Combining Meta-EAs and Racing for Difficult EA Parameter Tuning Tasks

Bo Yuan and Marcus Gallagher

School of Information Technology and Electrical Engineering
University of Queensland, Qld. 4072, Australia
boyuan@itee.uq.edu.au, marcusg@itee.uq.edu.au

Summary. This chapter presents a novel framework for tuning the parameters of Evolutionary Algorithms. A hybrid technique combining Meta-EAs and statistical Racing approaches is developed, which is not only capable of effectively exploring the search space of numerical parameters but also suitable for tuning symbolic parameters where it is generally difficult to define any sensible distance metric.

1 Introduction

One of the major issues in applying Evolutionary Algorithms (EAs) is how to choose an appropriate parameter setting. The importance of parameter tuning is at least threefold. Firstly, EAs are not completely parameter robust and may not be able to solve challenging problems effectively with inappropriate values. Secondly, when two or more EAs are compared, arbitrarily specified parameter values may make the comparison unfair and conclusions misleading. Finally, finding the optimal setting may be also helpful for better understanding the behavior of EAs.

There has been a long history of work on finding the optimal parameter values of EAs [11]. However, it has shown that this kind of optimal setting does not exist in general and there are different optimal values for different problems [17]. To better understand the relationship between parameter setting and performance, for each pair of algorithm and problem, a performance landscape could be defined with one dimension for each parameter and an extra dimension for the performance. As a result, the global optimum of this landscape corresponds to the optimal parameter setting of the EA on that particular problem. Note that, the structure of such landscapes is unlikely to be always trivial (e.g., monotonic or separable).

One principled approach to parameter tuning is to exhaustively explore the performance of an EA with systematically varied parameter values. Statistical techniques such as ANalysis Of VAriance (ANOVA) could then be applied on

the results to formally investigate the influence and contribution of each parameter towards the algorithm's performance as well as the interaction among parameters [4, 5, 16], which may be used as heuristic knowledge in choosing the appropriate parameter setting. However, a large number of experiments are usually required to collect enough data for such statistical analysis, which is inefficient for the purpose of parameter tuning. Alternatively, mathematical modeling techniques such as Response Surfaces could be employed to search for the optimum parameter setting by only testing a relatively small number of parameter combinations [2].

If the performance landscape is suspected to be multimodal and there is no prior knowledge to come up with a good starting position, global optimization methods could be applied to conduct parameter tuning. A typical example is to use an external EA, called a Meta-EA [1, 8] as the high-level optimization procedure to conduct searching on the performance landscape. In these search-based methods, all parameters to be tuned are usually encoded into a vector and each instantiation of this vector/individual/algorithm instance corresponds to a fully-specified EA. Consequently, the process of parameter tuning is to find the individual or algorithm instance that yields the best performance.

It is worth mentioning that, in typical parameter tuning, algorithm parameters are chosen in advance and remain fixed during evolution. Another branch of research is parameter *control* in which algorithm parameters are allowed to vary with time as the solution vectors of the objective function are optimized [7]. 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. Furthermore, not all algorithm parameters can be conveniently adapted in this manner. After all, finding the optimal settings in different situations may also provide valuable information for designing better parameter control strategies.

The major contributions of this chapter are listed as follows. Firstly, the properties of traditional search-based methods are given a critical analysis. We point out that, in addition to being very time-consuming in evaluating each candidate, they have inherent difficulty in tuning nominal or symbolic parameters where it is very hard to define any sensible distance metric over the parameter space. For these parameters, no search-based methods could be expected to be better than the exhaustive or random search.

Secondly, a novel statistical technique called Racing [12], which is originally proposed to solve the model selection problem in Machine Learning, is introduced to reliably choose the best parameter setting. The advantage of Racing is that it does not require a well-defined parameter space but is still much more efficient than the exhaustive search.

Finally, with a good understanding of the properties of the two distinct classes of tuning techniques, a framework of hybridization is proposed, which is not only able to handle symbolic parameters but also to search and explore

the parameter space efficiently. Two case studies on tuning the parameters of a Genetic Algorithm (GA) are also conducted to highlight the effectiveness of the proposed techniques.

2 The Challenge of Symbolic Parameters

As mentioned above, in search-based methods, the EA to be tuned is often encoded as a vector and each individual/algorithm instance corresponds to a unique point in the parameter/search space. The major reason that these search-based methods can be expected to outperform random or exhaustive search is largely due to the assumption of the existence of some regular structure in the performance landscape. In other words, there should be some kind of "path" in the landscape that could be exploited by the search operators to effectively drive the overall tuning process.

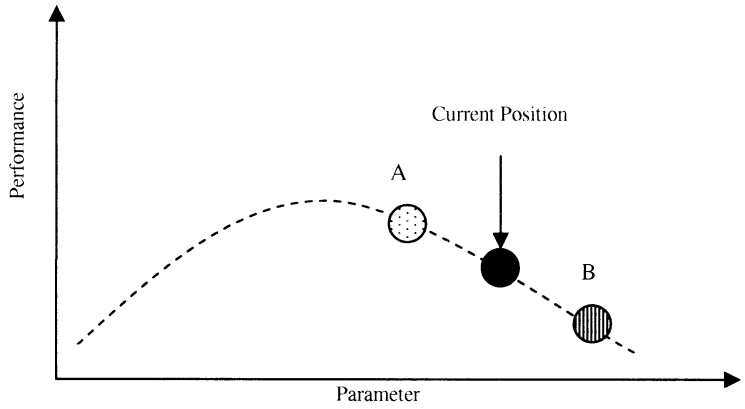

Fig. 1. An example of the tuning process conducted by a hill-climbing method.

Figure 1 shows the tuning process conducted by a hill-climbing method working on a single parameter. After evaluating the two neighbors A and B of the current individual, it turns out that A is better than B as well as the current individual. As a result, in the next step, the hill-climbing method will move its current position to A (i.e., towards the optimum).

Although this example is very simple, it does reveal a key factor of all non-trivial search-based methods: a meaningful distance metric among individuals, which could create a performance landscape compatible with the search operators. In general, the distance metric in the search space (i.e., the spatial relationship among parameter settings) is determined by the way that parameters are encoded. For example, if the continuous crossover rate and

mutation rate parameters are directly encoded using their original data type, it would be reasonable to say that two algorithm instances similar to each other in terms of these two parameters are also close to each other in the search space defined by the Euclidean distance. However, if they are encoded into binary strings, these two algorithm instances may be quite distant from each other in terms of the Hamming distance.

The major issue here is that EAs with similar parameter settings should be represented by individuals that are also close to each other in the search space. Note that the actual distance between two individuals in terms of searching is also influenced by the specific search operator in use [10]. If the distance metric does not match the similarity among algorithm instances, the fitness values/performance of individuals close to each other in the corresponding search space may not be closely correlated.

From the landscape point of view, this means that the performance landscape may present a significant level of noisy structure, which could reduce the effectiveness of almost all search-based methods. In fact, finding the appropriate encoding scheme is also a well-known common issue in the field of Evolutionary Computation.

Unfortunately, unlike those numerical parameters such as population size and mutation rate, some parameters are nominal and only have symbolic meanings, which cannot be encoded in a natural way. For instance, an EA could adopt one of many different selection strategies but it is hard to design a meaningful encoding scheme for this parameter. The reason is that there is usually no clear knowledge about the relationship among these selection strategies and it is difficult to quantify the distance between each pair of them (e.g., the distance between the Tournament selection and the Truncation selection). In fact, it is even difficult to say which one is more similar to which one in some general sense. Certainly, it is always possible to apply any arbitrary encoding scheme on this parameter but the shape of the performance landscape could also be arbitrary, which can hardly be guaranteed to be solved efficiently by a given search-based method.

Suppose that there are 10 candidate selection strategies and the performance of the EA with each strategy is shown in Figure 2 (top), labeled according to a certain encoding scheme compatible with their inherent similarity. Note that this is an idealized situation and if an arbitrary encoding scheme is applied, selection strategies that produce similar performance could be mapped into quite different individuals (i.e., large distances along the horizontal axis). As a result, the original smooth landscape may turn into something pretty nasty as shown in Figure 2 (bottom).

More specifically, in Figure 2 (top), selection strategies close to each other usually result in similar algorithm performance and the global optimum could be easily reached by the hill-climbing method starting from any position. By contrast, in Figure 2 (bottom), although the algorithm's performance with each selection strategy is kept unchanged, the spatial relationship has been

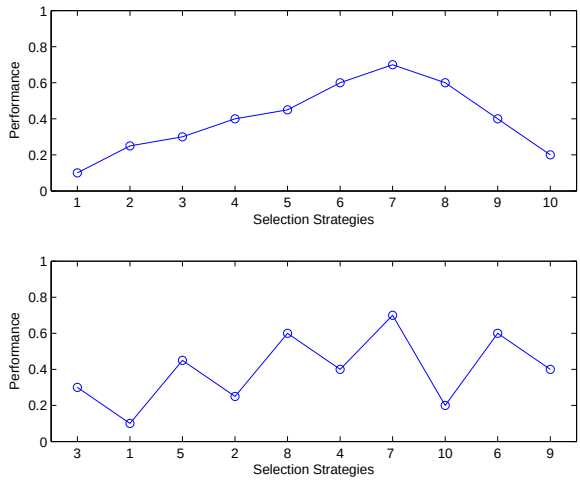

Fig. 2. The influence of the encoding scheme on the performance landscape: A smooth landscape defined in consistence with the similarity of selection strategies (top) and a nasty landscape defined by reordering the sequence (bottom).

altered. As a result, the previous smoothness is broken and the new landscape contains multiple local optima.

There are further challenges associated with applying search-based methods to parameter tuning as shown in previous research. Firstly, it is possible that some parameters only play a role in certain situations. As a result, the encoding scheme may have to contain some redundancies and the tuning algorithms also need to be specifically customized to handle these parameters. Secondly, in order to evaluate each individual, the corresponding EA must be run for several trials in order to get reliable performance. However, in practice, it would often require an enormous amount of computational time and may become impractical in many cases.

3 Racing and Model Selection

Racing was originally proposed to efficiently solve the model selection problem in Machine Learning [12, 13]. Typically, the performance of a group of M candidate supervised learning models is evaluated on a data set of interest containing N test points using a validation approach. The general mechanism of the leave-one-out cross validation (LOOCV) is to use all but a single test point to train the model. After training, the model is tested on the test point absent from the training. This process is repeated N times with each time a unique test point left out and the model with the lowest average error is

regarded as the best model. In other words, the performance of each model is determined by the mean of its performance distribution, which consists of the outputs from a series of experiments.

The most straightforward approach would be to exhaustively test each model on all available test points. With this brute force method, after finishing N experiments, the true performance E_{true} of a certain model could be found and after totally $M \times N$ experiments, it will guarantee to find the best model according to the LOOCV measure. However, the complexity of this method is $O(M \cdot N)$, which may become quite time consuming for large M and/or N values. After all, each experiment may involve quite heavy computation as the model needs to be trained on $N - 1$ test points.

By contrast, Racing techniques approach this model selection problem from a different angle: whether it is necessary to test a model for N times? Certainly, in principle, each model must undertake all tests to reveal its E_{true}. However, the ultimate goal here is *not* to examine E_{true} but to find the best model. From a statistical point of view, the errors of a model on all N test points create a population and it is possible to estimate the population mean E_{true} based on E_{sam}, which is the mean of a random sample of size N' (i.e., $N' < N$). Usually, it is more helpful to estimate the true difference between two models based on the difference of their sample means.

The basic idea behind Racing (Table 1) is to test all available models in parallel on a single unseen test point at each step. The errors of each model on all test points that have been selected are maintained and used in some statistical test. If there is enough evidence, as indicated by the result of the statistical test, that model i is significantly worse than another model, it will be removed from the model selection process and undertake no further testing because it is unlikely to be the best model. By doing so, models are forced to race against each other and only promising ones could survive through to the next iteration. As a result, the overall computational cost could be reduced compared to the brute force approach described above by avoiding running unnecessary experiments on inferior models. Figure 3 gives an illustration of a

Table 1. The framework of Racing for model selection problems.

Repeat steps 1-4 until there is no more unseen data point or only one model is remaining:
1. Randomly select a new data point
2. Calculate the LOOCV error of each remaining model on it
3. Apply some statistical test to conduct pair-wise comparison
4. Remove models that are significantly worse than others
5. Return the remaining model with the lowest average error

virtual Racing experiment with 50 starting models and totally 200 test points where the length of each bar indicates when a model was removed from the

Racing process. It is clear that after being tested on 20 test points quite a few models had already been removed and only 12 models could pass through 60 iterations. There were only two models (i.e., No. 34 and No. 47) left after 102 iterations and model No. 34 won the competition against No. 47 after being tested on additional 24 test points and thus the Racing process was terminated. Note that, in practice, it is also possible that some models are indistinguishable even at the end of Racing.

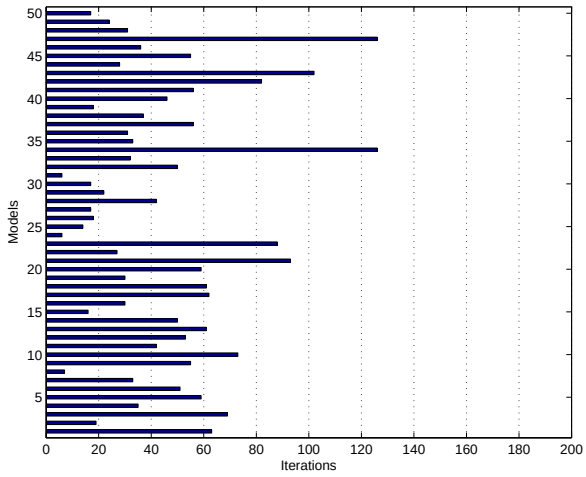

Fig. 3. An illustration of a virtual Racing experiment with 50 candidate models and 200 test points in the dataset. It shows the number of iterations that each model has gone through before being removed from the Racing process.

The advantage of Racing is obvious from the above example: there is no need to test all models thoroughly on 200 test points and most of the models were actually tested on far less number of test points before being discarded. The cost of Racing is indicated by the area of the bars (i.e., 2,264 tests), compared to the cost of the brute force method, which is equal to the entire box area (i.e., 50×200=10,000 tests). The heart of the Racing technique, as shown in Table 1, is the statistical test used to judge whether one model is significantly worse than another, which directly determines the efficiency and reliability of Racing. Depending on the properties of the performance distributions, there are two major tests that have been used in previous work: the non-parametric Hoeffding's bounds [9] and the parametric Student-t test [14]. For a single model, suppose its errors are bounded within $[b, a]$. Then, for $a - b > \xi > 0$, the probability of E_{sam} being more than ξ away from E_{true} after n iterations is:

$$P(|E_{\text{sam}} - E_{\text{true}}| \geq \xi) \leq 2e^{\frac{-2n\xi^2}{(a-b)^2}} \tag{1}$$

Based on Eq. 1, it is easy to derive the value of ξ for a given significance level α:

$$\xi = \sqrt{\frac{(a-b)^2 \log(2/\alpha)}{2n}} \tag{2}$$

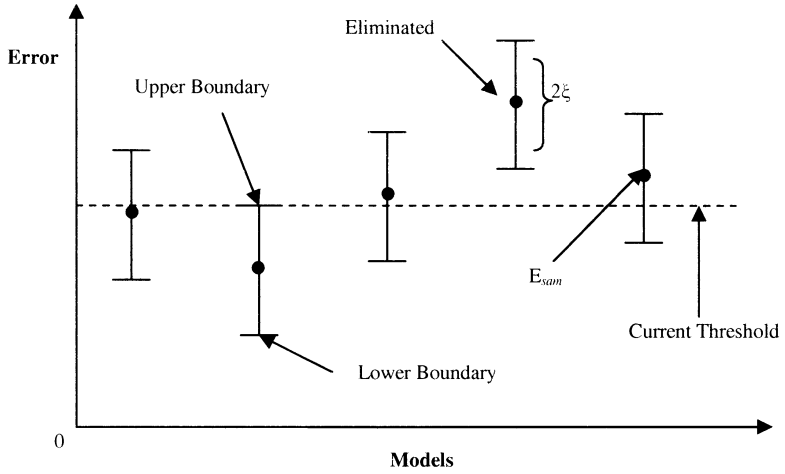

Fig. 4. An example of Hoeffding Racing at a specific iteration with 5 remaining models left. The current threshold is indicated by the dashed line.

According to Eq. 2, at each step, the upper and lower boundary of E_{true} is estimated for each remaining model: $[E_{\text{sam}} - \xi, E_{\text{sam}} + \xi]$. The eliminating rule is to remove models with lower boundaries (i.e., best possible errors) still higher than the upper boundary (i.e., worst possible error) of the best model [12]. Figure 4 shows the competition among five models at a certain stage of Racing where the dashed line is the threshold above which all models are to be eliminated (e.g., the second rightmost model).

In addition to Eq. 2, it would also be possible to utilize another Hoeffding's bound for the difference of two sample means with regard to the difference of the true means [9]:

$$P(|E^1_{\text{sam}} - E^2_{\text{sam}} - (E^1_{\text{true}} - E^2_{\text{true}})| \geq \xi) \leq 2e^{\frac{-n\xi^2}{(a-b)^2}} \tag{3}$$

The major advantage of Hoeffding's bound is that it has no assumption on the underlying probability distribution and can be applied in a variety of situations. However, the bounds acquired are typically not very tight and

consequently quite a lot of test points may be needed in order to distinguish models [18].

If it is reasonable to assume that the errors are approximately normally distributed, the parametric Student-t test is usually more powerful. Under the null hypothesis that there is no difference between the true means and the assumption of equal variances, the test statistic is given by:

$$z = \frac{E^1_{\text{sam}} - E^2_{\text{sam}}}{s\sqrt{2/n}} \tag{4}$$

In Eq. 4, s is the pooled estimator of the standard deviation σ and the test statistic z follows the Student-t distribution with $2n-2$ degrees of freedom. The null hypothesis is to be rejected if the value of z falls into the critical region based on the desired significance level.

If there is reason to doubt the assumption of equal variances, the test statistic is then given by (i.e., s_1 and s_2 are the estimators of σ_1 and σ_2):

$$z = \frac{E^1_{\text{sam}} - E^2_{\text{sam}}}{\sqrt{(s_1^2 + s_2^2)/n}} \tag{5}$$

The corresponding degree of freedom is approximated by:

$$\text{d.f.} \approx \frac{(s_1^2 + s_2^2)^2}{\frac{s_1^4 + s_2^4}{n-1}} \tag{6}$$

Figure 5 shows an example with five remaining models at a certain stage of Racing where the performance distributions are approximately normal but have different variances. In this case, based on the Student-t test, the model in the middle is likely to be eliminated.

Note that in the Hoeffding Racing and the Student-t Racing with equal variances, a model only needs to be tested against the currently best model because if it is not significantly worse than the best model, it cannot be eliminated by any other models. However, in the Student-t Racing with unequal variances, a model that cannot be eliminated by the best model with a large variance might be eliminated by a good model with a small variance. As a result, each model needs to be compared against all models that have better sample errors.

Furthermore, in Table 1, all remaining models are tested on the same data point in each step. However, both statistical tests above assume that samples are independently drawn from the populations and take no advantage of this paired experiment. Nevertheless, for statistical tests with blocking design, this feature could be explicitly exploited to reduce the variance among test points and increase the power of statistical tests.

In summary, the major advantage of Racing is that it only works on the statistics of the experimental results instead of the structure of models and could be applied in a much wider situation than many other methods.

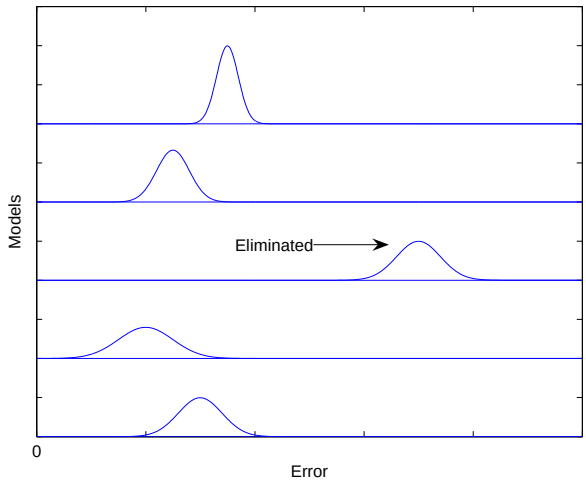

Fig. 5. An example of Racing based on the Student-t test (5 remaining models).

4 Racing Evolutionary Algorithms

4.1 Overview

There is a clear similarity between the model selection problem in Machine Learning and the task of parameter tuning in EAs. In each case, there is a meta-optimization problem: to find values for all of the adjustable parameters of the model or algorithm in order to produce the best results when applied to the problem of interest. More importantly, the performance of a model or an algorithm is often defined as the mean value of the outputs from a series of random experiments or trials, which provides the basis for applying statistical tests. In fact, the set of EAs to be raced do not necessarily need to be different instances of the same EA. Hence, Racing can be used in quite a general sense in an attempt to reduce the experimental effort required whenever a comparison is needed over a range of experimental configurations provided that the evaluation of each configuration involves iterating over a number of independent trials (restarts).

In addition to some early attempts in applying Racing in the field of EAs [3, 19], there are a few important differences between model selection and evaluation of EAs that must be made clear.

Firstly, the number of test points in model selection problems is fixed, which means that it is always possible to find the true performance of each model (i.e., assume that its behavior is not stochastic). However, for EAs, the number of trials that could be conducted for each pair of algorithm and problem is unlimited, which means that it is impossible to know the true

performance of EAs. Since, in practice, 50 to 100 trials are usually believed to be sufficient to produce trustable results, it is reasonable to put a similar upper boundary on the maximum number of trials in Racing experiments. Otherwise, it may need an extremely long period to terminate.

For example, in scientific studies, it is preferable to compare the best model or algorithm found by the brute force method and that found by Racing to show its reliability (i.e., whether the best model/algorithm was eliminated incorrectly at some stage). Although it is very straightforward in model selection as the performance is deterministic, in EAs, the only claim that can be made is to say whether the top EAs found by the brute force method (i.e., certainly it is based on a limited number of trials and the results are still stochastic) are also available at the end of Racing.

Secondly, in model selection, all models could be tested on the same test point at each time and the blocking techniques may improve the power of statistical tests. However, in the experiments of EAs, the output from the i^{th} trial of one EA typically has nothing to do with that of another EA, which means that there is no explicit correspondence/relationship between the results. Consequently, it is not beneficial to use any paired statistical tests in the comparison of EAs with regard to random trials.

Thirdly, in some model selection problems, the error range is known in advance. For example, in classification problems, the model's error at each test point is either 0 or 1, representing correct or incorrect classification. As a contrast, it is usually not easy to give a tight estimation of the range of the performance of EAs on test problems, which may make it difficult to apply the Hoeffding's method.

Finally, the number of test points in a typical dataset could easily reach a few hundreds or even thousands while the number of trials to be conducted in an EA experiment is usually no more than 100. A major implication is that, in model selection, it is acceptable to start Racing after a number of, say 30, test points have been selected and this would not have a significant impact on the efficiency of Racing methods. The advantage is that, although it is usually not known in advance whether the errors of those models are normally distributed, it is still reasonable to apply those statistical tests based on the assumption of normality such as the Student-t test as long as the sample size is relatively large. However, for EAs, this would make Racing methods require at least 60% cost of the brute force method if the maximum number of trials is 50.

4.2 Statistical Tests

Based on the above analysis, a good statistical test for racing EAs should be able to work with small sample sizes and not rely on the assumption of normality. Traditionally, the nonparametric Wilcoxon rank-sum test is used as the alternative of the Student-t test. However it still assumes that the two underlying populations follow distributions with similar shapes, which is not always reasonable.

To overcome the difficulty faced by many existing statistical tests, a hypothesis test based on the bootstrap method is explained next. The bootstrap method is a computationally intensive statistical technique based on the resampling of data [6]. It could be used to calculate many important statistics such as standard errors, bias and confidence intervals without the need of using complicated mathematical models. When used in hypothesis testing, it could avoid the danger of making various assumptions such as normality and equal variances required by other statistical tests.

The basic procedure of the bootstrap method is described in Figure 6. Given a sample X containing n data points, B bootstrap samples of the same size are generated by sampling from X with replacement. This means that X is regarded as the whole population and each X^* is a sample from it (i.e., each data point in X has equal probability to be selected). For each bootstrap sample $X^*(i)$, the bootstrap replication of the statistic of interest θ denoted by $\theta^*(i)$ is calculated from a certain function S such as the mean or the median of the sample. Finally, all such replications θ^* are combined together to, for example, calculate the standard error of θ or create a bootstrap table like the Student-t table for estimating confidence intervals.

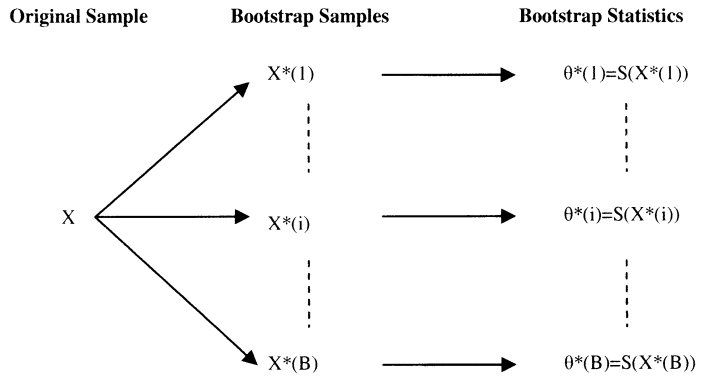

Fig. 6. The basic procedure of the bootstrap method.

In general, better estimation could be achieved with larger B values such as B=2000. Obviously, this requires much more computational resource compared to classical statistical tests. As a result, in the following experiments, the bootstrap test is applied on samples with less than 30 data points combined with the Student-t test in other situations to try to find a balance between reliability and computational cost.

4.3 Racing Experiments

In this section, Racing is applied to the situation where researchers are interested in finding out which algorithm out of a set of candidates performs best on a particular benchmark problem. For this purpose, we chose the well-known Rastrigin function, which is a multimodal problem (i.e., to be minimized) with the global optimum at the origin:

$$f(X) = 10n + \sum_{i=1}^{n}(X_i^2 - 10\cos(2\pi X_i)), X_i \in [-5, 10] \quad (7)$$

The algorithm to be tuned is a continuous EDA (Estimation of Distribution Algorithm) based on Gaussian distributions, which is similar to RECEDA [15] but with some additional features.

Table 2. The framework of the continuous EDA based on Gaussian distributions.

	Initialize and evaluate the starting population P
	`while` stopping criteria not met
1.	Select top n individuals P_{sel} (Truncation selection)
2.	Fit a multivariate Gaussian $G(\mu, \Sigma)$ to P_{sel} $\mu = \frac{1}{n}\sum_{i=1}^{n} P_{\text{sel}}$ $\Sigma = \frac{1}{n}(P_{\text{sel}} - \mu)^T \cdot (P_{\text{sel}} - \mu)$
3.	Update μ towards the best individual X^{best} $\mu' = (1 - \alpha) \cdot \mu + \alpha \cdot X^{\text{best}}$
4.	Update Σ by an amplification scalar γ $\Sigma' = \gamma^2 \cdot \Sigma$
5.	Sample a set of new individuals P' from $G(\mu', \Sigma')$
6.	Evaluate individuals in P'
7.	Choose the best individuals from the union of P and P' to form the new population P
	`end while`

A set of $5 \times 4 \times 3$=60 candidate algorithms were created by systematically varying the following three parameters described in Table 2 (i.e., each of these algorithm instances corresponds to a unique parameter setting):

- Selection ratio (step 1): [0.1, 0.3, 0.5, 0.7, 0.9]
- Learning rate (step 3): [0.0, 0.2, 0.5, 1.0]
- Amplification scalar (step 4): [1.0, 1.5, 2.0]

Other experimental configurations were fixed as: dimensionality=5, population size=100, number of generations=100 and maximum number of trials=100. The fitness value of the best individual found in each trial was recorded as the performance criterion.

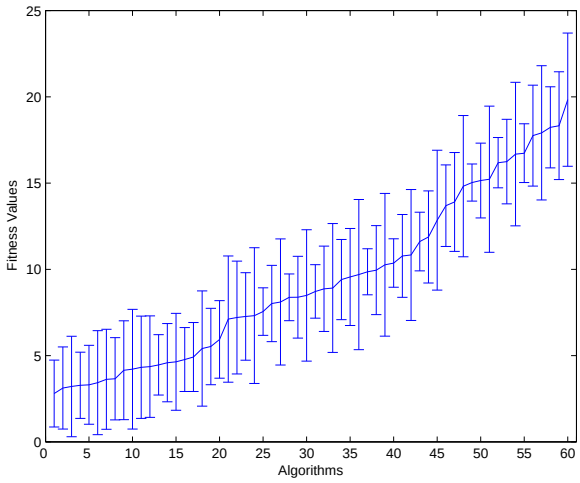

Fig. 7. The performance distributions of 60 candidates on Rastrigin's function.

In order to provide a benchmark for the performance of Racing, an exhaustive experiment was conducted by running all 60 algorithms for 100 trials (i.e., 6,000 trials in total) and the overall performance (i.e., sorted based on the mean values) is summarized in Figure 7 with error bars (i.e., one standard deviation above and below the mean). It is possible to inspect these results visually and determine, for example, which algorithms are most likely to find good solutions. Note that there is clear evidence that the variances tend to be different, which violates the widely adopted assumption of equal variances.

Using Racing techniques, we aim to make a similar observation at a fraction of the computational effort required for the above exhaustive experiment. The procedure operates here by eliminating poorly performing algorithms on the basis of a small number of restarts. The significance level α was 0.05 and the Racing algorithm was started after 5 trials, taking into account the limited maximum number of trials (100).

Note that since the brute force search has already been conducted, in the Racing experiment, the performance of an algorithm instance in each trial was retrieved as a sample from the corresponding performance population without actually re-running the EA (i.e., certainly this is only applicable for comparison purpose). Since the sequence of such samples may have more or less impact on the performance of Racing, the Racing experiment was repeated by 10 times with random sequences of samples.

The average number of algorithm instances remaining at each step is shown in Figure 8 from which it is clear that Racing eliminated a large portion of algorithms in the early stage and usually there were only around 5 algorithm instances remaining after 20 restarts. The average number of candidate algo-

rithms left after 100 restarts was 1.6 and sometimes Racing was terminated even without the need of fully testing any candidate for 100 trials. The efficiency of Racing can be visually examined by comparing the area under the curve and the area of the whole rectangular region.

From a quantitative point of view, the average cost of Racing is 10.89% of the cost of the brute force method in terms of the number of trials that have actually been conducted. In other words, Racing reduced the cost by almost 90%, which could be very significant for expensive experiments. Certainly, the actual efficiency of Racing depends on the properties of the specific performance distributions encountered.

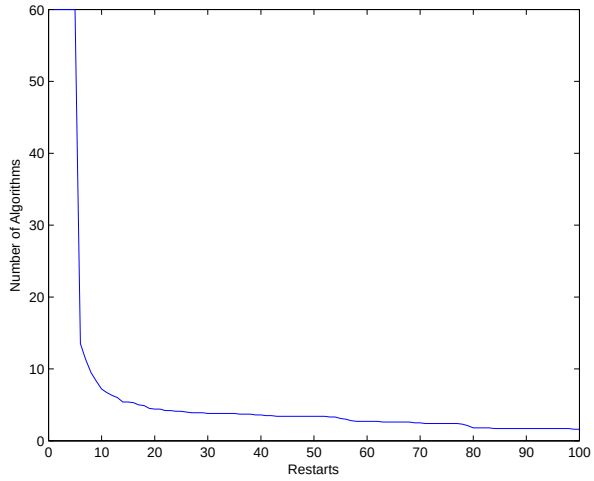

Fig. 8. The average number of remaining algorithm instances during Racing. The cost of Racing is indicated by the area below the curve.

Another important performance metric of Racing is reliability: whether the top algorithms could be preserved and avoid being removed incorrectly. In the above Racing experiments, the best algorithm based on the exhaustive experiment was remaining till the end of Racing in 9 out of 10 times, showing pretty impressive robustness. After all, for the single Racing experiment in which the best problem was eliminated, three other top algorithms were still able to survive through to the end of Racing. If greater reliability is desired, a higher significance level could be used to make Racing more conservative in eliminating models. Furthermore, Racing could be allowed to start after more samples have been collected in order to improve the accuracy of estimation.

5 Hybrid Techniques

5.1 Racing vs. Meta-EA

As shown in the last section, Racing does not work on the parameters of EAs in the way that a search-based method might. Instead, Racing is based on the statistical comparison of different algorithm instances, which are regarded as atomic black boxes (i.e., only the performance evaluation of the instances is required). By doing so, the requirement of defining an appropriate encoding scheme or distance metric is removed. However, 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. If Racing is to be applied to the task of parameter tuning, the optimal parameter setting must be included in the initial set. Otherwise, there is no hope to find it by Racing.

Unfortunately, there is usually no such guarantee in practice. One solution would be to start with an exhaustive set containing all possible combinations of parameter values. However, for continuous parameters, the candidate set might be very large assuming certain discretization of these parameters, which makes this approach impractical. Also, the accuracy of searching may also suffer accordingly.

In summary, the major challenge faced by all search-based methods is the difficulty in defining a distance metric over those nominal/symbolic parameters. Without a sensible distance metric, the resulting landscape may present high level irregularity and no search algorithm is expected to do well on it. For this reason, these parameters are generally referred as non-searchable parameters. Another practical issue is that in search-based methods each individual is to be evaluated independently of others and need to be tested thoroughly on the benchmark problem, just like the brute force method. If the size of the search space is relatively small, the advantage of search-based methods may not be significant as they still need to evaluate a large portion of all possible candidates (i.e., this is particularly true for those population-based methods such as Meta-EAs).

By contrast, the difficulty in Racing is mainly due to the fixed initial set of algorithm instances. If the number of all possible candidates is not too large, say a few hundreds, Racing is expected to work reasonably well. If the number is far beyond this figure, it would be difficult to apply Racing directly on such a huge set of candidates (e.g., even testing one million candidates for a single trial would require an enormous amount of time). In this situation, it is only practical to require Racing to work on a small number of candidates at each time.

5.2 Hybridization

It is easy to see that the properties of Racing and search-based methods are complimentary to each other. In order to efficiently handle situations where

the search space is potentially very large and/or contains symbolic parameters, it is useful to consider a hybrid tuning framework combining Racing and search-based methods. In this section, two examples are given to demonstrate how this hybrid approach works.

In the first scheme, a relatively simple situation is considered where a $(1+\lambda)$ ES, which is a special case of EAs, is combined with Racing and it is assumed that all parameters to be tuned are numerical and thus searchable. At each step, a set of λ new individuals/algorithm instances are sampled from a Gaussian distribution centered at the current solution.

The major computational cost in this method is on finding the best individual from a set of candidates, which is exactly what Racing is designed for. In the proposed hybrid approach, instead of sequentially evaluating each individual, all individuals are brought into Racing to find out the best one and as shown in previous experiments, it is expected that many of them may be eliminated after just a few trials, saving a large portion of computational resource (Figure 9).

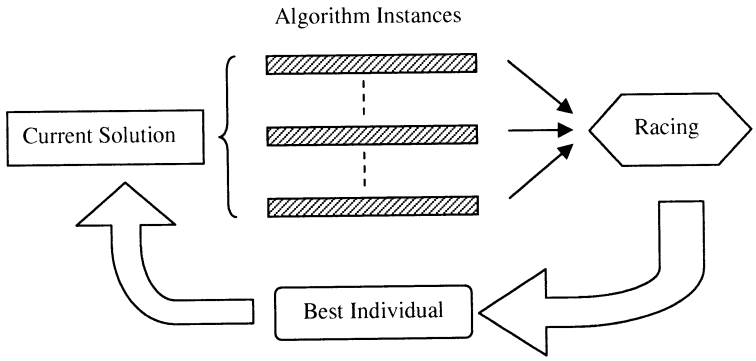

Fig. 9. The framework of the hybrid $(1+\lambda)$ ES + Racing scheme.

The second scheme is proposed to handle the more general and complicated situation where parameters to be tuned include symbolic ones, which are generally not searchable. Also, it is assumed that a more general and population-based Meta-EA is in use.

The general idea is to treat these two classes of parameters separately: only searchable parameters are encoded into individuals in the population of the Meta-EA and each individual no longer corresponds to a fully specified algorithm instance. Instead, it represents a set of complete EAs sharing the same searchable parameters and different from each other on those non-searchable parameters. Note that the size of the set depends on the number of

non-searchable parameters and their cardinalities. In order to evaluate each individual, Racing is applied to find the best algorithm instance from the corresponding set of EAs and its performance is returned as the fitness of the individual (Figure 10). An alternative approach would be to apply the brute force method to evaluate those algorithm instances, which may be very time-consuming.

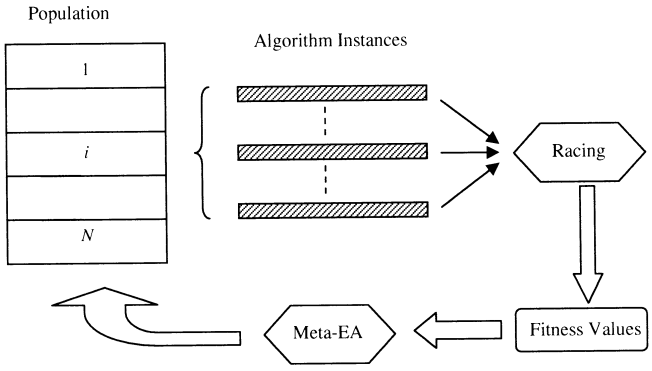

Fig. 10. The framework of the hybrid Meta-EA + Racing scheme.

5.3 Case Studies

In this section, a standard GA with binary representation is used in the experiments to illustrate the hybrid approach. Two parents from the current population are selected at each time and two offspring are generated through recombination 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 bit by flipping it from 0 to 1 and vice versa with probability P_m. 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.

There are a number of parameters to be specified. 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 cause premature convergence while a very large one may result in a slow convergence rate. The crossover rate P_c and mutation rate P_m are both continuous parameters within [0, 1], which are important for controlling the balance between exploration and exploitation. The selection

strategy (S) is a symbolic parameter, which contains various candidates (e.g., Tournament Selection) combined with their corresponding parameters (e.g., Tournament size). Also, a symbolic 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. The feasible values of S, C & E are given in Table 3.

Table 3. Feasible values of S, C & E.

Parameters	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)

According to the above discussion, the GA could be fully specified by a six-element tuple: $< P, P_c, P_m, S, C, E >$. Note that some parameters 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 symbolic, which created a mixed-value optimization problem.

In the first case study, a simple situation was considered in which only P, P_c and P_m were to be tuned while the rest of the parameters were set to some fixed values. In this experiment, although all parameters were searchable, evaluating each individual using the brute force method could still be very time-consuming. As a result, the hybrid (1+λ) ES+Racing approach was applied to help reduce this computational burden.

The (1+λ) ES in use was based on a fixed Gaussian distribution with a diagonal covariance matrix. The boundaries of the search space and the standard deviations are given in Table 4, which were chosen based on some general knowledge without any specific tuning. Note that for some values of the population size, it is possible that, given the fixed number of fitness evaluations (i.e., 500), the number of generations may not be an integer. The solution adopted was to increase the population size in the final generation to accommodate those extra individuals to maintain the same number of fitness evaluations among all candidates.

The 100-bit One-Max problem was used as the benchmark problem. The (1+λ) ES with λ =20 started from the middle of the search space while other GA parameters were arbitrarily set to (i.e., not necessarily a good choice) <"Truncation", 0.5, "One-Point", 1>. Note that, in addition to identifying the best candidate, its performance/fitness value is usually needed in search-based

Table 4. Definition of search space and standard deviations.

	Range	Standard Deviation
P	[20, 100]	5
P_c	[0, 1]	0.1
P_m	[0, 0.2]	0.02

methods. As a result, the starting individual was evaluated exhaustively. The best candidate found by Racing (i.e., significance level 0.10) was also required to go through all trials (100) to find out its mostly reliable performance for the comparison against the currently best one even if it was the only one left.

The parameter settings found during evolution in a typical trial (i.e., 20 generations) and the corresponding performance of the GA are shown in Figure 11. It is clear that this approach could find much better parameter settings than the initial one and the average performance of the GA increased from 71.51 to 83.97 within 15 generations.

In the meantime, all three parameters under tuning showed a clear pattern. The population size dropped from 60 to 20 while the crossover rate increased from 0.5 to 1.0. The mutation rate also quickly settled down to a very small value around 0.02. Since there is no dependence among the variables of the One-Max problem and it has a simple structure with only one optimum, it seems unnecessary to employ a large population to maintain the genetic diversity and a small mutation rate is well justified in this case. As for the crossover rate, it plays a major role in GAs in combining building blocks and a quite large value is often recommended.

Obviously, the hybrid approach did a good job in this case study and found some very good parameter settings for the GA. After all, it only took around 13.4% of the cost of an ordinary $(1+\lambda)$ ES (i.e., individuals are exhaustively evaluated) in terms of the number of trials conducted. Note that the lower limit of the cost of Racing in this experiment is around 9.75% as all 20 candidates needed to be tested for at least 5 trials and the best one needed to be fully tested for 100 trials.

Next, we will show how the GA's performance can be further improved through the simultaneous tuning of all six parameters.

In the second case study, a $(\mu+\lambda)$ ES with $\mu=\lambda=20$ was employed as the Meta-EA (i.e., the initial population was randomly generated) in which each individual consisted of three parameters P, P_c and P_m and represented a total of 66 algorithm instances (i.e., 11 selection strategies $\times$ 3 crossover operators $\times$ 2 options of elitism). Note that although E is a binary parameter, it was still included in the Racing so that the Meta-EA only needed to handle a continuous optimization problem. As a result, the role of Racing here was to, for each individual, find out the best algorithm instance out of a set of 66 candidates as efficiently as possible and return its performance as the fitness

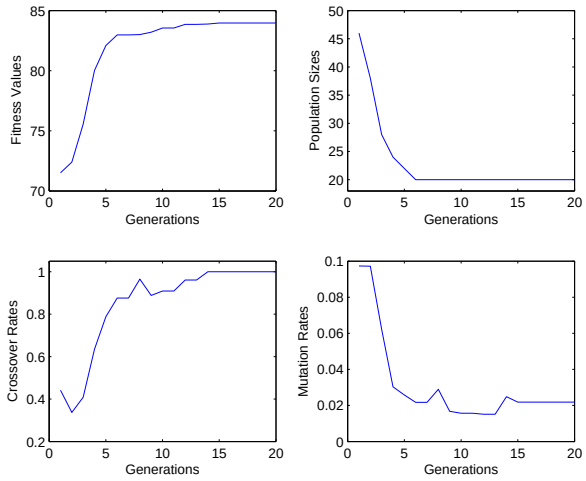

Fig. 11. The evolution process of the hybrid (1+λ) ES + Racing scheme.

value of that individual. Also, the best algorithm was to be fully tested even if it was the only one left.

Figure 12 presents the evolution process of this hybrid scheme in which all results were averaged over the population. It is clear that much better parameter settings were found while the superiority of small population sizes, large crossover rates and small mutation rates was again evident. Furthermore, the uniform crossover dominated the final population while a strong selection pressure was created by either the Tournament selection with tournament size 6 or the Truncation selection with selection ratio 0.2. As to Elitism, it was not shown to be beneficial in this case. The cost of Racing was only around 7.4% of the cost of the brute force method, had it been applied to find the best algorithm instances.

6 Summary

In this Chapter, the major issue addressed is how to efficiently find the best parameter settings for EAs through experimental studies, assuming no specific knowledge on those parameters. Two classes of tuning techniques: search-based methods and statistical Racing techniques, which are based on distinctly different mechanism, are given a critical review on their strengths and weaknesses.

We point out that the applicability of Racing is inherently restricted by its lack of exploration capability while it could be very difficult for typical search-based methods such as Meta-EAs to efficiently handle nominal parameters

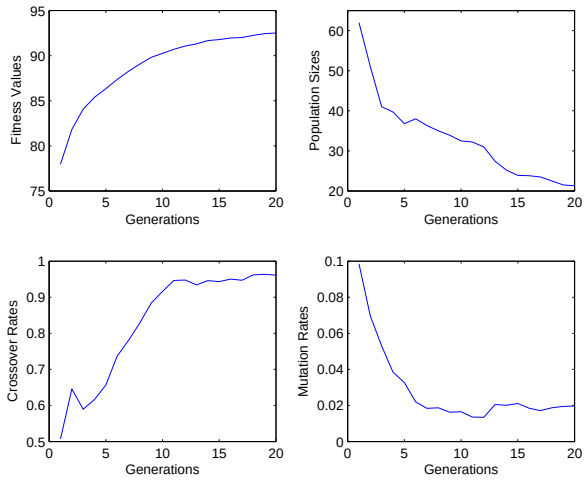

Fig. 12. The evolution process of the hybrid Meta-EA + Racing scheme.

that only have symbolic meanings. In order to take the advantages of both methods, two hybrid schemes are proposed in which search-based methods are responsible for exploring the parameter space while Racing is applied, in the place of the brute force method, to significantly reduce the cost associated with finding the best performing algorithm instance from a moderate size of candidates.

In the case studies, Racing has shown to be able to reduce the cost of the brute force method by around 90% while maintaining quite high reliability. The major direction for future work would be to establish some theoretical framework for Racing in order to conduct principled analysis of the influence of various experimental factors. It is also an important topic to investigate the possibility of combining more features into Racing from other tuning techniques.

Finally, some rules of thumb are given as follows to assist in choosing appropriate parameter tuning techniques:

1. If the size of the parameter space is small (e.g., ten candidates or even less), it is usually straightforward and reliable to conduct an exhaustive search to find the best parameter setting.
2. If the size of the parameter space is moderate (e.g., up to a few hundreds of candidates), Racing could be applied to significantly reduce the computational time required, regardless of whether the parameter space is searchable or not.
3. If the sizes of the searchable parameter space and non-searchable parameter space are both large, it is worthwhile to consider the proposed scheme of Meta-EA + Racing.

4. If the size of the searchable parameter space is large while there is no nominal parameter or the size of the non-searchable parameter space is small, it is possible to use Meta-EAs only (i.e., apply random/exhaustive search on nominal parameters if necessary).
5. An exception of rule No. 4: if the Meta-EA in use is similar to the $(1 + \lambda)$ or $(1, \lambda)$ ES, which requires choosing the best individual from a set of candidates, it is still possible to consider the scheme of Meta-EA + Racing.

References

1. T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.
2. T. Bartz-Beielstein. Experimental analysis of evolution strategies - overview and comprehensive introduction. Technical Report Reihe CI 157/03, SFB 531, Universität Dortmund, 2003.
3. M. Birattari, T. Stutzle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 11–18, 2002.
4. P. A. Castillo-Valdivieso, J. J. Merelo, and A. Prieto. Statistical analysis of the parameters of a neuro-genetic algorithm. *IEEE Transactions on Neural Networks*, 13(6):1374–1393, 2002.
5. A. Czarn, C. MacNish, K. Vijayan, B. Turlach, and R. Gupta. Statistical exploratory analysis of genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(4):405–421, 2004.
6. B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.
7. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.
8. J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on System, Man, And Cybernetics*, 16(1):122–128, 1986.
9. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
10. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, 1995.
11. K. De Jong. *The Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.
12. O. Maron and A. W. Moore. Hoeffding races: accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems 6*, pages 59–66, 1994.
13. O. Maron and A. W. Moore. The racing algorithm: model selection for lazy learners. *Artificial Intelligence Review*, 11(1):193–225, 1997.
14. D. S. Moore. *Introduction to the practice of statistics*. W. H. Freeman, New York, 4th edition, 2003.
15. T. K. Paul and H. Iba. Real-coded estimation of distribution algorithm. In *Proceedings of the 5th Metaheuristics International Conference (MIC2003)*, pages 61–66, 2003.

16. I. Rojas, J. Gonzalez, H. Pomares, J. J. Merelo, P. A. Castillo, and G. Romero. Statistical analysis of the main parameters involved in the design of a genetic algorithm. *IEEE Transactions on Systems, Man, And Cybernetics-Part C*, 32(1):31–37, 2002.
17. D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
18. F. Y-H. Yeh and M. Gallagher. An empirical study of heoffding racing for model selection in k-nearest neighbor classification. In J. Hogan M. Gallagher and F. Maire, editors, *Sixth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'05)*, volume 3578 of *Lecture Notes in Computer Science*, pages 220–227. Springer, 2005.
19. B. Yuan and M. Gallagher. Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In X. Yao et al., editor, *Proc. Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 172–181. Springer, 2004.