Applied Soft Computing 12 (2012) 2740–2755

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Repeated weighted boosting search for discrete or mixed search space and multiple-objective optimisation[☆]

Scott F. Page[a,1], Sheng Chen[b,c,*], Chris J. Harris[b], Neil M. White[b]

[a] Waterfall Solutions Ltd., Guildford, Surrey GU2 9JX, UK
[b] Electronics and Computer Science, Faculty of Physical and Applied Sciences, University of Southampton, Southampton SO17 1BJ, UK
[c] Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia

ARTICLE INFO

Article history:
Received 24 November 2010
Received in revised form 15 February 2012
Accepted 5 March 2012
Available online 24 April 2012

Keywords:
Global optimisation
Constrained optimisation
Discrete and mixed search space
Multiple-objective optimisation
Pareto-optimality
Genetic algorithms
Repeated weighted boosting search

ABSTRACT

Repeated weighted boosting search (RWBS) optimisation is a guided stochastic search algorithm that is capable of handling the difficult optimisation problems with non-smooth and/or multi-modal cost functions. Compared with other alternatives for global optimisation solvers, such as the genetic algorithms and adaptive simulated annealing, RWBS is easier to implement, has fewer algorithmic parameters to tune and has been shown to provide similar levels of performance on many benchmark problems. In its original form, however, RWBS is only applicable to unconstrained, single-objective problems with continuous search spaces. This contribution begins with an analysis of the performance of the original RWBS algorithm and then proceeds to develop a number of novel extensions to the algorithm which facilitate its application to a more general class of optimisation problems, including those with discrete and mixed search spaces as well as multiple objective functions. The performance of the extended or generalised RWBS algorithms are compared with other standard techniques on a range of benchmark problems, and the results obtained demonstrate that the proposed generalised RWBS algorithms offer excellent performance whilst retaining the benefits of the original RWBS algorithm.

1. Introduction

Guided stochastic search algorithms are widely used to solve complex optimisation problems where gradient based methods are ineffective, such as when the cost function is non-smooth and/or multi-modal. Two such global optimisation methods, the genetic algorithms (GAs) [1–4] and the adaptive simulated annealing (ASA) [5–8], are well-known within the engineering community and have now become classic global optimisation solvers. More recently, repeated weighted boosting search (RWBS) [9] was proposed as a guided search or meta-heuristic global optimisation algorithm. RWBS is essentially a multi-start search technique [10], where the local optimisation mechanism is based on an iterative, adaptive, weighted convex combination. Interestingly, this convex combination is analogous to the crossover operator in a GA. In conjunction with a reflection operator, the convex combination generates new solutions in a manner similar to the simplex method. The adaptive weight update process is a modified boosting technique [11]. The advantages of RWBS [9] include ease of implementation, very few number of tuning parameters, and capable of achieving the levels of performance comparable with the standard benchmark techniques, such as the GA and ASA. A number of applications have since been reported, covering the diverse fields of machine learning, chaotic system stabilisation, image and signal processing as well as wireless communication designs [9,12–27].

Whilst the original RWBS algorithm is a very useful optimisation tool for diverse applications, its original form proposed in [9] is restricted to unconstrained, single-objective optimisation problems with continuous search spaces. Many real-life problems are however constrained, multiple-objective optimisation problems with discrete and/or mixed search spaces. An example of such more generic problems is the optimisation encountered in the authors' research on optimal sensor scheduling [28]. With this motivation, this contribution proposes a number of novel extensions to the original RWBS algorithm to facilitate its application in a more general context. In particular, the aim is to exploit RWBS to solve the problems which can be formulated in the following generic framework:

$$\min_{\mathbf{u}\in\mathrm{U}} f(J_1(\mathbf{u}), J_2(\mathbf{u}), \ldots, J_N(\mathbf{u})) \tag{1}$$

$$\text{s.t.}\quad \begin{array}{l}\mathbf{g}(\mathbf{u}) = \mathbf{0}\\ \mathbf{h}(\mathbf{u}) \le \mathbf{0}\end{array}$$

☆ The authors acknowledge the support of Phase 1 of the UK MOD DIF DTC under Project 8.1 *Active Sensor Management*.

* Corresponding author at: Electronics and Computer Science, Faculty of Physical and Applied Sciences, University of Southampton, Southampton SO17 1BJ, UK.
E-mail addresses: scottpage888@gmail.com (S.F. Page), sqc@ecs.soton.ac.uk (S. Chen), cjh@ecs.soton.ac.uk (C.J. Harris), nmw@ecs.soton.ac.uk (N.M. White).
[1] The author was with Electronics and Computer Science, University of Southampton, when the work reported in this paper was carried out.

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.03.056

where $\mathbf{u} = [u_1\, u_2 \cdots u_n]^{\mathrm{T}}$ is the n-dimensional vector of bounded decision variables to be optimised, U denotes the feasible set of $\mathbf{u}$, $\mathbf{g}(\mathbf{u})$ is a vector of equality constraints, $\mathbf{h}(\mathbf{u})$ is a vector of inequality constraints, $J_i(\mathbf{u})$ is the ith objective function, and f is the objective preference function which may or may not be present. Note that U can represent a continuous space, a discrete space or a mixed space. Evaluation of the objective function may be analytic or procedural, and the cost function is not necessarily continuous or differentiable.

Specifically, the paper begins by providing new and extensive performance analysis results for the original RWBS algorithm. Subsequently, in order to permit optimisation over discrete and mixed search spaces, a new method is proposed which allows the convex combination operator to generate new points in the discrete space. The resulting RWBS algorithm for discrete and mixed search spaces is shown to have a similar efficiency to the original RWBS for continuous search spaces. Additionally, a number of methods for incorporating variable constraints are briefly discussed, including penalty functions and repair operators. The original RWBS algorithm is then modified for the use in multiple-objective problems where no objective preference structure is available. The resulting algorithm maintains a set of Pareto-optimal solutions for subsequent inspection by the designer, similar to the well-known non-dominated sorting genetic algorithm (NSGA-II) algorithm [29]. The performance of the resulting algorithm, the new 'Pareto-RWBS' algorithm, is assessed using some well-known benchmark problems, and the modified RWBS algorithm is shown to offer promising levels of performance, whilst retaining the attractive properties of the original version.

The reminder of this paper is organised as follows. Section 2 describes the original RWBS algorithm and compares its performance quantitatively with the following conventional techniques: a multi-start gradient-search (MSGS) algorithm with the quasi-Newton method as its local search technique; random search; and a GA. A number of parameter sensitivity experiments are subsequently presented which lead to tuning recommendations for the RWBS. Section 3 proposes a new method for extending the RWBS algorithm to discrete and mixed search spaces. The efficiency of the resulting algorithm is compared with the original RWBS. The computational complexity of the algorithm is also discussed. Constraint handling techniques are briefly discussed in Section 4. Section 5 derives the novel multiple-objective Pareto-RWBS algorithm and presents a number of experiments, comparing the Pareto-RWBS algorithm with the NSGA-II algorithm [29]. Our conclusions and discussions on further work are offered in Section 6.

2. Original RWBS optimisation algorithm

The optimisation problem considered in [9] is re-produced here for convenience[2]:

$$\min_{\mathbf{u}\in \mathrm{U}} J(\mathbf{u}) \tag{2}$$

where $\mathbf{u} = [u_1\, u_2 \cdots u_n]^{\mathrm{T}}$ is the n-dimensional vector of continuous-valued decision variables to be optimised, U is the feasible set of $\mathbf{u}$ and $J(\mathbf{u})$ is the cost function which can be non-smooth. The original RWBS algorithm [9] is a multi-start search method with the weighted boosting search (WBS) as its local optimiser, which is first presented below.

[2] This will then be extended to the generic optimisation problem (1) outlined in Section 1.

2.1. Weighted boosting search as a local optimiser

Consider a population of P_s points: $\mathbf{u}_i \in \mathrm{U}$ for $1 \le i \le P_s$. These P_s points are initially chosen at random. Define

$$i_{\mathrm{best}} = arg \min_{1\le i\le P_s} J(\mathbf{u}_i) \quad \text{and} \quad \mathbf{u}_{\mathrm{best}} = \mathbf{u}_{i_{\mathrm{best}}}, \tag{3}$$

$$i_{\mathrm{worst}} = arg \max_{1\le i\le P_s} J(\mathbf{u}_i) \quad \text{and} \quad \mathbf{u}_{\mathrm{worst}} = \mathbf{u}_{i_{\mathrm{worst}}}. \tag{4}$$

Next, a (P_s+1)th point is generated by performing a convex combination of $\{\mathbf{u}_i\}_{i=1}^{P_s}$:

$$\mathbf{u}_{P_s+1} = \sum_{i=1}^{P_s} \delta_i \mathbf{u}_i, \tag{5}$$

where the weights satisfy $\delta_i \ge 0$, $1 \le i \le P_s$, and

$$\sum_{i=1}^{P_s} \delta_i = 1. \tag{6}$$

The point $\mathbf{u}_{P_s+1}$ is always within the convex hull defined by $\{\mathbf{u}_i\}_{i=1}^{P_s}$. A mirror image of $\mathbf{u}_{P_s+1}$ is then generated with respect to $\mathbf{u}_{\mathrm{best}}$ and along the direction defined by $\mathbf{u}_{\mathrm{best}} - \mathbf{u}_{P_s+1}$:

$$\mathbf{u}_{P_s+2} = \mathbf{u}_{\mathrm{best}} + (\mathbf{u}_{\mathrm{best}} - \mathbf{u}_{P_s+1}). \tag{7}$$

If $\mathbf{u}_{P_s+1}$ or $\mathbf{u}_{P_s+2}$ are outside U, they can be projected back to U. The best of $\mathbf{u}_{P_s+1}$ and $\mathbf{u}_{P_s+2}$ replaces $\mathbf{u}_{\mathrm{worst}}$ in the population, according to their cost function values. This process is repeated until the population converges. The convergence can be assumed, for example, if

$$\|\mathbf{u}_{P_s+1} - \mathbf{u}_{P_s+2}\| < \xi_{\mathrm{B}}, \tag{8}$$

where the small positive scalar, ξ_{B}, is the chosen accuracy.

The weights, $\{\delta_i\}_{i=1}^{P_s}$, are adapted according to a boosting technique [9]. In the first iteration ($t=0$), the weights are distributed uniformly:

$$\delta_i(0) = \frac{1}{P_s}, \quad 1 \le i \le P_s. \tag{9}$$

In subsequent iterations ($t>0$) weights are updated using the following procedure. First, the cost function values are normalised:

$$\bar{J}_i = \frac{J_i}{\sum_{j=1}^{P_s} J_j}, \quad 1 \le i \le P_s, \tag{10}$$

then a weighting factor $\beta(t)$ is computed according to:

$$\eta(t) = \sum_{i=1}^{P_s} \delta_i(t-1)\bar{J}_i, \quad \beta(t) = \frac{\eta(t)}{1-\eta(t)}. \tag{11}$$

The weights are then updated for $1 \le i \le P_s$:

$$\delta_i(t) = \begin{cases} \delta_i(t-1)\beta(t)^{\bar{J}_i} & \text{if } \beta(t) \le 1, \\ \delta_i(t-1)\beta(t)^{1-\bar{J}_i} & \text{if } \beta(t) > 1, \end{cases} \tag{12}$$

and normalised according to:

$$\delta_i(t) = \frac{\delta_i(t)}{\sum_{j=1}^{P_s} \delta_j(t)}, \quad 1 \le i \le P_s. \tag{13}$$

2.2. Repeated weighted boosting search as a global optimiser

The WBS procedure described above is repeated multiple times with random population initialisation in order to search for a global optimum. Each repetition is termed a generation and an elitist based initialisation is used, which keeps the best solution found in the previous generation. The RWBS global optimisation algorithm can then be described as follows [9].

Specify the algorithmic parameters: population size P_s, number of generations in the repeated search N_g, number of iterations in the WBS N_B and accuracy for terminating the WBS ξ_B.

```
○ Outer loop: generations for $g = 1 : N_g$
  – Generation initialisation: Initialise the population by setting $\mathbf{u}_1^{(g)} = \mathbf{u}_{best}^{(g-1)}$ and randomly generating the rest of the population members $\mathbf{u}_i^{(g)}, 2 \le i \le P_s$, where $\mathbf{u}_{best}^{(g-1)}$ denotes the solution found in the previous generation. If $g = 1$, $\mathbf{u}_1^{(g)}$ is also randomly chosen.
  – Weighted boosting search initialisation: Assign the initial weights $\delta_i(0) = \frac{1}{P_s}, 1 \le i \le P_s$, for the population, and calculate the cost function value of each point:
    $J_i = J(\mathbf{u}_i^{(g)}), \quad 1 \le i \le P_s.$
  – Inner loop: weighted boosting search for $t = 1 : N_B$
    • Step 1: Boosting
      (1) Find:
          $i_{best} = arg \min_{1 \le i \le P_s} J_i,$
          $i_{worst} = arg \max_{1 \le i \le P_s} J_i.$
          Denote $\mathbf{u}_{best}^{(g)} = \mathbf{u}_{i_{best}}^{(g)}$ and $\mathbf{u}_{worst}^{(g)} = \mathbf{u}_{i_{worst}}^{(g)}$.
      (2) Normalise the cost function values:
          $\bar{J}_i = \frac{J_i}{\sum_{j=1}^{P_s} J_j}, \quad 1 \le i \le P_s.$
      (3) Compute the weighting factor $\beta(t)$ according to:
          $\eta(t) = \sum_{i=1}^{P_s} \delta_i(t-1)\bar{J}_i, \quad \beta(t) = \frac{\eta(t)}{1 - \eta(t)}.$
      (4) Update the weights for $1 \le i \le P_s$:
          $\delta_i(t) = \begin{cases} \delta_i(t-1)\beta(t)^{\bar{J}_i} & \text{for } \beta(t) \le 1, \\ \delta_i(t-1)\beta(t)^{1-\bar{J}_i} & \text{for } \beta(t) > 1, \end{cases}$
          and normalise them:
          $\delta_i(t) = \frac{\delta_i(t)}{\sum_{j=1}^{P_s} \delta_j(t)}, \quad 1 \le i \le P_s.$
    • Step 2: Parameter updating
      (1) Construct the $(P_s + 1)$th point using the formula:
          $\mathbf{u}_{P_s+1} = \sum_{i=1}^{P_s} \delta_i(t)\mathbf{u}_i^{(g)}.$
      (2) Construct the $(P_s + 2)$th point using the formula:
          $\mathbf{u}_{P_s+2} = \mathbf{u}_{best}^{(g)} + (\mathbf{u}_{best}^{(g)} - \mathbf{u}_{P_s+1}).$
      (3) Compute the cost function values $J(\mathbf{u}_{P_s+1})$ and $J(\mathbf{u}_{P_s+2})$ for these two points, and find:
          $i_* = arg \min_{i=P_s+1, P_s+2} J(\mathbf{u}_i).$
      (4) The pair $\{\mathbf{u}_{i_*}, J(\mathbf{u}_{i_*})\}$ then replaces $\{\mathbf{u}_{worst}^{(g)}, J_{i_{worst}}\}$ in the population.
    • If $\|\mathbf{u}_{P_s+1} - \mathbf{u}_{P_s+2}\| < \xi_B$, exit Inner loop
  – End of Inner loop The solution found in the gth generation is $\mathbf{u}_{best}^{(g)}$
○ End of Outer loop This yields the solution $\mathbf{u}_{best}^{(N_g)}$
```

Comments: RWBS shares a number of similarities with GAs that employ elitism. Both are population-based techniques which combine a local search based on current members of the population (i.e. convex combination in RWBS and crossover in a GA), and a stochastic search component (the outer loop in RWBS and mutation in a GA), designed to prevent the algorithm from converging towards local optima. The performance of these two techniques can, therefore, be expected to be similar in general. Indeed, the experiments reported in [9] do confirm this. RWBS is most closely related to a GA which employs a convex crossover operator, single offspring selection, and elitism.

Computational complexity: The computational complexity of the RWBS algorithm can easily be analysed in the following manner. There are at most N_B inner loop iterations, each consisting of $\mathcal{O}(P_s)$ operations, where the notation $\mathcal{O}$ represents 'order of'. Each outer loop iteration or 'generation', of which there are N_g, therefore, consists of $\mathcal{O}(N_B P_s)$ operations. The total computational complexity of the RWBS is, therefore, given by $\mathcal{O}(N_g N_B P_s)$.

2.3. Convergence to global optimality

There are two main factors which affect the global convergence performance of RWBS. In the simplest terms, RWBS is capable of reaching the global optimality due to the stochastic search component. Since there is always a population member generated using a uniform distribution over the decision space domain U, then the algorithm will, in the limit, converge to a global minimum with probability one, assuming that a global minimum lies within the chosen domain. The convergence rate of RWBS is, therefore, lower bounded by that of pure stochastic search:

$$C_{RWBS} \ge C_{RS}, \tag{14}$$

where C_{RWBS} and C_{RS} are the convergence rates of RWBS and pure random search (with the same generating distribution), respectively.

The convergence rate of RWBS is, however, expected to exceed that of random search due to the local search component in Eq. (5). A successful local search technique should converge to the local optimum very rapidly and thus the overall performance of RWBS will, in part, hinge on the combined convergence rate of the convex combination and reflection operators. However, it will be shown that the definition of 'locality' that is appropriate for RWBS is different from that associated with conventional local search such as steepest-descent.

The steepest-descent algorithm is designed to find the local minimum which lies within the (convex) 'basin of attraction' of a single point. It proceeds by computing the gradient of the objective function at a point, and moving in the direction of the negative gradient. The weighted boosting technique in RWBS, however, is a *population*-based technique designed to find a local minimum within the convex hull defined by the population of points $\{\mathbf{u}_i\}$. In many cases, this region will represent a larger subspace than the basin of attraction around a single point, especially if the search space is large and multi-modal. It should also be noted that due to the reflection operator in Eq. (7), the size of the convex hull that is assessed by the RWBS inner loop can increase if $J(\mathbf{u}_{P_s+1}) > J(\mathbf{u}_{P_s+2})$, and the solution found at the end of the inner search loop can in fact be outside of the convex hull defined by the initial population at the beginning of the inner loop. The local search in RWBS is, therefore, technically less restrictive than hill climbing or algorithms directly based on Newton's method, as it is able to explore a larger subspace. In application, RWBS exhibits a *smoothing* or *low-pass* effect which guides the optimisation towards the optimum of a smoothed version of the cost function. Thus, in general, it is conjectured that RWBS is expected to perform well in the cases were the cost function is approximately globally convex. A rigorous proof of this concept is the subject of further work.

2.4. Benchmark convergence experiments

This section presents a number of benchmark convergence experiments which analyse the performance of the original RWBS algorithm in more depth than the presentation in [9]. More specifically, the performance of the RWBS algorithm is compared to a random search, the MSGS, and a GA. A number of well known test functions with different attributes are used.

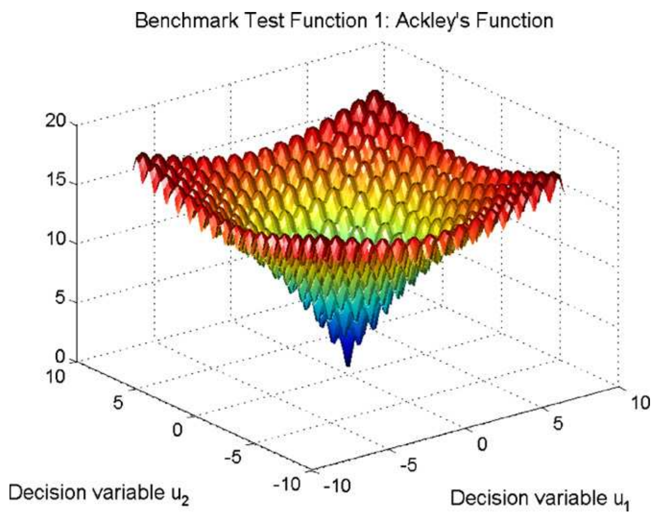


Fig. 1. Two-dimensional Ackley function: there are multiple local minima and one global minimum at $\mathbf{u}_{\text{global}} = [0\,0]^{\text{T}}$ with $J(\mathbf{u}_{\text{global}}) = 0$.

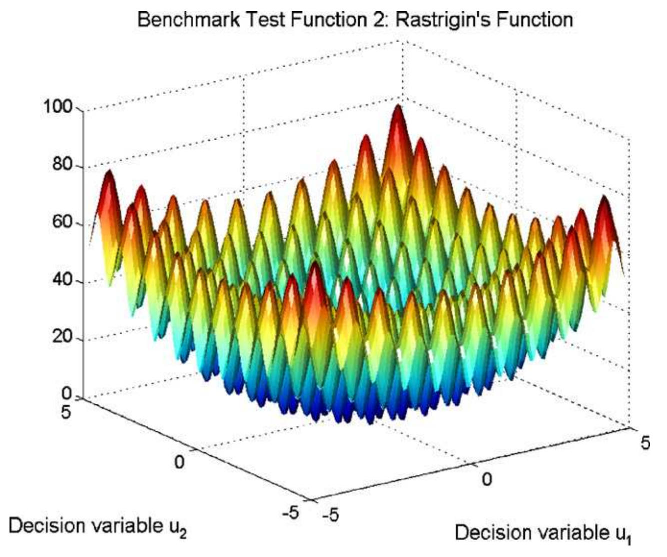


Fig. 2. Two-dimensional Rastrigin function: there are multiple local minima and one global minimum at $\mathbf{u}_{\text{global}} = [0\,0]^{\text{T}}$ with $J(\mathbf{u}_{\text{global}}) = 0$.

The first test function considered is a two-dimensional version of the widely used Ackley function:

$$J(\mathbf{u}) = -\,20\exp\left(-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{2}u_i^2}\right) - \exp\left(\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi u_i)\right) + \,20 + e, \tag{15}$$

where $\mathbf{u} = [u_1\, u_2]^{\text{T}}$. This function is illustrated in Fig. 1. For points outside the interval $\{-10 \le u_j \le 10\}$, $j = 1, 2$, the cost function is assigned a value of 100.

The second test function considered is the well-known two-dimensional Rastrigin function:

$$J(\mathbf{u}) = 20 + u_1^2 + u_2^2 - 10(\cos(2\pi u_1) + \cos(2\pi u_2)). \tag{16}$$

The surface of this function is illustrated in Fig. 2. For points outside the interval $\{-5 \le u_j \le 5\}$, $j = 1, 2$, the cost function is again assigned a value of 100.

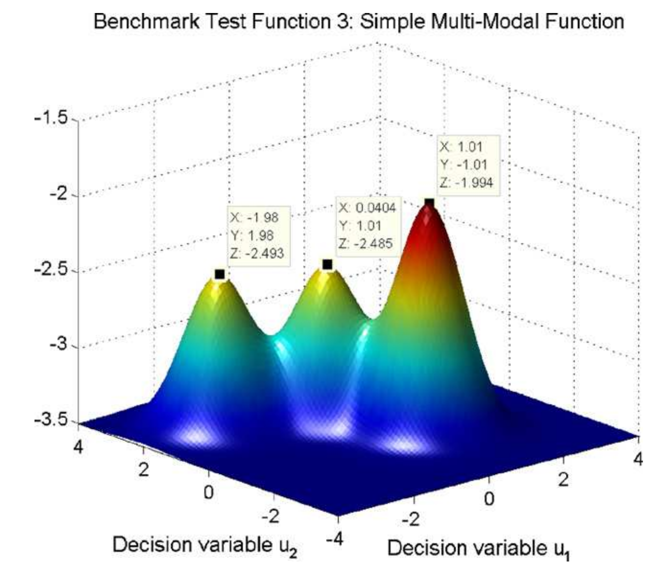


Fig. 3. Two-dimensional simple multi-modal benchmark cost function (the plot is inverted for clarity): there are two local minima corresponding to $\mathbf{u}_{\text{local1}} \approx [0\,1]^{\text{T}}$ with $J(\mathbf{u}_{\text{local1}}) \approx 2.5$ and $\mathbf{u}_{\text{local2}} \approx [-2\,2]^{\text{T}}$ with $J(\mathbf{u}_{\text{local2}}) \approx 2.5$, respectively, as well as the single global minimum at $\mathbf{u}_{\text{global}} \approx [1\,-1]^{\text{T}}$ with $J(\mathbf{u}_{\text{global}}) \approx 2.0$.

The above two functions are standard benchmark cost functions widely used in assessing the performance of optimisation algorithms. Both are non-smooth and have multiple local optima which should present difficulties to the MSGS. RWBS and GAs are generally expected to provide superior performance over the MSGS in these types of problems.

Thirdly, a simple two-dimensional function with two local minima and one global minimum is tested, and the cost function considered is given by

$$J(\mathbf{u}) = 1.5(1 - \exp(-(u_1 - 1)^2 - (u_2 + 1)^2)) + 1 - \exp(-(u_1 + 2)^2 - (u_2 - 2)^2) + 1 - \exp(-u_1^2 - (u_2 - 1)^2). \tag{17}$$

The surface of this cost function is illustrated in Fig. 3. For points outside the interval $\{-4 \le u_j \le 4\}$, $j = 1, 2$, the cost function is assigned a value of 100. This simpler cost function is smooth and its global optimum has a large basin of attraction, which should not present much difficulty to the MSGS. This function was chosen in order to compare the performance of RWBS with the MSGS on simple problems where the MSGS technique is known to perform well.

In order to create a basic MSGS algorithm, single sample is repeatedly taken from a uniform distribution over the predefined variable intervals and used to seed the MATLAB Optimisation Toolbox function *fminunc* [30]. Thus, in this application, the local gradient search optimiser of the MSGS adopts the optimisation algorithm *fminunc*, which employs a medium-scale algorithm based on the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) Quasi-Newton method with a mixed quadratic and cubic line search procedure [30]. The random search method employed is also based on repeated uniform sampling over the respective variable intervals. A GA was created using the MATLAB function *ga* from the Genetic Algorithm and Direct Search Toolbox. This high performance GA is based on the following components: a scattered crossover, zero-mean Gaussian mutation, stochastic uniform selection, double-vector real coding, migration, and a relative rank-based fitness scaling process.

Fig. 4 presents the results of the performance comparison on Ackley function. The number of cost function evaluations is plotted against the mean best cost function value found so far, averaged

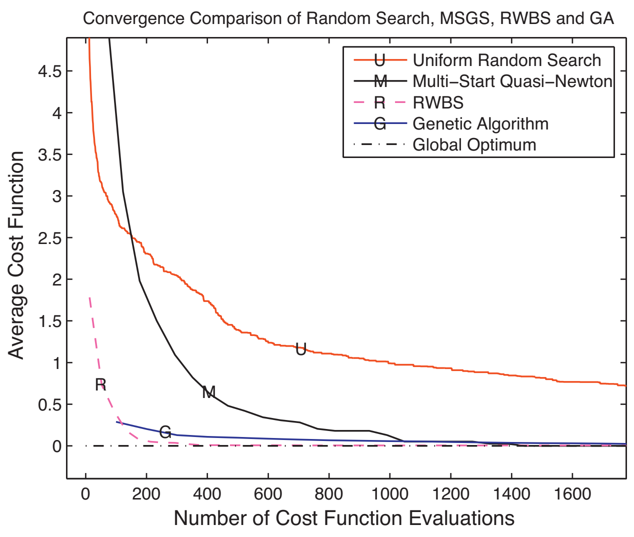


Fig. 4. Convergence performance comparison of the random search, MSGS, RWBS and GA on the test problem, Ackley function (15).

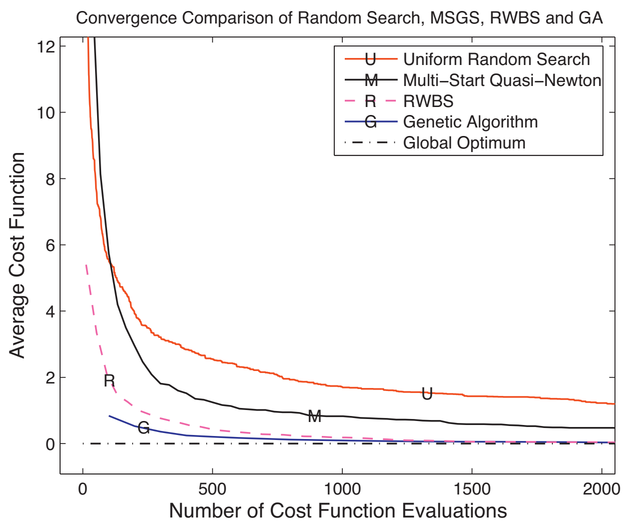


Fig. 5. Convergence performance comparison of the random search, MSGS, RWBS and GA on the test problem, Rastrigin function (16).

over 100 Monte-Carlo experiments of each algorithm. The RWBS algorithm was used with the following settings which were found to be appropriate using very rapid coarse tuning experiments: $P_s = 11$, $N_g = 150$, $N_B = 15$ and $\xi_B = 0.02$. The GA settings were mixed default values and empirically chosen ones. The default parameters of the GA used are: crossover fraction = 2, elitism count = 2 and migration interval = 20, which remained constant throughout all the experiments. The initial population range and the population size of 100 by contrast were chosen using coarse tuning experiments. Default settings for the maximum number of iterations and termination tolerances were used for the MSGS algorithm as they were deemed appropriate for the problem. From Fig. 4, it can be seen that the convergence rates of the GA and RWBS are much faster than that of the MSGS, while the random search has the worst convergence rate and appears to have the difficulty to converge to the global optimum. On average, RWBS finds a solution within 0.05 of the global optimum after only 253 cost function evaluations. The GA requires 1200 cost function evaluations in order to reach the same accuracy, while the MSGS requires 1327 cost function evaluations. The random search procedure remains at least 0.6 away from the global optimum after 2000 cost function evaluations.

It is noted that a number of rule-of-thumb-based approaches have been proposed for tuning GAs, for example in [31]. One such approach for choosing the GA population size is to simply use approximately 10 times the number of dimensions, corresponding to a population size of 20 in this case. This method has been shown to provide good performance in a number of cases. However, the experiments presented herein were repeated with a population size of 20 and the GA results were, in general, found to be inferior to those generated using the above-described parameters. One possible explanation for this observation is that the rule-of-thumb may make assumptions about the nature of the GA (e.g. regarding coding, crossover and mutation operators, and fitness scaling mechanisms etc) which are not justified in this case. Additional rules-of-thumb for selecting GA parameters, such as crossover rate, mutation rate, and number of generations have also been proposed in the literature. Investigating the relative performance of RWBS against the GA tuned using appropriate rules is, therefore, an interesting area of future work. All the results presented in this section are based on direct trial and error parameter tuning for the RWBS, GA and MSGS, rather than using rules-of-thumb. For the benchmark problem of Ackley function (15), the RWBS with only brief tuning

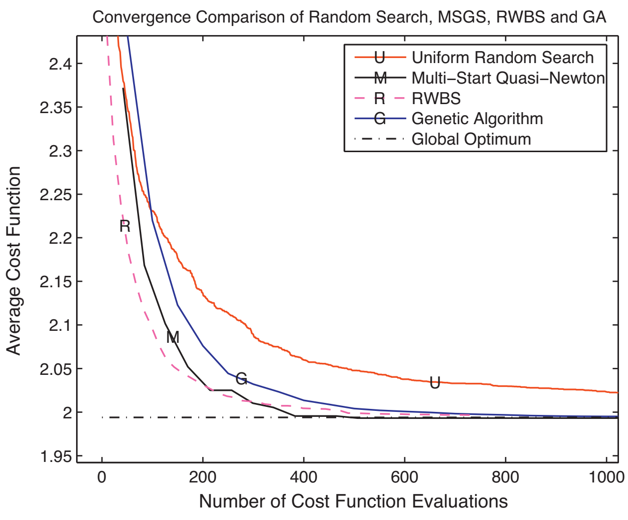


Fig. 6. Convergence performance comparison of the random search, MSGS, RWBS and GA on the simple multi-modal test problem of (17).

of the two major parameters, P_s and N_B, performs slightly better than the GA and significantly better than the MSGS algorithm.

Fig. 5 shows the results generated with Rastrigin function. In this case the RWBS algorithm was again used with the following settings which were found to be appropriate: $P_s = 11$, $N_g = 150$, $N_B = 15$ and $\xi_B = 0.02$. Once again, the default GA settings were used, except for the initial population range and the population size of 100 which were chosen empirically. Default settings for the maximum number of iterations and termination tolerances were again used for the MSGS algorithm. For this example, the initial convergence rate of the GA is slightly faster than that of the RWBS, but the RWBS finds a value within 0.05 of the global optimum within 1773 cost function evaluations while the GA reaches the same solution accuracy within 1900 cost function evaluations. As expected, the MSGS requires significantly more cost function evaluations (in fact larger than 6900) to reach a similar level of accuracy. Again, the random search has the worst convergence rate.

Fig. 6 illustrates the performance comparison using the function defined in Eq. (17). In this case, the RWBS algorithm was applied with the following settings which were empirically found to be appropriate: $P_s = 6$, $N_g = 150$, $N_B = 5$ and $\xi_B = 0.02$. The GA population

Table 1
Convergence performance comparison: mean number of cost function evaluations required to identify solution within the accuracy of 0.05 from the global optimum.

Algorithm\test function	Ackley	Rastrigin	Simple multi-modal
RWBS	**253**	**1773**	**173**
GA	1200	1900	300
MSGS	1327	>6900	214
Random Search	>2000	>6900	>2000

Boldface obviously indicates the best value among the four values.

size was once again set to 100. Default settings for the maximum number of iterations and termination tolerances were again used for the MSGS algorithm. It is clear from Fig. 6 that neither the RWBS nor the GA can significantly outperform the MSGS algorithm. This is expected as the basin of attraction for the global minimum in this example is very large, compared to that of the global minimum of Ackley or Rastrigin function. The MSGS is, therefore, likely to find the global minimum quite rapidly. In fact, both the RWBS and MSGS algorithms outperform the GA in this case. More specifically, the results show that the RWBS and MSGS find solutions within 0.05 of the global minimum after 173 and 214 cost function evaluations, respectively, while the GA requires 300 cost function evaluations to achieve the same solution accuracy.

Table 1 summarises the performance comparison for the three benchmark test problems. In each case, the RWBS converges to within the accuracy of 0.05 from the true global minimum faster than all the other algorithms. This provides additional support to the conjecture that the RWBS is a very promising algorithm for black-box optimisation problems, especially considering the very minor tuning requirements involved and the ease of implementation. In most cases examined herein the RWBS algorithm was found to perform well with the two key parameters set to $5 \le P_s \le 15$ and $5 \le N_B \le 20$. Coarse tuning within these ranges can be used to fine tune the algorithm performance very quickly.

2.5. RWBS algorithmic parameter sensitivity

The performance of RWBS on a given problem is dependent on the cost function to be optimised, and on the chosen RWBS algorithmic parameters. In order to make efficient use of RWBS, therefore, the robustness or sensitivity of the algorithm's performance to the algorithmic parameter variations is investigated. As the original RWBS algorithm has a total of four tuning parameters: the population size P_s, the number of generations N_g, the number of boosts or iterations in the WBS N_B and the WBS termination threshold ξ_B, the parameter sensitivity is a surface on the four-dimensional parameter space. If each parameter is tested over N discrete values, then the total number of optimisation runs extends to N^4. Moreover, it is very difficult to visualise a high-dimensional surface. However, practical experience has suggested that the key algorithmic parameters of interest are the population size, P_s, and the number of boosts, N_B. In practice, the number of generations, N_g, must be chosen to be sufficiently large to ensure a global convergence, in a similar way to which the number of generations used in a GA is chosen. Varying the number of generations in this context would not provide significant insight as it will simply effect whether or not the algorithm finds a solution, namely, converges successfully or not. The termination threshold, ξ_B, is of more direct interest with respect to the convergence rate. Varying the termination threshold would, therefore, provide additional insight into the rate of convergence. However, it is clear that one can simply remove the inner loop convergence test and only terminate the inner loop when the number of boosts, N_B, has been reached. In this way, there would be no need to specify the parameter ξ_B, and the parameter, N_B, would become the remaining control mechanism for convergence rate.

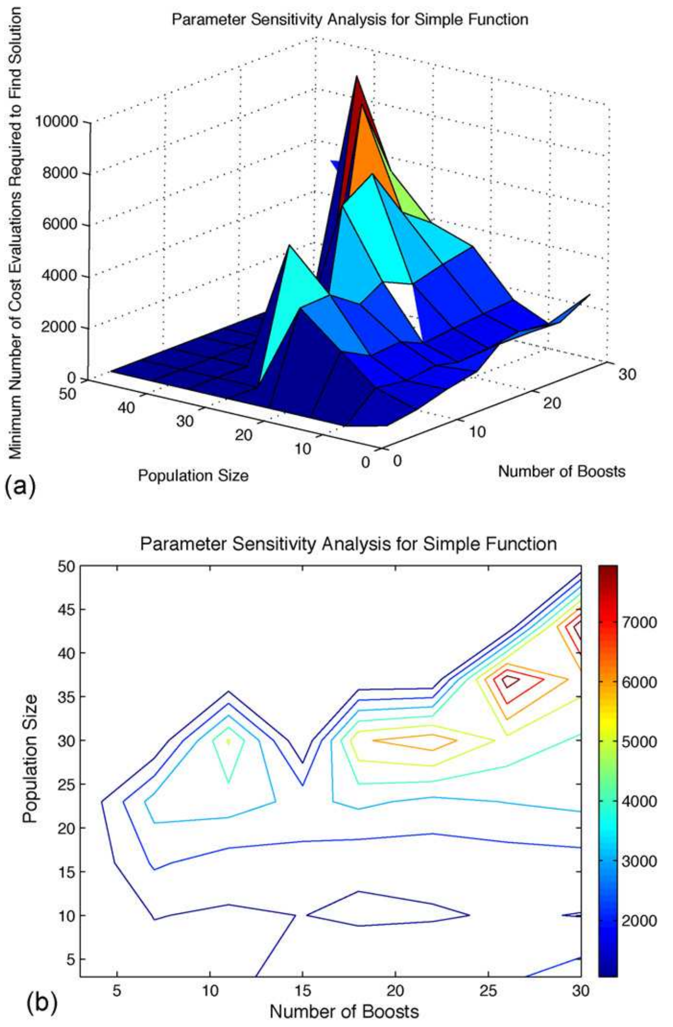


Fig. 7. Parameter sensitivity surface (a) and contour plot (b) for the RWBS applied to the simple multi-modal test problem of (17).

This analysis also supports the empirical experience suggesting that P_s and N_B are the key algorithmic parameters.

Therefore, we test the RWBS algorithm over a range of values for the two key parameters, P_s and N_B. It is informative to perform this process in order to provide some pragmatic suggestions for approaches to parameter choice. Specifically, for all the three benchmark cases, the algorithm was tested over the parameter ranges of $3 \le P_s \le 50$ and $3 \le N_B \le 30$, and the results were averaged over 50 Monte-Carlo simulations. The number of generations, P_s, was set to a very high value, and the termination threshold, ξ_B, was kept constant. For each pair of P_s and N_B, we recorded the minimum number of cost function evaluations required to maintain the performance level as described in the previous section.

Fig. 7(a) and (b) presents the parameter sensitivity surface and its contour plot, respectively, obtained by applying the RWBS algorithm to the simple multi-modal test problem of (17). It should be pointed out that the areas of the surface that appear to require zero cost evaluation correspond to those simulations where a solution of the required accuracy was not found. It is immediately noticeable from Fig. 7 that, in general, the larger the population size, the larger the number of cost function evaluations required. This provides justification for the suggestion to use small population sizes given in [9]. It is interesting to note that the optimum performance appears to occur when a population size of around 10 is chosen. As the population size increases, the algorithm requires more cost

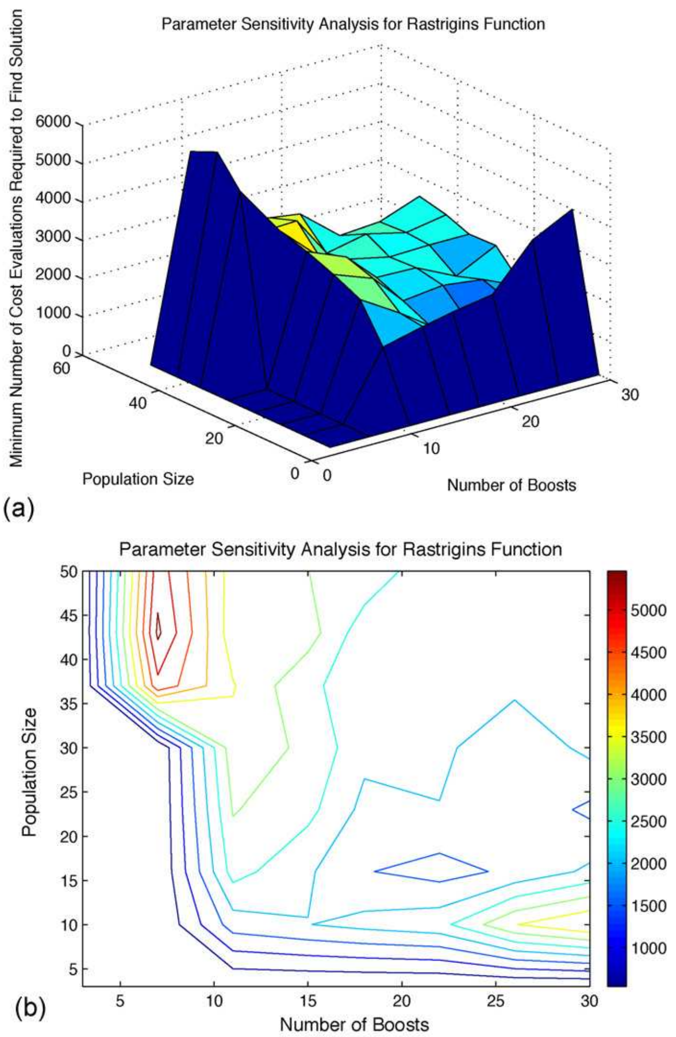


Fig. 8. Parameter sensitivity surface (a) and contour plot (b) for the RWBS applied to the benchmark test problem of Rastrigin function (16).

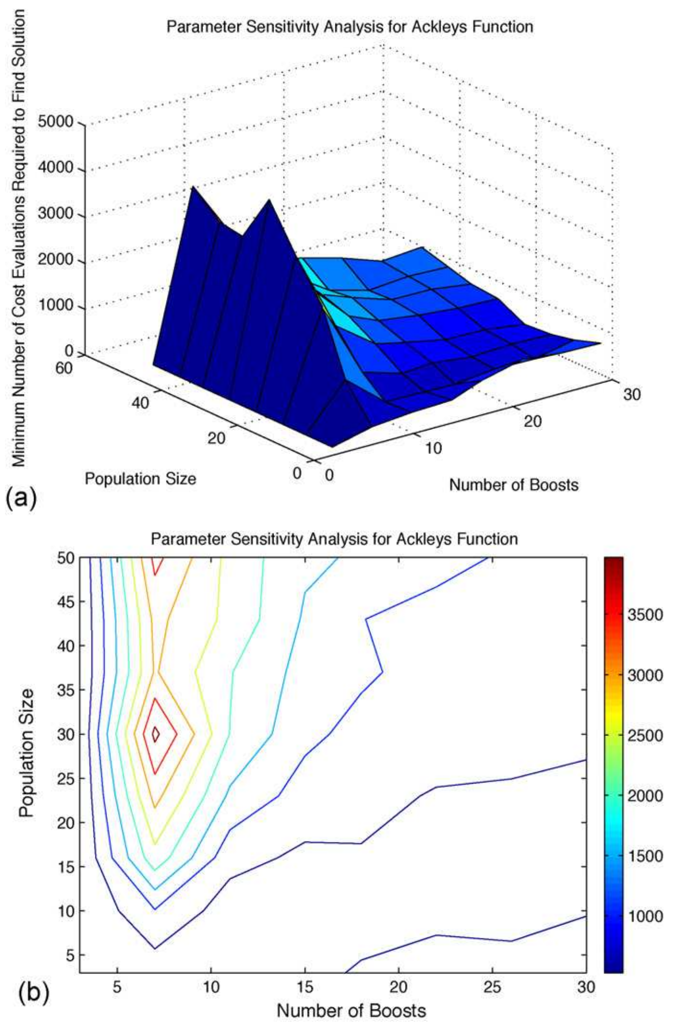


Fig. 9. Parameter sensitivity surface (a) and contour plot (b) for the RWBS applied to the benchmark test problem of Ackley function (15).

evaluations to converge. The results of Fig. 7 also indicate a level of relative insensitivity to the number of boosts for this example, and this provides evidence for the assertion that the population size is a more dominant key parameter, in terms of effect on the algorithm's performance.

Fig. 8(a) and (b) plots the parameter sensitivity surface and its contour plot, respectively, obtained by applying the RWBS algorithm to the benchmark test problem, Rastrigin function of (16). Unlike the case of the simple multi-modal test problem, a different parameter sensitivity surface is observed, in that the algorithm now exhibits more sensitivity to the number of boosts for small and large population sizes. However, it is noted that if the population size is set to an appropriate value (sizes less than 20 seem to offer reasonable performance), there is still a level of relative insensitivity to the number of boosts.

Fig. 9(a) and (b) shows the parameter sensitivity surface and its contour plot, respectively, obtained by applying the RWBS to the benchmark test problem, Ackley function of (15). Similar results to the previous examples are observed, suggesting the relative insensitivity to the number of boosts and the good levels of performance achievable with low population sizes.

In summary, therefore, the parameter sensitivity results indicate that the dominant key parameter of importance is the population size, P_s. If an appropriate choice of this parameter is made, the algorithm exhibits a level of insensitivity to the number of boosts, N_B. In addition, the results demonstrate that small population sizes are capable of providing good levels of performance across a range of different cost functions, further strengthening the potential of the RWBS algorithm for black-box optimisation, in problems such as those relevant to the autonomous sensor management [28]. Moreover, the following trade-off in parameter tuning can be drawn.

Like most stochastic search algorithms, the RWBS must be tuned to each particular problem to achieve the best results. The main trade-off in relation to convergence *success* is between the number of generations, N_g, and the number of boosts, N_B, which balance the stochastic component of the algorithm with its local search component. Appropriate choice of the algorithmic parameters depends on whether the cost function is suspected to contain multiple local minima. As the population size increases, the effect of each boosting iteration is attenuated. This is because a larger population is likely to cover a larger subspace and thus it takes longer for the boosting process to focus on a particular region of interest. Experiments suggest that larger populations, therefore, demand higher values for N_B, and as a consequence, the convergence rate may become slower. The advantage of a larger population is that the probability of encompassing the basin of attraction of the global minimum is increased. If the cost function is also relatively convex then the probability of locating the global minimum in a shorter time period is also increased. Initial

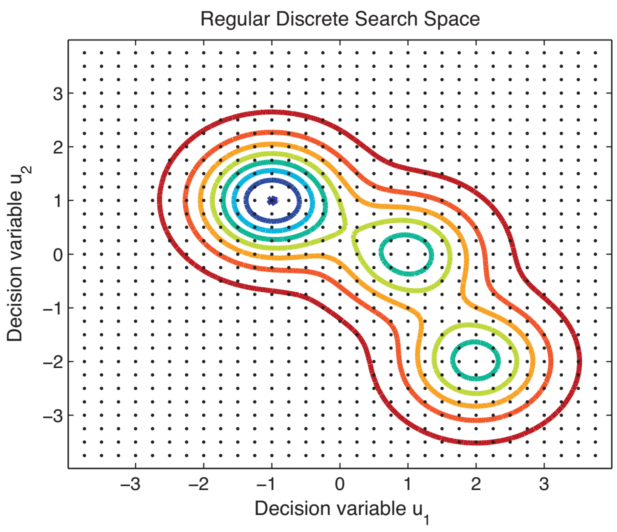


Fig. 10. Example of regular discrete decision variable space overlaid with the contour plot of the cost function (17), where black dot markers correspond to possible decision variable values which are located on the grid with the increment of 0.25.

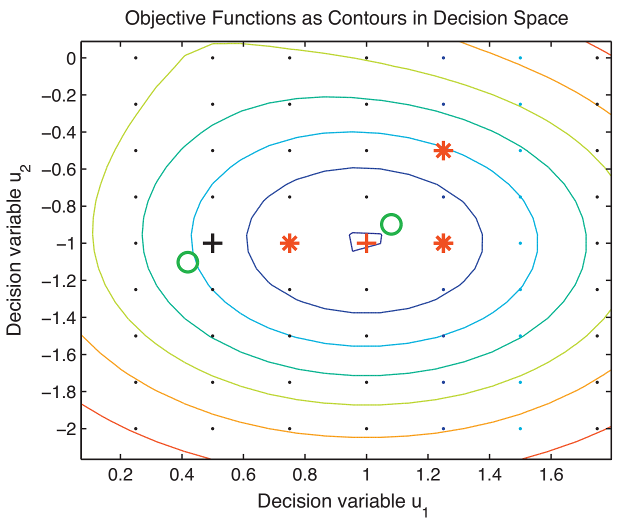


Fig. 11. Illustration of the NNRWBS parameter update on the simple multi-modal function of (17), using a regular discrete search space with feasible points located on the grid with the increment of 0.25 as in Fig. 10. (For interpretation of the references to colour in the text, the reader is referred to the web version of the article.)

experience with the algorithm suggests that smaller populations often provide more efficient results for non-smooth cost functions. The results presented above also indicate that if the population size is chosen appropriately, the algorithm will exhibit relative insensitivity to the number of boosts.

3. Extension to mixed search spaces

The RWBS algorithm described in Section 2 is only applicable to problems where the decision variables are defined over a continuous search space. This is because the weighted convex combination in the local search inner loop, Eq. (5), is a continuous-valued function. However, in many problems, the decision variables are defined over a discrete space, or a mixture of continuous space and discrete space, often described as a 'mixed' space. It should also be noted that in some cases, the cost function may only be defined or evaluated on the discrete points and, therefore, methods based on continuous-valued optimisation followed by discretisation are not always applicable. Consider the case where the search space U is purely discrete, such as that illustrated in Fig. 10. In this case, the search space is regular and feasible points are located on the grid with the increment of 0.25. In order to 'convert' the RWBS algorithm to operate over discrete search spaces, the convex combination operator must be constrained in some way to generate discrete solutions. A modified parameter update step is proposed here and the resulting algorithm is termed the nearest-neighbour RWBS (NNRWBS). This new algorithm retains the original convex combination operator but modifies the generated point so that it is assigned to a feasible value in the discrete space. It is assumed that the underlying search space U is a regular grid in an n-dimensional space.

3.1. Nearest-neighbour parameter update

A simple and efficient method to generate a new discrete point from a weighted combination of points in discrete space is given as follows. Firstly, generate an intermediate point, $\mathbf{v}_1$, by computing the weighted convex combination as in (5), and a second intermediate point, $\mathbf{v}_2$, is then generated by reflection using Eq. (7). Next, assign or 'snap' the new points to the nearest discrete points according to some distance measure, e.g. Euclidean norm. The resulting NNRWBS algorithm is identical to the original RWBS with the exception of the parameter update step which is replaced with the following (this particular example uses the Euclidean norm as the distance metric).

- *Step 2*: NNRWBS parameter update
  (1) Construct the (P_s+1)th point by first generating the intermediate point using:

$$\mathbf{v}_1 = \sum_{i=1}^{P_s} \delta_i(t)\mathbf{u}_i^{(g)}, \tag{18}$$

  and then assigning the new point to the discrete grid using:

$$\mathbf{u}_{P_s+1} = \underset{\mathbf{u}\in \mathrm{U}}{argmin}\|\mathbf{u}-\mathbf{v}_1\|. \tag{19}$$

  (2) To construct the (P_s+2)th point, first use the formula[3]

$$\mathbf{v}_2 = \mathbf{u}_{\text{best}} + (\mathbf{u}_{\text{best}} - \mathbf{v}_1), \tag{20}$$

  and then assign the new point to the discrete grid using:

$$\mathbf{u}_{P_s+2} = \underset{\mathbf{u}\in \mathrm{U}}{argmin}\|\mathbf{u}-\mathbf{v}_2\|. \tag{21}$$

  (3) Compute the cost function values $J(\mathbf{u}_{P_s+1})$ and $J(\mathbf{u}_{P_s+2})$ for these two points, and find:

$$i_* = arg \min_{i=P_s+1,P_s+2} J(\mathbf{u}_i). \tag{22}$$

  (4) The pair $\{\mathbf{u}_{i_*}, J(\mathbf{u}_{i_*})\}$ then replaces $\{\mathbf{u}_{\text{worst}}^{(g)}, J_{i_{\text{worst}}}\}$ in the population.

Note that the 3rd and 4th stages of the new parameter update step remain in the original form, but the first two stages are now modified. The NNRWBS parameter update step is illustrated in Fig. 11. The green circle markers indicate the two intermediate points, $\mathbf{v}_1$ and $\mathbf{v}_2$. The red asterisk markers illustrate the original members of the population, and the red and black cross markers indicate the (P_s+1)th and (P_s+2)th points generated, respectively.[4]

[3] One could equally use $\mathbf{u}_{P_s+1}$ instead of $\mathbf{v}_1$.

[4] Note that if the search space U is a regular grid then the (P_s+1)th point is guaranteed to be within the convex hull defined by $\{\mathbf{u}_i^{(g)}\}_{i=1}^{P_s}$.

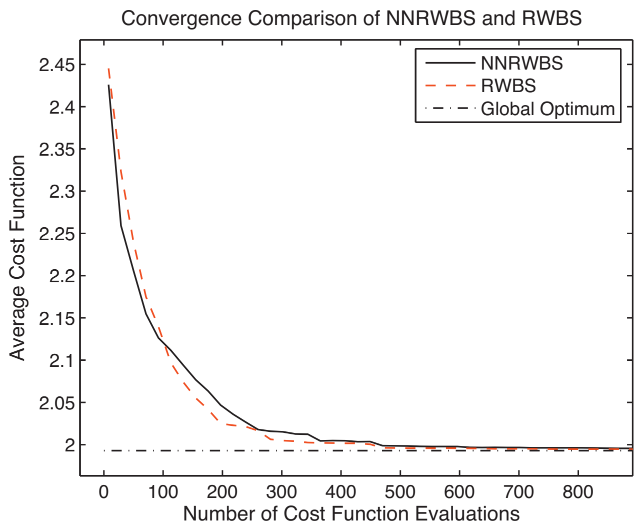


Fig. 12. Convergence performance comparison of the nearest-neighbour RWBS and the original RWBS on the simple multi-modal test problem of (17).

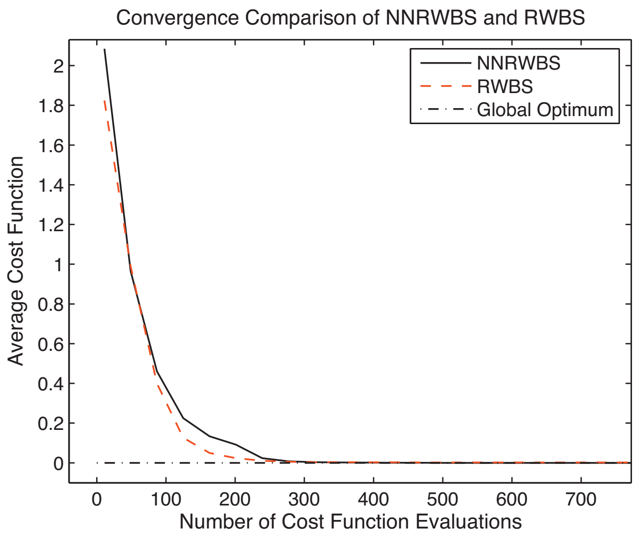


Fig. 13. Convergence performance comparison of the nearest-neighbour RWBS and the original RWBS on the test problem, Ackley function of (15).

Computational complexity: The computational complexity of the NNRWBS algorithm only differs from that of the original RWBS due to the extra two quantisation operations within each inner loop (and only in discrete dimensions). If the structure of the discrete space is known, then the quantisation can, in some cases, be performed relatively cheaply. For instance, in a regular Euclidean space, a simple rounding operation is sufficient. In such cases, the extra computation involved is small, and the overall complexity remains as $\mathcal{O}(N_g N_B P_s)$. In general, however, the complexity will depend on the difficulty involved in assigning the intermediate points to feasible points. If a highly irregular search space is used, a grid-based look up table may be used to identify candidate feasible points and an exhaustive search employed to select the new point.

RWBS for mixed search space: For an optimisation task over a mixed search space U, some of the n decision variables have discrete or integer values, while the rest of the decision variables have continuous values. Application of the RWBS algorithm for optimisation over such a mixed search space can be achieved by applying the NNRWBS parameter update step in the appropriate discrete dimensions, while retaining the original parameter update step in the rest continuous dimensions.

3.2. Benchmark convergence experiments

The performance of the proposed NNRWBS algorithm is assessed by comparison with the original RWBS algorithm operating on an equivalent continuous search space. The simple multi-modal test problem (17) and the benchmark test problem of Ackley function (15) were used in the comparison. The NNRWBS algorithm was applied using a regular discrete search space with feasible points located on the grid with the distance of 0.01 in the both dimensions, and the same algorithmic parameter values as used in the RWBS simulation of Section 2.4 were adopted.

Fig. 12 shows the results obtained for the simple multi-modal cost function (17), while Fig. 13 depicts the results generated for the test problem, Ackley function (15). In the both cases, the performance of the NNRWBS are very similar to those of the original RWBS algorithm for continuous search space, indicating that the modified parameter update step offers a reasonable approach to apply the RWBS in discrete space cases. In addition, as pointed out previously, if the search space is regular, then the quantisation operations in Eqs. (19) and (21) can be computed relatively

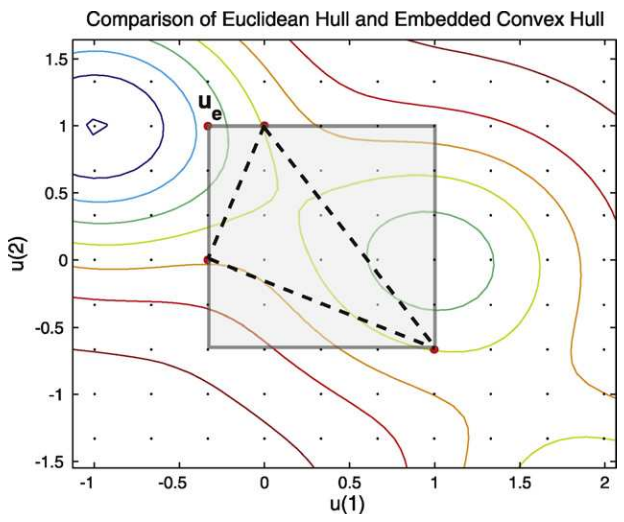


Fig. 14. Comparison of the Euclidean convex hull and discrete space embedded convex hull, where the shaded grey area is the discrete embedded convex hull while the Euclidean convex hull is depicted by the dashed trace.

easily. This experiment thus suggests that the NNRWBS algorithm retains the efficiency of the original RWBS algorithm.

3.3. Embedded hull parameter update

While the convex combination in the nearest-neighbour parameter update step can only generate intermediate points within the Euclidean convex hull defined by the population, the $(P_s + 1)$th point can, of course, be outside of the Euclidean convex hull. Despite of this, the use of the Euclidean convex operator may be considered restrictive as it does not account for the differences between the notions of continuous and discrete convexity. When discrete spaces are modelled with cell complexes, a number of different types of convexity can be defined, each with different properties [32]. The embedded convex hull is particularly useful in this context due to its simple shape. By identifying the embedded convex hull defined by the population, it is possible to relax the parameter update so that it operates over a larger subspace, potentially leading to superior performance. An illustration of this concept can be found in Fig. 14. Some of the points in the embedded convex hull are unreachable through the use of the nearest-neighbour parameter update, for example, the point $\mathbf{u}_e$ in Fig. 14. By augmenting the population with a set of new points, such that the overall population forms a Euclidean hull that covers the embedded hull, all points become reachable. An alternative discrete space parameter update step which results in the embedded hull RWBS (EHRWBS) algorithm, therefore, operates in the following manner: (a) identify the embedded discrete convex hull defined by the current population; (b) find a sparse set of new points such that the overall enlarged population creates a convex hull in Euclidean space which completely covers the discrete embedded hull; and (c) perform the nearest-neighbour convex combination algorithm on the resulting enlarged population, and retain the P_s population members after the update.

Generating a sparse set of points to enclose the discrete embedded hull can be achieved in various ways. A simple method is proposed which exploits the fact that the embedded convex hull will always be an n-dimensional hyper-cuboid. To find a set of points which form a Euclidean hull that encloses the embedded hull, all that is required is to find the intersection of the lines defined by the minimum and maximum points in each decision variable dimension. The number of points required to represent this hull is $n+1$, although in many cases the number of points that needs to be added to the population, m, will be less than $n+1$, as some of the nodes of the hyper-cuboid will already be members of the population. The resulting EHRWBS algorithm is, like the NNRWBS, identical to the original RWBS algorithm apart from the parameter update step in the inner loop. The EHRWBS parameter update step proceeds as follows.

- *Step 2*: EHRWBS parameter update
  (1) Identify the minimum and maximum points in all the n dimensions.
  (2) Identify the m nodes of the embedded hull hyper-cuboid that are not already members of the population, and augment the population with these points to create an intermediate enlarged population of size P_s+m.
  (3) To construct the th point, first generate an intermediate point using:

$$\mathbf{v}_1 = \sum_{i=1}^{P_s+m} \delta_i(t)\mathbf{u}_i^{(g)}, \tag{23}$$

  and then assign the new point to the discrete grid using:

$$\mathbf{u}_{P_s+1} = arg\min_{\mathbf{u}\in U}\|\mathbf{u}-\mathbf{v}_1\|. \tag{24}$$

  (4) To construct the (P_s+2)th point, first generate an intermediate point using the formula

$$\mathbf{v}_2 = \mathbf{u}_{\text{best}} + (\mathbf{u}_{\text{best}} - \mathbf{v}_1), \tag{25}$$

  and then assign the new point to the discrete grid using:

$$\mathbf{u}_{P_s+2} = arg\min_{\mathbf{u}\in U}\|\mathbf{u}-\mathbf{v}_2\|. \tag{26}$$

  (5) Compute the cost function values $J(\mathbf{u}_{P_s+1})$ and $J(\mathbf{u}_{P_s+2})$ for these two points, and find:

$$i_* = arg\min_{i=P_s+1,P_s+2} J(\mathbf{u}_i). \tag{27}$$

  (6) The pair $\{\mathbf{u}_{i_*}, J(\mathbf{u}_{i_*})\}$ then replaces $\{\mathbf{u}_{\text{worst}}^{(g)}, J_{i_{\text{worst}}}\}$ in the enlarged population.
  (7) Remove the m worst points from the enlarged population of size P_s+m, so that the size of the population remains P_s.

4. Extension to constrained optimisation

There has been a wide range of studies investigating methods for incorporating constraints into stochastic search processes, in particular, for evolutionary algorithms [33,34]. One of the simplest methods to handle constraints in stochastic search technique is to simply reject any infeasible solutions which are generated, and this is sometimes known as the death-penalty method. The most popular approach, however, is the application of less drastic penalty functions [35]. Penalty function based techniques transform the original constrained problem into an unconstrained problem, where the penalty function is added to the original cost function. Sometimes, the penalty is made proportional to the total constraint violation [33]. Various kinds of penalty functions have been investigated, and they can be broadly classified according to whether they are static or dynamic, and whether they are adaptive or non-adaptive. Dynamic penalty functions reduce the effect of the constraints as the optimisation process evolves. The main criticism of the penalty function approach is that it is very difficult to tune the penalty functions. However, the authors of [34] have shown that by searching both the original decision space and an associated Lagrange-multiplier subspace, that some of the problems associated with tuning penalty functions can be avoided.

Both the death-penalty and penalty-function methods can be used directly with the RWBS. However, it is noted that experimentation suggests that the reflection operator in the algorithm often yields infeasible solutions, and thus the question arises as to how to deal with such points. Simply deleting the infeasible points generated in the reflection operation may amount to question the legitimacy of this operator, and will at least reduce the effectiveness of the reflection operation. An alternative approach is to use a repair method, whereby infeasible points are assigned a value in the decision space which places them at the boundary of the constraint. Application of such methods is the subject of future work.

5. Extension to multiple objective problems

It is well known that if a priori information regarding the relative importance of different optimisation objectives is available, the multiple-objective optimisation problem can be reformulated as a single-objective problem, such as in the simple weighting method. Techniques which operate in this context can be termed 'non-Pareto methods' as they search for solutions to surrogate problems. However, if preference information is not available or the nature of the Pareto-frontier is of direct interest, then the optimisation algorithm must generate a *set* of Pareto-optimal solutions. Ideally, the solutions should be well distributed across the Pareto-frontier.[5] These methods can be termed 'Pareto methods'. The solution set can then be used to consider which solution is most appropriate for the particular problem and to implicitly infer some relative importance of the objectives. Several methods have been proposed to adapt common population based stochastic search techniques, in particular GAs, to generate Pareto-optimal sets. An introduction to this topic can be found in [36]. Other multiobjective evolutionary algorithms can be found in [37,38], and detailed review of multi-objective evolutionary algorithms is given in [39,40].

There are two main aspects to designing an efficient algorithm for Pareto-optimisation. Firstly, the algorithm needs to embody a mechanism which drives solutions towards the Pareto-frontier and, secondly, there needs to be a mechanism which ensures a good distribution of solutions across the frontier. Typically, a form of Pareto-ranking or Pareto-sorting is used to guide the

[5] What is meant by 'well distributed' in this context is often problem specific, but a reasonably uniform distribution over the Pareto-frontier may be deemed desirable.

optimisation towards the frontier [36]. These techniques effectively modify the cost value or fitness value for a solution depending on its performance *relative* to other solutions in the set, in contrast to the absolute notion of optimality used in conventional optimisation. Solutions which are 'non-dominated' or mildly dominated (i.e. only dominated by a limited number of other solutions) are attributed a higher fitness or lower cost than those which are strongly dominated. This promotes the generation of more non-dominated solutions. Distribution of solutions across the Pareto-frontier is commonly achieved using 'sharing' or 'niche' methods [36,41]. Sharing methods distribute an individual's fitness depending on how many solutions are nearby it,[6] thus encouraging spread and avoiding the problems associated with 'genetic drift'. The difficulty with sharing techniques is that the user must define the so-called 'sharing parameter' [36]. In general, manual fixing of the sharing parameter requires knowledge of the objective function and adds to the tuning complexity of the optimisation algorithm. However, the authors of [43] have shown that if the sharing process is considered from a density estimation viewpoint, the sharing parameter is analogous to a smoothing parameter, and thus the sharing process can be automated by modelling the population density using density estimation techniques, such as Parzen window density estimator [44]. The authors of [43] further show that with appropriate choice of kernel function, it is possible to create a parameterless sharing process. It should be noted, however, that the density estimator uses recommended heuristics for its internal parameters. In contrast, a distance based measure is used in [29] which is completely parameterless.

As the RWBS is a population-based stochastic search method, it can be readily adapted to the Pareto-optimisation case by a number of modifications. These include the addition of a Pareto-ranking process and a mechanism which encourages distribution as well as the modified elitism process that retains a larger set of solutions between generation, instead of the single point in the original algorithm.

5.1. Elitism count, pareto ranking, distribution and cost mapping

In the original RWBS algorithm for single-objective optimisation, the elitism process retains the single best solution to the next generation. This elitism process must be modified so that a larger set of solutions is retained. More specifically, in order to identify a suitable set of Pareto-optimal solutions, a record of potential solutions must be retained during each generation. To achieve this, the elitism process is extended so that a larger proportion of the current population is kept between each generation. This introduces a new parameter, P_e, known as the 'elitism count', which specifies how many population members are kept between generations.

All the population members are ranked relatively, in terms of Pareto-dominance, according to the 'fast-non-dominated-sort' procedure proposed in [29]. To encourage a good spread across the Pareto-frontier, the resulting Pareto-ranking of the ith population member, R_i, and the mean distance from all the other points, D_i, as well as a scaling parameter, P_r, are used to compute a distance and ranking adjusted cost according to

$$\hat{J}_i = \frac{P_r R_i}{D_i}, \quad 1 \le i \le P_s. \tag{28}$$

[6] Sharing can take place either in the decision space or the fitness space, although in some cases decision space sharing is preferable [42].

5.2. Pareto repeated weighted boosting search algorithm

The proposed Pareto-RWBS algorithm is constructed as follows. Specify the following algorithmic parameters[7]: population size P_s, number of generations in the repeated search N_g, number of iterations in the WBS N_B, Pareto-ranking scaling P_r, and elitism count P_e.

```
○ Outer Loop: generations for $g = 1 : N_g$
  – Pareto generation initialisation: Initialise the population by setting $\mathbf{u}_i^{(g)} = \mathbf{u}_{\text{best},i}^{(g-1)}$ for $1 \le i \le P_e$, and randomly generating the rest of the population members $\mathbf{u}_i^{(g)}$ for $P_e + 1 \le i \le P_s$, where $\{\mathbf{u}_{\text{best},i}^{(g-1)}\}_{i=1}^{P_e}$ denotes the set of the 'best' $P_e$ solutions found in the previous generation. If $g = 1$, $\mathbf{u}_i^{(g)}$, $1 \le i \le P_e$, are also randomly chosen.
  – Weighted boosting search initialisation: Assign the initial weights $\delta_i(0) = \frac{1}{P_s}$, $1 \le i \le P_s$, for the population. Calculate the cost function values for each point of the population set $\{\mathbf{u}_i^{(g)}\}_{i=1}^{P_s}$ and for each objective function
      $$J_{i,o} = J_o(\mathbf{u}_i^{(g)}), \quad 1 \le o \le N, \tag{29}$$
    where $1 \le i \le P_s$ and $N$ is the number of the objective functions.
  – Inner Loop: weighted boosting search for $t = 1 : N_B$
    • Step 1: Pareto Boosting
      (1) Perform Pareto Ranking, Distance Measure and Cost Mapping for the current population $\{\mathbf{u}_i^{(g)}, J_{i,o}, 1 \le o \le N\}_{i=1}^{P_s}$. Specifically,
          (a) Calculate the Pareto-ranking for each member of the population:
              $$\{R_i\}_{i=1}^{P_s} = \textbf{FastNonDominatedSort}\{J_{i,o}, 1 \le i \le P_s, 1 \le o \le N\}, \tag{30}$$
              using the method proposed in [29].
          (b) For each member of the population, compute the mean Euclidean distance to all the other points in the decision variable space:
              $$D_i = \frac{1}{P_s} \sum_{j \ne i} \|\mathbf{u}_i^{(g)} - \mathbf{u}_j^{(g)}\|, \tag{31}$$
              for $1 \le i \le P_s$.
          (c) Compute the distance and ranking adjusted costs:
              $$\hat{J}_i = \frac{P_r R_i}{D_i}, \quad 1 \le i \le P_s. \tag{32}$$
      (2) Find
          $i_{\text{best}} = arg \min_{1 \le i \le P_s} \hat{J}_i$,
          and denote $\mathbf{u}_{\text{best}}^{(g)} = \mathbf{u}_{i_{\text{best}}}^{(g)}$.
      (3) Normalise the distance and ranking adjusted cost values:
          $\bar{J}_i = \frac{\hat{J}_i}{\sum_{j=1}^{P_s} \hat{J}_j}, \quad 1 \le i \le P_s.$
      (4) Compute a weighting factor $\beta(t)$ according to
          $$\eta(t) = \sum_{i=1}^{P_s} \delta_i(t-1)\bar{J}_i, \quad \beta(t) = \frac{\eta(t)}{1 - \eta(t)}.$$
```

[7] As the aim is to find a set of solutions that are well spread across the Pareto-frontier, the accuracy for terminating the WBS, ξ_B, in the original RWBS becomes irrelevant, and the inner loop is simply terminated after the N_B boosts.

(5) Update the weights for $1 \le i \le P_s$

$$\delta_i(t) = \begin{cases} \delta_i(t-1)\beta(t)^{\bar{J}_i} & \text{for } \beta(t) \le 1, \\ \delta_i(t-1)\beta(t)^{1-\bar{J}_i} & \text{for } \beta(t) \le 1, \end{cases}$$

and normalise them:

$$\delta_i(t) = \frac{\delta_i(t)}{\sum_{j=1}^{P_s} \delta_j(t)}, \quad 1 \le i \le P_s.$$

- *Step 2*: Pareto parameter update
  (1) Construct the $(P_s + 1)$th point using the formula
  $$\mathbf{u}_{P_s+1} = \sum_{i=1}^{P_s} \delta_i(t)\mathbf{u}_i^{(g)}.$$
  (2) Construct the $(P_s + 2)$th point using the formula
  $$\mathbf{u}_{P_s+2} = \mathbf{u}_{\text{best}}^{(g)} + (\mathbf{u}_{\text{best}}^{(g)} - \mathbf{u}_{P_s+1}).$$
  (3) For these two new points, compute their objective function values: $J_{i,o}$, $1 \le o \le N$ and $i = P_s + 1, P_s + 2$.
  (4) For $i = 1:2$
    (i) Perform **Pareto Ranking, Distance Measure and Cost Mapping** for the enlarged population $\{\mathbf{u}_j^{(g)}, J_{j,o}, 1 \le o \le N\}_{j=1}^{P_s+2-(i-1)}$ to yields $\{\hat{J}_j\}_{j=1}^{P_s+2-(i-1)}$
    (ii) Find $j_{\text{worst}} = arg \max_{1 \le j \le P_s+2-(i-1)} \hat{J}_j$, and remove $\mathbf{u}_{j_{\text{worst}}}^{(g)}$ from the population
  This removes the two 'worst' points, and keeps the population size to P_s.

– **End of Inner loop** Choose the P_e best solutions, $\{\mathbf{u}_{\text{best},i}^{(g)}\}_{i=1}^{P_e}$:
  For $i = 1 : P_e$
  (i) Perform **Pareto Ranking, Distance Measure and Cost Mapping** for the population $\{\mathbf{u}_j^{(g)}, J_{j,o}, 1 \le o \le N\}_{j=1}^{P_s-(i-1)}$ to yields $\{\hat{J}_j\}_{j=1}^{P_s-(i-1)}$
  (ii) Find $j_{\text{best}} = arg \min_{1 \le j \le P_s-(i-1)} \hat{J}_j$, set $\mathbf{u}_{\text{best},i}^{(g)} = \mathbf{u}_{j_{\text{best}}}^{(g)}$, and remove $\mathbf{u}_{j_{\text{best}}}^{(g)}$ from the population

○ **End of outer loop** This yields the solution set $\{\mathbf{u}_i^{(N_g)}\}_{i=1}^{P_s}$

5.3. *Benchmark convergence experiments*

In order to evaluate the proposed Pareto-RWBS algorithm, its performance is compared with the NSGA-II algorithm on a number of test problems. The NSGA-II algorithm is a well-known state-of-the-art multiple-objective optimisation algorithm which has been shown to produce very good results on a wide range of problems [29]. The NGSA-II implementation used here utilises real-coding, binary tournament selection, binary crossover with probability 0.9, polynomial mutation with probability $1/n$, where n is the dimension of the decision variable space, and non-dominated sorting in conjunction with a crowding operator. In the experiments, the results were presented as individual simulations rather than multiple Monte-Carlo (MC) simulations for the Pareto-RWBS and NSGA-II.

5.3.1. *SCH function*

The first test function experimented is the one-dimensional 'SCH' function taken from [29], which exhibits a simple convex Pareto-frontier:

$$\begin{cases} J_1(u) = u^2, \\ J_2(u) = (u-2)^2. \end{cases} \tag{33}$$

The decision variable u in this case lies in the interval $[-1, 1]$. The following settings were used for the Pareto-RWBS: $P_s = 25$, $N_B = 10$, $N_g = 100$, $P_e/P_s = 0.8$, and $P_r = 10$. These algorithmic parameters were found to produce the best results based on trial and error. The population size and the number of generations for the NSGA-II were 30 and 50, respectively. These NSGA-II settings were also tuned using trial and error to provide the best performance.

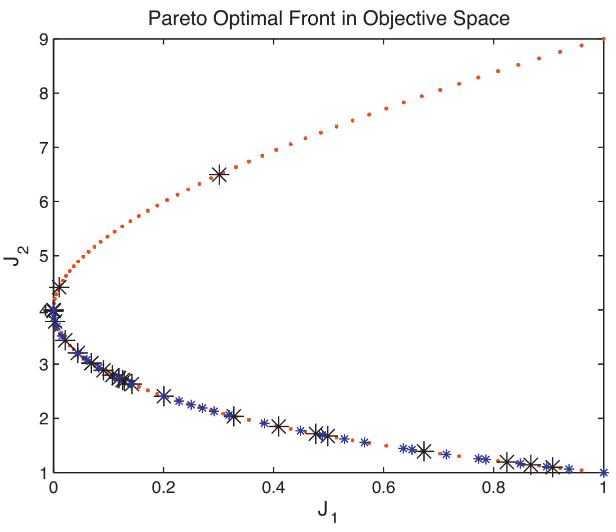


Fig. 15. Objective space performance comparison of the NSGA-II and Pareto-RWBS on the convex test problem of SCH function (33), where red dot markers indicate the feasible solutions generated by MC simulation, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and black larger asterisk markers are the Pareto-RWBS candidate solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

The results obtained for this test function are illustrated in Fig. 15, which shows the resulting objective space solutions. In this figure and all the other objective-space based figures in this section, red dot markers indicate the feasible solutions generated by multiple MC simulations based on random sampling in the decision space which help to visually locate the Pareto-frontier, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and black larger asterisk markers are the candidate solutions generated by the Pareto-RWBS. It is noted immediately from Fig. 15 that the Pareto-RWBS algorithm is capable of finding solutions across the Pareto-frontier. However, the distribution of the solutions found by the Pareto-RWBS is inferior to that of the NSGA-II results, as it is less uniform across the Pareto-frontier.

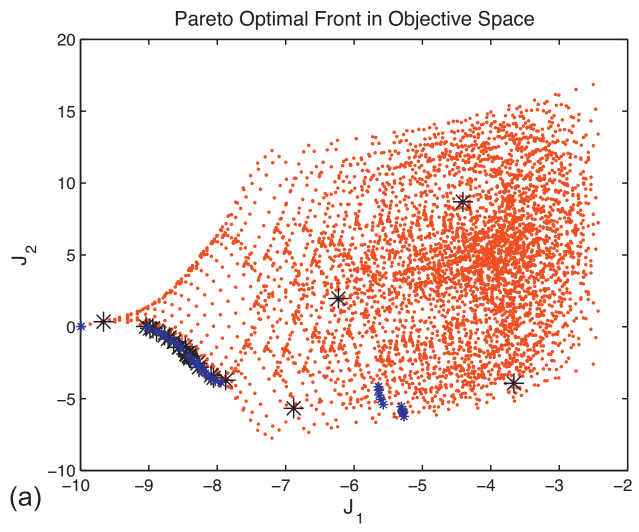
5.3.2. *KUR function*

This test function, the two-dimensional 'KUR' function, is again taken from [29], and is an example where the Pareto-frontier is non-convex:

$$\begin{cases} J_1(\mathbf{u}) = \sum_{i=1}^{n-1} -10\exp\left(-0.2\sqrt{u_i^2 + u_{i+1}^2}\right), \\ J_2(\mathbf{u}) = \sum_{i=1}^{n} \left(|u_i|^{0.8} + 5\sin(u_i^3)\right). \end{cases} \tag{34}$$

In this case, $n = 2$, $\mathbf{u} = [u_1\ u_2]^{\mathrm{T}}$, and the both decision variables lie in the interval $[-5, 5]$. The following settings were used for the Pareto-RWBS: $P_s = 25$, $N_B = 10$, $N_g = 100$, $P_e/P_s = 0.8$ and $P_r = 10$. The population size and the number of generations for the NSGA-II were 30 and 50, respectively. As in the first test problem, these parameters were chosen through trial and error. The results for this test function are illustrated in Figs. 16 and 17.

Similar to the first test problem of SCH function (33), both the NSGA-II and Pareto-RWBS algorithms focus on the same convex region of the Pareto-frontier, as can be seen from Fig. 16. A close-up of the objective space in the region where the majority of the solutions are located is illustrated in Fig. 16(b), which reveals that



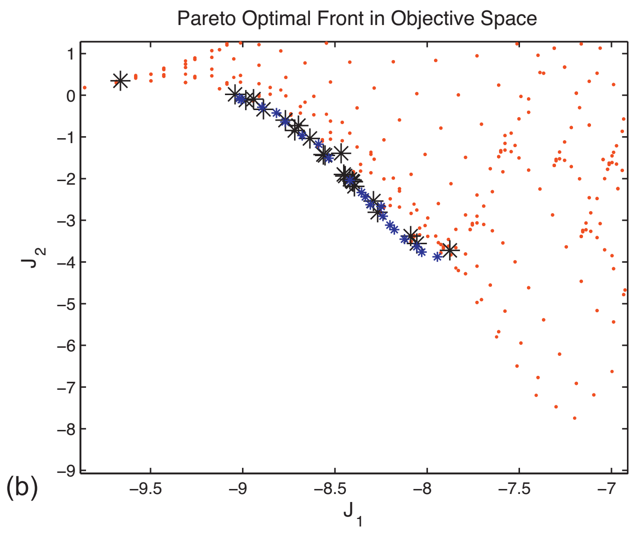


Fig. 16. Comparison of full objective space performance (a) and close-up objective space performance (b) for the NSGA-II and Pareto-RWBS on the non-convex test problem of KUR function (34), where red dot markers indicate the feasible solutions generated by MC simulation, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and black larger asterisk markers are the Pareto-RWBS candidate solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

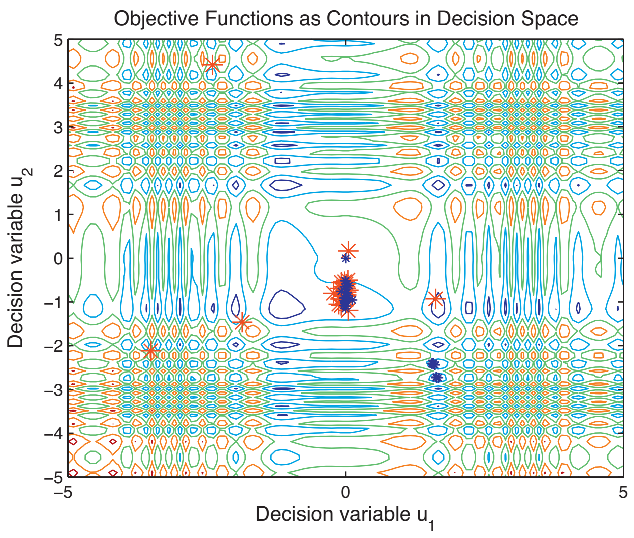


Fig. 17. Decision variable space comparison of the NSGA-II and Pareto-RWBS on the non-convex test problem of KUR function (34), where the overlaid contours represent the objective functions, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and red larger asterisk markers are the Pareto-RWBS candidate solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

the Pareto-RWBS algorithm approaches the Pareto-frontier successfully and the solutions are distributed across a similar region as the NSGA-II candidate solutions. The distribution of the Pareto-RWBS solutions across this region, however, is less optimal (i.e. less uniform) in comparison with that of the NSGA-II results, as was also observed with the first test problem.

The decision variable space results, plotted in Fig. 17, and all the subsequent decision variable space based figures in this section should be interpreted as follows: the overlaid contours represent the objective functions, and blue smaller asterisk markers indicate the NSGA-II candidate solutions, while red larger asterisk markers indicate the Pareto-RWBS candidate solutions. From Fig. 17, it can be observed that the Pareto-optimal solutions lie in a very small region of the decision variable space and the Pareto-RWBS algorithm has identified a very similar region to that of the NSGA-II. However, the Pareto-RWBS solutions are slightly more spread out in the decision space in comparison with the NSGA-II solutions, indicating that there are scopes for further improvements in the Pareto-ranking and cost adjustment process of the Pareto-RWBS.

5.3.3. Multi-modal function

The performance of the two algorithms in the case where the Pareto-frontier is multi-modal is examined using the following two-dimensional test function adopted from [42]:

$$\begin{cases} J_1(\mathbf{u}) = u_1, \\ \quad g(u_2) = 2.0 - \exp\left(-\left(\dfrac{u_2-0.2}{0.004}\right)^2\right) \\ \qquad\qquad -0.8\exp\left(-\left(\dfrac{u_2-0.6}{0.4}\right)^2\right), \\ J_2(\mathbf{u}) = \dfrac{g(u_2)}{u_1}, \end{cases} \tag{35}$$

where $\mathbf{u} = [u_1\ u_2]^T$, $u_1 \in [0.1, 1]$ and $u_2 \in [0, 1]$. Again, the following algorithmic parameters were used for the Pareto-RWBS: $P_s = 25$, $N_B = 10$, $N_g = 100$, $P_e/P_s = 0.8$ and $P_r = 10$, while the population size and the number of generations of the NSGA-II were set to 30 and 50, respectively. The results obtained for this test function are illustrated in Figs. 18 and 19.

This optimisation problem has multiple modes, an attribute which is known to cause difficulties for many multiple-objective optimisation methods. The Pareto-RWBS algorithm demonstrates the ability to identify a range of modes and, in some regions of the Pareto frontier, outperforms the NSGA-II algorithm, as can be seen in Fig. 18. Once again, a reasonable area of the Pareto-frontier is identified by the Pareto-RWBS, but the distribution of the solutions is less uniform than that of the NSGA-II results. The relative positions of the candidate solutions in the decision variable space, as depicted in Fig. 19, are not as informative in this case, and it is difficult to infer insight into the operation of the Pareto-RWBS or NSGA-II from them. However, armed with a priori knowledge regarding the true Pareto-optimal region of the decision space, it may be possible to gain a deeper understanding, and this is an interesting area for future research.

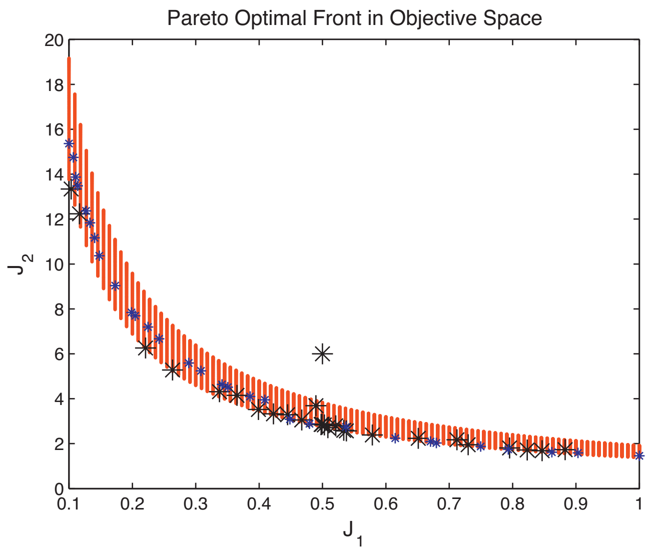


Fig. 18. Objective space performance comparison of the NSGA-II and Pareto-RWBS on the multi-modal test problem (35), where red dot markers indicate the feasible solutions generated by MC simulation, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and black larger asterisk markers are the Pareto-RWBS candidate solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

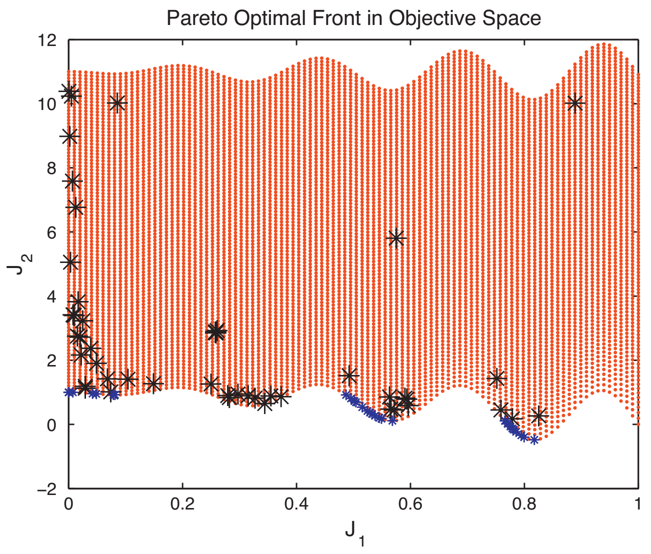


Fig. 20. Objective space performance comparison of the NSGA-II and Pareto-RWBS on the discontinuous test problem (36), where red dot markers indicate the feasible solutions generated by MC simulation, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and black larger asterisk markers are the Pareto-RWBS candidate solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

5.3.4. *Discontinuous function*

The following two-dimensional test function is an example where the Pareto-frontier is discontinuous [42]:

$$\begin{cases} J_1(\mathbf{u}) = u_1, \\ \quad g(u_2) = 1 + 10u_2, \\ J_2(\mathbf{u}) = g(u_2)\left(1 - \left(\dfrac{J_1(\mathbf{u})}{g(u_2)}\right)^{\alpha} - \dfrac{J_1(\mathbf{u})}{g(u_2)}\sin(2\pi q J_1(\mathbf{u}))\right), \end{cases} \tag{36}$$

where, in this case, $\alpha = 2$, $q = 4$, $\mathbf{u} = [u_1\ u_2]^{\mathrm{T}}$, and the both decision variables lie in the interval [0, 1]. The following settings were found empirically to provide the best results for the Pareto-RWBS: $P_s = 50$, $N_B = 20$, $N_g = 100$, $P_e/P_s = 0.8$ and $P_r = 10$. For the Pareto-RWBS algorithm, this particular problem required a larger population size and a larger number of boosts per generation, most likely due to the challenging nature of the problem. This two-objective optimisation problem has a discontinuous Pareto-frontier, an attribute which is known to challenge multiple-objective optimisation techniques. The NSGA-II was applied using the same settings as in the previous problems. The results obtained for this test function are illustrated in Figs. 20 and 21.

It is observed from Fig. 20 that the Pareto-RWBS converges towards four of the primary Pareto-optimal regions, while the NSGA-II algorithm only identifies three of the regions in this

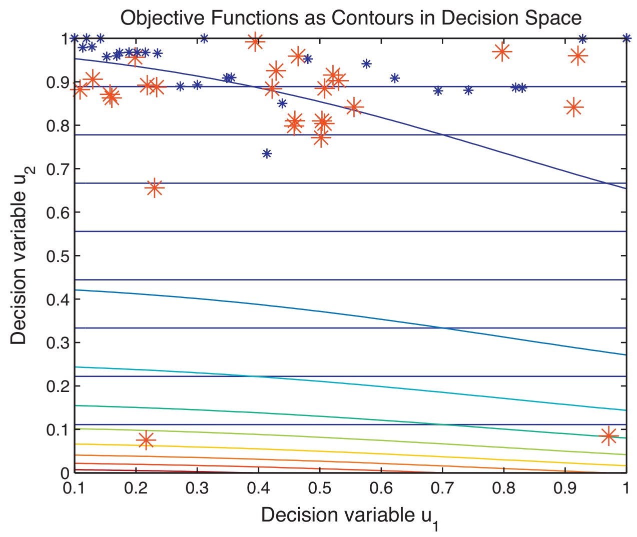


Fig. 19. Decision variable space comparison of the NSGA-II and Pareto-RWBS on the multi-modal test problem (35), where the overlaid contours represent the objective functions, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and red larger asterisk markers are the Pareto-RWBS candidate solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

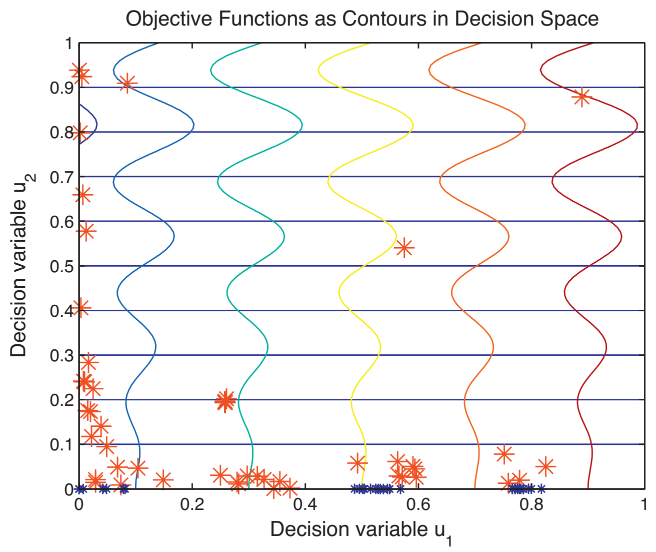


Fig. 21. Decision variable space comparison of the NSGA-II and Pareto-RWBS on the discontinuous test problem (36), where the overlaid contours represent the objective functions, blue smaller asterisk markers indicate the candidate solutions generated by the NSGA-II, and red larger asterisk markers are the Pareto-RWBS candidate solutions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

particular simulation. The performance of the NSGA-II algorithm within the three regions located by the algorithm is, however, superior to that of the Pareto-RWBS for these three regions, in terms of solution distribution, similar to the previous test cases. Fig. 21 also offers some intuition regarding the performance of the two algorithms. In this case, the Pareto-RWBS is observed to identify a larger area of the Pareto-frontier, in the form of four modes compared with the three modes identified by the NSGA-II, but the solutions of the Pareto-RWBS are located further from the frontier than the NSGA-II solutions. This also justifies the assertion that the Pareto-RWBS exhibits promising general performance characteristics, but the Pareto-ranking and cost mapping aspects of the algorithm could be improved.

In summary, the Pareto-RWBS algorithm proposed in this paper has demonstrated clear potential as a flexible, high-performance multiple-objective optimisation technique. The algorithm has been shown to converge reliably towards the Pareto-frontier in a range of test problems with various challenging attributes. The algorithm is observed to be capable of identifying a large area of the Pareto-frontier in each case, comparable with the NSGA-II algorithm which is a well-known state-of-the-art multiple-objective GA. In particular, for the test case of discontinuous Pareto-frontier, the Pareto-RWBS provides a superior performance, in terms of locating more discontinuous regions of the Pareto-frontier. There are scopes, however, to further improve the algorithm, both in terms of the distribution of its solutions along the Pareto-frontier, and the accuracy of the solutions in terms of their distances to the Pareto-frontier. The convergence accuracy hinges on the Pareto-ranking process used in this particular Pareto-RWBS algorithm, and the impact it has on the performance of the convex combination operator. A method for improving the performance of the Pareto-RWBS in this regard is outlined in the following.

5.4. *Selective combination*

As the RWBS algorithm generates new members through the convex operator in Eq. (5), the standard approaches to Pareto-ranking procedure, such as the one used in Section 5.2, may be sub-optimal. For instance, ranking methods which simply assign all non-dominated candidates a similar rank will usually work reasonably well with a GA, as the combination operator will simply select a number of members of the population to combine together and the rank value is not used directly in the local search operator, i.e. during crossover. The RWBS, however, *weights* the all candidates according to their rankings and, therefore, all the members of the same front will receive equal weighting in the combination (ignoring any sharing or other distribution related mechanisms). Thus, the efficiency of the convex operator may be reduced, which suggests that a larger number of boosts have to be used per generation. Additionally, the RWBS only generates a single new member (two if the one generated by the reflection operator is also counted) in each inner loop boosting stage, unlike a GA which can create several new members at the local search stage. With orders of population size required for many multiple-objective optimisation problems, a large number of boosts may be needed. An alternative approach to that outlined in Section 5.2, therefore, is to use a selection operator to select which members are used in a *set* of convex combinations at each stage (similar to the way a GA proceeds). This would create a number of new individuals in each generation as well as reducing the number of solutions in each boosting stage, thus reducing the required number of boosts. It is hypothesised that this approach would help to improve the algorithm's performance in terms of the accuracy to which the Pareto-frontier is located. Investigation of this alternative Pareto-RWBS algorithm is the subject of future work.

6. Conclusions and discussion for future work

This paper has proposed a number of extensions to the repeated weighted boosting search optimisation algorithm in order to facilitate its use in a wider class of optimisation problems. The extensions permit the use of the RWBS algorithm in the problems which have discrete or mixed search spaces, decision variable constraints and multiple objectives. It has been shown that a simple addition to the convex operator renders the algorithm capable of successful optimisation in discrete search spaces. Initial analysis with constraint handling techniques suggest that the death penalty method may be unsuitable for the application in this context, but that repair operator may provide a suitable method. Multiple-objective problems have been dealt with by using a Pareto-ranking scheme combined with a sharing process. The resulting Pareto-RWBS algorithm performs on par with the NSGA-II algorithm, in terms of identifying the Pareto-frontier. All the extensions proposed retain the attractive properties of the original RWBS algorithm proposed in [9], namely, simplicity, ease of implementation and small number of tuning parameters, while facilitating its use to a wider range of optimisation problems. This makes the RWBS algorithm a powerful tool for solving various complex optimisation problems, arising from diverse applications, with minimal implementation effort.

There are many avenues of future work to be investigated, regarding the further development of the RWBS algorithm. In particular, a rigorous theoretical analysis of the convergence properties, along with the interactions between the tuning parameters would prove very useful. It may also be possible to improve the performance of the algorithm in discrete search spaces by utilising other notions of discrete convex hulls, such as those detailed in [32] in order to expand the search space that is captured by the convex operator. Other future work includes detailed quantitative convergence analysis of the Pareto-RWBS algorithm, and modifications to improve the spread of the identified Pareto-solutions. Recently, bio-inspired computational intelligence methods, such as ant colony optimisation [45,46] and particle swarm optimisation [47,48] as well as the differential evolution algorithm [49,50], have attracted wide interests from all walks of science and engineering. A comprehensive performance comparison between the RWBS algorithm and these bio-inspired optimisation techniques in benchmark application problems would be an interesting subject of future work. Initial results in the context of wireless communication application are reported in [51].

References

[1] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[2] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[3] L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[4] K.F. Man, K.S. Tang, S. Kwong, Genetic Algorithms: Concepts and Design, Springer-Verlag, London, 1998.

[5] A. Corana, M. Marchesi, C. Martini, S. Ridella, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm, ACM Transactions on Mathematical Software 13 (3) (1987) 262–280.

[6] L. Ingber, B. Rosen, Genetic algorithms and very fast simulated reannealing: a comparison, Mathematical and Computer Modelling 16 (11) (1992) 87–100.

[7] L. Ingber, Simulated annealing: practice versus theory, Mathematical and Computer Modelling 18 (11) (1993) 29–57.

[8] S. Chen, B.L. Luk, Adaptive simulated annealing for optimization in signal processing applications, Signal Processing 79 (1) (1999) 117–128.

[9] S. Chen, X.X. Wang, C.J. Harris, Experiments with repeating weighted boosting search for optimization in signal processing applications, IEEE Transactions on Systems, Man and Cybernetics, Part B 35 (4) (2005) 682–693.

[10] F. Schoen, Stochastic techniques for global optimization: a survey of recent advances, Journal of Global Optimization 1 (3) (1991) 207–228.

[11] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.

[12] S. Chen, X.X. Wang, D.J. Brown, Orthogonal least squares regression with tunable kernels, Electronics Letters 41 (8) (2005) 484–486.

[13] X.P. Zong, Y. Xu, L. Hao, X.L. Huai, Camera calibration based on the RBF neural network with tunable nodes for visual serving in robotics, in: Proc. 2006 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, Beijing, China, October 9–15, 2006, 2006, pp. 5708–5712.

[14] S. Chen, X.X. Wang, X. Hong, C.J. Harris, Kernel classifier construction using orthogonal forward selection and boosting with Fisher ratio class separability measure, IEEE Transactions on Neural Networks 17 (6) (2006) 1652–1656.

[15] X.X. Wang, S. Chen, D. Lowe, C.J. Harris, Sparse support vector regression based on orthogonal forward selection for the generalised kernel model, Neurocomputing 70 (1–3) (2006) 462–474.

[16] J.X. Yu, H.Q. Cao, Y.B. He, A new tree structure code for equivalent circuit and evolutionary estimation of parameters, Chemometrics and Intelligent Laboratory Systems 85 (1) (2007) 27–39.

[17] M. Zhang, J.G. Zhou, L.H. Fu, T.T. He, Hybrid wavelet model construction using orthogonal forward selection with boosting search, in: Proc. 4th Int. Conf. Fuzzy Systems and Knowledge Discovery, Haikou, China, August 24–27, 2007, 2007, pp. 341–345.

[18] S. Chen, X.X. Wang, C.J. Harris, NARX-based nonlinear system identification using orthogonal least squares basis hunting, IEEE Transactions on Control Systems Technology 16 (1) (2008) 78–84.

[19] M. Abuthinien, S. Chen, L. Hanzo, Semi-blind joint maximum likelihood channel estimation and data detection for MIMO systems, IEEE Signal Processing Letters 15 (2008) 202–205.

[20] M. Zhang, L. Fu, G. Wang, T. He, Improved orthogonal least-squares regression with tunable kernels using a tree structure search algorithm, IEEE Signal Processing Letters 15 (2008) 653–656.

[21] S.A. Hoseini, R.P. Torghabeh, M. Kaveh, H. Khaloozadeh, Designing stabilizing regulators for chaotic systems using repeated weighted boosting search method, in: Proc. 2nd Int. Conf. Computer, Control and Communication, Karachi, February 17–18, 2009, 2009, pp. 1–6.

[22] M. Zhang, J. Zhou, L. Fu, T. He, Hybrid wavelet model construction using orthogonal forward selection with boosting search, International Journal of Business Intelligence and Data Mining 3 (4) (2009) 437–450.

[23] S. Chen, X. Hong, B.L. Luk, C.J. Harris, Construction of tunable radial basis function networks using orthogonal forward selection, IEEE Transactions on Systems, Man, and Cybernetics, Part B 39 (2) (2009) 457–466.

[24] J. Jiang, X.-P. Zhang, A new player-enabled rapid video navigation method using temporal quantization and repeated weighted boosting search, in: Proc. 2010 IEEE Computer Society Conf. Computer Vision and Pattern Recognition Workshops (CVPRW), San Francisco, CA, June 13–18, 2010, 2010, pp. 64–71.

[25] L.H. Fu, M. Zhang, H.W. Li, Sparse RBF networks with multi-kernels, Neural Processing Letters 32 (3) (2010) 235–247.

[26] J. Zhang, S. Chen, X. Mu, L. Hanzo, Joint channel estimation and multi-user detection for SDMA/OFDM based on dual repeated weighted boosting search, IEEE Transactions on Vehicular Technology 60 (7) (2011) 3265–3275.

[27] S. Wang, S. Chen, A.H. Wang, J.P. An, L. Hanzo, Joint timing and channel estimation for bandlimited long-code-based MC-DS-CDMA: a low-complexity near-optimal algorithm and the CRLB, under review.

[28] S.F. Page, A.N. Dolia, C.J. Harris, N.M. White, Multiple objective optimization for active sensor management, in: Proc. SPIE: Multisensor, Multisource Information Fusion – Architectures, Algorithms, and Application, Orlando, FL, March 30, 2005, vol. 5813, no. 269, 2005, pp. 269–280.

[29] K. Deb, A. Pratap, S. Agarwa, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NGSA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[30] The MathWorks Inc., MATLAB Optimization Toolbox User Guide, 3rd edition, September 2006.

[31] K.A. De Jong, An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Dissertation, University of Michigan, 1975.

[32] A. Roy, J.G. Stell, Convexity in discrete space, in: W. Kuhn, M.F. Worboys, S. Timpf (Eds.), Proc. Conf. Spatial Information Theory (Ittingen, Switzerland), September 24–28, 2003, Lecture Notes in Computer Science, vol. 2825, Springer-Verlag, 2003, pp. 268–285.

[33] K. Deb, A. Pratap, T. Meyarivan, Constrained test problems for multi-objective evolutionary optimization, in: E. Zitzler, K. Deb, L. Thiele, C.A.C. Coello, D. Corne (Eds.), Proc. 1st Int. Conf. Evolutionary Multi-Criterion Optimization (Zurich, Switzerland), March 7–9, 2001, Lecture Notes in Computer Science, vol. 1993, Springer-Verlag, 2001, pp. 284–298.

[34] B.W. Wah, Y.X. Chen, Constrained genetic algorithms and their applications in nonlinear constrained optimization, in: Proc. 12th IEEE Int. Conf. Tools with Artificial Intelligence, Vancouver, Canada, November 13–15, 2000, 2000, pp. 286–293.

[35] C.M. Fonseca, P.J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms. Part I. A unified formulation, IEEE Transactions on Systems, Man, and Cybernetics, Part A 28 (1) (1998) 26–37.

[36] C.M. Fonseca, P.J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evolutionary Computation 3 (1) (1995) 1–16.

[37] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.

[38] M. Laumanns, L. Thiele, E. Zitzler, K. Deb, Archiving with guaranteed convergence and diversity in multi-objective optimization, in: Proc. Genetic and Evolutionary Computation Conf., New York, July 9–13, 2002, 2002, pp. 439–447.

[39] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.

[40] C.A. Coello Coello, Evolutionary multi-objective optimization: a historical view of the field, IEEE Computational Intelligence Magazine 1 (1) (2006) 28–36.

[41] C.M. Fonseca, Multiobjective Genetic Algorithms with Application to Control Engineering Problems, Ph.D. Dissertation, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, September 1995.

[42] K. Deb, Multi-objective genetic algorithms – problem difficulties and construction of test problems, Evolutionary Computation 7 (3) (1999) 205–230.

[43] C.M. Fonseca, P.J. Fleming, Multiobjective genetic algorithms made easy: selection sharing and mating restriction, in: Proc. 1st Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications, Sheffield, UK, September 12–14, 1995, 1995, pp. 45–52.

[44] E. Parzen, On the estimation of a probability density function and mode, Annals of Mathematical Statistics 33 (3) (1962) 1065–1076.

[45] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B 26 (1) (1996) 29–41.

[46] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004.

[47] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. 1995 IEEE Int. Conf. Neural Networks, Perth, Australia, November 27–December 1, 1995, vol. 4, 1995, pp. 1942–1948.

[48] J. Kennedy, R. Eberhart, Swarm Intelligence, Morgan Kaufmann, 2001.

[49] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer-Verlag, Berlin, 2005.

[50] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.

[51] J. Zhang, S. Chen, X. Mu, L. Hanzo, Stochastic optimization assisted joint channel and multi-user detection for OFDM/SDMA, in: Proc. VTC 2012-Fall (Québec, Canada), September 3–6, 2012, 5 pp.