Information Sciences 178 (2008) 2857–2869

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A genetic algorithm calibration method based on convergence due to genetic drift

Matthew S. Gibbs *,[1], Graeme C. Dandy, Holger R. Maier

School of Civil, Environmental and Mining Engineering, The University of Adelaide, Adelaide SA 5005, Australia

ARTICLE INFO

Article history:
Received 3 February 2008
Received in revised form 18 March 2008
Accepted 27 March 2008

Keywords:
Genetic algorithms
Calibration
Parameter estimation
Optimization
Genetic drift

ABSTRACT

The selection of Genetic Algorithm (GA) parameters is a difficult problem, and if not addressed adequately, solutions of good quality are unlikely to be found. A number of approaches have been developed to assist in the calibration of GAs, however there does not exist an accepted method to determine the parameter values. In this paper, a GA calibration methodology is proposed based on the convergence of the population due to genetic drift, to allow suitable GA parameter values to be determined without requiring a trial-and-error approach. The proposed GA calibration method is compared to another GA calibration method, as well as typical parameter values, and is found to regularly lead the GA to better solutions, on a wide range of test functions. The simplicity and general applicability of the proposed approach allows suitable GA parameter values to be estimated for a wide range of situations.

1. Introduction

The fact that Genetic Algorithms (GAs) are guided search methods that are capable of carrying out optimization in conjunction with any simulation model has allowed them to be adopted in a wide range of areas, such as engineering, numerical optimization, robotics, classification, pattern recognition and design [17]. However, before a GA can be applied, the user must specify a number of parameter values, such as population size, probability of crossover, and probability of mutation. Several studies have shown that a GA can tolerate some variation in its parameter values without dramatically affecting its overall performance [20]. However, even though the GA is quite robust with respect to some parameters settings, this does not imply that the GA will work equally well regardless of the settings. The No Free Lunch Theorem [39] stipulates that a global set of optimization algorithm parameters that will be effective for every problem does not exist.

A number of approaches have been proposed to assist in the setting of GA parameter values. The different methods typically fall into three categories: (i) empirical experiments; where results from GA experiments are used to develop empirical relationships, (ii) parameter control; where the GA parameter values are changed as the GA is solving the problem, and (iii) dimensional analysis; where simple models are used to represent how certain aspects of a GA will behave. Each approach has a number of drawbacks; either the results do not generalize to other fitness functions, additional parameters are introduced to control the GA parameters, or the models used to represent the GA over-simplify reality. Hence, there still does not exist an accepted approach to determining GA parameter values.

* Corresponding author. Tel.: +61 8 8303 3303; fax: +61 8 8383 4359.
E-mail addresses: mgibbs@civeng.adelaide.edu.au (M.S. Gibbs), gdandy@civeng.adelaide.edu.au (G.C. Dandy), hmaier@civeng.adelaide.edu.au (H.R. Maier).

[1] Supported in part by an Australian Postgraduate Award from the Australian Commonwealth Department of Education, Science and Training, and in part by the Cooperative Research Centre for Water Quality and Treatment, Project 2.5.0.3.

0020-0255/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2008.03.012

In this paper, a simple equation is proposed to allow the most suitable GA population size to be determined, based on when the population is expected to converge due to genetic drift. The only information required to determine the population size is the problem size (i.e. the length of the GA string) and the number of function evaluations that are available before a solution is required. The approach is applicable to any algorithm that implements two parent tournament selection, and the other operators do not significantly alter the population variance. Relationships are also proposed for the remaining GA parameter values, resulting in a complete GA calibration methodology.

Section 2 provides an overview of the background relevant to GA calibration. The derivation of the relationship to determine the population size is given in Section 3, before an investigation into suitable relationships for the remaining GA parameters is undertaken in Section 4, producing a complete GA calibration methodology. A comparison of the proposed GA calibration method with other methods that are available is given in Section 5, and a discussion of the results and concluding remarks are made in Sections 6 and 7, respectively.

2. Background

Typically, GA parameter values are found using a trial-and-error approach. This involves manually tuning the parameter values, where all parameters are initially given typical values, and then one parameter at a time is changed until a suitable set of values is identified. However, the GA operators will interact with each other, and therefore the behavior produced by the value of one parameter will be dependent on all the others. Due to computational requirements, only a limited number of parameter combinations can be trialled; therefore it is unlikely that a trial-and-error approach will identify parameter values close to those that will produce the best performance from the selected algorithm.

A number of alternate approaches have been developed in an attempt to provide information on the best GA parameters for a given problem. The different methods typically fall into three categories: (i) empirical experiments; where results from GA experiments are used to develop empirical relationships, (ii) parameter control; where the GA parameter values are changed as the GA is solving the problem, or (iii) dimensional analysis; where simple models are used to represent how certain aspects of a GA will behave. Each approach is considered in more detail in this section.

2.1. Empirical experiments

The simplest method for determining useful relationships considering all aspects of a GA is to perform empirical analyses on a range of test functions [6]. Empirical relationships provide useful results for the situations they have been applied to, however, the applicability of the results is limited to the situations that have been considered, and may not extend to the general case.

2.2. Parameter control

Recently, a number of feedback based adaptive methods have been developed to alter the population size [1,2,9,33]. [9] proposed a method to adjust the population size based on how frequently improvements in the best solution are found. They concluded that their method was the most effective population adjustment method available. However, the method introduces four more parameters that must be calibrated, including the increase and decrease factors and minimum and maximum population sizes, potentially adding to the difficulty of the GA calibration problem, rather than helping to solving it. [33] used feedback from the optimization process to alter the number of solutions in each population of a multi-population GA, however the total number of solutions in the populations was set arbitrarily. A feedback based approach to determine the probability of crossover and mutation, based on the entropy of the population, was used by [30], however the population size was again set arbitrarily. These studies highlight the difficulty in determining a suitable population size for a GA. Feedback based approaches such as these are not specific to the field of GAs. For example, population variation has been used for Genetic Programming [19], where the number of generations and fitness function values are used to control the population size, and minimize the computational effort required to find suitable solutions.

Setting GA parameter values can be considered a poorly structured, poorly defined, complex problem. GAs are generally considered to perform better than other methods on these types of problems [8]. Consequently, there may be an advantage in allowing a GA to calibrate itself. This can be achieved by incorporating the GA parameters into the solution to the problem, so each individual in the population contains values for the set of decision variables for the optimization problem, as well as a set of parameter values for the GA. As the optimization progresses, the good solutions are retained in the population, as well as the good parameters that produced these solutions, enabling the algorithm to find better solutions with the most effective parameters. However, this approach is only suitable for parameters applied on an individual scale, such as the probability of mutation and crossover, and it is not clear how to apply the method for parameters that are applied at a population scale, such as the population size or number of elite solutions.

2.3. Dimensional analysis and theoretical modeling

A number of theoretical modeling studies have been performed to provide an insight into the complex mechanisms at work in a GA run, to assist with the calibration of certain parameters. To enable any meaningful results to be drawn,

generally the models can consider the effect of only one or two aspects of a GA. Some of the more important studies consider convergence due to genetic drift [25,29], convergence due to selection [36], the interaction between selection and crossover [20], population sizing [14], and the effect of mutation alone [3,24].

Goldberg et al. [13] and Harik et al. [14] modeled the supply of good combinations of decision variable values, or building blocks, in the initial population, along with making the correct decision between building blocks during a GA run, to determine a relationship for the optimal population size. The analogy uses the gambler's ruin model in the analysis, and the following relationship was obtained [14]:

$$N = -2^{k_{BB}-1}(\ln \alpha)\frac{\sigma_{BB}\sqrt{\pi m_{BB}}}{d}, \tag{1}$$

where N is the optimum population size, k_{BB} is the building block size, α is the probability that the GA selects a sub-optimal building block, σ_{BB} is the building blocks' fitness variance, m_{BB} is the number of building blocks, and d is the building blocks' fitness signal. The relationship was validated experimentally for relatively simple test functions where the required parameter values could be determined. However, generally this is not the case, and little is known about the parameters in (1) for most optimization problems.

As GAs are derived from observations about natural selection, it is not surprising that results from Quantitative Genetics have been applied to the GA field. One of the most common relationships used in GA theories is the Genetical Theory of Natural Selection [11]. This relationship has been used in a number of studies [20,26,29,36] to predict the convergence of GAs. Further results from Quantitative Genetics may be useful to explain the dynamic behavior of GAs. Due to the influence of selection, σ_P changes from generation to generation in accordance with $\sigma'_P = k\sigma_P$ [10], where k is a constant with $0 < k < 1$. Due to the repetitive application of a selection operator in GAs, the population will eventually converge to a single solution. This process is termed *genetic drift*. Rogers and Pruegel-Bennett [29] present a method of calculating the rate of genetic drift in terms of the change in population fitness variance, $\sigma'_P = k\sigma_P$. k is found in terms of the population size, N, and for a simple GA with tournament selection $k = 1 - \frac{1}{N}$.

Based on the results derived from a number of dimensional analysis studies, [15] proposed the 'Parameterless GA'. Lobo [20] observed that, according to the Schema Theorem [18], the growth of building blocks is given by $s(1 - p_c)$, where s is the selection pressure and p_c is the probability of crossover. Therefore, the Parameterless GA implements $s = 4$ and $p_c = 0.5$, producing a net growth of building blocks of 2. To remove the population sizing problem from the user, the initial population size is set to a very small value, such as $N = 4$, and after the population converges, the population size is doubled and reinitialized, with a copy of the best solution found in the previous GA run inserted into the new population. The process is then repeated until no improvement in solution quality is observed from one GA run to the next, or a user specified criterion is met. Minsker [23] included a mutation operator in the approach, with the probability of mutation taking on the larger value of $1/l$ (where l is the length of the solution string), as proposed by [24,3], and $1/N$.

Reed and Yamaguchi [27] proposed a similar calibration methodology for real valued decision variables, specifically Differential Evolution [34], however, the authors suggest that the approach can be easily adopted for any Real Coded GA (RCGA). However, a number of studies [5, for example], have found that one GA run with a large population sizes is more beneficial than a number of GA runs with smaller population sizes, and Eiben et al. [9] found that the Parameterless GA was much slower than a traditional GA with a constant population size on a number of multi-modal problems.

From the review of literature presented, it is evident that there does not exist an accepted method to determine a suitable GA population size, without resorting to a trial-and-error approach. Either: (i) generally unknown parameters must be quantified, as in (1), (ii) a deterministic rule, such as doubling the population size each GA run must be used, without any consideration of the performance of the algorithm, or (iii) the population size needs to be adapted based on the performance of the algorithm, albeit with the introduction of more parameters that must be calibrated to control the adaptation. In order to address this problem, a relationship is proposed to determine the largest population size that can be expected to converge due to genetic drift. This is followed by an empirical study to identify relationships and suitable values for the remaining GA parameters, resulting in a complete GA calibration method.

3. Convergence due to genetic drift

As outlined in Section 2, a number of GA modeling studies have made use of the Genetical Theory of Natural Selection [11]. The rather intuitive result of this relationship is that there will be no more improvement in the fitness function values when there is no more diversity in the population.

The selection operator will decrease the variance in the fitness function values, by replacing the worst solutions by better ones. As outlined in Section 2, Quantitative genetic theory indicates that this decrease will be a constant value, k, each generation [10]. This results in an exponential decay of the variance in the fitness function values in terms of the number of generations. Therefore, the time until the GA converges to a single solution is dependent on the rate of decay of the population variance, and can be computed by

$$\begin{aligned}
&\sigma_{P,g} = \sigma_{P,0} k^g, \\
&\therefore \log\left(\frac{\sigma_{P,g}}{\sigma_{P,0}}\right) = g\log(k), \\
&\therefore g = \frac{\log(\sigma_{P,g}) - \log(\sigma_{P,0})}{\log(k)},
\end{aligned} \tag{2}$$

where $\sigma_{P,g}$ is the standard deviation of the fitness function values after g generations, $\sigma_{P,0}$ is the standard deviation of the fitness function values of the initial population, and k is a constant with value $0 \leqslant k \leqslant 1$. Given a specified number of function evaluations before a solution is required, the population size can be determined as $N = FE/g_{\text{conv}}$, where g_{conv} is the number of generations when convergence occurs. In order to use this relationship, expressions for k, $\sigma_{P,0}$, and $\sigma_{P,g}$ are required.

Beyer and Deb [4] proposed that the population standard deviation, σ_p, can be computed by

$$\sigma_p = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{l}(x_{i,j} - \bar{x}_j)^2}, \tag{3}$$

where $x_{i,j}$ is the value of the jth decision variable in the ith member of the population. All variables are standardized into the range $[0, 1]$ to ensure that one decision variable that happens to occur over a larger range does not dominate the computed value of σ_p. For a real coded GA, the decision variable values are randomly sampled from a uniform distribution over the search space, to initialize the population. Hence, the expected value of the variance for each decision variable is $\sigma^2 = \frac{(1-0)^2}{12} = \frac{1}{12}$. The approach used to compute the population variance, given in (3), involves summing the variance over each decision variable and hence the initial standard deviation in the population, $\sigma_{p,0}$, is given by

$$\sigma_{p,0} = \sqrt{\frac{l}{12}}. \tag{4}$$

It would be expected that g_{conv} occurs when there is no more variance in the population, occurring when $\sigma_{p,g} = 0$. However, in practice, the term $\log(0)$ cannot be computed in (2), and an approximation to signify when the population has adequately converged must be adopted. In this work, the GA has been deemed to have converged once $\sigma_{p,g} = 10^{-M}$.

Rogers and Pruegel-Bennett [29] presented a method of calculating the rate of genetic drift in terms of the change in population fitness variance, $\sigma'_p = k\sigma_p$. k is found in terms of the population size, N, and for a simple GA with tournament selection, $k = 1 - \frac{1}{N}$. By substituting these relationships into (2), the value of N can be obtained:

$$\begin{aligned}
&\frac{FE}{N} = \frac{\log(10^{-M}) - \log\left(\sqrt{\frac{l}{12}}\right)}{\log\left(1 - \frac{1}{N}\right)}, \\
&\therefore \frac{FE}{N}\log\left(1 - \frac{1}{N}\right) = -M - \log\left(\sqrt{\frac{l}{12}}\right).
\end{aligned} \tag{5}$$

Therefore, by solving the implicit equation (5) for N, the largest population size that will converge due to genetic drift in the number of function evaluations that are available for a given problem size can be determined. However, it should be pointed out that in many cases, GA convergence will occur due to selection pressure before it will occur due to genetic drift. Convergence due to selection pressure is dependent on the fitness function itself, and therefore is much more difficult to quantify. Hence, the population size computed by (5) can be considered to provide a minimum bound on the population size, where even under no selection pressure the GA can be expected to converge in the given number of function evaluations.

This estimate of the best population size to use for a given problem is very useful, however it is only one of the many GA parameter values that must be determined before the algorithm can be applied. Hence, the following section adopts an empirical approach to determine the best parameter values to use, before the proposed calibration method is outlined.

4. The relationship between GA parameters

The aim of this section is to provide guidelines to set the GA parameters, other than the population size (which can be determined using (5)). It is likely that the most suitable GA parameter values are related to each other. To determine if any relationships exist between the best values of the GA parameters, a large-scale parametric study was undertaken to identify the best performing combinations of parameter values. The resulting combinations of parameter values were then interrogated to determine if any relationships were present between the values. The following sections outline: (i) the algorithm used, (ii) the test functions adopted, and (iii) the method adopted. Finally, the results are presented.

4.1. Genetic algorithm

The GA used in this research uses real coding to represent the decision variables, as all the optimization problems considered have real valued decision variables. This approach has avoided any further complexities being added to the problem by the encoding scheme, such as Hamming Cliffs, or discretization issues. To allow (5) to be adopted, tournament selection

with a tournament size of two has been used [22]. An elitist strategy is used to preserve good solutions, where the worst solution in the population each generation is replaced by the best solution found so far.

A one-point distributed crossover operator has been used, as neighborhood-based crossover operators such as this have been found to exploit the numerical nature of Real Coded GAs (RCGAs) [16]. The crossover operator used for this work is similar to the Simulated Binary Crossover operator [7] and the Fuzzy Recombination Crossover operator [37]. If crossover occurs, a random crossover point is generated, and a new value of each decision variable is generated from a normal distribution centered on one parent's solution values, p_1, before the crossover point, or centered on the second parent's solution values, p_2, after the crossover point, as seen in Fig. 1. The spread of the distribution is determined from a parameter related to the distance between the two parent solutions, c.

A uniformly distributed mutation operator has been adopted, producing any value over the range of each decision variable. A Gaussian mutation operator is commonly used for RCGAs, however, this would provide a very similar search mechanism to the crossover operator adopted for this work. This operator has been selected as it has the potential to generate values all over the entire search space and it does not introduce additional parameters to be calibrated.

Therefore, for the operators selected, the following parameters values are required: population size, probability of crossover, probability of mutation, number of elite solutions, and standard deviation of the distribution used for the crossover operator. In this work the probability of mutation, p_m, used is the probability that mutation is applied to a solution string, rather than a decision variable in the solution string, p'_m. If mutation is applied, the decision variable to be mutated in that string is chosen randomly. Therefore, $p_m = 1$ corresponds to the empirical mutation rate for a binary-coded GA under bitwise mutation, $p'_m = 1/l$ [24].

In order for (5) to be applicable, selection must be the only GA operator to significantly alter the population variance. As the crossover operator is based on a normal distribution centered around the parent values, on average it would not be expected to change the variance of the population. The remaining GA operators, mutation and elitism, are applied at very low probabilities, and therefore are also not expected to alter the population variance significantly. Therefore, (5) is expected to apply to this GA, and to extend to any RCGA with two parent tournament selection, provided the remaining GA operators do not significantly alter the population variance.

4.2. Test functions

Table 1 lists the test problems that have been adopted for this research. F1 is the sphere function [6], F2 is the sine-envelope sine-wave function [31], F3 and F4 are functions proposed in [28], F5 is the Rastrigin Function [26], and F6 is a variation of Griwank's Function [38]. These functions have been selected as they provide a diverse mix of problem characteristics, including: structure toward the optimal solution, such as 'big bowl' problems (e.g., F1 and F5), compared with relatively flat search spaces (F2); differently scaled contributions from each decision variable (F3 and F4); and separable (e.g., F1 and F3) and non-separable functions (F2 and F6). All functions are simply scaled into higher dimensions, with the exception of F2, which has been extended into higher dimensions by summing sub-functions of the original 2-dimensional problem.

Along with the common benchmark optimization problems, F4, F5, and F6 have had parameters introduced to allow their characteristics to be altered. These parameters allow for different degrees of epistatic interaction, roughness in terms of the size and frequency of the local optima, and the scaling of the variables to all be considered. By changing these parameters, the size of the set of test functions is increased, and by considering many different function characteristics the results are more likely to generalize to other situations. Each function has been constructed using the guidelines proposed by [38], which allow non-linear, non-separable, scalable functions to be developed, providing a realistic challenge to any optimization algorithm. F5A allowed for changes in the degree of epistasis of the problem. All practical combinations of 1, $l/2$ and l interactions between decision variables that are 1, $l/2$ and $l-1$ positions apart in the solution string have been considered,

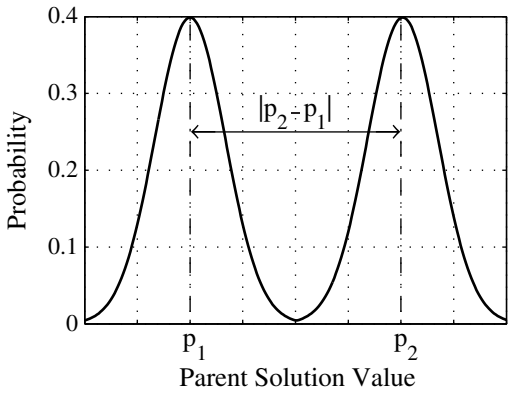

Fig. 1. The distribution for the crossover operator implemented.

Table 1
The test functions used in the parametric study

Function to be minimized	
F1	$\sum_{i=1}^{l} x_i^2$
F2	$\frac{l}{4} + \sum_{i=1}^{l/2} \left(\frac{\sin^2\left(\sqrt{x_{2i-1}^2 + x_{2i}^2}\right) - 0.5}{1 + 0.001\left(x_{2i-1}^2 + x_{2i}^2\right)}\right)$
F3	$\sum_{i=1}^{l} i\lvert x_i\rvert$
F4A	$\sum_{i=1}^{l-1} i^A x_i + x_l^p$
F4B	$\sum_{i=1}^{l-2} i^A \left(\frac{x_i + x_{i+1}}{2}\right) + \left(\frac{x_{l-1} + x_l}{2}\right)^p$
F5A	$\sum_{i=1}^{l}\sum_{j=i}^{l} \begin{cases} \left(\frac{x_i + x_j}{2}\right)^2 - B\cos\left(f\pi\frac{x_i + x_j}{2}\right) + B & \text{if } i,j \text{ interact} \\ 0 & \text{otherwise} \end{cases}$
F5B	$\sum_{i=1}^{l-1}\left(\left(\frac{x_i + x_{i+1}}{2}\right)^2 - B\cos\left(f\pi\frac{x_i + x_{i+1}}{2}\right)\right) + x_l^2 - B\cos(f\pi x_l) + Bl$
F5C	$\sum_{i=1}^{l-1}\left(\left(\frac{x_i + x_{i+1}}{2}\right)^2 - 10\cos\left(2\pi\frac{x_i + x_{i+1}}{2}\right)\right)^{p_i} + x_l^2 - 10\cos(2\pi x_l) + 10l$
F6A	$\frac{1}{4000}\sum_{i=1}^{l} i^p x_i^2 - C\left(\prod_{i=1}^{l}\cos\left(\frac{x_i}{\sqrt{i}}\right) - 1\right)$
F6B	$\frac{l^p x_l^2}{4000} + \frac{1}{4000}\sum_{i=1}^{l-1} i^p \left(\frac{x_i + x_l}{2}\right)^2 - C\left(\cos\left(\frac{x_l}{\sqrt{l}}\right)\prod_{i=1}^{l-1}\cos\left(\frac{\left(\frac{x_i + x_l}{2}\right)}{\sqrt{i}}\right) - 1\right)$

producing a total of six versions of F5A that were tested. The influence of adding more epsitatic interactions between decision variables can also be seen between F4A and F4B, as well as F6A and F6B.

Variations in the multi-modality and roughness of the Rastrigin function have been considered in function F5B. Values of $B = 10, 20$, and $f = 2, 5$ have been considered, where all combinations of B and f have been tested. An increase in the value of B will produce an increase in the roughness of the function. The effect of an increase in the value of f is an increase in the multi-modality of the function. Similarly, the C parameter can be used to control the roughness of F6, and values of $C = 2, 3$ have been considered.

Variations in the salience of the decision variables for the Rastrigin Function, or the contribution of each variable to the fitness function values, have been considered in the different cases considered for F5C in changes to the p_i parameter. The following four cases have been tested: (a) $p_i = 1$ for all decision variables, (b) $p_1 = 2$, (c) both $p_1 = 2$ and $p_2 = 2$, and (d) $p_i = 2$ for all decision variables. The p parameter can be used to alter the salience of the variables of F6, and values of $p = 2, 3$ have been considered. From Table 1 it can be seen that the parameter A can be used to change the size of the incremental dominance of each adjacent variable for F4. Parameter p can be used to control the salience of the last decision variable in the solution string for F4, x_l, by altering the magnitude of the order of its relationship with the fitness function value. Values of $A = 1$, l and $p = 0$, 1 and 2 have been used in the study.

The interval of $[-5.12, 5.12]$ has been considered for each decision variable for each function, with the exception of F2 and F3, where the interval $[-100, 100]$ has been used. Problems sizes of l = 5, 10, 20, and 30 have been considered for every variation of each function. Every combination of each of the parameter values considered for F4, F5 and F6 have been tested. Therefore, a total of 148 functions have been used to identify relationships between GA parameter values.

4.3. Parametric study

To determine the most effective GA parameter values, a large-scale parametric study has been undertaken. Population sizes of N = 10, 50, 100 and 200 have been considered. Based on each population and problem size considered, the appropriate stopping criterion, *FE*, was determined from (5) with $M = 3$, as seen in Table 2. The values tested for the remaining GA parameters are given in Table 3. Values of p_m between 0 and 1 at increments of 0.2 have been considered, to test a wide range of mutation probabilities. Generally, higher probabilities of crossover are found to perform well, hence values of p_c = 0.7, 0.85 and 1 have been included. For the distance for the standard deviation of crossover, a value of $c = 6$ has been tested, as it provides a normal distribution of three standard deviations around each parent value. Shorter distances have not been tested, as for this case offspring solutions are more likely to be produced far away from the parent solutions, hence the operator would behave more like mutation than crossover. A value of $c = 18$ produces offspring solutions much closer to the parent

Table 2
Stopping criteria, *FE*, for combinations of population size, *N*, and problem size *l*

l	*N*			
	10	50	100	200
5	1000	16,000	64,000	258,000
10	1000	17,000	68,000	272,000
20	1000	18,000	71,000	286,000
30	1000	18,000	73,000	294,000

Table 3
The GA parameter values used for the parametric study

Parameter	Values
Probability of mutation, p_m	0, 0.2, 0.4, 0.6, 0.8, 1.0
Probability of crossover, p_c	0.7, 0.85, 1.0
Distance for the standard deviation of crossover, c	6, 18
Elite solutions per generation, e	0, 1

solutions, and has also been included in the parametric study. Finally, the GA both with ($e = 1$) and without ($e = 0$) elitism has also been considered.

Each combination of parameter values was tested 30 times with different sequences of random numbers to evaluate the performance. This resulted in a total of 8640 GA runs for each problem size considered for each function. As 148 functions have been considered, a total of 1,278,720 separate GA runs were required to produce the results presented.

The average and standard deviation of the fitness function values of the best solution found over the 30 different GA runs for each GA parameter set were recorded after the relevant *FE* (Table 2) for each function. A 2-tailed Student's *t*-test with a 95% confidence interval was used to compare parameter sets, and identify those that were significantly better for each function in each problem size. The combinations of parameter values that produced the best results were stored, to allow any relationships between the best performing parameter values to be identified.

4.4. Results

Mutual information (MI) has been used to determine if any relationships exist between the best performing GA parameter values identified from the parametric study. MI is a non-linear correlation static, based on the concept of Entropy [32]. The MI was computed between the values found for each pair of GA parameters to identify any relationships. If the best values for two GA parameters are related to each other then there will be a high MI between their values, otherwise if the parameter values are independent of each other, the MI computed will be close to zero. The MI was computed between every pair-wise combination of the GA parameters, to determine if there were any relationships between their best performing values. The results from this analysis are presented in Table 4, where each pair of GA parameters have been ranked by their MI value, as a higher value indicates a stronger relationship. The values in brackets are the 95% confidence levels produced from 1000 bootstraps of the data for each pair of parameters. This involved computing the MI between a random reordering of the data, removing any true interaction between the parameters. This process was repeated 1000 times, and the MI value that was greater than the MI computed for a random reordering between the parameter values 95% of the time is presented in brackets in Table 4. Therefore, if the MI computed between two GA parameters is less than or equal to the significance level, it can be concluded that the interaction between the parameters is not significant, and is due to chance alone.

The results presented in Table 4 indicate that there exists a strong interaction between the best values for the population size and probability of mutation for the functions considered, with $I(N, p_m) = 0.313$. The interactions between the remaining pairs of GA parameters are less significant than that between the population size and the probability of mutation, with the next highest mutual information being between the population size and the number of elite solutions, with $I(N, e) = 0.109$. However, most MI values are greater than the significance levels computed from bootstrapping the data, indicating that there is a weak relationship between all parameters, apart from the population size and the probability of crossover. The relationships between the GA parameters and the frequency of occurrence of each of the GA parameter values tested are considered in more detail in the remainder of this section.

4.4.1. Probability of mutation

A number of previous studies [31, for example], have observed a relationship between the population size and the most suitable value for the probability of mutation. The MI results presented in Table 4 confirm this relationship, with the highest

Table 4
GA parameter interaction results

Rank	X_1	X_2	$I(X_1, X_2)$	Significance level
1	N	p_m	0.313	0.003
2	N	e	0.109	0.001
3	p_m	e	0.105	0.001
4	p_m	c	0.029	0.001
5	p_m	p_c	0.013	0.002
6	N	c	0.007	0.001
7	e	c	0.007	0.001
8	p_c	c	0.006	0.001
9	p_c	e	0.003	0.002
10	N	p_c	0.002	0.002

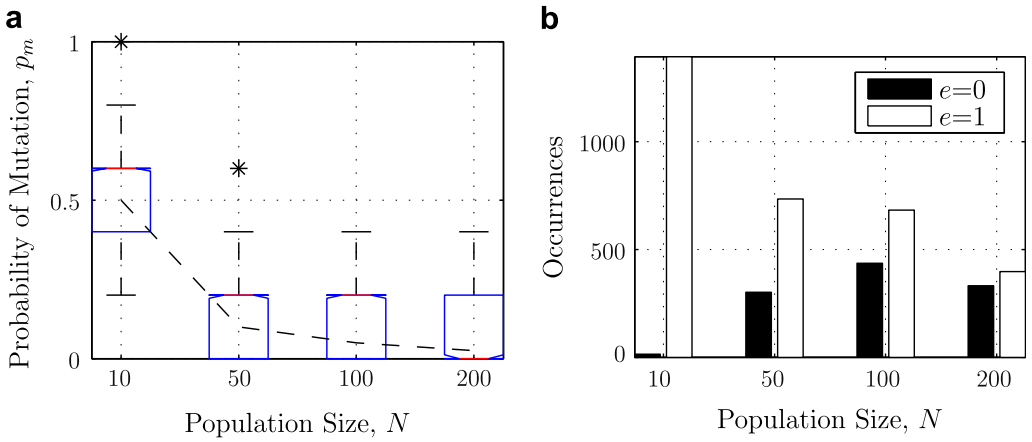

Fig. 2. (a) Probabilities of mutation occurring for each population size, and (b) the number of elite solutions for each population size.

values of $I(N, p_m) = 0.313$. From the combinations of GA parameter values that located the best solutions, a boxplot of the probabilities of mutation which coincided with each population size is presented in Fig. 2a. In Fig. 2a, the stars represent the combination of parameter values that fall outside the upper and lower quartiles represented on the box plot, however, as they were observed in the parametric study, they have been included on the plot as outliers.

From Fig. 2a, it can be seen that there is indeed a strong inverse relationship between N and p_m in this dataset, with large p_m only suitable for small N, and *vice versa*. Therefore, it is proposed that the probability of mutation be calculated from the population size, which is first determined by (5). The dashed line in Fig. 2a is the line $p_m = 5/N$, and it can be seen that this relationship provides a good fit to the box plot for each population size. As the probability of mutation used in this work is the probability of mutating each solution string in the population, this relationship suggests that there should be five mutations per generation, irrespective of the population size or dimension of the problem.

4.4.2. Elitism

The next strongest relationship identified between the GA parameters by the MI analysis was between the number of elite solutions, e, and population size, and the number of elite solutions and the probability of mutation, with $I(N, e) = 0.109$ and $I(p_m, e) = 0.105$, respectively. Given that N and p_m were strongly related to each other, if one parameter is related to the elitism operator, it is not surprising that both parameters are related to the elitism operator.

The number of occurrences for a GA with ($e = 1$) and without elitism ($e = 0$) for different population sizes is presented in Fig. 2b. It can be seen from Fig. 2b that $e = 1$ performs the best with all the population sizes considered in the parametric study. The difference is largest for the smaller population sizes, which also perform best with high mutation rates, where good solutions are easily lost without an elitist operator. Therefore, $e = 1$ is used in the GA calibration methodology proposed in this work.

4.4.3. Standard deviation of crossover

The GA parameter that had the strongest relationship with the standard deviation of the crossover operator was the probability of mutation, with $I(p_m, c) = 0.029$, as seen in Table 4. Both these GA parameters control the amount of variation of the population, as both a high probability of mutation and a smaller fraction of the distance between the parent values for c will generally produce solutions that are further away in the search space from their parent solutions. However, the mutual information value between these two parameters was quite low, indicating the relationship is not very strong, but is still an order of magnitude higher than the 95% significance level of 0.001.

The number of occurrences for each value of c considered for each mutation rate in the parametric study is presented in Fig. 3a. As was the case for the elitism parameter, it can be seen from Fig. 3a that $c = 6$ performed the best for all p_m considered.

4.4.4. Probability of crossover

The final GA parameter value that must be determined before the GA can be applied is the probability of crossover. The MI results indicate that this parameter also had the strongest relationship with the probability of mutation, with $I(p_m, p_c) = 0.013$. While this MI value is greater than the corresponding significance level, the MI values suggest that the relationship is very weak, and unlikely to effect the performance of the GA.

The number of occurrences of each probability of crossover for each probability of mutation considered in the parametric study is given in Fig. 3b. The results indicate that $p_c = 1$ produced the best results for all values of p_m, and the MI analysis indicates that the parameter was not significantly related to the value of any of the other GA parameters. Therefore, $p_c = 1$ has been adopted for the GA calibration methodology. This result implies that every solution in the population is altered

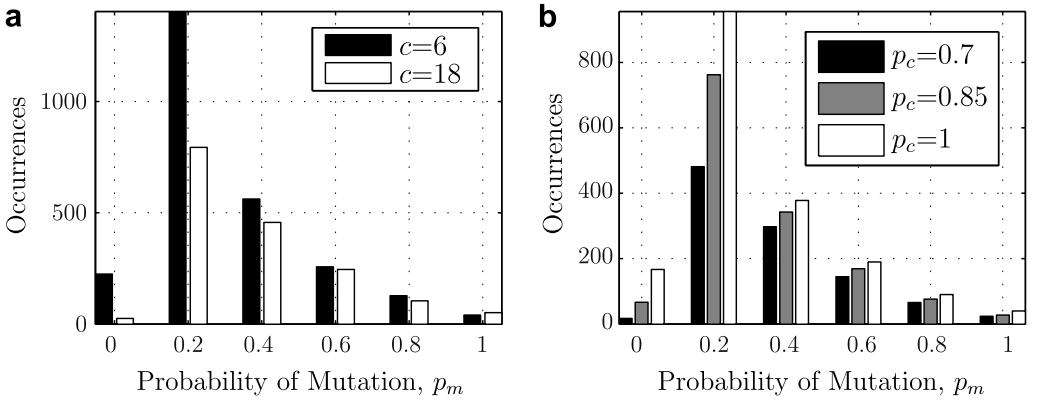

Fig. 3. (a) Standard deviation of crossover, and (b) probability of crossover for each probability of mutation.

every generation by crossover, therefore making the most efficient use of *FE* available. Conversely, $p_c < 1$ will result in a number of solutions being re-evaluated every generation, provided a solution tracking operator is not implemented to ensure that only solutions that change from one generation to the next are evaluated.

4.5. Summary

By combining the results presented so far in this paper, a complete GA calibration methodology is proposed. The approach is summarized as follows:

(i) Determine *FE*, by dividing the computer time available by the average time to compute the fitness function;
(ii) Solve (5) to determine N;
(iii) Compute $p_m = 5/N$;
(iv) Use $e = 1$, $p_c = 1$, and $c = 6$.

This approach assumes that the true optimal solution is never found, hence the implication is that a longer run time will locate a better solution. The GA parameters e, p_c, c did not have a significant relationship with any of the other parameters, and therefore the best values identified in Section 4 are used in the calibration methodology.

5. Evaluation of proposed GA calibration method

In this section, the utility of the proposed GA calibration method is assessed by comparing its performance with that of another approach to determining the GA parameter values, as well as 'typical' GA parameter values. Next, the different GA calibration methodologies that have been tested are outlined, before the range of test functions they are applied to are presented. This is followed by a comparison of the solution quality found by the different GA calibration methods.

5.1. Outline of methodologies

Three different methods for determining the GA population size have been selected to be tested on a range of fitness functions. Other approaches for self adaptation of the population size are available [1,2,9]. However, these approaches result in replacing the population size with one or more parameters to control the change in population size, hence they potentially contribute to the GA calibration problem, rather than solve it. Each population sizing method has been used with set GA parameter values as well as self-adaptive parameter values, producing a total of six methods considered, as outlined in Table 5. The first GA

Table 5
The different calibration methods considered

Method	N	e	p_m	p_c	c
1	Drift	1	$5/N$	1	6
2	Parameterless	1	1	0.5	6
3	100	1	1	0.9	6
4	Drift	1	Self adaptive		
5	Parameterless	1	Self adaptive		
6	100	1	Self adaptive		

calibration method to be compared is the methodology proposed in this paper, and has been referred to as 'Drift' in the analysis of the results.

The second method selected to be tested was the 'Parameterless' GA calibration method. This GA calibration methodology was first proposed by [15], followed by a number of studies, including [21,27,23]. Reed and Yamaguchi [27] applied this GA calibration methodology to real coded EAs, specifically differential evolution, and therefore a similar approach is adopted in this work. As is the case in this work, Reed and Yamaguchi [27] assumed that the population size is the key parameter controlling the reliability and efficiency of EAs. Consequently, the approach involves doubling the population size after convergence is achieved, and standard values are used for the remaining EA parameters. The initial starting population used in this work was $N = 10$. As proposed by Reed and Yamaguchi [27], the GA has been deemed to have converged after a minimum of $g = Nl$ generations, and after the improvement in the fitness function value from one generation to the next is less than 1%. After the GA has converged, the population size was doubled and reinitialised with random solutions, along with the best solution from the last GA run. The process was continued until the available number of *FE* have been completed.

The GA calibration methodology proposed by [15] suggests that to ensure building block growth in line with the Schema Theorem, the probability of crossover must be $p_c \leqslant (1 - s)/s$, where s is the selection pressure. The relationship has been applied to binary-coded GAs, however, as it is derived from the schema theorem, it is also applicable to RCGAs [12,40]. For this work, the parameter values proposed by Minsker [23] to satisfy this relationship are adopted, namely $s = 2$ and $p_c = 0.5$. Minsker [23] suggested that the bitwise probability of mutation should be the maximum value of $p_m = 1/N$ and $p'_m = 1/l$, where $p'_m = 1/l$ per bit used in that work is, on average, equivalent to $p_m = 1$ per string used in this work. Therefore, for the parameterless GA calibration methodology, $p_m = 1$ has been adopted. For the remaining GA parameters $c = 6$ and $e = 1$ have been used.

The third GA calibration method tested was used to represent the GA parameter values used typically, and is therefore referred to as 'Typical'. The values adopted for this method were; $N = 100$, $p_m = 1$ per string, $p_c = 0.9$, $c = 6$, and $e = 1$.

Each of these three GA calibration methods has also been used in a self-adaptive framework. The three methods described above have been used to determine the population size, and for each method $e = 1$ has been adopted. These parameters are applied at the population level, not the individual solution level, and therefore there is no clear way for the GA to self adapt these parameter values. For the remaining parameter values, each solution in the population also included a value for p_m, p_c and c to be used by the GA for that solution. When crossover occurred, the crossover parameter values to be used were randomly selected with an equal probability from one of the two parent solutions to be crossed over. The range for each parameter considered was $[0 \leqslant p_m \leqslant 1]$, $[0.5 \leqslant p_c \leqslant 1]$, and $[0.01 \leqslant c \leqslant 36]$.

5.2. Test functions

To avoid biasing the results toward the proposed GA calibration methodology, a set of fitness functions that have not been used in the calibration of the methodology have been selected. The test functions developed for the special session on real-parameter optimization at the 2005 IEEE Congress on Evolutionary Computation [35] have been used as the test functions to compare the different GA calibration methods. These functions have been composed to posses a number of different properties, including [35]: shifted functions (i.e. each variable has a different value at the optimum), rotated functions, varying structure in the fitness landscape, continuous and non-continuous functions, a range of the number of local optima, a range in the size and shape of local optima basins with respect to the basin of attraction for the global optimum, additively non-separable or non-decomposable functions, global optimum at the search boundary for some variables, and scalability into higher dimensions.

Formal definitions of the functions used can be found in [35]. Each function has been considered for problem sizes of l = 10, 30, and 50. For the different GA calibration methods, convergence times corresponding to $FE = 1e3$, $1e4$, and $1e5$ have been considered, along with $3e5$ for each problem with $l = 30$, and $5e5$ for each problem with $l = 50$. Each function has been optimized over the range $x_i \in [-5, 5]$ with the exception of f1–f6 and f14, which have the range $x_i \in [-100, 100]$, f8 over $x_i \in [-32, 32]$, f11 over $x_i \in [-0.5, 0.5]$, and f12 over the range $x_i \in [-\pi, \pi]$. Functions that include random noise in the fitness function value (f4, f7, f24) or that do not have a set search space (f17, f25) have been excluded from the analysis, producing a total of 20 test functions used in the comparison. The analysis methods proposed by [35] to compare the results from different calibration methods for the test functions have been adopted in this work. For every GA calibration methodology tested, each function in each problem size for each stopping criterion has been run 25 times with different seeds for the random number generator, therefore a total of 12,000 GA runs were required to produce the results presented in Section 5.4.

5.3. Population size for the drift method

The population sizes (rounded to the nearest 10 solutions) computed by (5) with $M = 3$ for each case considered in the comparison of the GA calibration methodologies are presented in Table 6. The self-adaptive calibration method adds three more decision variables to the problem (p_m, p_c, and c), however, this slight increase in problem size did not influence the population sizes presented in Table 6.

Table 6
Population sizes, N, predicted to converge due to genetic drift, for each value of l and FE

l	FE				
	$1e3$	$1e4$	$1e5$	$3e5$	$5e5$
10	10	40	120	–	–
30	10	40	120	200	–
50	10	40	110	–	260

Table 7
Overall mean ranking of each GA calibration method, and the average ranking over all fitness functions, for each combination of FE ($1e3$ to $5e5$) and l (10–50) considered

Method	Mean ranking	$1e3, 10$	$1e3, 30$	$1e3, 50$	$1e4, 10$	$1e4, 30$	$1e4, 50$	$1e5, 10$	$1e5, 30$	$1e5, 50$	$3e5, 30$	$5e5, 50$
1	2.0	1.4	1.5	1.9	2.1	2.5	2.8	2.7	1.8	2.0	1.7	1.8
2	2.9	2.2	1.9	1.8	3.4	2.6	2.3	3.4	3.4	3.3	3.4	3.9
4	3.4	4.1	3.7	3.3	4.0	3.8	3.8	3.1	3.2	3.2	2.9	2.7
6	3.9	4.4	5.0	5.5	2.3	3.1	3.9	3.4	3.6	3.6	4.0	4.2
5	4.0	4.1	3.7	3.3	3.9	4.2	4.0	3.9	4.2	4.3	4.2	3.8
3	4.9	5.0	5.3	5.2	5.3	4.9	4.3	4.6	4.9	4.7	4.8	4.7

5.4. Overall solution quality comparison

For each function, the mean and standard deviation of the difference between the true optimal solution for each function and the solution found by the GA with each calibration method has been computed for each of the 25 runs with different random number seeds. Based on these statistics, the performance of each GA calibration method over each of the 20 fitness functions has been ranked in order of decreasing ability to locate the best solutions. The ranking has been performed using a Student's t-test between the mean values found by the different methods, and if there was not a significant difference between two methods with a 95% confidence level, they have been given the same rank. The GA calibration methods have been sorted by the mean of the average rankings over each case of FE and l considered, producing an overall ranking of the performance of the different GA calibration approaches considered.

It can be seen from Table 7 that overall, the proposed GA calibration methodology with set values for the GA parameter values (Method 1) performed best, with a mean ranking of 2.0. The second best GA parameter calibration method overall was the Parameterless method with set values for the remaining GA parameters (Method 2). However, with an overall rank of 2.9, it was consistently outperformed by the Drift method, which made use of a constant population size. The Parameterless method with self-adaptive parameters (Method 5) was the second worst performing calibration method, with a mean ranking of 4.0, which was even slightly outperformed overall by the typical population size with self-adaptive parameters, with a ranking of 3.9 (Method 6). Not surprisingly, the Typical GA parameter calibration method (Method 3) performed worst, with a mean rank of 4.9, highlighting the importance of calibrating the GA parameter values for each individual problem. By allowing the GA to self-adapt the values for p_m, p_c and c, the typical GA parameter setting of $N = 100$ was able to improve the mean ranking of this method to 3.9.

When each method is considered on a function by function basis, the Drift GA calibration method with set parameter values (Method 1) consistently ranked as the best approach, with the method producing the best solutions for 13 of the 20 functions considered. The Parameterless method with set parameter values (Method 2) produced the second best performance, locating the best solution on average for 5 of the 20 functions. The Drift method with self-adaptive parameter values (Method 4) located the best solutions for f14, while the Parameterless method with self-adaptive parameter values (Method 5) located the best solutions on average for f9. However, in those cases where the Drift method with set parameter values was not ranked as the best calibration method, it was the second best, suggesting that it is a robust approach.

Somewhat surprisingly, the calibration methods with set values for p_m, p_c and c consistently outperformed the corresponding methods with self-adaptive parameter values. The only exception to this was the typical GA parameter setting, where the ranking improved from 4.9 to 3.9 by allowing the parameter values to self adapt. This was most likely due to the probability of mutation of $p_m = 1$ (one mutation per solution) used for the typical setting, which was able to be reduced by the self-adaptive method when appropriate. This probability of mutation was also used for the Parameterless GA calibration method with set values, which may explain its relatively poor performance. However, if only the self-adaptive calibration methods are considered, and therefore the values of the GA parameters other than the population size and number of elite solutions are left to the GA, it can be seen from Table 7 that the Drift method still performed the best of the three methods used to determine the population size.

6. Discussion

From Table 7, it can be seen that generally the static population sizes determined by the Drift method outperformed the Parameterless GA calibration method. Eiben et al. [9] also found that the Parameterless GA was much slower than a

traditional GA with a constant population size on a number of multi-modal problems. From a small initial population size ($N = 10$), the GA calibrated by the Parameterless method would be expected to converge quickly to a local optimum, and then by injecting this local optimum into the population for larger population sizes, the GA is lead back toward this local optimum in the first few generations, and therefore potentially away from better regions in the search space. This result suggests that in general, one GA run with a large population size can identify better solutions than multiple restarts with smaller population sizes for the same *FE*. Other studies [5, for example] have also come to this conclusion. The proposed calibration method has put a value on how large is 'large', to estimate the largest population size that can be expected to converge in the *FE* available. Therefore, fitness function evaluations are not wasted by the GA population prematurely converging, but at the same time the population will converge close to one solution, allowing the best possible solution to be found in the time available.

Eq. (5) provided the largest population size that can be expected to converge due to genetic drift for a given *FE* and *l*. However, generally the population will converge due to selection pressure before it will randomly converge due to genetic drift. Convergence due to selection pressure is much more difficult to predict, as the convergence rate is dependent on the fitness function characteristics, which are not immediately clear. For example, for a flat fitness function, the convergence will be due to genetic drift, as the fitness function value has no influence on the solutions chosen during selection. However, for a function with any structure, it would be expected that the fitness function values would cause the GA population to converge more quickly than what will occur randomly due to genetic drift. Therefore, as fewer generations are available before the GA population converges, a larger population size than that predicted by (5) would be expected to converge based on a given *FE*. Further work is required to firstly identify these characteristics, and then use them to predict the population convergence.

Eiben et al. [8] suggested that self-adaptive parameter control methods were the most promising method to determine the most suitable GA parameter values. However, the results presented in this paper indicate that the use of set parameter values for p_m, p_c and c outperformed the self-adaptive methods. A similar result was found by Bäck et al. [2], who concluded that the performance obtained when self-adapting p_m and p_c was disappointing when used on its own (i.e.; without any control of the population size). The most likely reason for this is that time spent on searching for good parameter values is time taken away from finding the optimum. While a self-adaptive approach provided a mechanism for the GA parameters to adjust to more suitable values as the GA converges to smaller regions of the search space, it is unlikely that the GA will find better values for the parameters through mutation alone once there is little diversity in the population.

7. Summary

A complete GA calibration methodology has been proposed in this paper. One of the main advantages of this approach is its simplicity; the only information required to apply the method is the problem size and the number of function evaluations that are available to find a solution. The population sizing equation is derived from a number of theoretical results based on quantitative genetics and GA convergence, and is therefore applicable to many algorithms provided that two parent tournament selection is the only operator that will significantly alter the population variance. For the GA used in this work, a complete GA calibration method is developed, where $p_m = 5/N$, $p_c = 1$, $e = 1$ and $c = 6$ are used for the remaining GA parameter values.

The proposed GA calibration method was compared with a number of other GA calibration methods that are currently available. A wide range of functions, problem sizes, and convergence criteria were considered. The results indicate that the proposed GA calibration methodology produced the best overall results of the six different GA calibration methods considered, while the typical GA parameter values produced the worst results. These findings highlight the importance of calibrating the GA parameter values to the characteristics of the fitness function.

The population sizing equation based on convergence due to genetic drift can be considered to provide a minimum bound on the most suitable population size to use, where even on a flat fitness function, the population can be expected to converge to a solution. An approach to provide a prediction of the population size that can be expected to converge due to the selection pressure, based on the characteristics of the fitness function, is the focus of further work.

Acknowledgments

The authors would like to thank the South Australian Partnership for Advanced Computing for the use of their facilities in generating the results presented in this paper. The authors would also like to thank the anonymous reviewers for their comments and suggestions, which improved the quality of this paper.

References

[1] J. Arabas, Z. Michalewicz, J.J. Mulawka, GAVaPS – a genetic algorithm with varying population size., in: International Conference on Evolutionary Computation, 1994, pp. 73–78.
[2] T. Bäck, A.E. Eiben, N. van der Vaart, An empirical study on GAs "without parameters", in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature—PPSN VI of Lecture Notes in Computer Science, vol. 1917, Springer-Verlag, Paris, France, 2000, pp. 315–324.
[3] T. Bäck, H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimisation, Evolutionary Computation 1 (1) (1993) 1–23.
[4] H.G. Beyer, K. Deb, On self-adaptive features in real-parameter evolutionary algorithms, IEEE Transactions on Evolutionary Computation 5 (3) (2001) 250–270.

[5] E. Cantú-Paz, D.E. Goldberg, Are multiple runs of genetic algorithms better than one?, in: E.A. Erick Cantú-Paz (Ed.), Genetic and Evolutionary Computation—GECCO 2003 of Lecture Notes in Computer Science, vol. 2723, Springer, 2003, pp. 801–812.
[6] K.A. De Jong, An analysis of the behaviour of a class of genetic adaptive systems, Ph.D. Thesis, University of Michigan, Ann Arbor, MI, 1975.
[7] K. Deb, R. Agrawal, Simulated binary crossover for continuous search space, Complex Systems 9 (1995) 115–148.
[8] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 124–141.
[9] A.E. Eiben, E. Marchiori, V.A. Valko, Evolutionary algorithms with on-the-fly population size adjustment, in: X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervós, J. Bullinaria, J. Rowe, P. Tino, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature—PPSN VIII of Lecture Notes in Computer Science, vol. 3242, Springer, 2004, pp. 41–50.
[10] D.S. Falconer, Introduction to Quantitative Genetics, second ed., Longman Scientific and Technical, Harlow, Essex, UK, 1981.
[11] R.A. Fisher, The Genetical Theory of Natural Selection, Clarendon Press, Oxford, 1930.
[12] D.E. Goldberg, Real-coded genetic algorithms, virtual alphabets, and blocking, Complex Systems 5 (2) (1991) 139–168.
[13] D.E. Goldberg, K. Deb, J. Clark, Genetic algorithms, noise, and the sizing of populations, Complex Systems 6 (4) (1992) 333–362.
[14] G. Harik, E. Cant-Paz, D.E. Goldberg, B.L. Miller, The gambler's ruin problem, genetic algorithms, and the sizing of populations, Evolutionary Computation 7 (3) (1999) 231–253.
[15] G.R. Harik, F.G. Lobo, A parameter-less genetic algorithm, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, vol. 1, Morgan Kaufmann, Orlando,Florida,USA, 1999, pp. 258–265.
[16] F. Herrera, M. Lozano, A.M. Sanchez, A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study, International Journal of Intelligent Systems 18 (3) (2003) 309–338.
[17] F. Herrera, M. Lozano, J.L. Verdegay, Tackling real-coded genetic algorithms: operators and tools for behavioural analysis, Artificial Intelligence Review 12 (4) (1998) 265–319.
[18] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Cambridge, MA,, 1975.
[19] P. Kouchakpour, A. Zaknich, T. Braunl, Population variation in genetic programming, Information Sciences 177 (17) (2007) 3438–3452.
[20] F.G. Lobo, The parameter-less genetic algorithm: rational and automated parameter selection for simplified genetic algorithm operation, Ph.D. Thesis, Universidade Nova de Lisboa, Lisbon, Portugal, 2000.
[21] F.G. Lobo, D.E. Goldberg, The parameter-less genetic algorithm in practice, Information Sciences 167 (1–4) (2004) 217–232.
[22] B.L. Miller, D.E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise, Complex Systems 9 (1995) 193–212.
[23] B. Minsker, Genetic algorithms, in: P. Kumar, J. Alameda, P. Bajcsy, M. Folk, M. Markus (Eds.), Hydroinformatics: Data Integrative Approaches in Computation Analysis and Modeling, CRC Press, Florida, USA, 2005, pp. 439–456.
[24] H. Mühlenbein, How genetic algorithms really work—Part I: Mutation and hillclimbing, in: R. Männer, B. Manderick (Eds.), Parallel Problem Solving from Nature—PPSN II, Elsevier Science Publishers, Amsterdam, North Holland, 1992, pp. 15–25.
[25] H. Mühlenbein, H. Asoh, On the mean convergence time of evolutionary algorithms without selection and mutation, in: Y. Davidor, H. Schwefel, R. Männer (Eds.), Parallel Problem Solving from Nature—PPSN III, of Lecture Notes in Computer Science, vol. 866, Springer Verlag, London, UK, 1994, pp. 88–97.
[26] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization, Evolutionary Computation 1 (1) (1993) 25–49.
[27] P. Reed, S. Yamaguchi, Simplifying the parameterization of real-coded evolutionary algorithms, in: World Water and Environmental Resources Congress, EWRI/ASCE, Salt Lake City, Utah, USA, 2004, pp. 1–9.
[28] S. Rochet, G. Venturini, M. Slimane, E.M. El Kharoubi, A critical and empirical study of epistasis measures for predicting GA performances: a summary, in: Proceedings of the 1997 3rd European Conference on Artificial Evolution, Lecture Notes in Computer Science, vol. 1363, Springer-Verlag, 1998, pp. 275–285.
[29] A. Rogers, A. Pruegel-Bennett, Genetic drift in genetic algorithm selection schemes, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 298–303.
[30] L. San Jose-Revuelta, A new adaptive genetic algorithm for fixed channel assignment, Information Sciences 177 (13) (2007) 2655–2678.
[31] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, R. Das, A study of control parameters affecting online performance of genetic algorithms for function optimization, in: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, pp. 51–60.
[32] C. Shannon, A mathematical theory of communication, Bell System Technical Journal 27 (1948) 379–423. 623–656.
[33] K. Srinivasa, K. Venugopal, L. Patnaik, A self-adaptive migration model genetic algorithm for data mining applications, Information Sciences 177 (20) (2007) 4295–4313.
[34] R. Storn, K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[35] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report, Nanyang Technological University, May 2005.
[36] D. Thierens, D.E. Goldberg, A. Pereira, Domino convergence, drift and the temporalsalience structure of problems, in: IEEE International Conference on Evolutionary Computation, IEEE Press, New York, NY, 1998, pp. 535–540.
[37] H.-M. Voigt, H. Mnhlenbein, D. Cvetkovic, Fuzzy recombination for the breeder genetic algorithm, in: L. Eshelman (Ed.), Proceedings of the sixth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1995, pp. 104–113.
[38] D. Whitley, K. Mathias, S. Rana, J. Dzubera, Building better test functions, in: L. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1995, pp. 239–246.
[39] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.
[40] A. Wright, Genetic algorithms for real parameter optimization, in: G. Rawlins (Ed.), Foundations of Genetic Algorithms 1, Morgan Kaufmann, San Mateo, CA, 1991, pp. 205–218.