Learning Control Strategies for High Performance Genetic Algorithms

Michael A. LEE
Computer Science Division, University of California, Berkeley

高木, 英行
Central Research Laboratories, Matsushita Electric Industrial Co., Ltd.

出版情報：Architecture and Applications for Knowledge Acquisition/Adaptation, pp.93-96, 1994-08-09. IEEE
バージョン：
権利関係：

Learning Control Strategies for High Performance Genetic Algorithms[1]

Michael A. LEE
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
lee@cnmat.berkeley.edu

Hideyuki TAKAGI
Central Research Laboratories
Matsushita Electric Industrial Co., Ltd.
3-4, Hikaridai, Seika
Souraku, Kyoto 619-02 Japan
htakagi@crl.mei.jp

Abstract - In this paper, we propose a method to learn high performance strategies for controlling genetic algorithms. In our proposal, control strategies are represented by fuzzy systems that dynamically control population sizing, crossover rates, and mutation rates. The control strategies are acquired and optimized according to online and offline measures using a genetic algorithm technique. We compare control strategies obtained using our methods with optimized static genetic algorithms and show performance improvements. In some experiments, these strategies use a combination of high crossover rates, fluctuating population size, and exponentially decreasing mutation rates to realize high online and offline performance.

1 Introduction

The relationship between genetic algorithm performance and genetic algorithm parameter settings, such as population size and crossover rates, is complex and has been the topic of many recent publications [1-9,12-14]. In an effort to further our understanding of these issues and to realize high performance genetic algorithms, we proposed the Dynamic Parametric GA; a genetic algorithm that uses a fuzzy knowledge-based system to control genetic algorithm parameters dynamically, such as population size, crossover rates, and mutation rates in a genetic algorithm (see Figure 1) [12,13]. In that work, we also proposed acquiring control strategies using the automatic fuzzy design technique proposed in [11]. The combination of these two approaches yielded a method to automatically discover genetic algorithms exhibiting high online and offline performance. In this paper we further compare results of acquired control strategies and reveal the dynamic nature of these automatically obtained strategies. Section 2 reviews the Dynamic Parametric GA framework and our automatic design technique. Section 3 presents results for online and offline strategies. Section 4 concludes and gives areas for further extensions.

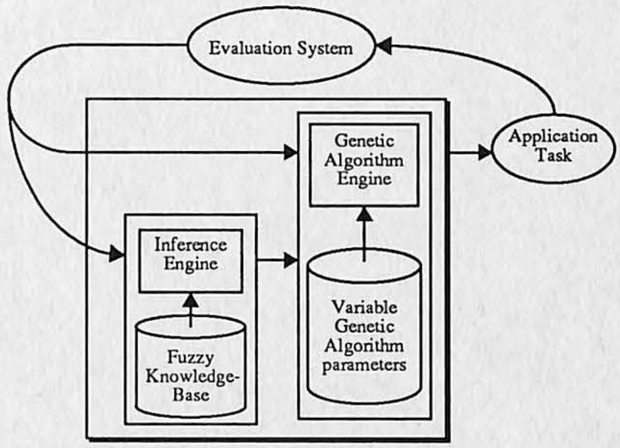

Figure 1. Dynamic Parametric GA: a genetic algorithm with dynamic parameters controlled by a fuzzy knowledge-based system. The fuzzy knowledge based system monitors performance measures from the evaluation system to control genetic algorithm parameters such as population size, mutation rate, or crossover rate.

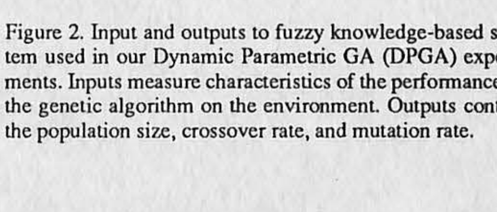

Figure 2. Input and outputs to fuzzy knowledge-based system used in our Dynamic Parametric GA (DPGA) experiments. Inputs measure characteristics of the performance of the genetic algorithm on the environment. Outputs control the population size, crossover rate, and mutation rate.

2 The Dynamic Parametric GA

2.1 DPGA Framework

Inputs to the fuzzy knowledge-based system for genetic algorithm control can be any combination of genetic algorithm performance measures or current control settings, and outputs can be any of the genetic algorithm control parameters, such as population size or mutation rate. Typical inputs might be population diversity measures such as the ratio of average fitness to best fitness, current population size, current mutation rate, or fitness variance. Rules in the fuzzy knowledge-based system reason about these measures and prescribe some control action. Typical rules in the knowledge base may include the following:

IF (average fitness)/(best fitness) is *big* THEN population size should *increase*.
IF (worst fitness)/(average fitness) is *small* THEN population size should *decrease*.

1. This paper to appear in the Proc. of 1994 IEEE/Nagoya University World Wisemen/Women Workshop (WWW'94), Nagoya, Aichi, Japan (August 9-10, 1994).

2.2 Learning GA Control Strategies

In our experiments, we automatically design DPGAs with inputs and outputs as shown in Figure 2. The initial population size, crossover rate, and mutation rate as well as the membership function parameters, rule consequents, and the number of fuzzy rules of the fuzzy knowledge base represent the parameters of a control strategy. The ranges of the outputs were set such that the population size change could not change by more than half the current population size and could not go below 2 or exceed 160. The crossover and mutation parameters were also restricted to change at most by half their current value and were bounded by [0.0001,1.0]. For example, a value of 1.5 for population size change would increase the size of the population by 50% of its current size up to a maximum value of 160. The fuzzy system uses triangular membership functions, the *min* intersection operator and correlation-product inference procedure. Defuzzification of the outputs is performed using the fuzzy centroid method [10]. Each of the system parameters is encoded as part of a concatenated binary string that is operated on by another, meta-level, genetic algorithm (see [13] for details on fuzzy system design using genetic algorithms).

The control strategies learned in our experiments were optimized according to performance measures and a five function test suite designed by DeJong [3]: *online* performance to measure ongoing performance and *offline* performance to measure convergence. Online performance is the running average of all evaluations performed up to a given time and may be appropriate in situations where the cost of evaluating a structure is related monotonically increasing to its fitness value (i.e., evaluating a poor solution is more expensive than evaluating a good one). Offline performance is the running average of the best performance value up to a given time and may be appropriate when there is no additional cost for evaluating poor structures. Both equations are given below:

$$x_{online}(s, e, T) = \frac{1}{T}\sum_{t=1}^{T} f_e(t)$$

$$x_{offline}(s, e, T) = \frac{1}{T}\sum_{t=1}^{T} f_e^*(t)$$

where s is the search strategy, e is the environment, $f_e(t)$ is the objective function value at time t, and $f_e^*(t)$ is the best function value obtained up to time t and T is the current number of evaluations.

3 Results

We design a separate Dynamic Parametric GAs for optimizing online and offline performance measures. In this section, we will look at the dynamic behavior of the DPGAs and compare the results with a simple static genetic algorithm proposed by DeJong (SSGA)[3], the optimized static online and offline genetic algorithms proposed by Grefenstette (OSGA)[7], and random search (see Table 1 for GA parameter settings).

Table 1: Genetic algorithm search parameter settings for simple static GA (SSGA), optimized static GA for online performance (OSGA online), and optimized static GA for offline performance (OSGA offline).

parameter	SSGA	OSGA online	OSGA offline
population size	50	30	80
crossover rate	0.6	0.95	0.45
mutation rate	0.001	0.01	0.01
generation gap	1.0	1.0	0.9
window size	7	1	1
selection strategy	Elite	Elite	Pure

The genetic algorithm used to design Dynamic Parametric GAs itself had fixed parameters of population size=10, crossover rate=0.8, mutation rate=0.0333. It used an elitist selection strategy and window sizes and generation gaps were fixed at 1 and 1.0 respectively. This genetic algorithm was allowed to evaluate 1000 Dynamic Parametric GAs.

3.1 Online Performance

The initial population of fuzzy systems and initial conditions used for determining the good online performance was seeded with an individual with static settings, i.e. no rules, as prescribed by 'OSGA online' given in Table 1. For each fuzzy system produced for the Dynamic Parametric GA, the generation gap, window size, and selection strategy were fixed at 1.0, 1, and Elitist. After evaluating 665 fuzzy systems, the meta-level genetic algorithm produced a fuzzy system with 59 rules and the following initial conditions[13]:

Initial Population Size:	10
Initial Crossover Rate:	0.942647
Initial Mutation Rate:	0.009903

The left plot of Figure 3 shows the online performance vs. evaluations for the DPGA, OSGA online, SSGA, and random search for DeJong Function 3. The data in the figure is averaged from running each GA with ten different initial conditions.

The plots on the right show the dynamic control of the population sizing, crossover rate, and the mutation rate for a typical run on DeJong Function 3. Both the population size and mutation rate decrease toward the minimum value while the crossover rate remains high. The strategy that this particular DPGA has chosen is a conservative approach. Because the elite selection strategy is enabled and the population size goes to two, the search becomes a greedy hill climber. A good solution is not abandoned until a better one is found. In addition, the low mutation rate keeps the exploration relatively local.

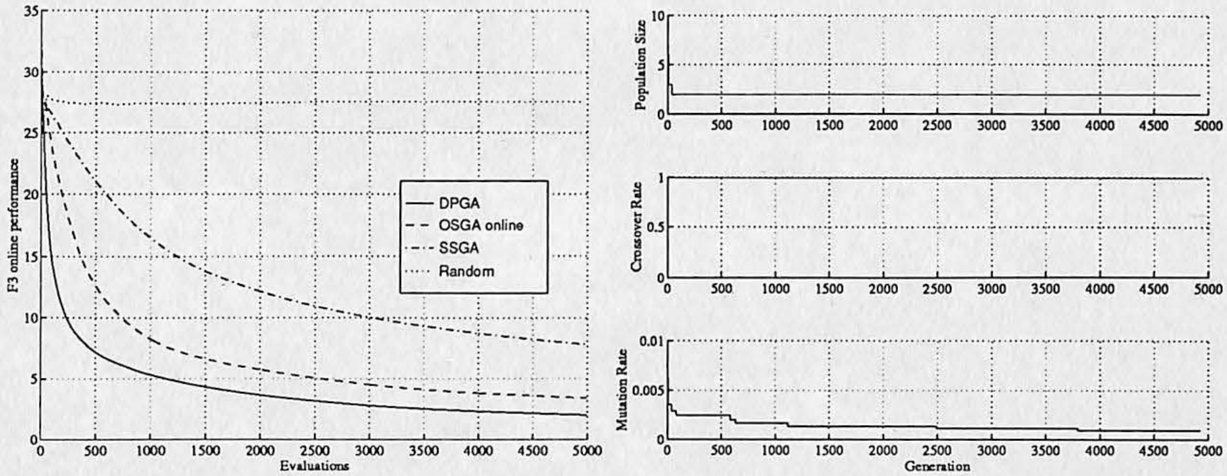

Figure 3. Online performance of the Dynamic Parametric GA (DPGA) optimized for online performance on DeJong Function 3. Also shown are the online performance measures of the optimized static GA (OSGA online), simple static GA (SSGA), and random search (see Table 1 for GA search parameters). The plots on the right show the dynamic control of the population size, crossover rate, and mutation rate for a typical DPGA run on DeJong Function 3.

3.2 Offline Performance

As with the online search, the initial population of fuzzy systems and initial conditions used for determining the good offline performance was seeded with an individual identical with static settings, i.e. no rules, as prescribed by 'OSGA offline' given in Table 1. For each fuzzy system produced for the Dynamic Parametric GA, the generation gap, window size, and selection strategy were fixed at 0.9, 1, and Pure. After evaluating 373 fuzzy systems, the meta-level genetic algorithm produced a fuzzy system with 68 rules and the following initial conditions[13]:

Initial Population Size:	4
Initial Crossover Rate:	0.922059
Initial Mutation Rate:	0.170671

Figure 4 shows the offline performance vs. evaluations for the DPGA, OSGA offline, SSGA, and random search for DeJong Function 3 (as with the online data, this data is averaged over ten runs).The plots on the right show the dynamic control of the population sizing, crossover rate, and the mutation rate. As in the online control strategy, the mutation rate decrease toward the minimum value while the crossover rate remains high. However, the population size increases toward the maximum value. As the number of evaluations increases, random search becomes more difficult to out-perform. Although we expected the mutation rate to increase over time (a move toward random search behavior) we found that the control strategy relied more on the crossover operator than the mutation as it continued its search.

4 Conclusions and Further Research

We have demonstrated a method to acquire and optimize genetic algorithm control strategies. The control strategies are represented using fuzzy systems and were automatically designed using a GA technique. These control strategies show improved performance over simple static and optimized static GAs. We have also shown the dynamical behavior of GA control strategies optimized for online and offline performance. Both the online and offline strategies prescribe high crossover rates and exponentially decreasing mutation rates.

We have performed additional experiments that deactivated the control of the mutation rate and population sizing and found that the combination of both are required to achieve the high performance exhibited when they are both active. We also found that the exponential decreasing mutation rate has a stronger effect than that of increasing or decreasing the population size. A further discussion of these behaviors is beyond the scope of this paper and will be addressed in future research.

By using an automatic technique, we gain the possibility of discovering new relations, which in turn, may offer insight to understanding the complex interaction between genetic algorithm control parameters and genetic algorithm performance. We would like to emphasize that the experimental results we report have been obtained for a specific instance of the class of Dynamic Parametric GAs; our technique can be applied to systems with other inputs and outputs. Research on eliminating useless rules, and determining relevant input variables should be explored and more analysis needs to be performed on the resulting systems. In addition, search performance metrics other than the offline and online measures used in this paper warrant investigation.

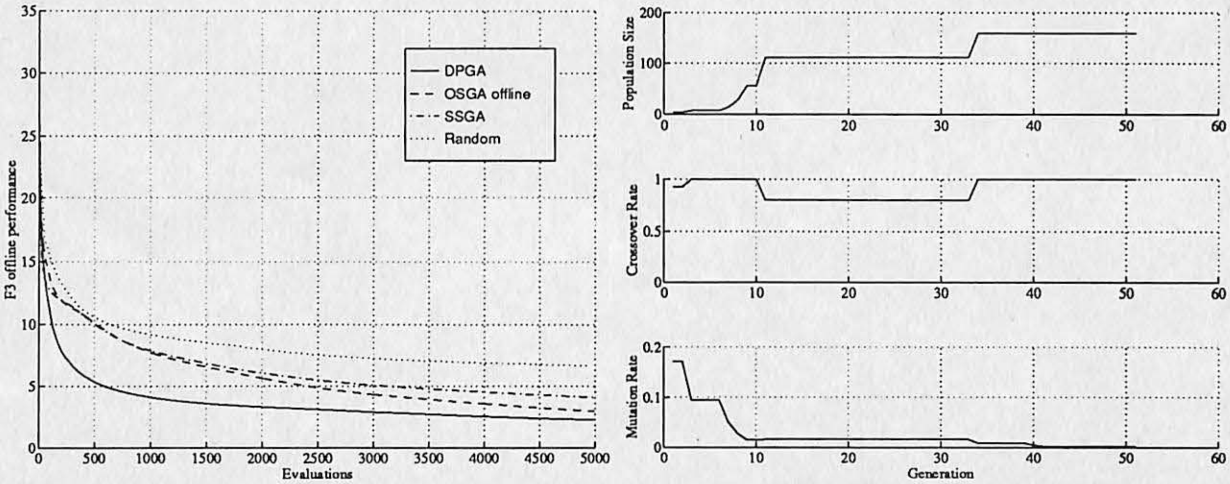

Figure 4. Offline performance of the Dynamic Parametric GA (DPGA) optimized for offline performance on DeJong Function 3. Also shown are the offline performance measures of the optimized static GA (OSGA offline), simple static GA (SSGA), and random search (see Table 1 for GA search parameters). The plots on the right show the dynamic control of the population size, crossover rate, and mutation rate for a typical DPGA run on DeJong Function 3.

Acknowledgments

This research is supported in part by NASA Grant NCC-2-275, MICRO State Program Award No.90-191, EPRI Agreement RP8010-34, and BISC program. The authors would also like to thank Prof. David Wessel and the Center for New Music and Audio Technologies at UC Berkeley and Silicon Graphics Inc. for use of computing resources.

References

[1] Back, T., "Self-adaptation in genetic algorithms," *Proceedings of the First European Conference on Artificial Life*, Paris, France, 1991, pp.263-271.

[2] Baker, J. E., "Adaptive selection method for genetic algorithms," *Proceedings of an International Conference on Genetic Algorithms* (ICGA'85), Pittsburgh, PA, 1985, pp.101-111.

[3] DeJong, K. A., *An analysis of the behavior of a class of genetic adaptive systems*, Ph.D. Thesis, University of Michigan, 1975.

[4] DeJong, K. A. and Spears, W. M., "An analysis of interacting roles of population size and crossover in genetic algorithms," *Parallel Problem Solving from Nature. 1st Workshop, PPSN 1 Proceedings*, Dortmund, West Germany, 1990, pp.38-47.

[5] Fogarty, T. C., "Varying the probability of mutation in the genetic algorithm," *Proceedings of the Third International Conference on Genetic Algorithms* (ICGA'89), Arlington, VA, 1989, pp.104-109.

[6] Goldberg, D. E., Deb, K. and Clark, J. H., "Genetic algorithms, noise, and the sizing of populations," *Complex Systems*, Vol.6, No.4, 1992, pp. 333-362.

[7] Grefenstette, J. J., "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.16, No.1, 1986, pp. 122-128.

[8] Hesser, J. and Manner, R., "Towards an optimal mutation probability for genetic algorithms," *Parallel Problem Solving from Nature. 1st Workshop, PPSN 1 Proceedings*, Dortmund, West Germany, 1990, pp.23-32.

[9] Holland, J. H., *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, 1975.

[10] Kosko, B., *Neural Networks and Fuzzy Systems*, Addison-Wesley, Englewood Cliffs, NJ, 1992.

[11] Lee, M. A. and Takagi, H., "Integrating design stages of fuzzy systems using genetic algorithms," *Proc. IEEE Int. Conf. on Fuzzy Systems* (FUZZ-IEEE '93), San Francisco, CA, 1993, pp.612-617.

[12] Lee, M. A. and Takagi, H., "Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques," *Proceedings of the Fifth International Conference on Genetic Algorithms* (ICGA'93), Urbana-Champaign, IL, 1993, pp.76-83.

[13] Lee, M.A., *Automatic Design and Adaptation of Fuzzy Systems and Genetic Algorithms using Soft Computing Techniques*, Ph.D. Thesis, University of California, Davis, 1994.

[14] Schaffer, J. D., Caruana, R. A., Eshelman, L. J. and Das, R., "A study of control parameters affecting online performance of genetic algorithms for function optimization," *Proceedings of the Third International Conference on Genetic Algorithms* (ICGA'89), Arlington, VA, 1989, pp.51-60.