2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

malan2014progressive

A Progressive Random Walk Algorithm for Sampling Continuous Fitness Landscapes

Katherine M. Malan
Department of Computer Science
University of Pretoria
South Africa
Email: kmalan@cs.up.ac.za

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
South Africa
Email: engel@cs.up.ac.za

***Abstract*—A number of fitness landscape analysis approaches are based on random walks through discrete search spaces. Applying these approaches to real-encoded problems requires the notion of a random walk in continuous space. This paper proposes a progressive random walk algorithm and the use of multiple walks to sample neighbourhood structure in continuous multi-dimensional spaces. It is shown that better coverage of a search space is provided by progressive random walks than simple unbiased random walks.**

I. Introduction

Over the last few decades, the evolutionary computation community has dedicated considerable research effort to the analysis of fitness landscapes and the link to problem difficulty. Following the limited success in finding a single measure of problem hardness [1], [2], [3], [4], recent research efforts have instead been focussing on measuring multiple problem characteristics [5], [6], [7], [8] with the aim of automatically selecting an appropriate algorithm for solving a problem [9], [10], [11]. For this approach to be successful, there is a need for multiple techniques that measure different aspects of fitness landscapes, and there is a wealth of these in the literature (for a survey see [12]).

Many fitness landscape techniques were originally defined based on a complete enumeration of a discrete search space, but this is obviously not feasible for most real-world discrete problems or any real-valued problems. A form of sampling of the search space is therefore required on which to base an approximation of some originally precise measure. There are techniques that require a simple random sample of solution points, such as fitness distance correlation [13], density of states [14], dispersion metric [15] and exploratory landscape analysis features [6]. Other techniques are based on a random walk through the search space to capture a sequence of neighbouring solutions, such as autocorrelation techniques [16], correlation length [17] and entropic measures of ruggedness and smoothness with respect to neutrality [18]. Adapting these for real-valued problems would require a method for performing a random walk in a continuous space.

This paper proposes a random walk algorithm for continuous spaces with the following aims:

1) The random walk should be unbiased in terms of fitness, that is, fitness information should not be used to influence the direction of the walk. This is in contrast to online sampling (where the points are generated by the search algorithm itself) and other sampling strategies that use a particular search operator to define neighbourhood (as is the case with techniques that measure evolvability such as fitness clouds [19], negative slope coefficient [20], fitness-probability clouds [21] and accumulated escape probability [21]). The reason for requiring an algorithm-independent sample is so that the analysis based on the sample could be applicable to any search algorithm. If the ultimate aim is to select an appropriate algorithm based on the characteristics of the problem, then the characterisation should be algorithm independent.
2) The points in a walk should be ordered based on some generic notion of neighbourhood, such as Euclidean distance. In this way, a walk captures neighbourhood information and the series of fitness values generated through a walk can be assumed to be related in space based on the distance metric used.
3) The set of walks used to sample the search space should attempt to provide as wide a coverage of the search space as possible within the constraints of acceptable computational cost. If the purpose of analysing a problem is to obtain information to guide the choice of an appropriate algorithm, then the computational cost of sampling should be significantly less than solving the problem with multiple algorithms using a trial-and-error approach.

Although requirement 3 above stresses that the computational effort of sampling and characterising a problem should be significantly less than the computational effort in using a trial-and-error approach with multiple algorithms, one could argue that this is not an essential feature. A trial and error approach to solving an unknown problem has no guarantee of producing a good solution to the problem. On the other hand, characterising a problem should lead to a deeper understanding of the problem and better choices of algorithms and therefore have an increased chance of producing a solution of higher quality than the uninformed application of multiple search algorithms.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

The proposed random walk algorithm discussed in this paper was implemented in a previous study [22] to investigate the ruggedness, funnels and gradients in fitness landscapes and the effect on the performance of a traditional particle swarm optimisation (PSO) algorithm. The progressive random walk algorithm was used as the basis for a ruggedness measure based on entropy and a variant on this random walk, called a Manhattan progressive random walk, was used as the basis of a measure for estimating gradients. Results showed that the ruggedness and gradient measures were moderately correlated to the performance of a traditional PSO on a range of benchmark problems [22] and a further study [23] showed that the gradient measure, in particular, was a good predictor of failure for different variations on the PSO algorithm. The purpose of this paper is to properly describe the progressive random walk algorithm and to highlight why such an approach is justified as the basis for sampling neighbouring fitness values in continuous spaces.

The remainder of the paper is organised as follows: Section II provides an overview of approaches to random walks in the literature, Section III looks at a simple unbiased random walk algorithm for continuous spaces and shows that the coverage of the search space is not adequate. Section IV proposes an alternative algorithm called a *progressive random walk* algorithm. The coverage of walks in higher dimensions is tested in Section V, followed by a conclusion in Section VI.

II. Random Walks

This section provides a brief overview of random walks in general and describes how random walks have been used in fitness landscape analysis.

A. Random walks in general

Random walks have been used to describe features and phenomena in a wide variety of fields such a physics, biology, chemistry, psychology and economics. Many of the models of random walks are defined for a particular dimension, such as a one-dimensional walk representing the price of a fluctuating stock, a two-dimensional walk for the search path of a foraging animal or a three-dimensional walk of a particle in gas (Brownian motion). In the domain of continuous optimisation problems, however, random walk models are n-dimensional to apply to the solution space of any n-dimensional problem.

The first description of a random walk in literature appeared in a letter to the journal *Nature* from Pearson in 1905 [24] and is described as follows: *"A man starts from a point O and walks I yards in a straight line; he then turns through any angle whatever and walks another I yards in a second straight line. He repeats this process n times."* Pearson's description is a fixed step length random walk in two-dimensional continuous space. The randomness is captured in the angle that the man turns at each point. The walk is isotropic, because the man can turn in any direction. If there is some restriction or bias on the size of the angle, then the walk would be anisotropic. Examples of models of anisotropic walks in two-dimensional continuous space include a random walk of a particle with directional memory [25] and the model of random motion with preferential direction [26]. The progressive random walk algorithm proposed in this paper generates an anisotropic random walk in n-dimensional continuous space.

When a random walk has random step lengths, the step length can be based on some probability distribution, such as uniform, Gaussian, or a heavy-tailed distribution (in the case of Lévy flights). The random walk algorithms described in this paper generate walks with step lengths based on a uniform distribution. Uniform distributions are commonly used for generating the stochastic elements of many optimisation algorithms, such as PSOs.

B. Random walks in fitness landscapes

Random walks featured as the basis of the earliest fitness landscape analysis techniques with Kauffman and Levin's [27] adaptive walk. An adaptive walk is a biased random walk, originally defined for binary spaces, where each next point in the walk has a better fitness value than the previous point. If the expected length of an adaptive walk is short, then this would signify an uncorrelated (or rugged) fitness landscape. Since then, a number of other fitness landscape analysis techniques have been proposed based on biased and unbiased random walks, such as autocorrelation measures [16], [28], correlation length [17], entropic measures [29], [18], [30] and the length of neutral walks [31]. All of these techniques assumed a discrete problem with each point in the space having a finite number of neighbouring points. For binary problems, a random walk could be implemented as follows [18]: start from a randomly chosen point, generate all neighbours of the current point by mutation (bit flip), choose randomly one neighbour as the next point, generate all neighbours of the new point, and so on. In the case of continuous search spaces, however, there is no equivalent set of all possible neighbours of any point, since the number of neighbours of a point is theoretically infinite. Section III-A defines a simple hypercube model of neighbourhood for n-dimensional space from which neighbours can be sampled using a uniform distribution spanning the neighbourhood in each dimension.

A recent contribution in fitness landscape analysis of continuous problems, based on random walks, is Morgan and Gallagher's [32] length scale property. This approach characterises continuous landscapes using the entropy of length scale values, sampled using a Lévy flight. Due to the heavy-tailed distribution used for choosing step lengths, this random walk in solution space has frequent small steps interspersed with infrequent large steps. For the purposes of this study, a Lévy walk was not considered as a suitable approach, since it does not meet the second aim outlined in Section I. Due to the occasional large steps, adjacent points on a Lévy walk could be positioned relatively far away from each other, resulting in neighbouring fitness values from very different parts of the fitness landscape.

All of the fitness landscape techniques discussed above are based on random walks that are created by generating

points in a sequence based on some concept of neighbourhood. A point in the search space is chosen as a starting point, then some algorithm is followed to repeatedly choose the next point in the neighbourhood until the required random sequence of points has been created. An alternative approach could be to generate a sample of points using a sampling strategy with a good coverage of the search space, such as Latin Hypercube sampling for continuous spaces, and then to connect the points into a walk using a form of pathfinding with nearest neighbour search. The algorithm proposed in this paper using the former approach, but it would be interesting to see if a computationally cheap way could be developed using the latter approach.

III. Simple Random Walk in Continuous Space

This section proposes an algorithm for generating an isotropic walk in n-dimensional continuous space. A definition of neighbourhood is defined for n-dimensional continuous space, the algorithm is proposed and sample runs are plotted for different step bounds.

A. Hypercube neighbourhood

The neighbourhood of a multidimensional point $\mathbf{x}$ is commonly defined as the set of points within the hypersphere with some small radius and centre $\mathbf{x}$ [33]. However, this approach requires Euclidean distance calculations to ensure that one point is in the neighbourhood of another point. To simplify the computational complexity of neighbourhood checking, a neighbourhood based on hypercubes is proposed. Formally, the neighbourhood set $N(\mathbf{x}_k)$ of an n-dimensional point $\mathbf{x}_k$ is defined as follows:

$$\mathbf{x}_j \in N(\mathbf{x}_k) \iff |x_{ki} - x_{ji}| < s, \ \forall i \in [1, \ldots, n] \quad (1)$$

where s is half of one length of the hypercube specifying the neighbourhood size. Note that this definition assumes that the neighbourhood size is equal in all dimensions. In some cases it may be desirable to define s, not as a scalar value, but as a multidimensional vector $\mathbf{s}$. For example, consider a two-dimensional search space where variables x_1 and x_2 have domains of $[0, 100]$ and $[0, 1]$, respectively. The neighbourhood size could be specified as $\mathbf{s} = [10, 0.1]$, so that although the neighourhood is technically a rectangle, it is a square relative to the seach space, since each side of the neighbourhood is 10% of the domain.

B. Simple Random Walk algorithm

Given the definition of neighbourhood in Equation 1, a simple approach to a random walk through a continuous space could be the following: start at a random position within the bounds of the multi-dimensional search space; take a step of random size and direction, within the bounds of the multi-dimensional hypercube defining the neighbourhood, always ensuring that the walk stays within the outer bounds of the search space, until the required number of steps are reached. This simple random walk algorithm is expressed more formally in Appendix A and Figure 1 plots the position vectors of sample runs for a two-dimensional space using different values of s. As can be seen, taking steps in random directions has a tendency for the points of a walk to be clustered in limited areas of the search space, which becomes more pronounced as the value of s decreases. This can result in poor coverage of the search space.

IV. Proposed Progressive Random Walk

To address the problem of points in the random walk being clustered in limited areas of the search space, a directional bias is introduced in the step to produce an anisotropic or *progressive random walk*. This is similar to Huang et al.'s [26] approach for two-dimensional space, where the walk has a preferential direction on a given axis.

The basic idea of a progressive random walk is as follows: A walk starts on the edge of the multi-dimensional search space, progresses in a random way, but with a bias in direction towards the opposite side of the search space. If a search space boundary is reached, the bias is changed to the opposite direction. Multiple walks are generated from different random starting positions on the outer boundaries of the search space. The details of the approach are specified below.

A. Starting Zones for Progressive Random Walks

As described in Table I, for an n-dimensional search space, there are 2^n non-overlapping starting zones. Each starting zone is identified using an n-bit string $b_1 \ldots b_n$, which specifies the corner point $(c_1, \ldots, c_n)$ of the starting zone as follows:

$$c_i = \begin{cases} x_i^{min} & \text{if } b_i = 0 \\ x_i^{max} & \text{if } b_i = 1 \end{cases} \quad (2)$$

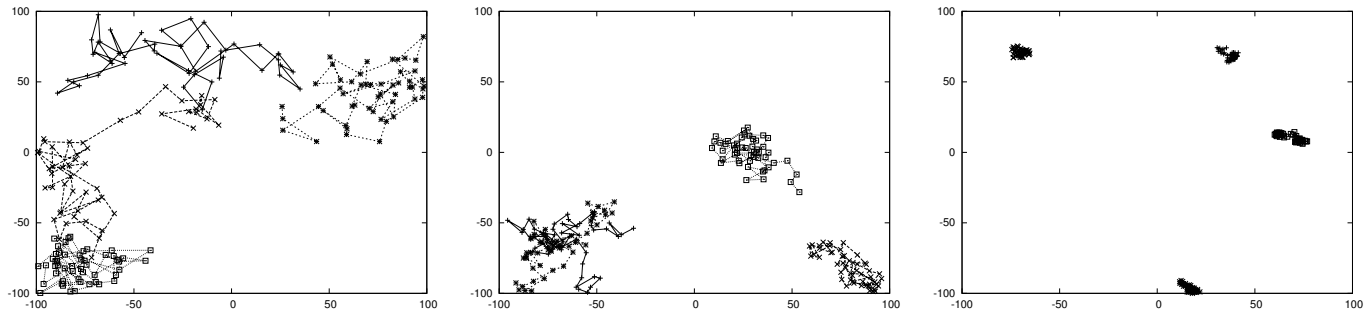
A point $(x_1, \ldots, x_n)$ is defined as being in the starting zone identified by the binary number $b_1 \ldots b_n$ with corner point $(c_1, \ldots, c_n)$ when the following hold:

$$\begin{array}{ll} \forall i \in [1, \ldots, n], & |x_i - c_i| < \frac{x_i^{max} - x_i^{min}}{2}, \text{ and} \\ \exists i \in [1, \ldots, n], & \text{where} \quad x_i = c_i \end{array} \quad (3)$$

For a one-dimensional search space, there are only two possible starting positions on the edge of the search space: the minimum and maximum points defining the problem domain. In two dimensions, there are four lines of boundary points; in three dimensions there are six planes of boundary points; and so on. In an attempt to provide a wide coverage of the search space, the set of points on the boundary is divided into non-overlapping zones, so that multiple random walks can start in different portions of the outer boundary. The proposed approach to dividing the boundary points into zones is described in Table I, where the domain for each dimension i is specified as $[x_i^{min}, x_i^{max}]$.

B. Progressive Random Walk Algorithm

Given a binary number specifying the starting zone, a random progressive walk starts by generating a random position in the specified starting zone. Steps are then based on random offsets in each dimension, within the neighbourhood, in the direction of the opposite boundaries. The algorithm is



(a) s = 20 (b) s = 10 (c) s = 2

Fig. 1. Plots of the position vectors of sample simple (isotropic) random walk runs for a two-dimensional space with differing values of s (step bound). Each sub-figure shows four independent sample walks of 50 steps each.

TABLE I
DESCRIPTION OF PROPOSED STARTING ZONES FOR DIFFERENT DIMENSIONS.

	Description of starting zones and random walk progression
1-D	Two starting zones, which are the single points (x^{min}), starting zone 0, and (x^{max}), starting zone 1.
2-D	Four starting zones, corresponding to each corner of the rectangular search space. Each starting zone is defined as the set of points falling on the two lines extending from the corner point up to the midpoints of the ranges along the axes, as illustrated in Figure 2. Starting zone 00 refers to the lines extending from corner (x_1^{min}, x_2^{min}). Similarly, zones 01, 10 and 11 refer to the lines extending from corners (x_1^{min}, x_2^{max}), (x_1^{max}, x_2^{min}), and (x_1^{max}, x_2^{max}), respectively.
3-D	Eight starting zones (identified as zones 000 to 111 in binary) corresponding to the eight corners of the cuboid defining the search space. Each starting zone is defined as the set of points on the planes falling on the outer boundaries of the search space extending from the given corner up to the midpoints of the ranges of the axes.
n-D	2^n starting zones corresponding with the 2^n corner points, where each corner point is of the form $(c_1, \ldots, c_n)$, such that $\forall i \in [1, \ldots, n], c_i \in \{x_i^{min}, x_i^{max}\}$. Each starting zone is defined as the set of points falling on the outer boundary of the search space extending from the given corner point up to the midpoints on the ranges of the axes.

expressed more formally in Appendix B, while Figure 3 plots the position vectors generated by sample runs of the algorithm for a two-dimensional space using different values of s.

In constrast to Figure 1, it can be seen that a better coverage is obtained by the progressive random walks than the simple random walks in two dimensions. It can also be seen that a step bound of 20 (10% of the range of the domain on a single dimension) provides a better coverage of the space than smaller bounds, while still maintaining a reasonably close proximity between points on the walk.

V. TESTING COVERAGE OF WALKS

This section proposes a technique for quantifying the coverage of a sample in continuous space in terms of deviation from the distribution of a uniform sample.

A. Estimating coverage

A histogram is a standard technique for visualising and estimating the distribution of a sample. For example Figure 4 shows the distributions of three samples of 10 000 points in a two-dimensional space with domain $[-100, 100]$. The frequencies are based on 100 (10×10) bins of equal size, resulting in a mean of 100 points per bin. Figure 4(a) shows the distribution of a uniform random sample of the space. It can be seen that the frequencies deviate slightly from the mean of 100. Figures 4(b) and 4(c) show the distributions of samples resulting from simple random and progressive random walks respectively. In both cases the samples were generated from four equal-length walks with a step bound of 20 (10% of the domain), with the progressive random walks starting in different starting zones. As can be seen, the distribution of samples produced by the progressive random walks is more similar to a uniform distribution than the simple random walks. The problem of clustering of points in the search space is clearly evident in the histogram of the simple random walks.

To test the coverage of the random walk algorithms in higher dimensions, experiments were conducted on search spaces with domain $[-100, 100]$ for dimensions (D) 1, 2, 3, 4, 6 and 10. Sample sizes were chosen to be in the order of $10^4 \times D$ (the maximum number of function evaluations sometimes used in real-parameter optimisation competitions [34]), but adjusted to allow for a set mean of 100 points in a bin, while also ensuring equal sized bins (k^D for some integer k). The number of points and bins for each dimension is given in Table II.

For the two random walk algorithms, 30 samples were generated through independent runs of the algorithm for each dimension. Each run consisted of 2^D walks using a step bound of 20 (10% of the domain). For the progressive random walk each walk of a sample started in a different starting zone. For each run in a sample, the standard deviation of the frequency in the bins was calculated and the mean of the standard deviations over the 30 runs was calculated. The deviations based on uniform random samples were also calculated and the results are given in Table II. As can be seen, a uniform random sample on average deviates by approximately 10 points in a bin from the mean of 100 in all dimensions. A sample generated by

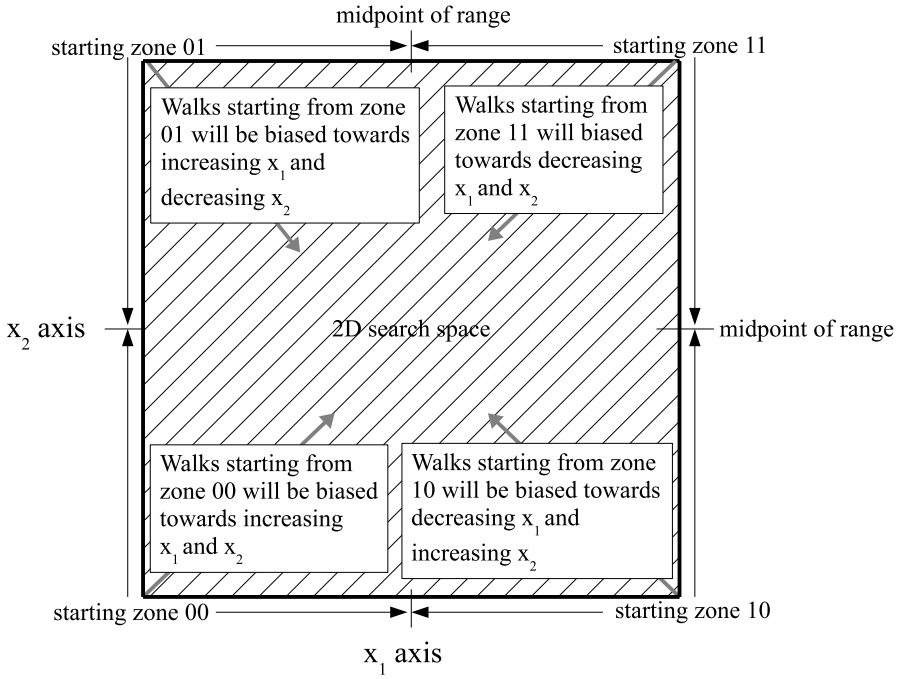
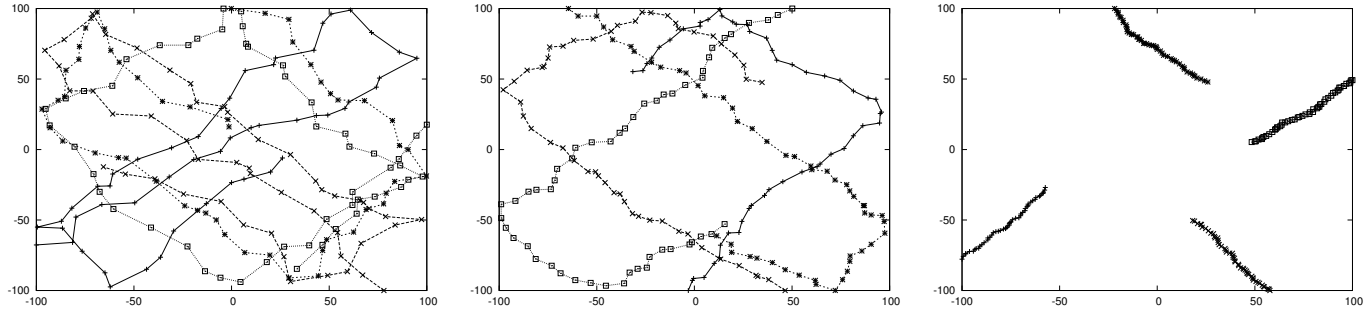
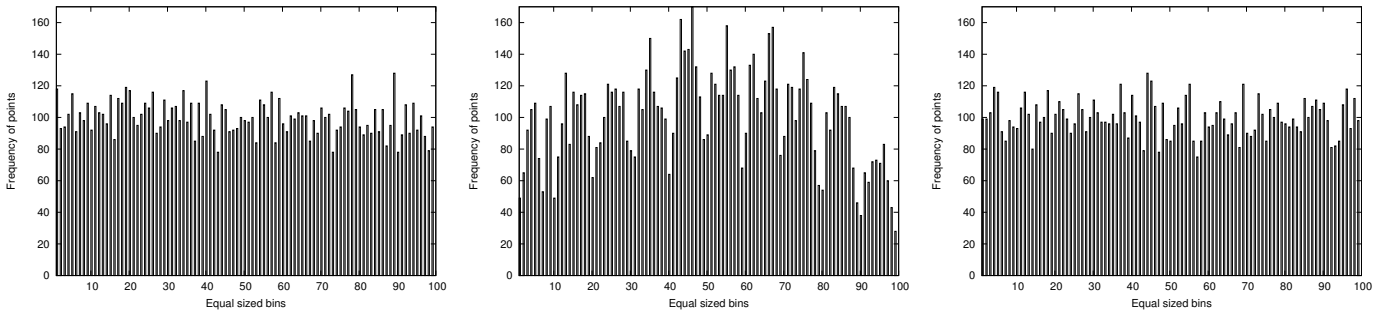


Fig. 2. Four starting zones for a two-dimensional search space.



(a) $s = 20$ (b) $s = 10$ (c) $s = 2$

Fig. 3. Plots of the position vectors of sample progressive random walk runs for a two-dimensional space with differing values of s (step bound). Each sub-figure shows four sample walks of 50 steps each, each starting in a different starting zone. Wider coverage of the search space is obtained than in the case of the simple random walks as illustrated in Figure 1



(a) Uniform sample (b) Simple random walks (c) Progressive random walks

Fig. 4. Histograms showing the distribution of samples of 10 000 points in a two-dimensional space. Frequency is in terms of numbers of points in 10 $\times$ 10 equal-sized bins.

a simple random walk has much higher average deviations, ranging from approximately 20 in 1 dimension to over 30 in 10 dimensions. Samples generated by progressive random walks have significantly smaller deviations on average than simple random walks.

These results show that a progressive random walk sample is more uniformly distributed than a simple random walk sample, but not as uniformly distributed as a uniform random sample. The advantage of progressive random walk samples over uniform random samples is that additional information is provided in the form of the neighbourhood captured in the sequence of points in the sample.

B. Multiple random walks in multi-dimensional space

In multi-dimensional spaces, the size of the search space increases exponentially as the dimension increases. When performing a random walk, larger search spaces require more walks to sample the space. When sampling an n-dimensional space using a random walk, the number of walks should ideally be equal to the number of starting zones, that is 2^n, as described in Table I. This is clearly infeasible for high dimensions, especially since the aim is to provide a characterisation of a problem that is less computationally expensive than actually solving the optimisation problem.

To keep within a limit of $10^3 \times n$ sample points as the basis for fitness landscape measures (10 times less than the maximum number of function evaluations commonly used in real-parameter optimisation competitions), the following approach is proposed for n-dimensional space:

- The number of walks performed is equal to the number of dimensions, so n independent walks are performed with each walk starting in a different starting zone. This ensures a linear growth in computation time as the dimension increases.
- To distribute different walks across the starting zones, every $(\frac{2^n}{n})^{th}$ starting zone is used as the starting point for a walk. For example, for a 4-dimensional space with 16 starting zones, 4 walks will be performed starting in zones 0000, 0100, 1000 and 1100.

Note that any sampling of a high dimensional space will provide inadequate coverage of the search space. The aim is to have a random sampling technique that can be used as the basis of measures that are reasonably reliable (that is, the resulting characterisation measures do not have very large standard deviations).

The use of multiple progressive random walks as described above was used in a previous study [22] to estimate the macro ruggedness of a range of benchmark functions in different dimensions. The ruggedness estimate was determined based on D progressive random walks of 10^3 steps, with a step bound of $(x^{max} - x^{min}) * 0.1$, where x^{max} and x^{min} define the bounds of the search space. Using a fairly large step bound such as this (up to 10% of the range of the domain) has the effect of estimating ruggedness on a larger scale (macro ruggedness), rather than micro ruggedness. Thirty independent runs of the ruggedness estimation algorithm were performed on each function and dimension combination. Figure 5 plots the mean ruggedness values for dimensions 1, 2, 5, 15, and 30 for a number of benchmark functions (definitions and one-dimensional plots of the functions are provided in [22]).

It can be observed from Figure 5 that in 30 dimensions, there are three distinct groups of functions with similar macro ruggedness values. Functions Ackley, Rastrigin, Salomon and Schwefel 2.26 have the highest macro ruggedness values. Functions Griewank, Spherical and Step have similar ruggedness values (on a macro level, these functions have the same basic shape) and Quadric has the lowest value. More importantly, it can be seen from Figure 5 that the standard deviations of the ruggedness measure do not in general increase with an increase in dimension, indicating that the walks on which the measure is based provide sufficient information to characterise the ruggedness for these benchmark problems.

VI. Conclusion

A random walk is used for sampling a fitness landscape when the relative fitness values of neighbouring points is of relevance. This paper proposes an algorithm for performing an anisotropic random walk in continuous spaces that could be used as the basis of techniques for approximating landscape topology features such as ruggedness, smoothness and neutrality. The algorithm is called a progressive random walk and works on the principle of a walk that starts at a random position on the boundary of a search space and progresses in a random way with a bias towards the boundaries at the other side of the search space. It is shown that, when used with an appropriate step bound, the distribution of progressive random walk samples is more uniform than the distribution of simple random walk samples and so provides a better coverage of the entire search space. In high dimensions, multiple random walks starting in different zones on the boundary can be used as the basis for obtaining sufficient information to distinguish between different classes of problems based on estimated characteristics.

References

[1] H. Guo and W. H. Hsu, "GA-Hardness Revisited," in *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, ser. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2003, vol. 2724, pp. 1584–1585.

[2] J. He, C. Reeves, C. Witt, and X. Yao, "A Note on Problem Difficulty Measures in Black-Box Optimization: Classification, Realizations and Predictability," *Evolutionary Computation*, vol. 15, no. 4, pp. 435–443, 2007.

[3] T. Jansen, "On Classifications of Fitness Functions," in *Theoretical aspects of evolutionary computing*. London, UK: Springer-Verlag, 2001, pp. 371–385.

[4] B. Naudts and L. Kallel, "A Comparison of Predictive Measures of Problem Difficulty in Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 1, p. 1, April 2000.

[5] P. Caamaño, F. Bellas, J. A. Becerra, V. Díaz, and R. J. Duro, "Experimental analysis of the relevance of fitness landscape topographical characterization," in *IEEE Congress on Evolutionary Computation*, June 2012, pp. 1–8.

[6] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory landscape analysis," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011, pp. 829–836.

TABLE II
AVERAGE STANDARD DEVIATIONS FROM THE MEAN FREQUENCY IN A BIN (MEAN EQUALS 100), BASED ON 30 RUNS, WITH STANDARD DEVIATIONS SHOWN IN BRACKETS.

Dimension	Number of points in sample	Number of bins	Average standard deviations from the mean of 100		
			Uniform random sample	Simple random walk	Progressive random walk
1	10,000	100	10.21 ($\pm$0.55)	20.02 ($\pm$4.34)	9.59 ($\pm$0.69)
2	19,600	196 (14^2)	10.03 ($\pm$0.64)	24.75 ($\pm$2.48)	12.75 ($\pm$2.33)
3	34,300	343 (7^3)	10.06 ($\pm$0.40)	26.36 ($\pm$1.88)	16.47 ($\pm$2.57)
4	62,500	625 (5^4)	9.93 ($\pm$0.27)	26.56 ($\pm$1.35)	18.34 ($\pm$1.59)
6	72,900	729 (3^6)	10.07 ($\pm$0.25)	31.07 ($\pm$1.47)	23.86 ($\pm$1.17)
10	102,400	1024 (2^{10})	9.98 ($\pm$0.26)	33.61 ($\pm$0.89)	19.67 ($\pm$0.62)

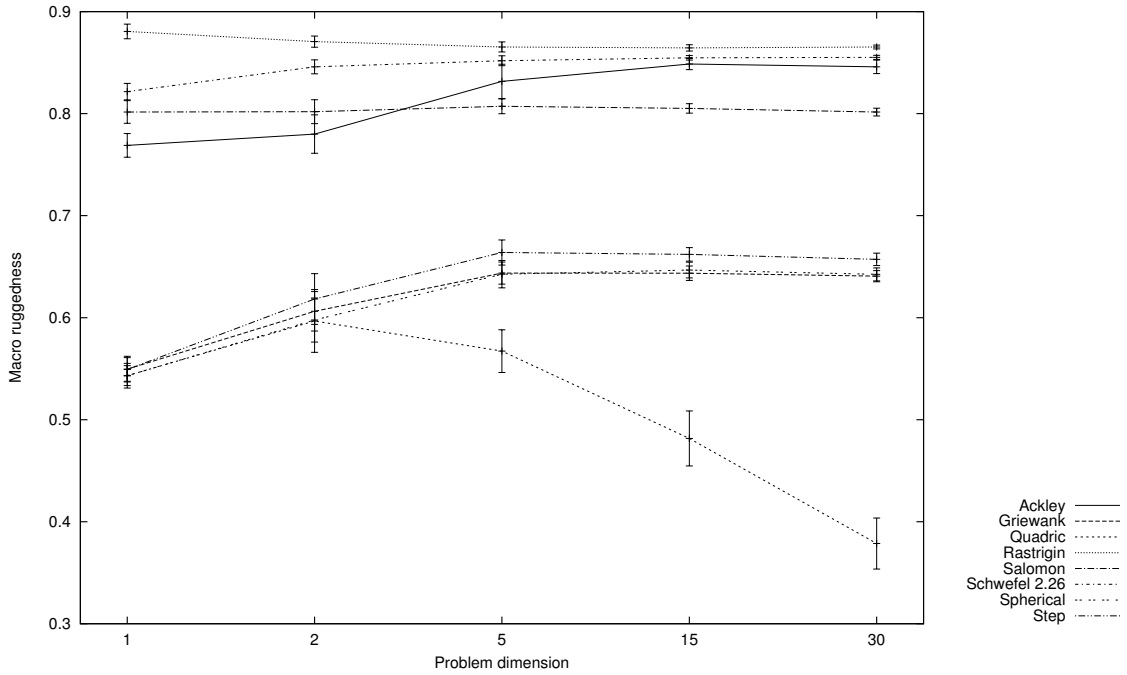


Fig. 5. Estimate of macro ruggedness for a number of benchmark functions in different dimensions. Values are means over 30 independent runs with standard deviations indicated as error bars.

[7] M. A. Muñoz, M. Kirley, and S. K. Halgamuge, "Landscape characterization of numerical optimization problems using biased scattered data," in *IEEE Congress on Evolutionary Computation*, June 2012, pp. 1–8.

[8] K. Smith-Miles and L. Lopes, "Measuring instance difficulty for combinatorial optimization problems," *Computers & Operations Research*, vol. 39, no. 5, pp. 875 – 889, 2012.

[9] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß, "Algorithm selection based on exploratory landscape analysis and cost-sensitive learning," in *Proceedings of the Fourteenth International Genetic and Evolutionary Computation Conference*, 2012, pp. 313–320.

[10] M. A. Muñoz, M. Kirley, and S. K. Halgamuge, "A meta-learning prediction model of algorithm performance for continuous optimization problems," in *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature – Part I*, 2012, pp. 226–235.

[11] K. Smith-Miles, "Towards insightful algorithm selection for optimisation using meta-learning concepts," in *Proceedings of the IEEE Joint Conference on Neural Networks*, 2008, pp. 4118–4124.

[12] K. M. Malan and A. P. Engelbrecht, "A survey of techniques for characterising fitness landscapes and some possible ways forward," *Information Sciences*, vol. 241, pp. 148–163, 2013.

[13] T. Jones and S. Forrest, "Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995, pp. 184–192.

[14] H. Rosé, W. Ebeling, and T. Asselmeyer, "The Density of States - a Measure of the Difficulty of Optimisation Problems," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1996, pp. 208–217.

[15] M. Lunacek and D. Whitley, "The dispersion metric and the CMA evolution strategy," in *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference*, 2006, pp. 477–484.

[16] E. Weinberger, "Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference," *Biological Cybernetics*, vol. 63, no. 5, pp. 325–336, September 1990.

[17] M. Lipsitch, "Adaptation on rugged landscapes generated by iterated local interactions of neighboring genes," in *Proceedings of the 4th International Conference on Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds., 1991, pp. 128–135.

[18] V. K. Vassilev, T. C. Fogarty, and J. F. Miller, "Smoothness, Ruggedness and Neutrality of Fitness Landscapes: from Theory to Application," in *Advances in Evolutionary Computing: Theory and Applications*. Springer-Verlag New York, Inc., 2003, pp. 3–44.

[19] S. Verel, P. Collard, and M. Clergue, "Where are bottlenecks in NK fitness landscapes?" in *Proceedings of the 2003 Congress on Evolutionary Computation*, vol. 1, 2003, pp. 273–280.

[20] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Verel, "Fitness Clouds and Problem Hardness in Genetic Programming," in *Proceedings of Genetic and Evolutionary Computation Conference*, ser. Lecture Notes in Computer Science, K. Deb, Ed. Springer Berlin Heidelberg, 2004, vol. 3103, pp. 690–701.

[21] G. Lu, J. Li, and X. Yao, "Fitness-Probability Cloud and a Measure

of Problem Hardness for Evolutionary Algorithms," in *Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science, vol. 6622. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 108–117.
[22] K. M. Malan and A. P. Engelbrecht, "Ruggedness, Funnels and Gradients in Fitness Landscapes and the Effect on PSO Performance," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2013, pp. 963–970.
[23] ——, "Steep Gradients as a Predictor of PSO Failure," in *Proceedings of the Fifteenth International Conference on Genetic and Evolutionary Computation Conference, Companion*, 2013, pp. 9–10.
[24] K. Pearson, "The problem of the random walk," *Nature*, vol. 72, p. 294, 1905.
[25] C. Tojo and P. Argyrakis, "Correlated random walk in continuous space," *Physical Review E*, vol. 54, no. 1, pp. 58–63, 1996.
[26] S.-Y. Huang, X.-W. Zou, and Z.-Z. Jin, "Directed random walks in continuous space," *Physical Review E*, vol. 65, no. 5, p. 052105, 2002.
[27] S. Kauffman and S. Levin, "Towards a General Theory of Adaptive Walks on Rugged Landscapes," *Journal of Theoretical Biology*, vol. 128, no. 1, pp. 11–45, September 1987.
[28] B. Manderick, M. K. de Weger, and P. Spiessens, "The Genetic Algorithm and the Structure of the Fitness Landscape," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds., 1991, pp. 143–150.
[29] V. K. Vassilev, T. C. Fogarty, and J. F. Miller, "Information Characteristics and the Structure of Landscapes," *Evolutionary Computation*, vol. 8, no. 1, pp. 31–60, 2000.
[30] K. M. Malan and A. P. Engelbrecht, "Quantifying Ruggedness of Continuous Landscapes using Entropy," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2009, pp. 1440–1447.
[31] C. M. Reidys and P. F. Stadler, "Neutrality in fitness landscapes," *Applied Mathematics and Computation*, vol. 117, no. 2-3, pp. 321–350, 2001.
[32] R. Morgan and M. Gallagher, "Length scale for characterising continuous optimization problems," in *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature – Part I*, 2012, pp. 407–416.
[33] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2009.
[34] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University, Singapore, Tech. Rep., May 2005.

APPENDIX A
SIMPLE RANDOM WALK ALGORITHM

Let p be the problem on which to base the walk, which encapsulates the number of dimensions, n, and domain for each dimension $[x_1^{min}, x_1^{max}], \ldots, [x_n^{min}, x_n^{max}]$
Let $numSteps$ specify the number of steps in the walk
Let $stepSize$ specify the bound on the step size for each dimension
Create an array of n-dimensional vectors for storing the walk (called *walk*) of size $numSteps + 1$
for all dimension i of n **do**
  Generate a random number r in the range $[x_i^{min}, x_i^{max})$
  Set $walk[0]_i$ to r
end for
for all step s from 1 **to** $numSteps$ **do**
  for all dimension i in n **do**
    repeat
      Generate a random number, r, in the range $[-stepSize, +stepSize)$
    until $walk[s-1]_i + r$ is in bounds of search space
    Set $walk[s]_i = walk[s-1]_i + r$
  end for
end for

APPENDIX B
PROGRESSIVE RANDOM WALK ALGORITHM

Let p be the problem on which to base the walk, which encapsulates the number of dimensions, n, and domain for each dimension $[x_1^{min}, x_1^{max}], \ldots, [x_n^{min}, x_n^{max}]$
Let $numSteps$ specify the number of steps in the walk and $stepSize$ the bound on the step size for each dimension
Let $startingZone$ be a binay array of size n (a bit for each dimension) specifying the starting zone of the walk and how the walk should progress
Create an array of n-dimensional vectors for storing the walk (called *walk*) of size $numSteps + 1$
for all dimension i of n **do**
  Generate a random number r in the range $[0, \frac{x_i^{max} - x_i^{min}}{2})$
  if $startingZone_i$ equals 1 **then**
    set $walk[0]_i$ to $x_i^{max} - r$
  else
    set $walk[0]_i$ to $x_i^{min} + r$
  end if
end for
Generate a random dimension, rD, in range $[0, \ldots, n]$
if $startingZone_i$ equals 1 **then**
  set $walk[0]_{rD}$ to x_i^{max}
else
  set $walk[0]_{rD}$ to x_i^{min}
end if
for all step s from 1 to $numSteps$ **do**
  for all dimension i in n **do**
    Generate a random number, r in the range $[0, stepSize)$
    if $startingZone_i$ equals 1 **then**
      set r to $-r$
    end if
    Set $walk[s]_i = walk[s-1]_i + r$
    if $walk[s]_i$ is out of bounds **then**
      set $walk[s]_i$ to mirrored position inside boundary
      Flip bit $startingZone_i$
    end if
  end for
end for