Predicting effective control parameters for differential evolution using cluster analysis of objective function features

Sean P. Walton[1] · M. Rowan Brown[2]

Abstract
A methodology is introduced which uses three simple objective function features to predict effective control parameters for differential evolution. This is achieved using cluster analysis techniques to classify objective functions using these features. Information on prior performance of various control parameters for each classification is then used to determine which control parameters to use in future optimisations. Our approach is compared to state-of-the-art adaptive and non-adaptive techniques. Two accepted bench mark suites are used to compare performance and in all cases we show that the improvement resulting from our approach is statistically significant. The majority of the computational effort of this methodology is performed off-line, however even when taking into account the additional on-line cost our approach outperforms other adaptive techniques. We also investigate the key tuning parameters of our methodology, such as number of clusters, which further support the finding that the simple features selected are predictors of effective control parameters. The findings presented in this paper are significant because they show that simple to calculate features of objective functions can help to select control parameters for optimisation algorithms. This can have an immediate positive impact on the application of these optimisation algorithms on real world problems, where it is often difficult to select effective control parameters.

Keywords Tuning evolutionary algorithms · Differential evolution · Unsupervised learning · Optimization methods · Evolutionary algorithms

✉ Sean P. Walton
s.p.walton@swansea.ac.uk

1 Department of Computer Science, College of Science, Swansea University, SA2 8PP Swansea, UK

2 Centre for Nanohealth, College of Engineering, Swansea University, SA1 8EN Swansea, UK

🖄 Springer

1 Introduction

Despite having a number of advantages over classical gradient based techniques, the performance of evolutionary algorithms depends both on the problem to be optimised and the algorithm being used (Wolpert and Macready 1997). To make matters worse, this performance also depends heavily on the selection of algorithm specific control parameters. This variability of performance makes the field hard to penetrate for users in industry who simply want to use an algorithm to solve a problem. Often the problem they wish to solve is not well understood before they start to solve it, which makes selecting an algorithm and control parameters all the more difficult. The motivation of the work, presented in this paper, is to automate this selection using simple machine learning techniques. Specifically the aim is to automatically select an effective set of control parameters for differential evolution for an unknown problem.

1.1 Terminology

The problem to be optimised is termed the objective function. This paper focusses on optimising continuous black box objective functions. We identify a number of features, β, that an objective function can be described by. An optimisation algorithm instance is determined by its control parameters **p**. Our aim is to classify objective functions using their features, in order to predict a set of effective control parameters which will result in a high performing algorithm for a particular objective function.

1.2 Background

When applying an evolutionary algorithm to a new application it is common to use the control parameters suggested in literature. These parameters are usually obtained from extensive studies on algorithm behaviour using suites of benchmark optimisation problems. Parameters which work well on common problem test suites will emerge (Eiben and Smit 2011) and this single set will end up being used in the majority of applications. The problem is that with truly novel applications there may be no understanding of which test suites, if any, correctly represents the real world problem. Strictly speaking, each time an algorithm is applied to a new application a parameter study should be undertaken, to both provide insight into the robustness of the parameters and perhaps squeeze out some additional performance. The reality is that these studies are often infeasible in real applications, where a single objective function evaluation may represent hours, or days, of computational time (Naumann et al. 2015; Walton et al. 2013a, b, 2015). Thus a great deal of research has been undertaken with the motivation to address this problem. We have identified three interrelated strands of research in the meta-heuristic optimisation community relevant to this problem. These are briefly discussed below, we then place our own approach in this context.

1.2.1 Automatic tuning algorithms based on performance modelling

A considerable body of work has shown that it is possible to build empirical performance models of algorithms (Hutter et al. 2014). These models can then be used to select tuning parameters with good predicted performance (Hutter et al. 2006). Sequential model-based optimization for general algorithm configuration (SMAC) (Hutter et al. 2011) and sequential parameter optimization (SPO) (Bartz-Beielstein et al. 2005) are both specific examples of this approach. In the case of SPO the approach facilitates manual tuning whereas SMAC is automated.

1.2.2 Feature based approaches

It is increasingly argued that we need to understand and use the characteristics and features of a problem to select a suitable algorithm, or tune it (Smith-Miles 2008). Feature based algorithm configuration (FBAC) (Belkhir et al. 2016) can be thought of as an extension of the automatic tuning algorithms mentioned above. It uses sophisticated objective function features to classify objective functions. They are able to accurately predict performance models for objective functions which could, in theory, be used to determine an effective set of control parameters. However, the features they use require a large number of samples of the objective function to calculate. This would lead to an excessive computational cost in real applications. Exploratory landscape analysis (ELA) (Mersmann et al. 2011) introduces ten features, which are relatively cheap to calculate and can be used to classify objective functions. These features are grouped into five classes which relate to different characteristics of objective functions. Promising results have been presented whereby the ELA features are used to train a one-sided support vector regression model to select an appropriate optimisation algorithm (Kerschke et al. 2016).

1.2.3 Adaptive algorithms

The most common strategy to address the problem of performance variability is to design algorithms with self-adaptive control parameters. In such algorithms the control parameters are themselves optimised, based on current performance, as the algorithm runs (Sarker et al. 2014; Zamuda and Brest 2015; Guo et al. 2014). A related field is hyper-heuristics whose goal is to automate the design of heuristic optimisation algorithms based on current performance (Burke et al. 2013; Li and Kendall 2015). These strategies are performed on a per-objective function basis and do not use knowledge of objective function features, or past performance on different objective functions.

1.2.4 Case study optimisation algorithm: Differential Evolution

To show the effectiveness of our approach we are forced to select a single optimisation algorithm. Differential Evolution (DE) (Storn and Price 1997) will be used to test the effectiveness of the predictive methodology. It is stressed that this approach is independent of the evolutionary algorithm, although some thought will be required if

an algorithm has any non-continuous control parameters. DE is popular and its control parameters are well studied. The algorithm is aimed at nonlinear non-differentiable continuous functions and has been designed to be a direct stochastic search method. The method has a small number of control parameters and applies a crossover and mutation operator based on the differences between randomly selected individuals of the population.

There are a number of alternative DE methods and many additions have been made to the algorithm. It is beyond the scope of this paper to explain these additions in detail, so instead we describe the algorithm used in this study and allow the reader to find detailed explanation in original papers.

To select new members of the population, a direct one-to-one competition scheme is employed in each generation. From the population of the current generation, a target member, $\mathbf{x}_{i,g}$, is selected, where i refers to the member's number and g the generation. A donor vector, $\mathbf{v}_{i,g}$, is generated using the current-to-pbest/1/bin approach (Zhang and Sanderson 2009). Three members of the population, distinct to that of the target member, are selected at random and $\mathbf{v}_{i,g}$ is calculated according to the relation

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + p_2(\mathbf{x}_{pbest,g} - \mathbf{x}_{i,g}) + p_2(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{1}$$

where p_2 is a control parameter usually referred to as the weighting factor. $\mathbf{x}_{r1,g}$ and $\mathbf{x}_{r2,g}$ are two members selected at random from the whole population and $\mathbf{x}_{pbest,g}$ is randomly selected from the top $q \times p_3$ ($q \in [0, 1]$). p_3 is the population size or number of parents. q is a control parameter which controls the greediness of the algorithm, to eliminate this parameter it is randomised as in the success-history based parameter adaptation for differential evolution (SHADE) algorithm (Tanabe and Fukunaga 2013). In addition, an external archive of previous members of the population is maintained and used to generate $\mathbf{x}_{r2,g}$ (Tanabe and Fukunaga 2013).

A cross over operator is applied to the target and donor vectors to form a trial vector. The elements of the target and donor vectors enter the trial vector with a probability p_1, a control parameter usually referred to as crossover constant. The target vector is compared with the trial vector and the vector with the best fitness value is selected for admission into the next generation. This iteration scheme repeats until a suitable stopping criterion is met (Storn and Price 1997).

DE has been applied, with success, to the fields of electrical power systems, electromagnetic engineering, control systems and robotics, chemical engineering, pattern recognition, artificial neural networks and signal processing (Das and Suganthan 2011). Storn (2016) suggests using the control parameters $p_1 = 0.900$, $p_2 = 0.500$ and $p_3 = 10D$ where D is the number of dimensions in the function. The effect of these parameters on algorithm performance is a well researched subject. For example, there appears to be complex relationships between problem dimensionality and the most appropriate population size (Piotrowski 2016).

We compare our proposed predictive technique to a state of the art adaptive technique: SHADE (Tanabe and Fukunaga 2013). This technique uses an historical memory of control parameters which have performed well to guide the selection of control parameters each generation. In the original study it was shown to have competitive performance compared to other state of the art algorithms using the CEC 2005

benchmarks which are used in this study. All control parameters used in our study are the same as used in the original SHADE study (Tanabe and Fukunaga 2013).

1.3 Contribution and motivation of this paper

The approach we have adopted is to select three simple to calculate features and use these to classify objective functions. Then as we optimise a series of objective functions a global memory of the performance, of various control parameters, for each of the classifications is stored. This information is then used to adapt control parameters for future optimisations. We do not create a performance model but directly use prior knowledge to adapt the optimisation algorithm. Thus our approach falls under the adaptive algorithm category and hence we compare our strategy to other adaptive strategies below. Our approach also falls under the feature based approach category since we are using objective function features to drive our adaptation. Our features are much simpler, and more crude, than those used in FBAC (Belkhir et al. 2016) and we use fewer than those identified in ELA (Mersmann et al. 2011). Our contribution is that even when using our deliberately simple approach there is a statistically significant improvement in performance when compared to algorithms which do not consider objective function features. The motivation for this is real world applications where it is infeasible to tune an algorithm each time a new objective function is considered, and where the form of the objective function may be unknown, making it difficult to relate to previous analyses of control parameters.

2 Methodology

2.1 Our approach: predicting effective control parameters for evolutionary algorithms using cluster analysis of objective function features

The aim of our approach is to automatically predict an effective set of control parameters for an unknown objective function. This is achieved by classifying objective functions using three simple to calculate features which are described in Sect. 2.1.2. A number of experiments are performed off-line with varying control parameters, across a range of objective functions. The algorithm performance is measured and recorded for each experiment, the performance metric used is described in Sect. 2.1.1. Functions are split into classifications using the unsupervised machine learning technique k-means++ (Arthur and Vassilvitskii 2007). All the experiments in a particular classification are ranked by performance and the mean values of the control parameters used in the top 10% of experiments is calculated. When a new function is to be optimised it is sampled, on-line, and its features calculated. It is then classified and the mean values calculated for its classification are used to optimise it.

2.1.1 Optimisation algorithm performance metric

There are a number of metrics which can be utilised to define the performance of an optimisation algorithm (Eiben and Smit 2011; Belkhir et al. 2016). The meaning of performance may change depending on the application (López-Ibáñez et al. 2016), but in general we wish to reduce the objective function value with a small number of objective function evaluations. In this work a performance metric, α, is defined as

$$\alpha = \frac{100(F_1 - F_G)}{F_1 N_g} \tag{2}$$

where F_1 is the lowest objective function value in the first generation, F_G is the lowest objective function value in the final generation, G, and N_g is the total number of function evaluations performed up to and including generation g. Generation g is the first generation at which the reduction in the objective function reaches 99% of the total reduction i.e.

$$\frac{F_G}{F_g} > 0.99 \tag{3}$$

This choice is justified as follows. In practice, an evolutionary optimisation algorithm is run until a maximum number of objective function evaluations is reached or a predetermined accuracy, or tolerance, is achieved. Dividing by N_g means that α gives us information on the efficiency of the optimisation algorithm. An algorithm which finds the optimum in the first few generations therefore has a larger α than an algorithm which found the optimum in the final generation. In real applications, practicalities such as objective function evaluation cost, limit the number of objective function evaluations (Naumann et al. 2015; Walton et al. 2013a, b, 2015). α is designed to reward algorithms which exhibit high convergence in the first few function evaluations.

It is not claimed that α is the correct metric for all situations, it is a choice depending on user requirements. In this study an attempt is made to model a situation where an engineer wishes to apply an optimisation algorithm to a real problem. One can imagine that such an engineer would simply select an algorithm and use the set of control parameters suggested in literature. In the authors experience, in applying optimisation algorithms to engineering applications, the proposed metric α is relevant for many engineers. Control parameters suggested in literature may not have been tuned with this metric in mind, despite this the engineer would likely use these parameters. In the results section convergence curves are presented to show the effect of this metric choice.

2.1.2 Objective function features

Functions are often described using features such as symmetry, smoothness, condition number or separability. It is well understood that these features affect the performance of optimisation algorithms. The challenge, therefore, is to formulate a set of features that can be calculated with a small number of objective function evaluations.

In this proof of concept study, the starting point for calculating these features will be a Latin hypercube sampling of the objective function search space. The number of

samples taken is referred to as σ. This sampling will be performed prior to the optimisation in this study, but in future this sampling could also be used as the first generation of the optimisation algorithm. The objective function values in this sampling are first normalised by subtracting the mean and dividing by the standard deviation. Three simple features have been selected to test the methodology.

1. β_1, is the number of dimensions of the function which is known to strongly effect algorithm performance.
2. β_2, is the interquartile range of the normalised data, which provides information on function variation within the domain. This feature will identify functions which have largely flat topology. This is identified as a feature which relates to curvature in Mersmann et al. (2011).
3. β_3, is the skew of the normalised data. The skew tells us how the function value is distributed, a skew of zero would indicate a normal distribution, whereas positive and negative values would indicate a tailed distribution. This feature could potentially identify functions with sharp optimum as well as give information regarding function symmetry. This is identified as a feature which relates to y-distribution in Mersmann et al. (2011).

Collectively these features make up the characteristics, $\boldsymbol{\beta}$, of a particular objective function.

2.1.3 Control parameter selection

DE requires a number of control parameters, stored in the vector $\mathbf{p}$, which defines a single instance of the algorithm. Running many $\mathbf{p}$ on many objective functions results in a number of data points in the form $(\mathbf{p}, \boldsymbol{\beta}, \alpha)$. This data is named the training data and is used to exploit any relationships between the control parameters, function features and performance.

The approach adopted is to apply the unsupervised clustering algorithm k-means++ (Arthur and Vassilvitskii 2007). The k-means algorithm takes an unlabelled data set and classifies it into a user specified number of groups, κ. Each group is defined by a cluster centroid, a data point belongs to the group whose centroid it lies closest to. The k-means++ variant of the algorithm carefully initialises these centroids in favour of random initialisation (Arthur and Vassilvitskii 2007).

Using the training set, the objective functions are classified by applying k-means++ to the $\boldsymbol{\beta}$ data points. For each classification the data points are sorted by α. The top 10% data points are identified and the mean $\mathbf{p}$ is calculated from that set and used to optimise new functions identified as belonging to that classification.

At the end of optimisation the new data point, $(\mathbf{p}, \boldsymbol{\beta}, \alpha)$, from that run is appended to the training data and the k-means++ algorithm is run to update the classifications and redetermine the best control parameters for each new centroid. The key idea is that the memory of good performing parameters are extended from a single optimisation run to the entire history of using the algorithm.

2.2 Experimental methodology

2.2.1 Procedure for a single optimisation

Each time a function was optimised the optimiser was limited to 10,000 objective function evaluations. To consider our approach fairly the on-line cost of calculating β, before the optimisation takes place, contributes to the number of objective function evaluations. In other words our approach has fewer objective function evaluations available when the optimisation starts. All optimisation runs were repeated 30 times, with 30 different random seeds used for all random number generation. With the same random seed the same control parameters would result in the same performance on the same function instance. This allowed pairwise comparisons between different control parameters. Repeating each optimisation run with 30 different random seeds ensured a 'lucky' seed was not selected which benefited a particular approach.

2.2.2 Test suites

Two established optimisation benchmark suites were used in this study.
Real-parameter black-box optimization benchmarking functions (BBOB) 2015 BBOB 2015 were used to train the predictive methodology. The 24 benchmark functions which make up the BBOB 2015 test suite are given in Finck et al. (2010) and Hansen et al. (2010). This suite includes separable functions, functions with low to high condition numbers and multi-modal functions with weak global structure. The same numbering system for the functions in Hansen et al. (2010) is used in this paper. All of these functions are defined for an arbitrary number of dimensions and have the same search domain. The test suite includes 15 instances for each function, for each instance a combination of optimal location shifting and linear transformations are applied. Each instance is shifted and rotated in the same manor on subsequent runs which enables direct comparison of performance. In the experiments presented here, a single test suite entails optimising each function instance at a range of dimensions (2, 10, 20, 30, 40, 50) using 30 different random seeds. The resulting number of tests in a single run of the suite is then 64,800.
IEEE congress on evolutionary computation CEC 2005 real-parameter optimisation benchmarks The CEC 2005 benchmark functions as detailed in Suganthan et al. (2005) make up the second test suite. These were used to test the effectiveness of problem specific tuning on objective functions different to the training set. The 25 functions were used with the same numbering system presented in the technical report (Suganthan et al. 2005). All functions were optimised at 2, 10, 30 and 50 dimensions using 30 different random seeds resulting in a total of 3000 tests.

2.2.3 Statistical methodology

Using the test suites described above allows pairwise comparison of α between different approaches. The approach for using nonparametric statistical tests described by Derrac et al. (2011) is followed here. The Wilcoxon signed ranks test is used to compare the predictive methodology to other approaches. The test results in the value

W, the sum of the ranks of the differences (zero-differences are split between positive and negative) which will be reported along with the two sided p value. For all the statistics presented a positive W indicates that the predictive methodology has performed better than the group it is compared to, a larger W indicates a more significant improvement.

2.2.4 Experiments

For each function in the BBOB 2015 suite a Latin hypercube sampling of $\mathbf{p}$ will be generated and each of these control parameter sets used to optimise that function. Since there are three control parameters, 30 sets of $\mathbf{p}$ will be generated each time. For these optimisation runs the number of samples used to calculate $\boldsymbol{\beta}$, was set to $\sigma = 1000$. The resulting data will be used as the initial training set for the problem aware tuning.

There will then be four methods for selecting the DE control parameters:

– The suggested parameters from literature,
– SHADE,
– The predictive methodology (using cluster analysis),
– Using the best performing control parameters from the training set.

The predictive methodology will be applied with varying σ and κ to gauge the sensitivity to these. Each time our approach is used, a new set of samples is generated to calculate $\boldsymbol{\beta}$ in order to simulate the use of the approach in practice. Each of these methods will be used to optimise both function suites. The non-parametric tests will then be utilised to compare the effectiveness of each method.

It needs to be stressed that, in the comparisons presented, $\boldsymbol{\beta}$ is recalculated and used for objective function classification in each optimisation run. This does mean that when comparing the number of function evaluations to other methods, the predictive methodology has σ additional evaluations. These function evaluations have been included in all measurements of performance as they indicate the cost of our methodology.

The goal of this paper is to show that features, such as those in $\boldsymbol{\beta}$, can be used as predictors for $\mathbf{p}$ in order to maximise α. If this is the case, future research into minimising the required σ to effectively approximate $\boldsymbol{\beta}$ can be undertaken, as well as research into different definitions for α.

3 Results

3.1 BBOB 2015 function suite

Figure 1 shows a two dimensional projection of the training data used in the following studies. This data resulted from optimising the BBOB 2015 function suite only. The marker colour indicates which classification each data point belongs to when $\kappa = 10$.

In the following experiments the BBOB 2015 suite is both the training suite and the testing suite. This is the most basic test for our approach. It is worth pointing out that $\boldsymbol{\beta}$ is recalculated using new random samplings for each optimisation experiment.
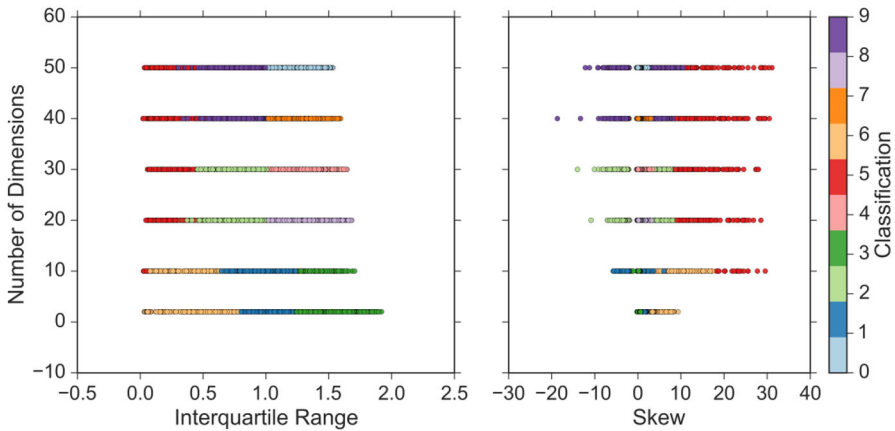
Fig. 1 The BBOB 2015 training data set. Markers are coloured according to their classification when $\kappa = 10$ (Color figure online)

Table 1 The control parameters used in the fixed control parameter optimisations

	p_1	p_2	p_3
Best in training set	0.624	0.744	336
Suggested in literature (Storn 2016)	0.900	0.500	10D

Table 2 BBOB 2015 function suite: Wilcoxon signed ranks test data comparing the predictive methodology to using the best performing control parameters overall in the training set (equivalent to $\kappa = 1$)

κ	σ	W	p value
10	10	12647560.5	< 0.0001
10	100	12000441.0	< 0.0001
10	1000	14409261.5	< 0.0001
100	1000	13138542.0	< 0.0001

3.1.1 Predictive methodology compared to picking the best from the training set

In the following study a single set of tuning parameters, the best performing in the initial training set, were used to optimise the BBOB 2015 test suite. The control parameters used in this study are presented in Table 1. The results of the statistical tests, shown in Table 2, show that predictive methodology results in a statistically significant increase of α compared to the best from the training set. This improvement was achieved regardless of the values of κ and σ.

3.1.2 Predictive methodology compared to using the best parameters from literature

The test suite was optimised using the control parameters suggested in literature, these parameters are shown in Table 1. These parameters are what most practitioners would use in practice as a *rule of thumb*. Table 3 shows the statistical comparison between the performance of these fixed parameters to the predictive methodology. In

Table 3 BBOB 2015 function suite: Wilcoxon signed ranks test data comparing the predictive methodology to using the control parameters most commonly used in literature

κ	σ	W	p value
10	10	64554898.0	< 0.0001
10	100	65968426.5	< 0.0001
10	1000	87498052.0	< 0.0001
100	1000	83669358.0	< 0.0001

Table 4 BBOB 2015 function suite: Wilcoxon signed ranks test data comparing the predictive methodology to SHADE

κ	σ	W	p value
10	10	50395288.5	< 0.0001
10	100	51910557.5	< 0.0001
10	1000	68985487.5	< 0.0001
100	1000	66556975.0	< 0.0001

all cases the predictive methodology performs significantly better. There is a jump in performance when σ increases from 100 to 1000 which indicates a sensitivity on the sampling of the objective functions.

3.1.3 Predictive methodology compared to SHADE

The test suite was optimised using SHADE. Table 4 shows the statistical comparison between the performance of SHADE to the predictive methodology. In all cases the predictive methodology performs significantly better than adaptive tuning. The performance is more significant when $\sigma = 1000$.

3.1.4 Convergence behaviour

In Fig. 2 convergence plots are presented for a number of functions in the BBOB 2015 suite. The objective function value is plotted against the number of function evaluations for each control parameter selection strategy. These data points were only recorded once per generation if an improvement in the objective function value was found. This means that the number of data points depends on the population size and is not the same for every curve. Each function is shown for different numbers of dimensions and the control parameters selected by the predictive methodology are presented. These functions were selected to show a range of cases, some where the predictive methodology performs well and some where it does not. Where the predictive methodology performs well it achieves rapid convergence early in the optimisation, which is what the metric α was designed to achieve.
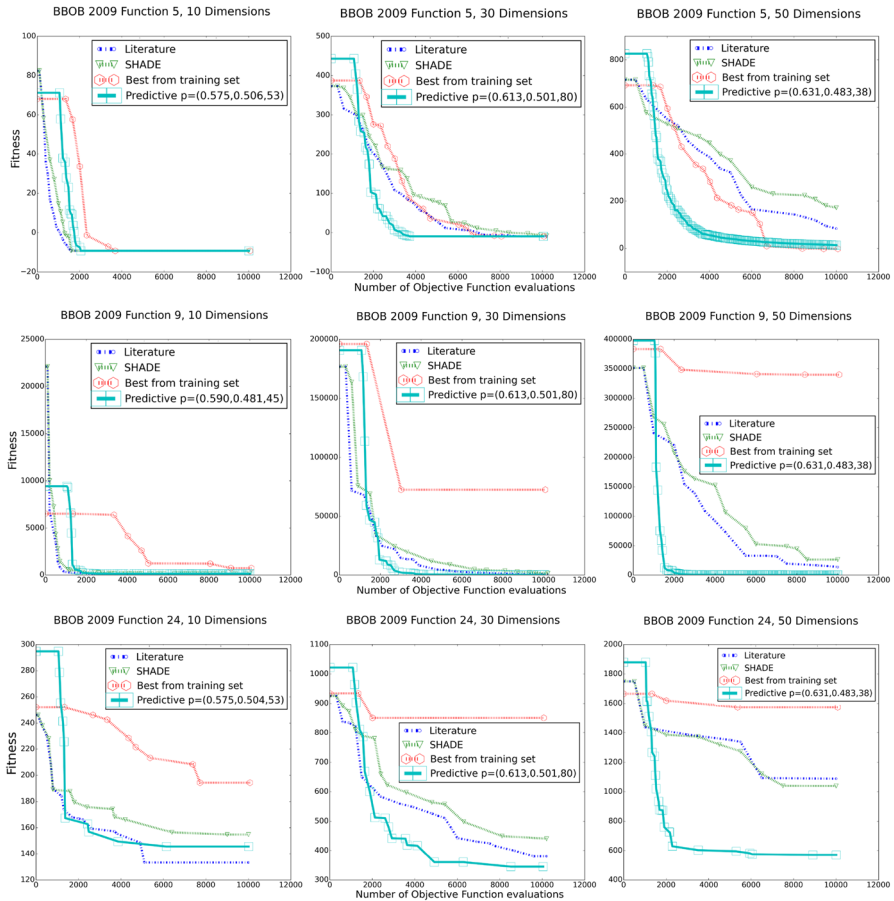
Fig. 2 Examples of the convergence behaviour using different control parameter selection strategies for the BBOB 2015 function suite. In the predictive methodology $\kappa = 10$ and $\sigma = 1000$

3.2 CEC 2005 function suite (predictive methodology trained using BBOB 2015)

3.2.1 Problem aware tuning compared to picking the best from the training set

The predictive methodology and the best control parameters from the initial training set were used to optimise the CEC 2005 benchmark functions. Table 5 shows the statistical tests for this comparison. For all κ and σ the predictive methodology performs significantly better with p values < 0.0001. The improvement observed when optimising the CEC 2005 function suite is comparatively less significant than the improvement observed when optimising the BBOB 2015 suite.

Table 5 CEC 2005 function suite: Wilcoxon signed ranks test data comparing the predictive methodology to using the best performing control parameters overall in the training set (equivalent to $\kappa = 1$)

κ	σ	W	p value
10	10	55436.0	< 0.0001
10	100	49811.5	< 0.0001
10	1000	58615.0	< 0.0001
100	1000	58971.5	< 0.0001

Table 6 CEC 2005 function suite: Wilcoxon signed ranks test data comparing the predictive methodology to using the control parameters most commonly used in literature

κ	σ	W	p value
10	10	220650.0	0.001
10	100	223100.0	0.003
10	1000	233488.0	0.075
100	1000	243996.0	0.528

Table 7 CEC 2005 function suite: Wilcoxon signed ranks test data comparing the predictive methodology to using SHADE

κ	σ	W	p value
10	10	183736.0	< 0.0001
10	100	187102.0	< 0.0001
10	1000	217967.0	< 0.0001
100	1000	206840.0	< 0.0001

3.2.2 Predictive methodology compared to using the best parameters from literature

The results of statistical tests comparing the predictive methodology to the fixed parameters suggested in literature are shown in Table 6. For all cases the predictive methodology outperformed the fixed parameters, but the increase in performance is not statistically significant when $\kappa = 100$.

3.2.3 The predictive methodology compared to SHADE

The results of statistical tests comparing the predictive methodology to the fixed parameters suggested in literature are shown in Table 7. For all κ and σ the predictive methodology significantly outperformed SHADE with p values all < 0.0001.

3.2.4 Convergence behaviour

Figure 3 compares convergence plots for functions from the CEC 2005 suite using the different control parameter selection strategies. Functions were selected to present a range of behaviours.
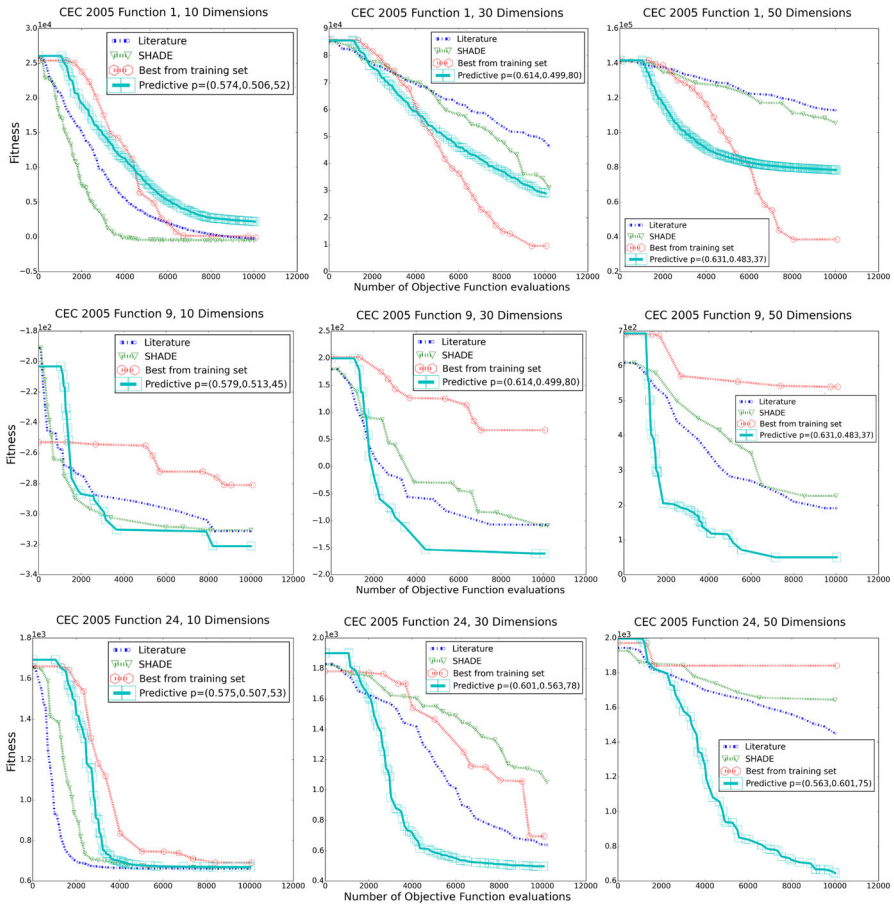
Fig. 3 Examples of the convergence behaviour using different control parameter selection strategies for the CEC 2005 function suite. The predictive methodology was trained using the BBOB 2015 function suite with $\kappa = 10$ and $\sigma = 1000$

4 Discussion

The results show that, when optimising the BBOB 2015 objective function suite, the predictive methodology out performed using fixed and adaptive tuning parameters for DE with p values < 0.0001. The predictive methodology is more likely to outperform other methodologies when the initial sampling size, σ, of the objective functions was increased. There was a slight drop in performance from $\kappa = 10$ to $\kappa = 100$. This indicates that having fewer large classifications of objective function are better than a more granulated approach. This trend does not continue to $\kappa = 1$, i.e. simply using the best parameters from the training set. The observed improvement from $\kappa = 1$ to $\kappa = 10$ and $\sigma = 100$ to $\sigma = 1000$ both support the claim that $\boldsymbol{\beta}$ can be used as a predictor for which control parameters to use in an optimisation algorithm.

Optimising the CEC 2005 benchmark functions, using only the BBOB 2015 suite for training, is a tough test for the suitability of β to act as a predictor for control parameters in the general case. In all cases the predictive methodology outperformed other approaches. The closest performing approach to the predictive methodology was simply using fixed control parameters suggested in literature, in this case two tests ($\kappa = 10, \sigma = 1000$ and $\kappa = 100, \sigma = 1000$) have p values 0.075 and 0.528 respectively. This could be explained by the fixed control parameters not requiring objective function evaluations to adapt, i.e. all objective function evaluations are used for optimisation. The high value of σ for these outliers further supports this explanation, when $\sigma < 100$ the p values are < 0.01 when comparing to fixed control parameters from literature. This effect is then compounded when $\kappa = 100$ which, as observed elsewhere, performs less well than $\kappa = 10$.

Overall the results show that the simple objective function features, β can act as a predictor for selecting appropriate control parameters for DE. The advantage of this approached can be observed when comparing it to SHADE. SHADE learns which control parameters are most effective during the optimisation process. The predictive methodology attempts to predict effective control parameters prior to the optimisation, therefore the benefit is felt from the first iteration. This prediction itself comes at the cost of objective function evaluations, which were accounted for in this study. The open problem, therefore, is how to effectively approximate these features with fewer objective function evaluations. In the future, an aim is to use these objective function evaluations as the first generation of the optimisation run in order not to waste them.

To improve performance, it may be possible to update the value of β as the optimisation runs and better samples the objective function. As the approximation of β improves the control parameters could be changed mid run. This would require thoughtful implementation to avoid introducing significant computational overhead. There is also scope for designing more sophisticated and varied objective function features. Performance may also be increased with a larger training data set. This does come with an additional overhead, as the computational cost of the k-means algorithm increases with the size of the training data. In the future there is no reason why the k-means calculation could not be performed using cloud computing with training data collected from many users of the algorithm.

4.1 Conclusions

The methodology proposed has been shown to offer statistically significant improvement over other approaches. This implementation shows that the concept has the potential to be a powerful addition to evolutionary optimisation algorithms. The method is general and could be applied to any evolutionary algorithm and any performance measure of interest. There are a number of avenues to investigate, discussed above, which may improve the methodology further. In particular an investigation into more sophisticated function features, such as those presented in ELA (Mersmann et al. 2011). The long term goal should be to extend this methodology to automatically select the most appropriate evolutionary algorithm for a problem, not just the control

parameters. Such an automation would be of great use for industrialists wishing to apply evolutionary algorithms to real world applications.

References

Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, pp. 1027–1035. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2007). http://dl.acm.org/citation.cfm?id=1283383.1283494

Bartz-Beielstein, T., Lasarczyk, C.W., Preuß, M.: Sequential parameter optimization. In: The 2005 IEEE Congress on Evolutionary Computation, 2005, vol. 1, pp. 773–780. IEEE (2005)

Belkhir, N., Dréo, J., Savéant, P., Schoenauer, M.: Feature based algorithm configuration: a case study with differential evolution. In: International Conference on Parallel Problem Solving from Nature, pp. 156–166. Springer (2016)

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(S12), 1695–1724 (2013). https://doi.org/10.1057/jors.2013.71

Das, S., Suganthan, P.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011). https://doi.org/10.1109/TEVC.2010.2059031

Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011). https://doi.org/10.1016/j.swevo.2011.02.002

Eiben, A., Smit, S.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**(1), 19–31 (2011). https://doi.org/10.1016/j.swevo.2011.02.001

Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE (2009). http://coco.gforge.inria.fr/doku.php?id=bbob-2015-downloads. Updated February 2010

Guo, H., Li, Y., Li, J., Sun, H., Wang, D., Chen, X.: Differential evolution improved with self-adaptive control parameters based on simulated annealing. Swarm Evol. Comput. **19**, 52–67 (2014). https://doi.org/10.1016/j.swevo.2014.07.001

Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions. Technical Report RR-6829, INRIA (2009). http://coco.gforge.inria.fr/doku.php?id=bbob-2015-downloads. Updated February 2010

Hutter, F., Hamadi, Y., Hoos, H.H., Leyton-Brown, K.: Performance prediction and automated tuning of randomized and parametric algorithms. In: Benhamou, F. (ed.) Principles and Practice of Constraint Programming—CP 2006, pp. 213–228. Springer, Berlin (2006). https://doi.org/10.1007/11889205_17

Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proceedings of LION-5, pp. 507–523 (2011)

Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K.: Algorithm runtime prediction: methods & evaluation. Artif. Intell. **206**, 79–111 (2014). https://doi.org/10.1016/j.artint.2013.10.003

Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Low-budget exploratory landscape analysis on multiple peaks models. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference—GECCO '16. ACM Press (2016). https://doi.org/10.1145/2908812.2908845

Li, J., Kendall, G.: A hyper-heuristic methodology to generate adaptivestrategies for games. IEEE Trans. Comput. Intell. AI Games **PP**(99), 1–1 (2015). https://doi.org/10.1109/TCIAIG.2015.2394780

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation—GECCO '11. ACM Press (2011). https://doi.org/10.1145/2001576.2001690

Naumann, D.S., Evans, B.J., Walton, S., Hassan, O.: A novel implementation of computational aerodynamic shape optimisation using modified cuckoo search. Appl. Math. Modell. **40**(7–8), 4543–4559 (2015). https://doi.org/10.1016/j.apm.2015.11.023

Piotrowski, A.P.: Review of differential evolution population size. Swarm Evol. Comput. (2016). https://doi.org/10.1016/j.swevo.2016.05.003

Sarker, R.A., Elsayed, S.M., Ray, T.: Differential evolution with dynamic parameters selection for optimization problems. IEEE Trans. Evol. Comput. **18**(5), 689–707 (2014). https://doi.org/10.1109/TEVC.2013.2281528

Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. **41**(1), 1–25 (2008). https://doi.org/10.1145/1456650.1456656

Storn, R.: Differential Evolution (DE) for Continuous Function Optimization. http://www1.icsi.berkeley.edu/~storn/code.html (2016). Accessed 6 Jul 2016

Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)

Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical Report. Nanyang Technological University, Singapore (2005)

Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation. IEEE (2013). https://doi.org/10.1109/cec.2013.6557555

Walton, S., Hassan, O., Morgan, K.: Reduced order mesh optimisation using proper orthogonal decomposition and a modified cuckoo search. Int. J. Numer. Methods Eng. **93**(5), 527–550 (2013a). https://doi.org/10.1002/nme.4400

Walton, S., Hassan, O., Morgan, K.: Selected engineering applications of gradient free optimisation using cuckoo search and proper orthogonal decomposition. Arch. Comput. Methods Eng. **20**(2), 123–154 (2013b). https://doi.org/10.1007/s11831-013-9083-7

Walton, S., Hassan, O., Morgan, K.: Strategies for generating well centered tetrahedral meshes on industrial geometries. In: Perotto, S., Formaggia, L. (eds.) New Challenges in Grid Generation and Adaptivity for Scientific Computing, pp. 161–180. Springer, Berlin (2015). https://doi.org/10.1007/978-3-319-06053-8_8

Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)

Zamuda, A., Brest, J.: Self-adaptive control parameters randomization frequency and propagations in differential evolution. Swarm Evol. Comput. **25**, 72–99 (2015). https://doi.org/10.1016/j.swevo.2015.10.007

Zhang, J., Sanderson, A.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009). https://doi.org/10.1109/tevc.2009.2014613