A Hybrid Approach to Parameter Tuning in Genetic Algorithms

Bo Yuan
School of Information Technology and Electrical Engineering
The University of Queensland
QLD 4072, Australia
boyuan@itee.uq.edu.au

Marcus Gallagher
School of Information Technology and Electrical Engineering
The University of Queensland
QLD 4072, Australia
marcusg@itee.uq.edu.au

Abstract- Choosing the best parameter setting is a well-known important and challenging task in Evolutionary Algorithms (EAs). As one of the earliest parameter tuning techniques, the Meta-EA approach regards each parameter as a variable and the performance of algorithm as the fitness value and conducts searching on this landscape using various genetic operators. However, there are some inherent issues in this method. For example, some algorithm parameters are generally not searchable because it is difficult to define any sensible distance metric on them. In this paper, a novel approach is proposed by combining the Meta-EA approach with a method called Racing, which is based on the statistical analysis of algorithm performance with different parameter settings. A series of experiments are conducted to show the reliability and efficiency of this hybrid approach in tuning Genetic Algorithms (GAs) on two benchmark problems.

1 Introduction

Evolutionary Algorithms (EAs) refer to a broad class of optimization algorithms under a unified framework. The total number of algorithms and their variations is very large and in practice a number of choices such as the crossover rate, the type of crossover and the population size etc. must be made in advance in order to fully specify a complete EA. The importance of these choices is at least threefold. Firstly, EAs are not completely parameter robust and EAs with inappropriate choices may not be able to solve problems effectively[10]. Secondly, when two or more EAs are compared, arbitrarily specified parameters may make the comparison unfair and conclusions misleading. Finally, finding the optimal setting may be also helpful for better understanding the mechanisms of EAs.

There has been a large amount work dedicated to finding the "optimal" parameters of EAs [6, 7]. However, it has shown that this kind of optimal setting does not exist in general. In fact, for different problems, there are different optimal specifications[17]. If each algorithm parameter is regarded as a variable and the performance of the EA is regarded as the objective value, there is a performance landscape of EAs on each problem and finding the best parameter setting simply means finding the global optimum.

There are a number of possible approaches to selecting algorithm parameters based on classical statistical methods such as Response Surfaces and Regression[14]. One major advantage is that by explicitly building a model of the relationship between algorithm parameters and the algorithm's performance, a deep insight into the interaction among parameters can be achieved. However, these methods are not efficient in terms of searching complex performance landscapes because in order to build a full model involving many variables, an extremely large number of sampling points are needed. On the other hand, building a local model using a small number of sampling points is equal to doing local search and may drive the tuning process to a local optimum. Please refer to [2]for detailed introduction and references.

An alternative approach to parameter selection is to employ another EA, called a Meta-EA[1, 9], to conduct searching because EAs are believed to be able to search complex landscapes efficiently. Traditionally, in the Meta-EA level, each individual contains all the necessary algorithm parameters and thus fully specifies an EA. The fitness of each individual is evaluated by running the EA on some test problems for a few trials. Results have shown that this Meta-EA approach could successfully find algorithm parameters that are better than default values recommended in the literature.

However, there are also some issues with the Meta-EA approach. Firstly, not all algorithm parameters are searchable by EAs. For example, suppose that there is a variable specifying the type of selection. Since it is not obvious how to define a reasonable distance metric along this dimension (i.e., the distance between each pair of selection operators), it may be very difficult to design appropriate search operators other than uniform/random search. Secondly, there may be also some variables that only require specification in certain situations. For example, suppose that there is a variable specifying the tournament size in Tournament Selection. However, when the current selection method is Truncation Selection, this variable is not applicable. In other words, the representation itself is not very efficient. Finally, this Meta-EA approach is usually very time-consuming because in order to evaluate each individual, the corresponding EA must be run for a large number of trials with different initial conditions due to the stochastic behaviour of these algorithms.

In this paper, we address the above issues by combining a statistical method called Racing [13] with the Meta-EA approach. Racing is originally proposed as a searching method to efficiently identify the best performing learning algorithm from a set of candidates and has also been applied in evaluating EAs[3, 18]. Usually, it only requires a small fraction of the cost of brute force methods as it could quickly identify weak candidates and remove them from the experiment based on statistical tests. It is particularly useful when the evaluation process of each algorithm contains a number of independent trials, which could be regarded as random samples from the unknown performance distribution. The major advantage of Racing is that it is independent of the internal structure of those algorithms, which means that it could be used to compare algorithms belonging to significantly different categories where no distance metric is available.

Note that, in parameter tuning, algorithm parameters are chosen in advance and remain fixed during evolution. Another class of approaches is called parameter control[7] in which those parameters are subject to evolution as the problem parameters are optimized. A typical example is the self-adaptive standard deviation values in Evolution Strategies [15]. The advantage is that different parameter values may be needed in different stages of evolution in order to achieve optimal performance. However, there are still some exogenous parameters used to control those self-adaptive parameters. Also, not all algorithm parameters can be changed in this manner. Furthermore, finding the optimal settings of EAs in different situations may also provide valuable information for designing better parameter control strategies.

The remainder of this paper is organized as follows. Section 2 gives the details of two benchmark problems and the Genetic Algorithm whose parameters are to be tuned. Section 3 introduces the framework of Meta-EA and Racing and shows how to combine these methods together. Experimental results are presented in Section 4 and this paper is concluded in Section 5.

2 Genetic Algorithms

A classical Genetic Algorithm [8, 11] is used in this paper, which is specified in Figure 1 (i.e., there are some other versions of GAs with more or less difference). Two parents from the current population are selected at each time and two offspring are generated through recombination of these two parents with probability P_c (the crossover rate). Otherwise, these two parents are kept unchanged and copied to the mating pool. When the mating pool is full, mutation is applied, which changes the value of each variable with probability P_m. For variables of binary problems, which are the case of this paper, mutation is applied by flipping a bit from 0 to 1 and vice versa. Finally, new individuals are evaluated and replace the old population. If elitism is applied, the best individual found so far in previous generations will be copied to the new population, replacing a randomly chosen individual.

```
Initialize and evaluate the population
 While stopping criteria not met
   While mating pool is not full
      Select two individuals
      Apply crossover with probability Pc
      Copy offspring to mating pool
   End While
   Apply mutation with probability Pm
   Evaluate new individuals
   Replace old population
 End While
```

Figure 1. The Framework of the Genetic Algorithm

Obviously, there are a number of parameters to be specified in the above framework. The population size (P) is a discrete parameter of high cardinality (i.e., it should be a positive even number because individuals are generated in pairs). It is well-known that a very small population may result in premature convergence while a very large population may result in a slow convergence rate. The crossover rate P_c and mutation rate P_m are both continuous variables within [0, 1]. These two values are important for controlling the balance between exploration and exploitation. The selection strategy (S) is a discrete variable, which contains various candidates (e.g., Tournament Selection) combined with their corresponding parameters (e.g., Tournament size). Note that it is also possible to use two parameters with one specifying the type and the other specifying the parameter. Please refer to [4] for detailed analysis and comparison of various selection strategies. Also, a discrete parameter (C) is required to specify the type of crossover (e.g., one-point or two-point). At last, a binary parameter (E) is used to indicate whether elitism is used or not.

According to the above discussion, a GA could be fully specified by a six-element tuple: <P, P_c, P_m, S, C, E>. Among those algorithm parameters, some are continuous (i.e., the population size could be regarded as a continuous parameter during optimization due to its high cardinality and then rounded up to the nearest feasible value) while others are discrete, which creates a mixed-value optimization problem. The feasible values of S, C & E are given in Table 1.

For example, there are three selection strategies: “Truncation”, “Tournament” and “Linear Ranking” and there are also 5, 3 and 3 parameter values associated with each of them respectively, creating 5+3+3=11 possible selection strategies in total.

	Values
S	“Truncation” {0.1, 0.2, 0.3, 0.4, 0.5} “Tournament” {2, 4, 6} “Linear Ranking” {1.5, 1.8, 2.0}
C	“One-Point”, “Two-Point”, “Uniform”
E	0 (without elitism), 1(with elitism)

Table 1. Feasible Values of S, C & E

Two well-known binary optimization problems are used in this paper. The first test problem is the One-Max problem whose fitness value is simply the number of 1s in each individual. It has a very simple structure in that there is no dependence among bits. Another test problem is the HIFF (Hierarchical-if-and-only-if) problem[16], which is much more challenging:

$$f(B) = \begin{cases} 1, & if\ |B|=1 \\ f(B_L)+f(B_R)+|B|, & if\ \forall i\{b_i = 0\ or\ b_i = 1\} \\ f(B_L)+f(B_R), & otherwise \end{cases} \quad (1)$$

In Eq.1, B is a block of bits $\{b_1, \ldots, b_n\}$ and |B| is the length of B (n). B_L and B_R are the left and right half-strings respectively (i.e., the length of the string must be an integer power of 2). This function regards the string as a binary tree and recursively decomposes it into left and right sub-strings. Each sub-string is a building block in a certain level and if all the bits in the sub-string are of the same value (i.e., all ones or all zeros), it is rewarded as the best block in its level. In the HIFF problem, two different best building blocks at one level can create a second best block at the next level when they are combined together, which makes it a difficult problem for GAs.

3 Meta-EA and Racing

3.1 Meta-EA

As discussed in Section 2, each GA corresponds to an instance of the six-element parameter tuple and finding the best parameter setting means finding the best value assignment through searching in this 6D solution space. Certainly, it is possible that two or more parameter settings may produce very similar performance. In the Meta-EA approach[1, 9], each element is treated as a tunable variable and each individual contains all such variables and thus represents a complete candidate GA. The fitness value of each individual is evaluated by decoding it into its corresponding GA and running this GA on the test problem(s) for a number of trials. In fact, the only difference between Meta-EAs and other EAs is that individuals represent parameters of EAs in Meta-EAs while they represent candidate solutions of the optimization problem in EAs. In practice, the algorithm used as the Meta-EA could be any EAs such as a GA [9] or a specifically designed EA [1].

One of the major difficulties in applying this Meta-EA approach is that some algorithm parameters are not searchable in general. A common feature of numerous EAs is that they all assume that the problem to be solved has some kind of searchable structure. A searchable space means that it is possible to measure the distance (similarity) between any two candidates so that a sensible search space (landscape) could be defined. In other words, for any candidate, it should be possible to find out which candidates are close to it and which candidates are far from it. Otherwise, it would be very difficult to apply genetic operators such as crossover and mutation. In this paper, S is a typical example of such non-searchable parameters (e.g., it is not easy to say whether Tournament Selection with size 2 or Linear Ranking Selection with maximum number of offspring 1.8 is closer to Truncation Selection with selection ratio 0.3). Previously, this kind of parameters are coded into binary strings in an arbitrary way [9] or only mutation is applied on them, which actually does pure random search[1].

Since random search is typically not an efficient and/or reliable method, one solution is to, instead of searching, use a brute force approach to enumerate all possibilities. For example, when it comes to evaluating an individual, which contains non-searchable parameters, it will first be expanded to a set of individuals containing all possible combinations of the values of those non-searchable parameters (i.e., all these individuals are identical in terms of other searchable parameters). All individuals are evaluated and the highest fitness value is returned as the fitness value of the original individual. The obvious advantage is that this method could guarantee to find the best values of non-searchable parameters, given the values of other parameters. However, when the number and/or the cardinality of the parameters are large, it could be a tedious process to fully evaluate all possible algorithms. After all, several trials are needed to get reliable results.

3.2 Racing

Racing [12, 13]is originally proposed to solve the model selection problem in Machine Learning: given a set of data points and a number of candidate learning algorithms (which could include multiple versions of the same algorithm with different, specified parameter values or algorithms belonging to different classes), which algorithm yields the minimum prediction error? In contrast to the brute force method, which is to sequentially evaluate all algorithms on all available data points and choose the best performing algorithm, the Racing method investigates all algorithms in parallel (See Figure 2).

Repeat the following steps until only one candidate left or no unseen data point

- Randomly select a new data point
- Test all available candidates on it
- Record corresponding results
- Apply statistical test
- Eliminate candidates (if any) worse than others at a predefined significance level

Figure 2. The Framework of Racing

In each step, all algorithms are tested on a single independently selected data point and their prediction errors on that point are calculated. The mean predication error of each algorithm on data points that have already been seen is also maintained. This error, E_{est}, is an estimation of the true prediction error E_{true} over the entire data set. As the algorithms are tested on more and more

data points, E_{est} approaches E_{true}. The fundamental mechanism of Racing attempts to identify and eliminate weak candidates on the basis of E_{est} as early as possible to minimize the number of unnecessary predication queries. In other words, candidates compete with each other for computational resources and only promising candidates survive to undertake further testing.

Racing can be also applied to EAs [3, 18] due to the similarity between the model selection problem in Machine Learning and the task of parameter tuning in EAs. In each case, the user is faced with a meta-optimization problem, which is to efficiently find values for all of the adjustable parameters of the model (algorithm) in order to produce the best results. Furthermore, the performance of each model (algorithm) needs to be evaluated based on a number of data points (trials). Since the Racing approach only utilizes the statistics of the results generated by a set of algorithms instead of their internal structure, it is very useful when the algorithm space is not searchable.

A closer look at Racing reveals that there is an inherent lack of any mechanism of exploration. In fact, Racing always starts with a fixed set of candidates and no other candidates can be reached. From this point of view, it is more like a selection method than a searching method. If Racing is to be directly applied to the parameter tuning task, the best algorithm must be within the initial set. Otherwise, there is no hope to find it by Racing.

Unfortunately, there is usually no such guarantee in practice. One approach is to use an exhaustive initial set containing all possible algorithms. However, for algorithms with continuous parameters, the total number of algorithm instances is numerous, which makes Racing impractical. Certainly, it is possible to discretize those continuous parameters but the accuracy of searching will inevitably suffer, especially if the algorithm's performance is sensitive to the values of these parameters and there is not enough *prior* knowledge about the value range.

3.3 Hybridization of Meta-EA and Racing

In Sections 3.1 & 3.2, we gave an introduction of two different approaches to parameter tuning: Meta-EA and Racing and pointed out their strengths and weaknesses. The Meta-EA has global optimization ability but suffers from the dilemma of non-searchable parameters. Note that it is certainly possible to encode these parameters one way or another but it is unlikely that the encoding may correctly reflect the underlying structure. By contrast, Racing could handle this issue without any difficulty but its performance is strongly dependent on the initial set of candidates. It is clear that the properties of these two approaches are complimentary to each other and better performance might by achieved by combining both approaches.

The general idea is to only encode those searchable parameters into individuals to be evolved by the Meta-EA (See Figure 3). At this stage, each individual does not specify a complete GA. Instead, it corresponds to a set of GAs sharing the same searchable parameters. Racing is then applied to efficiently identify the best GA within each set and its fitness value is returned as the fitness value of the corresponding high-level individual.

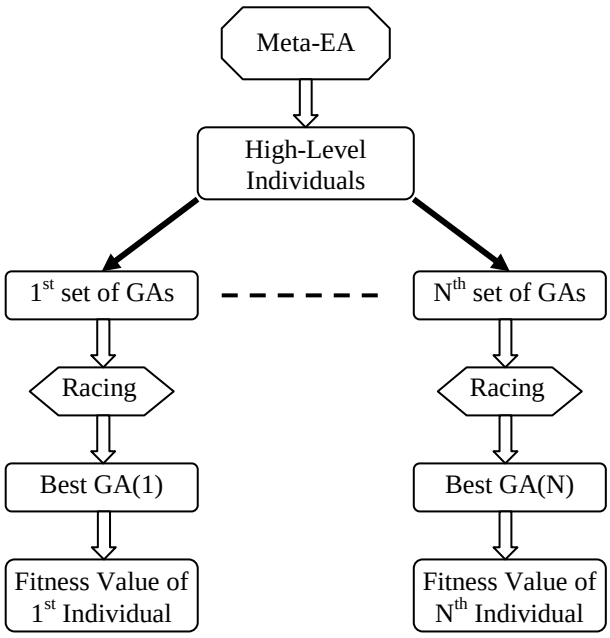

Figure 3. The Framework of the Hybrid Approach

4 Experiments

4.1 The Effect of GA Paramters

In this section, some preliminary experiments were conducted to demonstrate the effect of parameter setting with regard to the performance of GAs. More specifically, we focus on three points:

1. Different parameter settings may result in large difference in performance on a single problem.
2. A parameter setting that is good for one test problem may not be suitable for another one.
3. Dependences exist among GA parameters, making it inappropriate to tune parameters independently.

In order to demonstrate the first two points, two parameters settings were chosen (Table 2), which were different in selection strategy, crossover type and elitism.

	P	P_c	P_m	S	C	E
A	50	0.8	0.01	"Truncation" Ratio= 0.1	"Uniform"	0
B	50	0.8	0.01	"Tournament" Size=2	"Two-Point"	1

Table 2. Parameter Settings A and B

The GA described in Section 2 was tested on two maximization problems: the 100-bit One-Max problem and the 32-bit HIFF problem with the above two parameter

settings. The number of fitness evaluations was set to 1000, allowing 20 generations in each trial (i.e., the GA did not make any significant progress after 20 generations). For each experiment configuration (i.e., parameter setting + test problem), 50 trials were conducted and the distribution of the best solution in each trial is plotted in Figure 4 in which the Y-axis represents the fitness values (i.e., the global optimum value of 100-bit One-Max is 100 and the global optimum value of the 32-bit HIFF is 192). The box-whisker plots shows the upper quartile, median and lower quartile values of each distribution as well as the extent of the rest data.

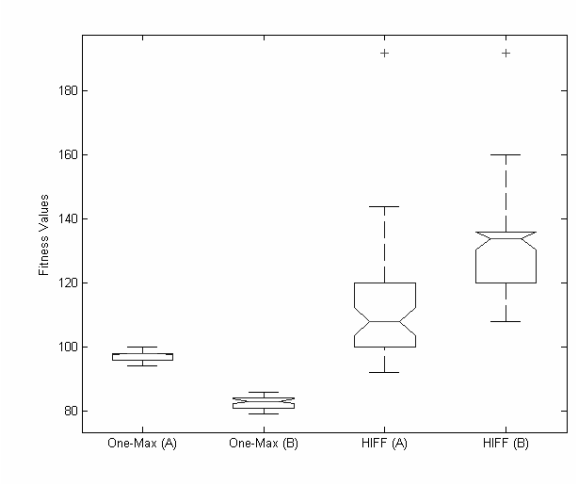

Figure 4. The Effect of Different Parameter Settings

In Figure 4, it is evident that parameter setting A produced a much better result than parameter setting B on the One-Max problem while the reverse is true on the HIFF problem. This clearly demonstrates the importance of parameter tuning in GAs and the necessity of doing specific tuning with regard to different problems.

A direct implication of this result is that in empirical studies of EAs, it is dangerous to establish any general conclusion with regard to the performance of algorithms on arbitrarily chosen test problems as well as arbitrarily chosen algorithm parameters.

Dependences among algorithm parameters can be demonstrated by the following experiments. Suppose that the interaction between P_m and S is of interest. Furthermore, suppose that Truncation Selection is in use but the corresponding selection ratio is to be determined. An incomplete algorithm specification is given by <50, 0.8, ? , "Truncation" ?, "Two-Point", 0> where "?" represents the parameter to be investigated. For each mutation level, a brute force method was used to find the best selection ratio for the Truncation Selection. The test problem was the 32-bit HIFF problem and 50 trials were conducted for each experiment configuration.

Experimental results (i.e., mean value ± one standard deviation) are summarized in Table 3 where the best results are marked in bold. With a small mutation rate 0.01, the best performance was achieved with selection ratio=0.5. By contrast, with a large mutation rate 0.20, the optimum selection ratio was 0.1, which shows clearly the correlation between the two algorithm parameters.

	S=0.1	**S**=0.3	**S**=0.5
$\mathbf{P_m}$ = 0.01	118.1±14.7	127.0±13.4	**136.7±17.9**
$\mathbf{P_m}$ = 0.20	**104.4±9.1**	96.2±9.1	92.6±8.5

Table 3. The Dependence between P_m and S

4.2 Racing vs. Brute Force

The core idea of the proposed hybrid approach is to use Racing to conduct searching on parameters not directly searchable. In this section, we empirically demonstrate the reliability (i.e., whether Racing can identify the best algorithms among a set of candidates) and efficiency (i.e., the number of trials conducted in Racing compared to the number of trials that need to be conducted in the brute force method) of Racing.

As mentioned in Section 2, there are six parameters to be specified including three continuous parameters (i.e., population size is regarded as a continuous variable due to its large cardinality). Although E is a binary variable, it is also included into Racing with S and C so that the Meta-EA only needs to handle a continuous optimization task. As a result, the task of Racing is, for each high-level individual, to find the best GA out of 11×3×2=66 candidates as efficiently as possible. Note that even if only one algorithm is left during Racing, it will still finish all 50 trials to have an accurate measure of its performance. In this paper, the Friedman test [5] was used in Racing with significance level α=0.10.

Four individuals were chosen that were different from each other in terms of $P/P_c/P_m$. For each individual, Racing and the brute force method were applied to find the best performing GA and its performance was returned as the fitness value. The test problem was the 100-bit One-Max problem and the number of evaluations was 500. In Table 4, the fitness values assigned by Racing and the brute force method are presented along with the ratio of the cost of Racing to the cost of the brute force method, measured by the number of trials actually conducted. It is clear that Racing was able to find GAs with similar quality as the brute force method but at a small fraction of its cost (i.e., the results of both methods could vary slightly due to the randomness of GAs). Certainly, Racing itself involves additional computational cost (e.g., statistical tests) but it is usually trivial compared to the cost of GA experiments.

P	$\mathbf{P_c}$	$\mathbf{P_m}$	**Fitness Value**		**Cost Ratio**
			Brute Force	**Racing**	
50	0.8	0.01	87.7	87.6	0.0833
50	0.8	0.20	73.7	73.4	0.1142
20	0.8	0.01	92.0	92.0	0.1179
20	0.5	0.01	90.0	89.3	0.1042

Table 4. Racing vs. Brute Force on One-Max

In order to have an intuitive understanding of the efficiency of Racing, a single Racing trial was picked up and the number of remaining algorithms at each step is presented in Figure 5. It shows that most of candidate algorithms were eliminated from the experiment and received no further testing after less than 10 trials, which saved a significant amount of computational effort. More specifically, the cost of the brute force method is equal to the whole box area (i.e., total number of algorithms × number of trials) while the cost of Racing is indicated by the much smaller area under the curve.

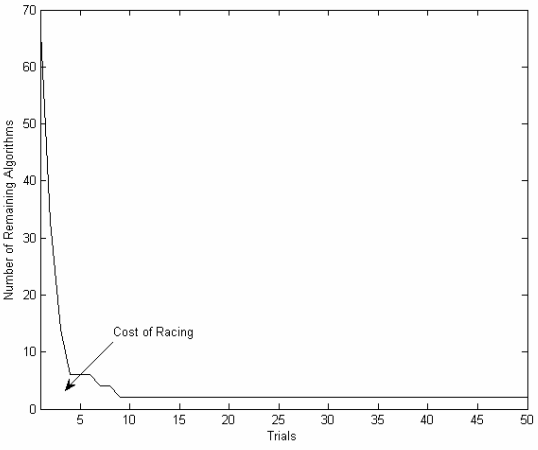

Figure 5. The Efficiency of Racing on One-Max

Similar experiments were also conducted on the 16-bit HIFF problem. Although the reliability of Racing was still very good, its efficiency generally decreased (i.e., it usually required 30% to 40% of the cost of the brute force method). It has been demonstrated that the efficiency of Racing is influenced by the performance distribution of those candidates with regard to the test problem[18]. The most favourable situation is that the performance of all algorithms is consistent across independent trials and the best algorithm is significantly better than others so that the majority of candidate algorithms could be quickly eliminated in early stages.

In practice, the effectiveness of Racing depends on a variety of factors. For example, the performance criterion in our work is the fitness of the best individual found and if all algorithms are allowed a large number of generations, it is possible that the performance of these algorithms may all be very good and thus similar to each other, which makes it a difficult task for Racing to distinguish them. On the other hand, if the problem is too difficult, all algorithms may perform very badly and it will also be difficult to distinguish them.

The property of HIFF may also account for the low efficiency. In the 100-bit One-Max problem, the possible fitness values are uniformly distributed within the range of [0, 100] with step size 1. By contrast, in the 16-bit HIFF problem, there are only 22 distinct values and also some big gaps among these values (e.g., the global optimum is valued at 80 while the next best value is 64). This means that in the 100-bit One-Max problem, algorithms could be characterized more precisely than in the 16-bit HIFF problem and thus may be more easily distinguished.

4.3 Meta-EA+Racing

In this section, the proposed hybrid approach was used to optimize the parameters of a GA on the 100-bit One-Max problem. There exist a number of EAs that could potentially be used as the Meta-EA. In this paper, a simple $(\mu+\lambda)$ ES [15] with diagonal Gaussian distribution was used due to its simplicity and capability of working with a small population. In our experiments, the population size was set to 20 (i.e., $\mu=\lambda=20$) and the Meta-EA was allowed to run for up to 30 generations. The boundaries and standard deviations for the three GA parameters are listed in Table 5. These values were chosen based on some general knowledge without any specific tuning. Other settings were the same as in Section 4.2.

Note that, for some values of the population size of the GA, it is possible that, given the fixed number of fitness evaluations, the number of generations may not be an integer. The solution adopted here was to increase the population size of the final generation to accommodate those extra individuals to maintain the same number of fitness evaluations among all candidates.

	Range	**Standard Deviation**
P	[20, 100]	5
$\mathbf{P_c}$	[0, 1]	0.05
$\mathbf{P_m}$	[0, 0.2]	0.02

Table 5. Definition of Search Space and Parameters of ES

The performance of the best GA and the mean performance of all 20 GAs in each generation are shown in Figure 6. Within the first 20 generations, the performance of the best GAs found increased from 89.2 to 93.98. The mean performance also increased from 78.23 to 93.38 during this period. A quick look into the final population shows that all individuals were very similar to each other and the Meta-EA has already converged.

Among those GAs in the final population, they all employed either Tournament Selection (size=6) or Truncation Selection (ratio=0.1). This indicates that it is preferable to have strong selection pressure to favour those promising individuals. Also, Uniform crossover dominated the crossover operator. Finally, elitism did not offer any significant benefit in this case study. Note that all GAs had very similar fitness values and due to their stochastic behaviour, it is not reasonable to regard any one of them as the best GA. The evolution process of the GA's parameters (i.e., averaged over the population in each generation) is presented in Figures 7-9 giving a clear indication of the advantage of small population size, large crossover rate and small mutation rate.

The cost of this hybrid approach was also calculated by adding up the relative cost of Racing in evaluating each Meta-EA individual. The total cost of Racing was 72.30,

giving an average cost of 72.30/ (20×30) =0.1205, which shows that Racing only required 12.05% of the effort required by the brute force method (i.e., this was at the similar level as those examples in Table 4).

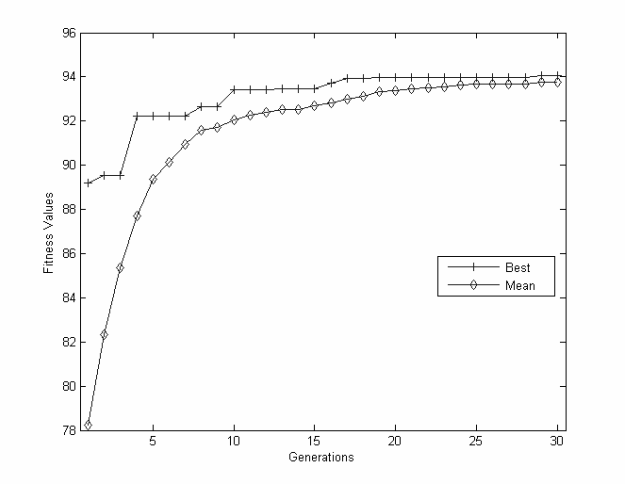

Figure 6. The Mean and Best Performance of GAs

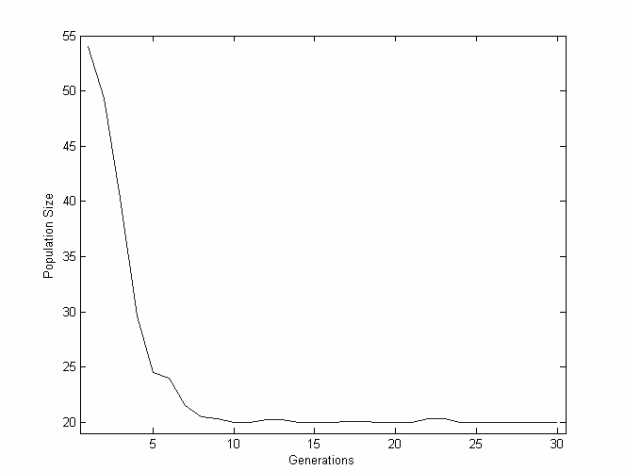

Figure 7. The Evolution Process of Population Size

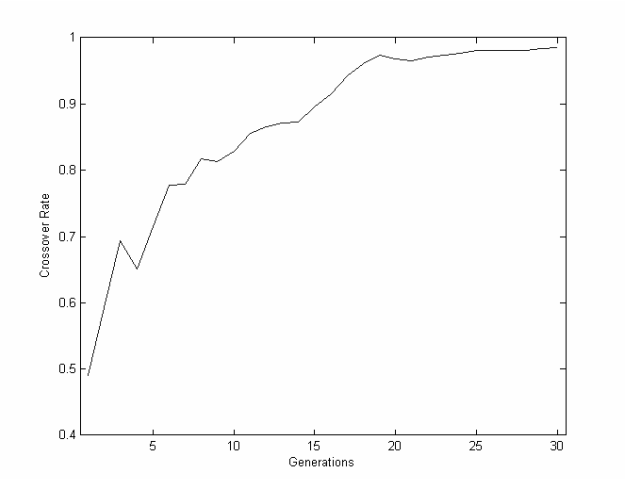

Figure 8. The Evolution Process of Crossover Rate

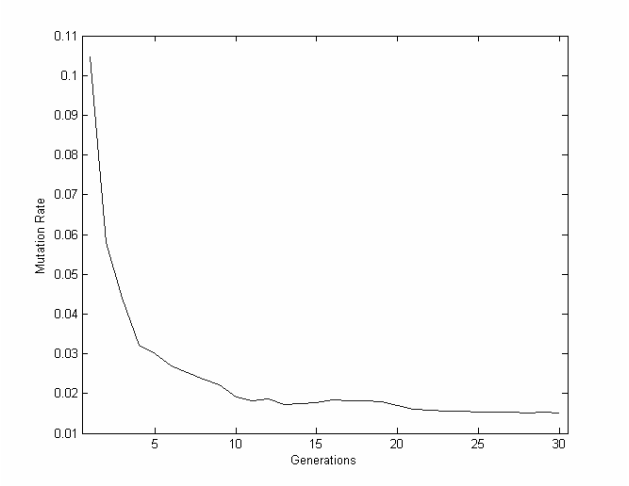

Figure 9. The Evolution Process of Mutation Rate

5 Conclusion

The major motivation of this paper is to investigate the issue of parameter tuning in GAs as well as other EAs. The traditional Meta-EA approach and a relatively new statistical Racing approach were analysed and their advantages and disadvantages were also discussed. A novel method was proposed by combining these two approaches in order to exploit their unique strengths while avoiding some inherent weaknesses. The core idea is to use the Meta-EA approach to optimize those tunable algorithm parameters while Racing is used to identify the best algorithm from a set of candidates different from each other only in terms of those non-searchable parameters. By doing so, this new method could enjoy both the global optimization ability of the Meta-EA and Racing's ability of handling non-searchable parameters.

A simple ES in combination with Racing was used to tune a GA with six parameters on the One-Max problem. Note that the focus here is not to argue which EA should be used as the Meta-EA. Instead, we intend to highlight the advantage of the hybrid approach as well as the efficiency of Racing. The estimated running time of the parameter tuning task is more than 70 hours on a P III 800 MHz PC without Racing (i.e., Meta-EA + Brute Force) while it actually only took about 9 hours with the help of Racing (i.e., Meta-EA + Racing).

Certainly, we are also aware that the parameter tuning of EAs is still a very time-consuming process especially for complex test problems, which prevents it from being widely applied in practice. One possible direction is to look for more effective statistical methods that could achieve better efficiency and reliability. After all, Racing is a general framework, which allows different techniques to be plugged into it. On the other hand, if the search space is known to be unimodal or a good starting point could be provided, classical local optimization methods may achieve faster convergence speed compared to Meta-EAs.

Acknowledgement

This work was supported by an Australian Postgraduate Award granted to Bo Yuan.

References

[1] Bäck, T. *Evolutionary algorithms in theory and practice*. Oxford University Press, New York, 1996.

[2] Bartz-Beielstein, T. *Experimental Analysis of Evolution Strategies - Overview and Comprehensive Introduction*. Technical Report 157/03, University of Dortmund, 2003.

[3] Birattari, M., Stutzle, T., Paquete, L. and Varrentrapp, K. A Racing Algorithm for Configuring Metaheuristics. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2002)*, 2002, 11-18.

[4] Blickle, T. and Thiele, L. *A Comparison of Selection Schemes used in Genetic Algorithms*. Technical Report Nr. 11, Swiss Federal Institute of Technology, 1995.

[5] Conover, W.J. *Practical Nonparametric Statistics*. John Wiley & Sons, Inc., 1999.

[6] De Jong, K. *The analysis of the behavior of a class of genetic adaptive systems*. PhD Thesis, University of Michigan, 1975.

[7] Eiben, A.E., Hinterding, R. and Michalewicz, Z. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, *3*, 2 (1999), 124-141.

[8] Goldberg, D.E. *Genetic Algorithms in search, optimization, and machine learning*. Reading, Mass. : Addison-Wesley, 1989.

[9] Grefenstette, J.J. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on System, Man, And Cybernetics*, *16*, 1 (1986), 122-128.

[10] Hart, W.E. and Belew, R.K. Optimizing an Arbitrary Function is Hard for the Genetic Algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, 190-195.

[11] Holland, J.H. *Adaptation in Natural and Artificial Systems*. University of Michigan, 1975.

[12] Maron, O. and Moore, A.W. Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. In *Proceedings of Advances in Neural Information Processing Systems 6*, 1994, 59-66.

[13] Maron, O. and Moore, A.W. The Racing Algorithm: Model Selection for Lazy Learners. *Artificial Intelligence Review*, *11* (1997), 193-225.

[14] Montgomery, D.C. *Design and analysis of experiments*. John Wiley & Sons, 1991.

[15] Schwefel, H.-P. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[16] Watson, R.A. and Pollack, J.B. Hierarchically-Consistent Test Problems for Genetic Algorithms. In *Proceedings of Congress on Evolutionary Computation*, 1999, 1406-1413.

[17] Wolpert, D.H. and Macready, W.G. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, *1*, 1 (1997), 67-82.

[18] Yuan, B. and Gallagher, M. Statistical Racing Techniques for Improved Empirical Evaluation of Evolutionary Algorithms. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, 2004, 172-181.