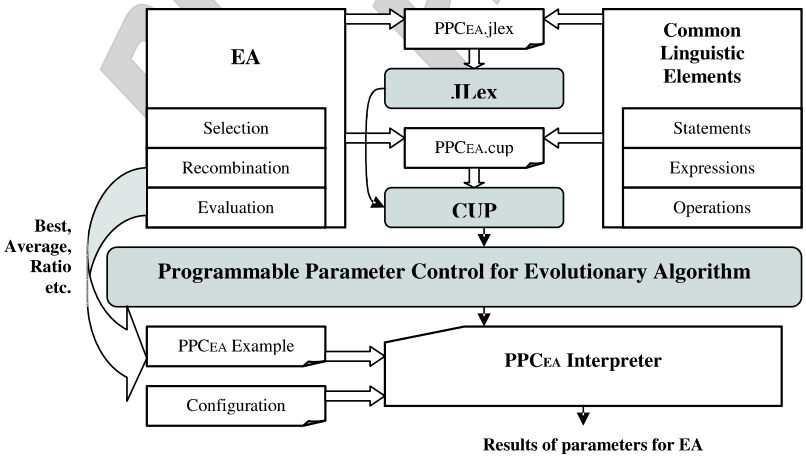

Figure 1. The framework of design and implementation of PPCEA.

PPCEA is in between the system programming language and scripting language [15]. PPCEA is plain, abstract and weakly-typed. Like many other scripting languages (e.g., shell script for UNIX), PPCEA includes simple expressions, control structures, extended functionality and runtime type checking. However, PPCEA uniquely focuses on the domain of EAs. The approach of writing a PPCEA source code for the interpreter, called metaprogramming, facilitates EAs to hide the implementation details and lift the adaptation of the control parameter settings up to the DSSL level. The major advantage of employing the metaprogramming approach to assess control parameter settings is that EA parameters can be evaluated dynamically by PPCEA's interpreter at runtime. For example, users are able to write if-then-else or while-loop statements to control the parameters at the DSSL level before or during the evaluation process.

In addition, PPCEA provides the configuration mechanism for a fitness function and parameter assignments. The fitness function in general is embedded into the evaluation statement of PPCEA. We use the file processing method to insert the fitness function into an EA. Therefore, users can easily alter the fitness function externally. On the other hand, the configuration mechanism for parameter assignments is equivalent to the predefined identifiers of PPCEA. The configuration mechanism and implementation of DSSL encapsulate the source code of EAs safely and reduce the trial-and-error approach efficiently.

Table 1. Initial values of parameters in the following examples

Parameter	*Value*	*Parameter*	*Value*	*Parameter*	*Value*
Maxgen	500	Popsize	50	Pmutation	0.1
Pxover	0.7	Epoch	10	t	0

4. Examples

We had applied the PPCEA approach to solve a routing problem [11] successfully. The routing problem determined the best route of a Mass Transit System from a set of roads. The best solution was decided by a fitness function within a limited budget. The fitness function was the sum of multiplication of weight and attribute (e.g., road length and construction cost) parameters. The PPCEA approach found the best route, which satisfied the constraint, from the roads, and appropriate control parameters. For simplicity, we provide a simpler algebraic fitness function. All examples are tested on the problem of the fitness function in [14],

$$f(x,y) = 0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{(1+0.001*(x^2+y^2))^2} \quad (1)$$

where $-100 \leq$ x, and y ≤ 100. The initial values of required parameters are in Table 1. However, Epoch is Maxgen in the example 4.1, and Pxover is 0

in examples 4.2 and 4.3. In the following examples, Pxover and Pmutation are the probability rates of crossover and mutation, respectively; Epoch is the counter variable that an EA is executed "Epoch" times without initialization of population; Ratio is the successful mutation ratio for 1/5 success rule [2]; Best and Average parameters are the maximum and average fitness values, respectively. For the sake of brevity, we omit the source code of the initialization of these parameters in the following examples.

4.1 Parameter Tuning

Let us start with a simple example where the best probability of crossover and mutation are searched in the predefined range. Values of parameters (Pmutation and Pxover) are not changed during the run of EA. This example corresponds to several and tedious runs of basic EAs. Figure 2 (parameter tuning) shows which parameters produce best results.

```
while (Pxover <= 0.9) do
    Pmutation := 0.1;
    while (Pmutation <= 0.2) do
        init; // initialize population
        call_EA; // run EA for one epoch
        writeresult;
        Pmutation := Pmutation + 0.01
    end;
    Pxover := Pxover + 0.05
end
```

4.2 Deterministic Control of Mutation Step

Deterministic parameter control employs certain deterministic rule to parameters. The example shows that the mutation step size is controlled deterministically by $P_m(t) = 1 - 0.9 * (t/T)$, where t is current generation number and T is maximum generation number.

```
init;
while (t <= Maxgen) do
    Pmutation := 1 - (0.9 * t)/Maxgen;
    call_EA; // run EA for one epoch
    writeresult;
    t := t + Epoch
end
```

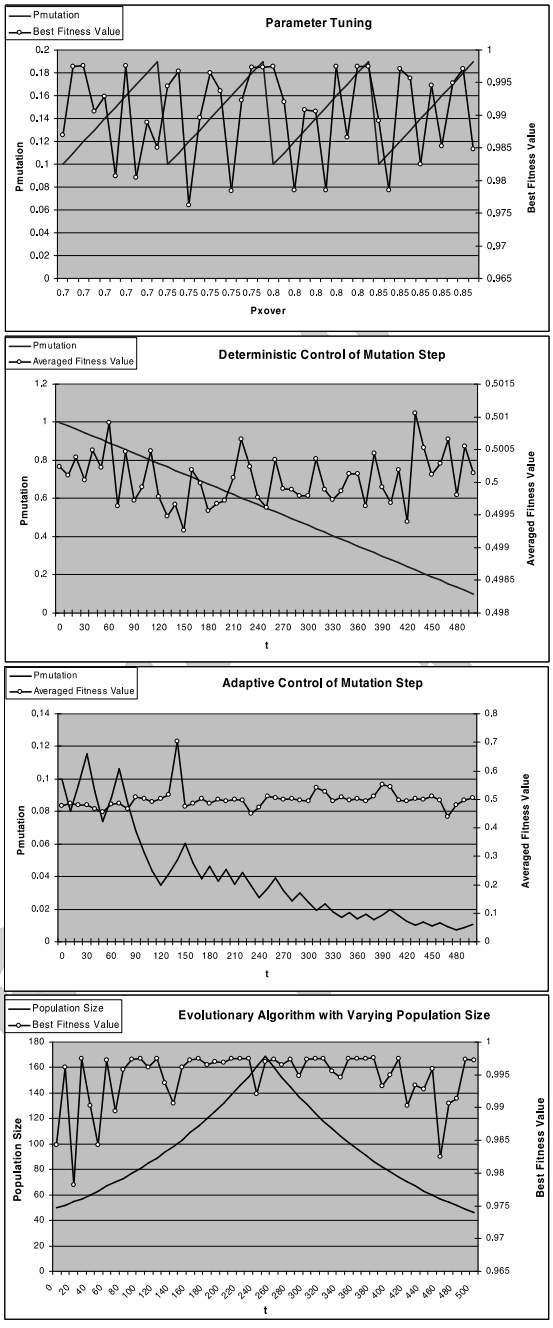

Figure 2. Experimental results of examples: Sect. 4.1, Sect. 4.2, Sect. 4.3, and Sect. 4.4.

4.3 Adaptive Control of Mutation Step

In adaptive control, the feedback from the evolutionary process controls the values of parameters. An example of simple adaptive control is 1/5 success rule [2] where the ratio of successful mutations to all mutations should be 1/5. If the ratio is greater (lower) then the mutation step should be increased (decreased). The example shows that PPCEA lifts the 1/5 success rule up to the DSSL level. Pmutation is controlled adaptively according to the successful mutation ratio.

```
init;
while (t <= Maxgen) do
   call_EA;
   writeresult;
   if (Ratio > 0.2) then
      Pmutation := Pmutation * 1.2
   else
      Pmutation := Pmutation * 0.8
   fi;
   t := t + Epoch
end
```

Self adaptation of the mutation step is also possible. In this case the mutation step size is encoded into the chromosome and as such goes through evolution. Therefore self adaptation is not at the level of PPCEA, but at the level of basic EA. However, the meta-evolutionary approach can be expressed with PPCEA.

4.4 Evolutionary Algorithm with Varying Population Size

In [14, 17], an EA with varying population size is described where the concept of the chromosome age has been introduced. The similar effect can be achieved with the following program in PPCEA. The example shows that Popsize is determined by the relationship of the best and averaged results of an EA. We eliminate the worst chromosomes by sorting if the new Popsize is less than the current one. Yet new members will be generated randomly if the new Popsize is larger than the present one. A more sophisticated formula for Popsize can be programmed exploiting best and average fitness values as well.

```
init;
while (t <= Maxgen) do
    call_EA;
    writeresult;
    if (Best < Average*Maxgen/(t+1)) then
        Popsize := Popsize * 1.05
    else
        Popsize := Popsize * 0.95
    fi;
    t := t + Epoch
end
```

Figure 2 shows the experimental results for all examples. Each figure includes average or best values of the fitness function. Necessary parameters such as Pmutation, Pxover, and Popsize also appear in the figures to assist users with deciding appropriate control parameter settings. From Fig. 2, users can easily acquire the best control parameter setting by analyzing the diagrams. For example 4.1, there are various combinations of Pxover and Pmutation to obtain the best fitness value. Example 4.2 exhibits that the best averaged fitness value is obtained when Pmutation is 0.23 at 430^{th} generation. For example 4.3, EA computes the highest averaged fitness value where Pmutation and generation are about 0.05 and 130, respectively. Example 4.4 reveals that diverse population sizes affect the best fitness value at varied stages.

The main advantage of our approach is that the problem of control parameter settings is lifted to the higher abstraction layer while keeping an EA simple as before. All control parameters are treated uniformly (e.g. the probability of crossover, population size, epoch), hence achieving a good flexibility of the proposed approach.

5. Conclusion

PPCEA applies the advantages of domain-specific languages and scripting languages to solve the long-lasting control parameter setting problems. A domain-specific language provides the abstract and repeatable features, since a simple EA evaluation statement (call_EA) covers obligatory selection, recombination and evaluation processes. On the other hand, a scripting language endows EAs with adaptable and controllable characteristics. Therefore, the joint features of PPCEA empower the usage and performance of EAs. PPCEA not only outperforms the trial-and-error approach, but also performs the adaptable, reusable and controllable solutions of control parameter settings for EAs in parameter tuning, deterministic, and adaptive aspects.

References

[1] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS - a Genetic Algorithm with Varying Population Size. In Proceedings of the 1^{st} IEEE Intl. Conf. on Evolutionary Computation, Orlando, Florida, 1994, pp. 73–78.

[2] T. Bäck and H.-P. Schwefel. *Evolution Strategies i: Variants and Their Computational Implementation*. Genetic Algorithms in Engineering and Computer Science. Editor: J. Périaux and G. Winter, John Wiley & Sons Ltd., Chichester, 1995.

[3] E. Berk. *JLex: A lexical analyzer generator for Java*, 1997. http://www.cs.princeton.edu/∼appel/modern/java/JLex/.

[4] A.J. Chipperfield and P.J. Fleming. *Genetic Algorithm Toolbox User's Guide*, 1995. http://www.shef.ac.uk/∼ gaipp/ga-toolbox/manual.pdf

[5] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Transascition on Evolutionary Computation*, 3:124–141, 1999.

[6] B. Freisleben and M. Härtfelder. In Search of the Best Genetic Algorithm for the Traveling Salesman Problem. In *Proceedings of* 9^{th} *International Conference on Control Systems and Computer Science*, Bucharest, Romania, 1993, pp. 485-493.

[7] J.J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transaction on Systems, Man & Cybernetics*, 16:122-128, 1986.

[8] G.R. Harik and F.G. Lobo. A parameter-less genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conf. GECCO '99*, Orlando, Florida, 1999, Vol. 1, pp. 258–265.

[9] S.E. Hudson. *CUP parser generator for Java*. 1999. http://www.cs.princeton.edu/∼appel/modern/java/CUP/

[10] K. De Jong. The Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, Department of Computer Science, University of Michigan, Ann Arbor, Michigan, 1975.

[11] S.-H. Liu. *Parameterized Genetic Algorithm*. Domain-specific Language: Project Report, 2004. http://www.cis.uab.edu/liush/DSL/final_project.pdf

[12] M. Mernik, M. Črepinšek, and V. Žumer. A Meta-evolutionary Approach in Searching of the Best Combination of Crossover for the TSP. In *Proceedings of Neural Networks NN '2000*, Pittsburgh, USA, 2000, pp. 32–35.

[13] M. Mernik, J. Heering, and A.M. Sloane. When and How to Develop Domain-Specific Languages. CWI Technical Report, SEN-E0309, 2003.

[14] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. 3^{rd} Edition. Springer-Verlag, 1996.

[15] J.K. Ousterhout. Scripting: Higher Level Programming for 21^{st} Century. *IEEE Computer*, 31(3):23-30, 1998.

[16] J.D. Schaffer, R.A. Caruana, Eshelman, and R. Das. A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In *Proceedings of the* 3^{rd} *Intl. Conf. on Genetic Algorithms*, Fairfax, Virginia, 1989, pp. 51–60.

[17] R. Smith and E. Smuda. Adaptively Resizing Populations: An Algorithm, Analysis, and First Results. *Complex Systems*, 1(9):47–72, 1995.