The Barrier Tree Benchmark: Many Basins and Double Funnels

Tim Blackwell
t.blackwell@gold.ac.uk
Department of Computing, Goldsmiths, University of London
New Cross, London, SE14 6NW, UK

ABSTRACT

The Barrier Tree Benchmark (BTB) is a principled generator of continuous real-valued landscapes: problems of known topography/critical point structure can be systematically designed and deployed in algorithm comparison studies. A previous BTB study focused on a single funnel and a double basin. This work demonstrates algorithm performance on BTB instances with many basins, and on double funnels. A methodology for principled algorithm comparison on families of problems of similar complexity and structure is proposed. It is hoped that the BTB will address a parameter tuning pathology of current problem benchmarks, namely, that common optimisation algorithms require widely different control parameter settings for optimal performance on differing problem classes. This pathology is traced to the irregular and arbitrary composition of standard benchmarks.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

optimisation, algorithm benchmarking, swarm intelligence

ACM Reference Format:
Tim Blackwell. 2023. The Barrier Tree Benchmark: Many Basins and Double Funnels. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583131.3590478

1 INTRODUCTION

Benchmarking, or the assessment of algorithm relative performance by application of a suite of test functions, is an integral part of evolutionary algorithm (EA) research. In the absence of theoretical analysis, the aim of benchmarking is a putative matching of algorithm to problem. The outcome is important for real-world application, and as a means of characterising algorithm behaviour [16, 20, 30].

The two main benchmarks available to the EA community are those provided by the Congress on Evolutionary Computation (CEC) [14] and the Comparing Continuous Optimiser suite (COCO) [9]. Although widely used, these benchmarks suffer numerous pathologies.

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07.
https://doi.org/10.1145/3583131.3590478

First, they are comprised of a hetereogeneous collection of problems. Since algorithms typically need careful control parameter tuning in order to deliver the best result on a particular problem, relative comparison across the benchmark based on a single control parameter setting, even if optimised on a smaller training set, is of dubious value. Even though the test functions in the CEC and COCO benchmarks can be grouped (unimodal, multi-modal, composition etc.), the classifications are broad and there is no guarantee that an algorithm will deliver good performance on a novel problem of a broad class given that it has some proven success on a representative of that class from the benchmark.

The second issue with existing benchmarks is that functions are specified by combinations of mathematical functions. This specification, although helpful for evaluating test positions, produces landscapes with only partial topographical information. For example, number of critical points and the number, depth and widths of its constituent basins are in not in general known, if not actually disputed [22]. This partial knowledge hinders relating algorithm to function property and prohibits the necessary narrower classification of function types.

The Barrier Tree Benchmark (BTB) [27] was initiated in order to address the above deficiencies. The underlying design principle is the assembly of a landscape from basis functions (funnel and basin). The construction procedure, by default, generates landscapes whose properties are known rather than calculated, inferred or guessed. The pay-off is a principled construction of test functions and an opportunity to relate algorithm performance to function topography.

This paper continues BTB development: a correction is made to the previous funnel function definition; bi-funnel and highly multi-modal Barrier Tree Functions (BTFs) are constructed; the performance of four established algorithms are compared on these BTFs; a population generalisation of a downhill walker algorithm is proposed and trialled.

Section 2 provides further background on benchmarks, function generators and particle swarm optimisation (PSO) performance in various scenarios. The paper continues with an overview of the Barrier Tree Benchmark (BTB) and an explanation of Barrier Tree Function (BTF) construction (Secs 3 and 4). A correction to the funnel function proposed in [27] is made in Sec. 4.

A high-level account of the BTF generator used for the experiments reported in this paper follows (Sec. 5). These experiments, which were not exhaustive, were designed to illustrate the deployment of the BTB in algorithm comparison. Methodology and results are reported in Secs 6 and 7 wherein a new swarm algorithm, the Swarm Walker (SW) is proposed, tested and (favourably) compared to PSO and differential evolution (DE). The paper closes with a proposal for the use of the BTB in algorithm control parameter tuning, Sec. 8, and overall conclusions, Sec. 9.

2 BACKGROUND

A number of benchmarking problem suites have been developed for the ongoing series of CEC conference optimisation challenges. Garcia provides a ten-year report up to 2017 [8]. The challenges have continued since that report; the 2023 CEC presentation will run nine competitions[1].

The original 2005 CEC continuous valued benchmark comprised twenty-five single-objective test functions: five unimodal functions, seven multimodal functions, two 'complex' multimodal functions, and eleven 'hybrid' functions, which were the weighted sums of ten basic test problems [25]. CEC2005 became a standard benchmark for several years until the 2013 competition test suite improved and supplemented the original function definitions [14] and became a new reference point. CEC2013, a suite of twenty-eight functions, includes five unimodal functions, fifteen basic multimodal functions, and eight composition functions.

A parallel suite of benchmark problems emerged in 2009 for the Black-box Optimization Benchmarking (BBOB) workshop series [10]. The sophisticated COCO platform, developed for the BBOB workshops, allows for considerable automation in running and testing optimisation algorithms. The current COCO benchmark comprises forty-eight single objective test functions, split evenly between noisy and noiseless.

Generators of random problems have been proposed. A polynomial test problem generator (NGLI) with known basins and saddle points allows for only partial control on modality since there is a strong coupling between dimensionality and modality of [17]. A max set of Gaussians (MSG) controlling the number of basins [7] and a general framework for generating Gaussian test functions with controllable properties have also been proposed [15]. A continuing interest in Gaussian basis functions is apparent in a modified Gaussian fitness landscape generator [13], and a non-separable test problem generator (N-Peaks) with randomly distributed basins [28].

The danger of benchmarking on these diverse problem sets lies in overfitting algorithm control parameters - particularly those in the whatever novel algorithm is being proposed (the comparison algorithms, which inevitably fare worse, do not enjoy the benefit of such careful tuning). The procedure, which relies on a relatively small number of test functions of wide heterogeneity, limits, generalisation to other problems [18].

The problem lies on the reliance of a function definition in a single expression which encompasses the entire search space. The number and variety of such functions is limited by human ingenuity and it is very hard to find expressions which generate a more homogeneous function set.

The complexity of benchmarks such as CEC and COCO may indeed limit the understanding of the strengths and weaknesses of these algorithms [20]. This paper points to the desirability of a simple generator of test landscapes of entirely known properties in order to understand the behaviour of the optimisation algorithms and their matching preferred function classes.

The BTB benchmark enables arbitrary funnel and basin structures. Particle swarm optimisation (PSO) performance, for example, in multi-funnel landscapes has been investigated by several authors [26] [6], [21] with the outcome that PSO struggles in multi-funnel scenarios. It has been suggested that basin size and configuration is more pertinent to PSO performance than modality and the presence of any funnel [31]. These studies are sporadic and inconclusive. A simple and principled investigation of PSO behaviour on numerous examples of functions of similar property is needed; this prerequisite is fulfilled by the Barrier Tree Benchmark.

Despite numerous studies, to the best of our knowledge, no research has achieved a comprehensive understanding of the relationship between optimiser performance and geometric quantities such as basin size, landscape topography and barrier tree complexity. The BTB is offered as a means of filling that gap.

[1] https://2023.ieee-cec.org/competitions/

3 BARRIER TREE BENCHMARK: OVERVIEW

A barrier tree is a hierarchical representation of a fitness landscape [24]. The tree is drawn vertically against a depth axis and nodes correspond to saddle points and basin optima. Apart from its use in evolutionary biology, barrier trees have been rediscovered as 'disconnectivity graphs' in chemical physics. For example, the highly complex pattern of local energy minima of the Lennard-Jones atom is illustrated in [5].

Although a barrier tree relates to optimisation difficulty, it ignores important topographic information such as basin size. An exceedingly narrow optimum basin will be harder to discover than a broad one, even if the respective trees are identical. The BTB supplies this missing information.

The basic concept is to flesh-out a bare barrier tree with funnel and basin functions which provide the detailed landscape that interpolates between critical points. These functions, which are described in the following section, must be constructed so that basin and saddle regions do not contain any maxima, and so that the function value at the boundaries of touching regions (i.e. basins and funnels) must equate (and in fact be equal to the saddle point value at the touching point). A further desideratum is for the pattern of level sets to follow the shape of region boundaries since otherwise smaller regions might be optimised in preference to larger ones. These requirements are not easily met, and not by a continuous landscape: Barrier Tree Functions (BTFs) are in general discontinuous at region boundaries. This restriction is not a hindrance to evolutionary algorithms, particle swarm optimisers or other non-gradient algorithms.

The construction is not complete with the choice of region function. Centres, dimensions and distributions of the regions must be specified. There are two possibilities: a regular placement, for example at points on a hypercubic lattice, or a random arrangement as in the aggregation of bubbles in a raft. This latter choice leads to instances of random BTFs i.e. to particular aggregations. The instances will typically be generated afresh at each run of an algorithm so that performance is measured across the spectrum of possibilities for a particular random BTF. This procedure is already followed in the random and noisy members of the CEC and COCO benchmarks.

4 BARRIER TREE FUNCTIONS

Consider a continuous search space X that can be divided into continuous, contiguous, non-overlapping regions, comprising basins $\mathcal{B}_i$ and funnels $\mathcal{F}_j$, and isolated saddle points $\overline{x}_k$: $X = \{\mathcal{B}_i\} \cup \{\mathcal{F}_j\} \cup$

$\{\overline{x}_k\}$. An objective function $f(X)$, which need not be continuous, is defined on X; the optimisation problem is the discovery of a member of the optimal set X_{opt} that (globally) minimises f. Frequently, optimisation algorithms can only find approximate solutions, or solutions that satisfy certain criteria.

The distinction between basins and funnels is made with reference to downhill paths i.e. to continuous, directed paths p, parameterised by $t \in [0, 1]$, that start at $p(0)$, end at $p(1)$ and such that $t' > t \implies f(t') \leq f(t)$.

A basin $\mathcal{B}$ comprises a single isolated minimum point x^* and a surrounding continuous set of points $\mathcal{B} \setminus x^*$ such that all downhill paths originating at $x \in \mathcal{B} \setminus x^*$ terminate at x^*. On the other hand, downhill paths commencing on a funnel position will leave the funnel and enter one of two or more regions. Neutral regions (funnels or basins), for explanatory convenience, are not considered.

The underlying idea for the construction of a BTF is to begin with a barrier tree and to proceed by specifying region topography. Nodes of a barrier tree correspond to critical points (local minima and saddles) and the tree is aligned against a vertical axis of depths i.e. of critical function values.

The upper part of Fig. 1 depicts a barrier tree of four minima and three saddle points (horizontal nodes). This tree has two funnels with two basins in each funnel. The whole structure sits inside the entire search space i.e. is surrounded by a larger enclosing funnel. A simple nomenclature for the tree in Fig. 1, deriving from pre-order traversal, is *fbbfbb*. The lower part of the figure shows a possible topography. In this two-dimensional representation, basins are discs and funnel boundaries are circles. The topography (choice of region function) of the interior of a region is arbitrary but must be such that funnels have no critical points and basins have a single minimum and no other critical point. Furthermore, all saddle points within a funnel have the same depth, and since saddle points lie on the surface of the inner sub-regions, all points on the inner surface also have equal depth.

Region formation, the arrangement of regions within regions, whether random or regular, is another arbitrary choice. The BTB generator allows for a random formation of sub-regions (*bubble formation*, in analogy to a raft of bubbles) and three regular formations (hypercube, orthoplex and simplex). These latter formations correspond to the three regular polytopes that exist in all dimensions [3].

A convenient choice of basin function is

$$f_i(x) = m_i|x - x_i| + d_i \tag{1}$$

where the basin is centred on the minimum point x_i and $m_i > 0$ is an arbitrary multiplier chosen so that no interior point has depth greater than a boundary point. The depth, d_i is specified by the underlying barrier tree.

Suppose that all basins are n-balls and all funnels are n-balls minus their inner regions. Then a suitable definition for the funnel function can be based on the distance to the nearest surface of the contained regions. The aim is that funnel topography should consist of iso-surfaces that follow the shape of the inner region boundary at close distances and become more and more spherical at at larger separations. Fig. 2 shows iso-surfaces surrounding two

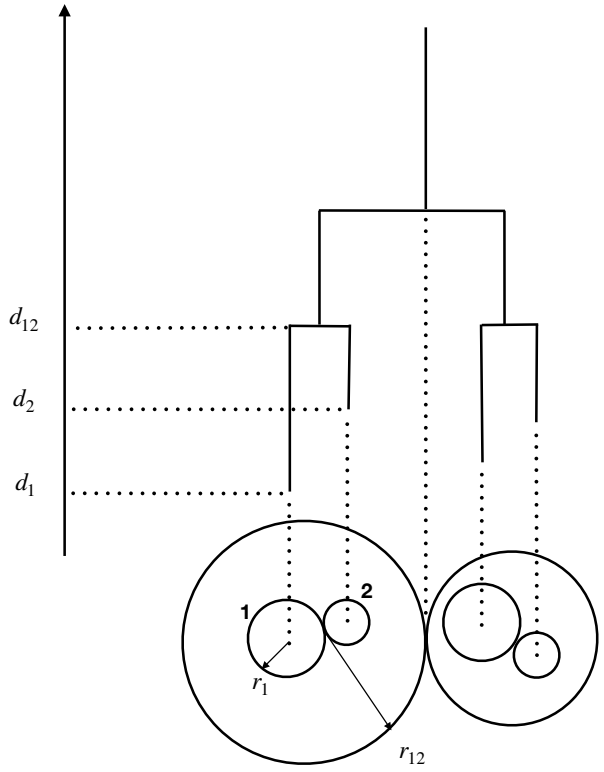

Figure 1: An fbbfbb barrier tree. Funnel and basin geometry are indicated below the tree.

contained regions. This aim is realised by the expression

$$f_j(x) = \min_{\mathcal{R}_i}\{m_j(|x - x_i| - r_i) + d_j\} \tag{2}$$

where $r_{\{i\}}$ are the centres and radiuses of regions $\mathcal{R}_{\{i\}}$ contained within funnel $\mathcal{F}_j$. $|x-x_i|-r_i$ is the nearest separation between inner region $\mathcal{R}_i$ and funnel point x and m_j and d_j are the gradient and depth of the funnel (the saddle value) respectively. The gradient is a suitable multiplier that ensures that no interior point has a value higher than a point on the funnel boundary. This definition replaces an earlier funnel function definition that led to distorted iso-surfaces with the effect that more numerous downhill paths could flow into small regions than larger regions in some circumstances [27]. The advantage of the new funnel function definition is that smaller regions are the destinations of fewer downhill paths so that, intuitively, the probability of a downhill path flowing into inner region $\mathcal{R}_i$ is determined by the surface area of $\mathcal{R}_i$.

A point x is evaluated by determining the containing region and applying either Eq. 1 or Eq. 2, depending on whether the region is a basin or a funnel, or, if x is a saddle point, by direct look-up on the underlying barrier tree.

5 A BTF GENERATOR

The barrier tree, in the BTF generator employed in this paper, is built from a list of depths in pre-order form. For example, the fbbfbb tree of Figure 1 is constructed from the string $L, d_{12}, , d_1, , d_2, d_{34}, , d_3, d_4$

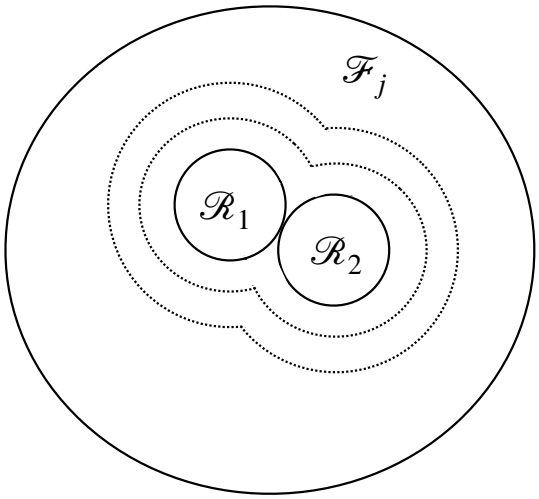

Figure 2: A funnel, $\mathcal{F}_j$ with two inner regions. The dotted lines are iso-surfaces of constant function value.

where the search space, X, is a box, $X = [-\frac{L}{2}, \frac{L}{2}]^n$, d_{12} and d_{34} are the depths of the funnels and $d_{1,2,3,4}$ are the basin depths. Region level is indicated by the number of preceding commas. Level 0 corresponds to X, level 1 to the funnels and level 2 to the basins.

Region radiuses are specified in pre-order and the radiuses and centres of n-ball regions are added. Bubble formations are constructed as follows. Suppose a region has n_{child} inner regions: child regions are added in random order to the parent. The centre of the first randomly chosen child region is positioned at the centre of the parent ball. Further child regions are added by selecting one of the existing child regions at random. A random position on the child boundary is found and the new child region is centred so it touches at this random position. The bubble 'raft' is then offset from the centre of the parent funnel by a random amount so that no child region contains the parent centre, and the entire structure of n-ball children is further rotated in n dimensions by application of a random rotation matrix. This procedure guards against any unintentional central tendency of the optimisation algorithm, and the random order of attachment and random touching points safeguards systematic prejudices which might otherwise occur if, for example, the final region of the depth string is optimal amongst the children and this final region would otherwise always occur at the edge of the bubble raft.

Since the procedure outlined above depends on pseudo-random numbers, the BTF generator produces an instance of the BTF random function specified by the depth and radius strings, and the choice of formation.

A particular barrier tree topology, for example fbbfbb (a function with two funnels of two basins), therefore requires further specification, and a huge range of problem difficulties is available. The optimal basin could be much deeper than the sub-optimal basin, or much narrower, or all basins might have equal depth. These differences are expected to be more or less challenging.

The depths and radiuses of bubble-BTFs with only a few regions can be specified by hand, but these parameters are better specified, for multi-region BTFs, by a formula. For example, $b_1 b_2 \ldots b_{100}$ denotes a multi-modal problem of 100 optima. BTF F5 in table 1 illustrates how depths might vary throughout the bubble structure (all basins having equal radius).

BTFs of arbitrary complexity (or, indeed, formal simplicity) can be built from the above procedure. The BTB replaces function definition by a single formula which defines the entire landscape, by formulas (or short list of constants) which specify region size and depth. The result is that function topography is chosen in advance rather than calculated or guessed from a prior definition. The underlying philosophy is that the salient landscape features for non-gradient optimisers (critical points, basin and funnel sizes and relative positions) are pre-determined. Optimiser performance can be calibrated by function class rather than averaged over a suite of highly heterogeneous objective functions.

6 METHODS

By means of illustration of BTB employment, a small suite of five BTFs was chosen (F1-5 in table 1).

F1 and F2 are two-funnel functions with the minimum number (2) of basins in each. F1, fbbfbb*, comprises two identical funnels; three basins of equal depth and size inhabit these basins and a fourth, optimal basin, has a slightly lower depth (-10.1 compared to -10.0). F2, fbbf*bb* investigates the situation when the optimal basin has a slightly smaller size (radius 1.0 compared to 1.1).

F3-5 comprise around 100 minima in different scenarios. F3 has 99 identical basins and an optimal basin of slightly lower depth; basin depths of F4 are a given by a formula that specifies a sequence of linearly decreasing depths to an optimum followed by a sequence of increasing depths. The random bubble construction will not necessarily place these basins in any particular order, but when averaged over a large number of instances, optimisers might be able to utilise fragments of large scale structure. F5 uses a similar depth formula to F4 but partitions the basins in two funnels. Basins in the optimal funnel have a slightly lower depth than the corresponding basins in the sub-optimal funnel. An optimiser would have both chose the optimal funnel and furthermore optimise a multi-modal problem within that funnel.

Five optimisers were chosen for the purposes of comparison: a simple non-population non-gradient algorithm, two forms of particle swarm optimisation (PSO), differential evolution and a novel optimsiser, Swarm Walker (SW).

The downhill walker algorithm (DHW), Algorithm 1, is a finite-step size implementation of a theoretical downhill path follower. The walker trials a random point on the hypersphere S of radius equal to the current steplength and moves if the function value at this point is not greater than the current position. S is centred on the current walker position, x. The walker has *tries* attempts to move; otherwise the step length is reduced under the assumption that finer detail exploration is required.

Table 1: BTF specification. Ordered pairs specify depth and region radius of funnels (f) and basins (b). For example f = (-1.0, 1.0) denotes a funnel of depth -1.0 and radius 1.0.

F1	fbbfbb*	f = (-1.0, 1.0), b = (-10.0, 0.01), b* = (-10.1, 0.01)
F2	fbbf*bb*	f = (-1.0, 1.1), f* = (-1.0, 1.0), b = (-10.0, 0.01), b* = (-10.1, 0.01)
F3	$b^{99}b^*$	b = (-1.0, 0.1), b* = (-1.01, 0.1)
F4	$b_1 b_2 \ldots b_{101}$	$b_i = (-1.0 + 0.01 \times \lvert i - 51 \rvert, 0.1)$, b* = b_{51}
F5	$fb_1 b_2 \ldots b_{51} fb_1^* b_2^* \ldots b_{51}^*$	f = (-1.0, 1.0), b_i =(-10.0 + 0.01 * |i - 26|, 0.001), b_i^* = (-10.01 + 0.01 * |i - 26|, 0.001)

Algorithm 1 DHW

```
attempts ← 0
initialise x
do
    attempts++
    if attempts > tries then
        step_length *= scale_factor
        attempts ← 0
    end if
    y ← random point on hypersphere S(x, step_length)
    if f(y) ≤ f(x) then
        x ← y
        attempts ← 0
    end if
while not terminated
```

Algorithm 2 SW

```
attempts ← 0
initialise a population of downhill walkers
do
    w ← next walker in swarm
    w.attempts++
    if w.attempts > tries then
        w.step_length *= w.scale_factor
        w.attempts ← 0
    end if
    p ← position of w's best ring neighbour
    δx = NI * (p − x)
    y ← random point on hypersphere S(x + δx, step_length)
    if f(y) ≤ f(x) then
        x ← y
        attempts ← 0
    end if
while not terminated
```

The DHW algorithm tests the topology of the BTF implementation. Since the DHW has no mechanism to skip between funnels[2] it will optimise the first funnel it chances upon so that, for example, in fbbffb*, the probability of finding the optimum at the centre of b^* should approximate 0.25. The DHW also provides a baseline algorithm that the population algorithms should surely beat.

The DHW converges very quickly on a basin [27] but, due to its single individual, it has no diversity. The Swarm Walker (SW, Algorithm 2) is a population generalisation that couples the fast DHW convergence properties with individual communication amongst individuals. Walkers in the swarm simply move towards their best neighbour in a preset communication topology (identical to PSO topologies), produce a trial step, and then retreat or remain, depending on the outcome. The factor $NI \in [0, 1]$ is Algorithm 2 determines how far the updating individual might move towards its best neighbour. SW with its simple and random distribution dynamics has a more transparent individual update rule than either PSO or DE and therefore provides a cleaner test of the influence of a swarm.

Two forms of PSO were trialled. Global particle topology (GPSO) and local, ring topology (LPSO). These are popular forms in applications and comparison studies [1].

The update rule for L/GPSO is [12, 19, 23]

$$\begin{aligned} v_i(t+1) &= w v_i(t) + cu_1 \circ (n_i(t+1) - x_i(t)) \\ &\quad + cu_2 \circ (p_i(t+1) - x_i(t)) \\ x_i(t+1) &= x_i(t) + v_i(t+1) \end{aligned} \tag{3}$$

[2]A jump between funnels would be possible if the walker were situated near the boundary a funnel and the step length was large enough to reach across to the second funnel.

was chosen where x_i, v_i, p_i are particle position, velocity and historical best position of particle i, $u_{1,2} \sim U(0, 1)$ are uniform random variables in $[0, 1]^D$ and $\circ$ is the Hadamard (entry-wise) product, n_i is the historical best position of the best neighbour in i's social network (an arbitrary choice is made in the case of a tie). The neighbourhood is fully connected in GPSO i.e. $n_i = g$, the swarm's best ever position. LPSO neighbourhoods are localised: we ran experiments with the ring neighbourhood in which each particle has access to two other particles.

Finally, DE was included in the mix because of its formal similarity to PSO. The DE/best/1 version DE variant was selected for these trials because of its competitive and robust performance [4]. The DE particle update rule for particle i at x_i is

$$\begin{aligned} &\text{if } u \sim U(0, 1) < CR \text{ or } d == r \\ &\quad y_d = g_d + F(x_{jd} - x_{kd}) \\ &\text{else} \\ &\quad y_d = x_{id} \end{aligned} \tag{4}$$

where g is the best particle, j, k are random particle indices such that $i \neq j \neq k$ and $r = U(\{1, 2, \ldots D\})$ is a random component. y replaces x if $f(y) \leq f(x_i)$.

6.1 Experiment procedure

1001 runs were executed on different instances of each 30D BTF F1-F5. The search space was $[-\frac{L}{2}, \frac{L}{2}]^n$ with $L = 100$ and $n = 30$. Runs were terminated at 150000 function evaluations or when the

error fell below 1e-10, whichever was sooner. A ceiling of 300000 trials per run was enforced in order to terminate runs in which trial positions repeatedly landed outside the search space (such trials were not evaluated).

The algorithms were initialised inside the top level funnel but outside the lower level funnels and basins. The number of runs for attainment of each basin was recorded, and from this figure, a success probability was calculated.

A run was deemed a success if the algorithm terminated with the best-found position in the optimal basin. The motivation for this performance metric is that all optimisers exhibit local convergence; given that the optimiser population (or individual in the case of the DHW) has found the optimum basin it will almost certainly continue to plumb its depths). The essential criterion is therefore: can an optimiser discover the optimal basin? This metric has the advantage of being directly comparable across a range of functions of varying scale where simple optimal value (or error, if the global optimum is known) comparison is problematic. A further runtime measurement was taken: the number of region jumps. This statistic, optimiser *mobility*, is defined as the ability to move between regions [1].

Statistical significance testing at P=0.05 was conducted to check if the null hypothesis (equal numbers of runs attain each basin) could be rejected with 95% confidence.

The control parameters for the standard algorithms were set at the optimal values for a bff function [27]. Otherwise, a step reduction factor of 0.5 was chosen for DHW and SW; the initial step length was 0.2L and *tries* set to 10. A swarm of 100 walkers was chosen for the SW and a neighbour influence of 0.25. All control values are available in Table 2.

7 RESULTS

Tables 3 and 4 provide experiment results. The first column of Table 3 lists the expected success probability under the assumption that all basins are equally likely. This assumption constitutes the null hypothesis.

DHW success probabilities do not significantly depart from the null assumption probabilities except for $F2$ which has a smaller optimal funnel; in fact the DHW success rate is significantly lowered for $F2$ and confirms the intuition that the probability that a downhill walker algorithm will enter a particular funnel depends on relative surface area. The DHW results for F1 and F3-5 confirm that the BTF implementation is not prejudicial towards any basin. Table 4 shows that DHW jumps are always equal to one less than the number of funnels (the search space minus all bubble structure is itself a funnel) which demonstrates zero algorithm mobility: a downhill walker never jumps between regions.

The swarm of downhill walkers significantly performs better than the baseline assumption in all cases. SW success rates on the two-funnels-of-two-basin functions are high; success probabilities fall in the highly multimodal cases (F3-5) but are always higher than G/LPSO and DE. The SW is the most mobile of the any algorithm (Table 4) and this feature could contribute to the winning performance of this algorithm in this limited trial. SW seems capable of maintaining a high enough population diversity to allow good funnel and basin mobility despite the lack of any explicit mechanism.

LPSO delivers better success rates than GPSO and DE except for DE/F4. The success probabilities are significant for all functions. LPSO is the next most mobile algorithm after SW (Table 4). SW and LPSO share the same ring-topology particle communication; this strategy slows the communication of information through the swarm and promotes diversity.

DE has significant success probabilities in three our of five functions, and GPSO in two out of five. These optimisers perform markedly less well on the double funnel functions than SW and LPSO and struggle (in common with SW and LPSO) on the multi-basin functions.

The multi-optima BTFs present more challenges to the optimisers than double-funnel-double-basin functions. SW seems to cope well when the optima are divided between funnels (F5) whereas LPSO performs less well and appears to be sensitive to funnel structure. The GPSO results for F3-5 are not significant and are close to the DHW result. The implication is that global information sharing promotes faster convergence and diversity reduction. DE is able to achieve statistical power in the case of a structured series of differing basin depths within a single funnel (F4); the algorithm is possibly exploiting any semblance of structure that remains after bubble randomisation.

These results and ensuing analysis illustrate how the BTB, success probability and mobility metrics can combine to provide insight on algorithm behaviour.

8 ALGORITHM TUNING

Optimisation algorithms have inevitably control parameters, and performance will depend, sometimes critically, on the settings of these parameters (e.g. [29]). Although some theoretical bounds, often applicable in idealised situations (for example, the convergence bounds for PSO parameters apply only when the particles do not interact [2]) have been imposed on control settings of some algorithms, almost invariably algorithms must be empirically tuned on a set of problems. Automatic parameter tuners are being developed to relieve the laborious task of manual tuning [11].

The effect of tuning, whether manual or automatic, will depend on the selection of tuning problems. The task is to adjust control parameters so that a pertinent statistic is optimised throughout the tuning set. Often, mean error is minimised across the tuning set.

A major issue in the implementation of such a process is the choice of tuning benchmark. If the tuning benchmark is broad, i.e. consists of a wide variety of scenarios, tuning will impart only average settings that are unlikely to deliver optimal performance on any particular problem; alternatively, a very narrow benchmark will produce algorithm settings that do not generalise. The choice of a suitable tuning benchmark is hampered by the heterogeneous collection of problems that comprise standard benchmarks such as CEC and COCO. Even taking more homogeneous subsets of these benchmarks is problematic because the functions, defined as they are by mathematical formulae, remain unclassified except in the broadest terms (unimodal, multimodal, composition, non-separable etc.).

Table 2: Algorithm control parameters

DHW	initial step length = $0.2L$, reduction factor = 0.5, tries = 10
SW	N=100, initial step length = $0.2L$, reduction factor = 0.5, tries= 10, neighbour influence = 0.25
L/GPSO	$N = 100, w = 0.729844, c_1 = c_2 = 1.49618$
DE	$N = 50, F = 0.8, CR = 0$

Table 3: Success rate. Emboldened success probabilities are significant at $P = 0.05$ i.e. cases in which a success probability of $= \frac{1}{nBasins}$ can be rejected with 95% confidence.

	$\frac{1}{nBasins}$	DHW	SW	LPSO	GPSO	DE
F1	0.25	0.246	**0.972**	**0.701**	**0.282**	**0.281**
F2	0.25	**0.189**	**0.839**	**0.443**	**0.181**	**0.0470**
F3	0.01	5.00E-3	**0.133**	**6.19E-2**	7.99E-3	1.50E-2
F4	9.90E-3	9.99E-3	**0.141**	**0.0659**	9.99E-3	**0.102**
F5	9.80E-3	0.013	**0.146**	**0.018**	0.005	0.012

Table 4: Mobility. The table records the mean and, in parentheses, the standard deviation of the number of jumps between regions.

	DHW	SW	LPSO	GPSO	DE
F1	2 (0)	33 (18)	16 (14)	2 (0)	6 (4)
F2	2 (0)	31 (22)	16 (14)	2 (0)	5 (3)
F3	1 (0)	60 (38)	17 (15)	1 (0)	10(6)
F4	1 (0)	13 (9)	8 (6)	1 (0)	8 (5)
F5	2 (0)	44 (22)	19 (16)	2 (0)	8(5)

The adoption of a benchmark whose representatives are constructed within a desired classification scheme is a possible solution to the above issue. BTB functions are by design already classified - according to barrier tree and topographical features such as basin and funnel width. This is not the only means of function classification, of course, but the BTB is suggestive of a principled tuning procedure.

Suppose that it is wished to tune an algorithm on a particular function class. For example, PSO on double-funnel-double-basin (bffbff) problems. A training set of bffbff functions would be produced by a BTB generator. Each member of the training set would differ by funnel and basin size and depth (these characteristics could be constrained, if desired, for example, by specifying basin depths with a certain range). An automatic tuner would push the algorithm through the training set and optimal parameter settings would be determined. After tuning, the algorithm is applied to a test set, generated just as for the training set, but consisting of different function class instances. The ability of the tuned algorithm to generalise is then assessed by its performance on this unseen test set.

This scheme is feasible in principle but its outcome is yet to be trialled and evaluated. There are many interesting issues such as the broadness of the function class and the appropriateness of the BTB classification. Whatever the outcome, the deployment of a benchmark such as the BTB can surely aid principled algorithm tuning.

9 CONCLUSIONS

This paper has defined the Barrier Tree Benchmark and improved a previous definition of the funnel basis function. Functions within the benchmark (BTFs) are constructed from a predefined barrier tree of minima depths, and basin and funnel dimensions. The operation of a BTF generator has been described and a demonstration series of trials of five algorithms on five BTB classes has been reported.

Performance, as gauged by the success probability of an optimiser to find the optimal basin, has been compared with algorithm mobility with the conclusion that the more mobile algorithms tend to perform better on the double funnel and highly multi-modal functions.

A new algorithm, the Swarm Walker has been advanced. This algorithm, as the name implies, is a swarm of downhill walkers which are, in turn, individuals that proceed downhill by a random and possibly reducing step length. The Swarm Walker has very simple dynamics and outperforms the versions of global and local PSO and differential evolution utilised in the trials. No algorithm has been tuned to these functions however and further studies are needed to understand the merits of each algorithm on these function classes.

The paper closes with a proposal for deployment of the BTB in algorithm parameter tuning. It is expected that use of a principled tuning benchmark, such as the BTB, will lead to a more rigorous algorithm tuning procedure and hence to more meaningful algorithm comparison.

REFERENCES

[1] Tim Blackwell and James Kennedy. 2018. Impact of communication topology in particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 23, 4 (2018), 689–702.

[2] Maurice Clerc and James Kennedy. 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation* 6, 1 (2002), 58–73.

[3] Harold Scott Macdonald Coxeter. 1973. *Regular polytopes*. Courier Corporation.

[4] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. 2011. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation* 15, 1 (2011), 4–31. https://doi.org/10.1109/TEVC.2010.2059031

[5] Jonathan PK Doye, Mark A Miller, and David J Wales. 1999. Evolution of the potential energy surface with size for Lennard-Jones clusters. *The Journal of Chemical Physics* 111, 18 (1999), 8417–8428.

[6] Ryan Forbes and T Nayeem Mohammad. [n. d.]. Particle swarm optimization on multi-funnel functions. *Computer Aided Optimum Design in Engineering XII* 255 ([n. d.]).

[7] Marcus Gallagher and Bo Yuan. 2006. A general-purpose tunable landscape generator. *IEEE transactions on evolutionary computation* 10, 5 (2006), 590–603.

[8] Carlos García-Martínez, Pablo D Gutiérrez, Daniel Molina, Manuel Lozano, and Francisco Herrera. 2017. Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. *Soft Computing* 21 (2017), 5573–5583.

[9] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2021. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software* 36, 1 (2021), 114–144.

[10] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*. Ph. D. Dissertation. INRIA.

[11] Changwu Huang, Yuanxiang Li, and Xin Yao. 2019. A survey of automatic parameter tuning methods for metaheuristics. *IEEE transactions on evolutionary computation* 24, 2 (2019), 201–216.

[12] J. Kennedy. 1999. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *In: Proceedings of the 1999, Congress of Evolutionary Computation*, Vol. 3. IEEE Press, 1931–1938.

[13] Ho Min Lee, Donghwi Jung, Ali Sadollah, and Joong Hoon Kim. 2020. Performance comparison of metaheuristic algorithms using a modified Gaussian fitness landscape generator. *Soft Computing* 24, 10 (2020), 7383–7393.

[14] JJ Liang, BY Qu, PN Suganthan, and Alfredo G Hernández-Díaz. 2013. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report* 201212, 34 (2013), 281–295.

[15] Jane-Jing Liang, Ponnuthurai Nagaratnam Suganthan, and Kalyanmoy Deb. 2005. Novel composition test functions for numerical global optimization. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* IEEE, 68–75.

[16] Olaf Mersmann, Mike Preuss, Heike Trautmann, Bernd Bischl, and Claus Weihs. 2015. Analyzing the BBOB results by means of benchmarking concepts. *Evolutionary computation* 23, 1 (2015), 161–185.

[17] Chi-Kong Ng and Duan Li. 2014. Test problem generator for unconstrained global optimization. *Computers & operations research* 51 (2014), 338–349.

[18] Adam P Piotrowski. 2015. Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions. *Information Sciences* 297 (2015), 191–201.

[19] R. Poli, J. Kennedy, and T. Blackwell. 2007. Particle Swarm Optimization: An overview. *Swarm Intellligence* 1 (2007), 33–57.

[20] Ronald L Rardin and Reha Uzsoy. 2001. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics* 7, 3 (2001), 261–304.

[21] Maziar Salahi, Ali Jamalian, and Akram Taati. 2013. Global minimization of multi-funnel functions using particle swarm optimization. *Neural Computing and Applications* 23, 7 (2013), 2101–2106.

[22] Yun-Wei Shang and Yu-Huang Qiu. 2006. A note on the extended Rosenbrock function. *Evolutionary Computation* 14, 1 (2006), 119–126.

[23] Y. Shi and R. Eberhart. 1998. A modified particle swarm optimizer. In *Congress on Evolutionary Computation*. 69–73.

[24] Peter F Stadler. 2002. Fitness landscapes. In *Biological evolution and statistical physics*. Springer, 183–204.

[25] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, Ying-Ping Chen, Anne Auger, and Santosh Tiwari. 2005. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL report* 2005005, 2005 (2005), 2005.

[26] Andrew M Sutton, Darrell Whitley, Monte Lunacek, and Adele Howe. 2006. PSO and multi-funnel landscapes: how cooperation might limit exploration. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 75–82.

[27] Itshak Tkach and Tim Blackwell. 2022. Measuring optimiser performance on a conical barrier tree benchmark. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 22–30.

[28] Simon Wessing, Mike Preuss, and Günter Rudolph. 2013. Niching by multiobjectivization with neighbor information: Trade-offs and benefits. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, 103–110.

[29] Dennis Weyland. 2008. Simulated annealing, its parameter settings and the longest common subsequence problem. In *Proceedings of the 10th annual conference on genetic and evolutionary computation*. 803–810.

[30] Darrell Whitley, Soraya Rana, John Dzubera, and Keith E Mathias. 1996. Evaluating evolutionary algorithms. *Artificial intelligence* 85, 1-2 (1996), 245–276.

[31] Bin Xin, Jie Chen, and Feng Pan. 2009. Problem difficulty analysis for particle swarm optimization: deception and modality. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*. 623–630.