Assessing Robustness of Optimisation Performance for Problems With Expensive Evaluation Functions

Evan J. Hughes, *Member, IEEE*

***Abstract*— In complex engineering problems, the objective functions can be very slow to evaluate, restricting the optimisation process to only a few hundred objective calculations. Often the optimisation process can only be performed once, requiring a good solution from the single run. Thus we need a robust approach to algorithm development and tuning.**

This paper introduces a new metric for *quantifying* the performance of different algorithms on different test functions relative to the range of performance expected from a random search. As a random search is a repeatable benchmark for any objective function, the metric can be applied as an absolute, rather than relative metric. The metric allows the best, worst and median performance of different algorithms to be compared directly, even for optimisation runs with only tens of evaluations.

Additionally a new optimisation algorithm, based on a Voronoi decomposition of the decision space, is presented that provides reliable optimisation performance, but with a very limited number of function evaluations. The paper evaluates the performance of the new algorithm with the new metric on a range of surfaces and against a typical evolutionary approach.

I. Introduction

Evolutionary Algorithms [1] are becoming a well established technique for solving hard engineering problems. Objective functions are becoming more complex and consequently can take a long time to evaluate. Problems such as aerodynamic optimisation and electromagnetic simulation often rely on finite element methods in order to simulate the systems of interest. These simulations can take from seconds to hours to run. The better the resolution and fidelity required, the longer the simulation time.

Many excellent algorithms have been developed (see [2] for an excellent survey) that tackle the problems associated with optimising expensive objective functions, but often they rely on generating a smooth local model of the anticipated fitness landscape in order to guide the selection of the next point to evaluate. With rough fitness landscapes (for example, problems which approximate a fractal surface, or surfaces which have very deep but narrow optima), a smooth approximation can be misleading.

Many publications in the area of expensive fitness evaluation are concerned with the direct comparison of a range of algorithms with each other. The trials are often suitable for assessing which algorithm is better on the chosen problems, but they do not indicate whether *any* of the algorithms are actually performing well.

Evan J. Hughes is with the Department of Aerospace, Power and Sensors, Cranfield University, DCMT, Shrivenham, Swindon, Wiltshire, England. SN6 8LA. (phone: +44 (0)1793 785255; fax: +44 (0)1793 785902; email: e.j.hughes@cranfield.ac.uk).

When the optimisation algorithms are applied to the real problem of interest, we need to be sure that even a bad run (given the stochastic nature of the EA) is likely to produce a useful result. Thus we require test functions that have similar characteristics to the problem of interest, allowing algorithms to be developed and tuned without the expensive optimisations; and also assessment metrics that alow us to quantify how *bad* the optimiser may be, as well as the typical performance.

This paper details a new metric, *Equivalent Random Search (ERS)*, that quantifies the performance of an algorithm on an objective function relative to the expected performance of a random search. The metric is calculated through a non-parametric statistical analysis of the best solutions found by the optimiser over an ensemble of trials. The performance of the optimiser is normalised using the cumulative probability density function of the objective surface, ensuring the metric is accurate for **all** functions, whether single peak or highly multi-modal.

This paper also proposes a new algorithm that is designed specifically to provide reliable and controllable exploration and exploitation of rough landscapes, but with few objective calculations. The algorithm is 'steady-state' rather than generational (it creates one search point at a time) and utilises all the objective calculations made when deciding where to place the next point in the hypercube that defines the search space. An 'ideal' base heuristic used to control the degree of global and local exploration.

Section II describes the ERS metric. Section III describes the 'ideal' heuristic for a global and local search algorithm and section IV describes the full Voronoi Optima Expansion algorithm. Section V describes three example test functions, section VI presents results of the optimisation trials and a comparison with a typical evolutionary approach, and section VII concludes.

II. The Equivalent Random Search Metric

The Equivalent Random Search (ERS) Metric assesses the performance of an algorithm, relative to the expected performance of a random search on the same objective function. The metric gives a direct indication of how many points would be needed in a random search to achieve the same solution quality. The metric can then be compared to the actual number of objective evaluations used by the optimiser in order to assess the effectiveness of the optimisation algorithm. For example, if 1000 objective calculations were used by the optimisation process, but the ERS metric reported that 10,000 points would be needed by a random search to

achieve equivalent results, then the algorithm is performing well. A similar metric based on comparisons to a uniform gridded search has been developed independently as part of the Huygens benchmarking suite [3]. Unfortunately a gridded search is dependent on the topology of the optima, and the interaction with the grid structure is highly complex and problem specific, preventing a simple generalised approach with a robust mathematical basis.

A. Objective Normalisation

To get a reliable assessment, the optimisation process is repeated M times and the best optima results, $Y_i : i = 1 \dots M$, gathered. Each of the M results are transformed via the cumulative probability density function (CDF) of the objective surface, $D(Y)$, to generate the probability of obtaining an objective value *better* than the best value observed in each of the M runs. This normalisation process allows even very rough, multi-modal and deceptive functions to be used for evaluating the optimisers.

The function $D(Y)$ can be generated for any objective function by performing a large uniform-random sample of the objective surface. The number of sample points, K, is typically a few orders of magnitude larger than the number of points generated by the optimisation algorithm. A modest number of samples may be used initially, and then extended if the calculation of the metric indicates insufficient resolution of $D(Y)$.

The set of K objective values from the random sampling process are sorted with the best solution labelled with a rank of 1, and the worst a rank of N, giving a sorted set $\mathcal{R}$. The function $D(Y)$ is then generated by finding the index number of the first member of set $\mathcal{R}$ which is worse than the optimised objective value Y. This index value is divided by K to provide an estimate of the probability of any single random evaluation yielding an objective value result that is less than Y. If a value of Y leads to an index value of 1 being identified as the first value of the set $\mathcal{R}$ that is larger, then the number of points, K, in $\mathcal{R}$ is too small and a larger sample set is required to prevent 'clipping' of the metric. Thus for test functions where the evaluation is expensive, the approximation of $D(Y)$ can be improved progressively until satisfactory resolution is obtained.

It is also possible (but not always trivial) to obtain an analytic solution for the CDF if the equations for the objective function are known. This allows the ERS metric to be calculated quickly, but more importantly, will allow functions that have a very low density of points at the Pareto surface to be analysed without resorting to massive Monte-Carlo searches.

B. Metric Calculation

Once we have calculated $D(Y)$, the probability of there existing a solution better than Y, we can compare the result directly to a random search. For the random search, if we generated a single random point, there would be a probability $D(Y)$ that the point would be better than the optima at Y, and a probability $1 - D(Y)$ that the point will be worse than Y. If we generate N independent random points, then we will not find a better solution than Y with a probability of $(1 - D(Y))^N$. Therefore the probability of finding *at least one* solution better than Y with an N point random search is given by:

$$D'(Y) = 1 - (1 - D(Y))^N \tag{1}$$

For example, if $D(Y) = 1/1000$ and we generated $N = 100$ random points, the probability of at least one of the N solutions being better than Y is $D'(Y) =$9.5%.

Importantly, the new cumulative density function, $D'(Y)$ in equation 1, describes the probability that a random search of N points would find an optimum value better than the value Y. Therefore if we repeated an N-point random search M times, $D'(Y)$ would describe the distribution of the M results. Thus the median value of Y from our M searches would be an approximation of the value of Y necessary to make $D'(Y) = 0.5$. We can exploit this property of $D'(Y)$ to create a metric that uses a simple random search as its reference. As the reference can be described analytically, we can use the metric to **quantify** the performance of any optimisation algorithm on any evaluation function.

The ERS metric is calculated by performing M independent runs of our optimisation algorithm under test, and then exploiting (1) to calculate a value for N, given the observed values for Y from our optimiser. By setting $D'(Y_{median}) = 0.5$, where Y_{median} is chosen to be the median result from our M trials of the optimiser, we can calculate the value for N to give us an equivalent size of random search that we would have to perform to achieve the same median result.

Therefore we can re-arrange (1) (and taking logarithms) to give:

$$N_{median} = \frac{\log(0.5)}{\log(1 - D(Y_{median}))} \tag{2}$$

Ultimately, the calculated value for N_{median} is only an estimate and is subject to sampling error (median is calculated by ranking the M values for Y and finding the central value). If we consider that the probability of the true value of N_{median} being less than the estimate is 0.5, and the probability of the true value being greater is also 0.5, we can describe the error in the estimate of the equivalent random search performance using a binomial distribution of the rank locations with the two probabilities being $p = 0.5$ and $q = 1-0.5$. A binomial distribution can be approximated by a normal distribution when $Mp \geq 5$ and $Mq \geq 5$. Thus a minimum value of $M = 10$ trials will suffice. The variance is given by $Mpq = M/4$ and therefore the standard deviation by $\sigma = \sqrt{M}/2$. The 95% confidence limits of a normal distribution are given by $\pm 1.96\sigma$. Therefore the the upper and lower bounds to give 95% confidence intervals on the estimate of the median correspond to the values of Y from the ranked data in indexes $(M + 1)/2 \pm 1.96\sqrt{M}/2$. Thus given the equivalent random search size from the median value, the confidence limits indicate the range of actual random search sizes that could potentially yield equivalent results to our optimiser.

We can also process other statistics such as the best and worst values of Y and associate them to the best and worst values expected from a random search.

For the random search, the probability given by $D'(Y)^M$ is the probability that M searches will all return values better than Y. Therefore the cumulative probability distribution in (3) is the distribution of probabilities that at least one worse value than Y will be found in M trials. The distribution $D''(Y)$ is therefore the distribution of the worst optimisation results from M trials.

$$D''(Y) = 1 - D'(Y)^M \tag{3}$$

Equation 4 shows (3) and (1) re-arranged to obtain a median estimate and the 95% confidence limits of the worst optimisation value.

$$\begin{aligned} N_{worst_{upper}} &= \frac{\log(1 - \sqrt[M]{0.025})}{\log(1 - D(Y_{worst}))} \\ N_{worst_{median}} &= \frac{\log(1 - \sqrt[M]{0.5})}{\log(1 - D(Y_{worst}))} \\ N_{worst_{lower}} &= \frac{\log(1 - \sqrt[M]{0.975})}{\log(1 - D(Y_{worst}))} \end{aligned} \tag{4}$$

A similar process may be used to establish the estimate of the equivalent random search based on the best results and is given in (5)

$$\begin{aligned} N_{best_{upper}} &= \frac{\log(0.025)}{M\log(1 - D(Y_{best}))} \\ N_{best_{median}} &= \frac{\log(0.5)}{M\log(1 - D(Y_{best}))} \\ N_{best_{lower}} &= \frac{\log(0.975)}{M\log(1 - D(Y_{best}))} \end{aligned} \tag{5}$$

It must be noted that care should be exercised if other statistics are to be used as some are not function independent and may be biased (for example the *mean* will only be unbiased for unimodal functions).

The metric N_{median} in (2) is the size of the random search optimisation that must be performed, that when repeated M times, will obtain a median optima Y_{median}. This metric is an indicator of typical algorithm performance (distance to the true global objective value) and a value of N_{median} larger than the actual number of function evaluations used indicates an optimisation algorithm well suited to the test function.

The metric N_{worst} in (4) is the size of the random search optimisation that must be performed, that when repeated M times, will obtain a worst optima of Y_{worst}. If this metric is larger than N_{median}, then the spread of the inferior solutions from the optimisation process (i.e. variance of inferior solutions) is smaller than the spread that would be obtained by a random search process. This is a desirable feature of optimisation algorithms as it suggests that if only a single run of the optimiser can be performed, there is confidence that a good solution will be identified. If N_{worst} is smaller than N_{median}, the optimiser is prone to premature convergence on poor solutions (highly undesirable).

The metric N_{best} in (5) is the size of the random search optimisation that must be performed, that when repeated M times, will obtain a best optima of Y_{best}. If this metric is larger than N_{median}, the optimisation algorithm is capable of identifying exceptionally good solutions occasionally. If N_{best} is less than N_{median}, the optimisation algorithm rarely finds exceptional solutions. In practice, as long as N_{best} is at least equivalent to the number of evaluations actually performed (i.e the extreme best solution found in M optimisation runs is of similar performance to the extreme best solution found in M random searches of the same number of function evaluations), the algorithm is quite satisfactory, but generally the confidence intervals of N_{best} are very large and it can only be used for indication purposes.

Any situations that give N_{median} lower than the actual number of evaluations used indicate that a random search would have most likely provided better results than from the optimiser.

III. The 'Ideal' Search Heuristic

The idealised heuristic forms the basis of the optimisation algorithm and is:

1) **Exploration:** Next point is the centre of the largest empty convex region.
2) **Exploitation:** Next point is the centre of the largest empty convex region that has a selected *good* point at one vertex.

The aim of the idealised heuristic is to reduce the size of unexplored regions, resulting in uniform search coverage, while still being able to focus on the areas where the local and global optima lie. With only a limited number of function evaluations available, every evaluation must count.

A similar concept based on Voronoi decomposition has been applied to multi-objective optimisation problems [4], and more recently a modified Genetic Algorithm that identifies large hyper-rectangles has been proposed [5].

A. Exploration

The exploration search step of the heuristic identifies the most unexplored region of the search hypercube, and places the next point at the centre of the region. The region could be described in a number of ways, the ideal being to find the largest convex region that will reside between the existing evaluation points.

Section III-E describes the Voronoi method for approximating the most unexplored region.

B. Exploitation

The exploitation step involves first identifying a good point. The method used in this paper is to first triangulate the decision surface, and then identify all solutions which are superior to all their neighbours: i.e. all local minima. The triangulation process can be achieved easily through Delaunay triangulation [6]

Once a point has been selected, the largest unexplored volume that contains the point at its edge is identified, and a new evaluation generated for the point corresponding to the centre of the volume.

C. Exploration versus Exploitation

The two phases of the algorithm, exploration and exploitation, must be controlled in order to provide effective coverage of the decision space. The algorithms must begin with an exploration phase to allow interesting regions to be identified, then the exploitation phase can be applied to refine the regions.

In evolutionary algorithms, the initial population provides pure exploration. The crossover operator also provides initial exploration, but the effect reduces as the algorithm converges. The selective pressure and crossover in subsequent generations provide exploitation, with a low level mutation providing continuing exploration of the decision space throughout the remaining optimisation process.

D. Largest Empty Convex Region

The idealised algorithm in section III relies on being able to identify the largest empty convex region either in the entire search space, or with a chosen point at its edge. The region may be approximated by finding the largest empty hypersphere that can be placed between the existing points. The new point would then be generated at the centre of the hypersphere. Finding the centre of the largest empty hypersphere is still not a trivial problem to solve.

E. Voronoi Diagrams

The *Voronoi diagram* [6], [7] can be used to identify the centre of the largest empty hypersphere. A typical Voronoi Diagram is shown in Fig. 1 with the largest empty circle indicated. The centre of the largest empty circle will always coincide with either a Voronoi vertex, or a vertex generated by the intersection of the Voronoi diagram with the convex hull of the set of points.

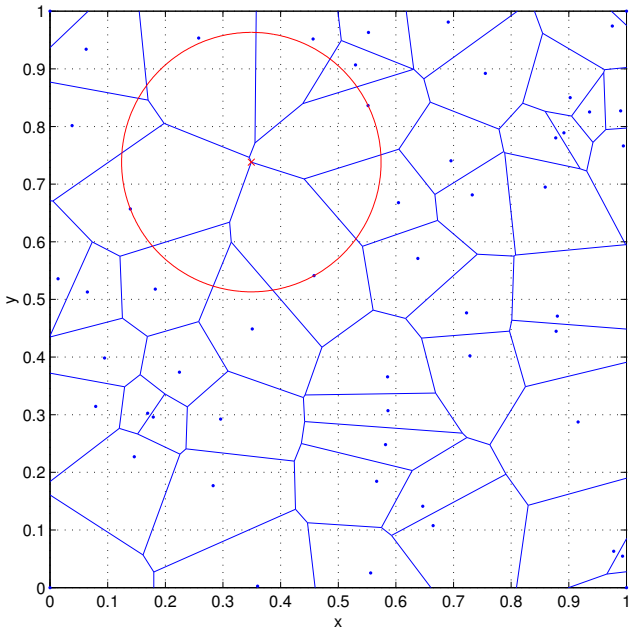

Fig. 1. Example 2D Voronoi diagram showing how centre of largest empty hypersphere lies at a Voronoi vertex

The Voronoi diagram divides a hyperspace containing points into regions, each region surrounding a single point. The space is divided so each point is associated with the region of space closest to it. If $P = p_1, p_2, \ldots, p_n$ is a set of points (or *sites*) in the hypervolume, the volume is partitioned by assigning every point in the volume to its nearest site. All those points assigned to p_i form the Voronoi region $V(p_i)$.

$$V(p_i) = \{x : |p_i - x| \leq |p_j - x|, \forall j \neq i\} \tag{6}$$

The Voronoi diagram is formed as the boundaries of the set of Voronoi regions. The Voronoi edges are points on the hyperplane that lie on the boundaries of the Voronoi regions and will be by definition equidistant from two sites. A Voronoi vertex is formed at the junction of multiple Voronoi hyperplanes. If the point at the centre of each Voronoi cell is joined by lines to the points at the centre of its neighbouring cells, the Delaunay triangulation results and can be used to identify the set of nearest neighbours to a point. The generation of Voronoi diagrams is computationally expensive and so direct use is only really possible for problems with low-dimensionality. Indirect calculation of the Voronoi diagram is still slow but can lead to useful optimisation systems [8]. It is possible to calculate the Voronoi diagram and Delaunay triangulation simultaneously and using an incremental approach [9], one point at a time, which is ideal for the algorithm described in this paper.

To simplify the processing for finding the largest empty hypersphere, a point is placed at each corner of the hypercube in the decision space, simplifying the calculation of the intersection of the Voronoi diagram with the convex hull of the points. The next point is then placed uniformly at random within the hypercube, allowing the Voronoi diagram to be generated and the optimisation process to begin. Thus only one point is placed at random, the structure of the remaining points is deterministic, but biased by the single random selection.

With a 10 dimensional problem, the hypercube has 1024 corners, therefore 1025 points would be required in the initial sampling of the decision space. For many engineering problems that are to be optimised on a single processor, the direct Voronoi approach is limited to problems with less than 10 dimensions due to a rapid expansion of computational complexity with increasing dimensionality. Unfortunately Voronoi decomposition is computationally expensive for even moderate numbers of variables, but the processing is small compared to the cost of evaluating the objective function

IV. Voronoi Optima Expansion

The *Voronoi Optima Expansion (VOE)* algorithm operates by first performing a global search phase for a fixed fraction of the available objective evaluations, and then performing a repeated local expansion of the best performing local minima in the Delaunay triangulation. Figure 2 shows a typical triangulation net after a run of the VOE algorithm with 10% global search.

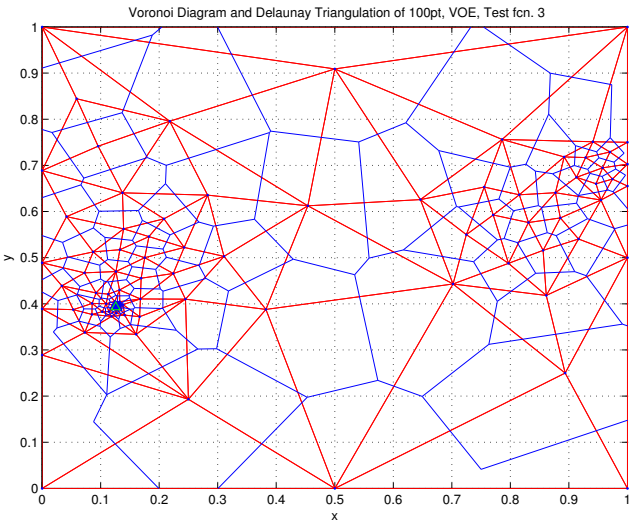

Fig. 2. Example Voronoi Diagram (blue) and Delaunay triangulation (red) of a VOE run on test function 3

At each local expansion iteration, L local optima are investigated with a local search step:

$$L = \max\left(1, \min\left(N_L, \frac{(I_{max} - I)}{S}\right)\right)$$

Where N_L is the actual number of local optima currently identifiable in the triangulation, I is the number of evaluations performed so far, I_{max} is the maximum number of evaluations to be performed, and S is a compression factor that limits the proportion of the available evaluations that may be used for this local search phase.

Each local optima is expanded in turn (starting with the worst performing). When more local optima are present than the expansion limit L, the best L are chosen for expansion. In practice, there will be a maximum of $I/2^D$ local optima at any one time, where D is the number of dimensions of the decision space and I is the number of evaluations performed so far. Initially, the number of local optima is often less than the maximum number of expansions allowed and all the optima get probed. As the remaining number of evaluations reduces, only the better optima get expanded, until in the final phase of the algorithm (approximately the last S evaluations) only the best solution is being expanded, effectively a local hill-climb.

The VOE algorithm is interesting in that most improvements in the objective values occur towards the end of the optimisation run when the algorithm is focussing aggressively on a very small number of optima, unlike many evolutionary approaches which provide very reasonable local solutions within the first few generations.

V. Objective Functions

Three objective functions have been used to examine the performance of the VOE optimisation process and the behaviour of the ERS metric.

A. Basic Test Functions

Two test functions were formulated as maximisation problems. Both having an optimum of 1.0. The first has a single optima, and the second many local optima. Both were formulated for a chromosome using two real-valued genes. Test function 1 (spike), described by equation 7 has a single narrow central spike as the global optimum point. Test function 2 (rings), described by equation 9 has a narrow central spike as the global optimum point and then a series of concentric ridges. The function is highly deceptive and it is very difficult for optimisation algorithms to identify the central optima.

1) Test function 1 - 'spike': Function 7 is simple and is used to form a 'lower' bound of complexity for the demonstration of the metric. The function has a very sharp 'spike' in the centre that requires a very large random search to identify accurately. Equation 8 is the analytic CDF for the spike function up to a probability of approximately 80% which is ample for even very small random searches.

$$Y = 1 - \sqrt[4]{x^2 + y^2}, \quad -10 \le x, y \le 10 \tag{7}$$

$$D(Y) = \frac{\pi(1-Y)^4}{400}, \quad 0 \le \sqrt{x^2 + y^2} \le 10 \tag{8}$$

2) Test function 2 - 'rings': Equation 9 shows a 2 dimensional multi-modal test function that has a central spike to be maximised, and 3 local optima surrounding it as 'rings', rather than discrete points. This objective function appears simple but can cause significant problems to optimisers due to the local objectives being plateaus widely distributed in decision space and forming very significant attractors in the objective domain. The central spike is in practice difficult to identify compared to the relative ease of identifying the local optima. Equation 10 shows the analytic Cumulative Probability Density Function (valid for greater than 95% of solutions).

$$\begin{aligned} Y &= \sum_{r=0}^{3} 0.998^r \exp\left[\frac{-(d - 4\pi r/5)^2}{0.3^2(r+1)}\right] \\ d &= \sqrt{\sum_{i=1}^{2} x_i^2}\ , \quad -10 \le x_i \le 10 \end{aligned} \tag{9}$$

$$\begin{aligned} D(Y) &= \frac{\pi}{(20^2)}\left[\left(-0.3^2 \ln(Y)\right) + \sum_{r=1}^{3}\left[(q \le 0.998^r)\right.\right. \\ &\quad \left.\left.\left[\left(\frac{4r\pi}{5} + q\right)^2 - \left(\frac{4r\pi}{5} - q\right)^2\right]\right]\right] \\ q &= \sqrt{-0.3^2(r+1)\ln\left(\frac{Y}{0.998^r}\right)} \end{aligned} \tag{10}$$

B. Fractal Surface Test Function

The third test function is a very rough fractal landscape formulated as a minimisation problem and is function 20_103 of the Huygens benchmarking suite [3] and is shown in figure 3. The global optima is unknown but the best found so

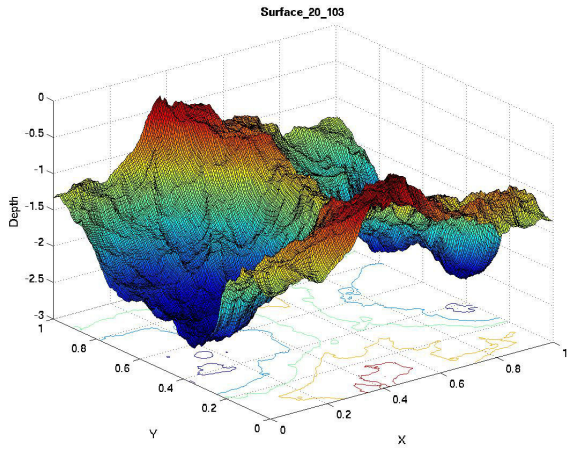

Fig. 3. Test objective surface of Huygens benchmark moon 20_103

far is -2.6114 at $[0.1006\ 0.4089]$. The function is formulated for a chromosome using two real valued genes, each lying in the range $[0, 1]$. Figure 4 shows the Cumulative Probability Density Function of the function.

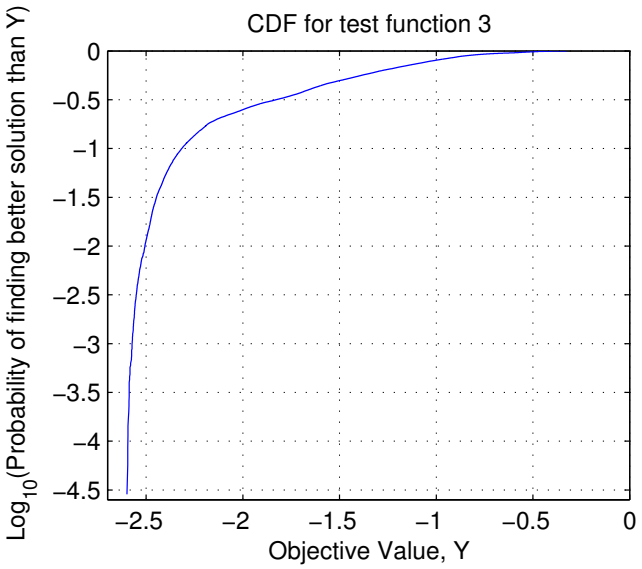

Fig. 4. Cumulative Probability Density Function of Huygens function 20_103 (based on $35,000$ evaluations)

VI. Results

The Voronoi Optima Expansion algorithm was applied to each of the three test functions, and as a comparison, Differential Evolution (DE) [10] (with rank selection) was also applied with the same limit on objective function evaluations in each case.

For each test function, the effect of global vs. local exploration was investigated by repeating the trials but with differing proportions of global and local search. For the VOE algorithm, linear steps were created. For DE the population size and number of generations were varied in order to achieve a similar contrast between local and global search. In DE linear steps were not possible. For both algorithms the case of total global search was evaluated (e.g. for DE a population of 100, 1 generation).

For each experiment to calculate the ERS metrics, 20 trials were performed for test functions 1 & 2, and 10 trials were performed for test function 3. The VOE algorithm is tuned by two parameters: The proportion of solutions used for global search, and the compression factor for the local search. A compression factor of $S = 16$ appears to be useful and has been applied in all the trials, with the ratio of global:local search being varied parametrically. The DE algorithm is tuned by the population size and number of generations, the search scaling factor F, and the crossover rate C. The population size and number of generations are varied in each experiment, and the values $F = 0.7$ and $C = 0.5$ were found to be a very good choice (there was little change in the algorithm performance with variation in the two parameters).

A. Test function 1 - 'spike'

As the function is so simple,100 objective evaluations for each trial proved more than adequate for both optimisers to identify good solutions. Figure 5 shows the ERS metrics for the VOE algorithm and figure 6 shows the ERS metrics for the DE algorithm. As expected in both plots, when the global search approaches 100%, the ERS matches that expected from a random search.

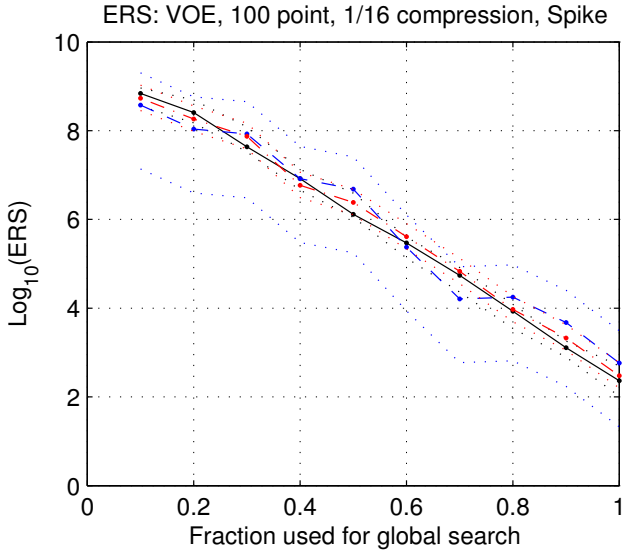

Fig. 5. ERS metric for VOE algorithm with 100 evaluations and 20 trials. Solid line is N_{Median}, dashed line is N_{Best} and dash-dot is N_{Worst}. Dotted lines show associated 95% confidence intervals

Figure 5 shows $\log_{10}(ERS)$ plotted versus the ratio of global to local search. It is clear from the VOE plot in figure 5 that small amounts of global search and large amounts of local search give best performance (approximately 10^9 evaluations would be required of a random search to match the performance of VOE with 100 evaluations).

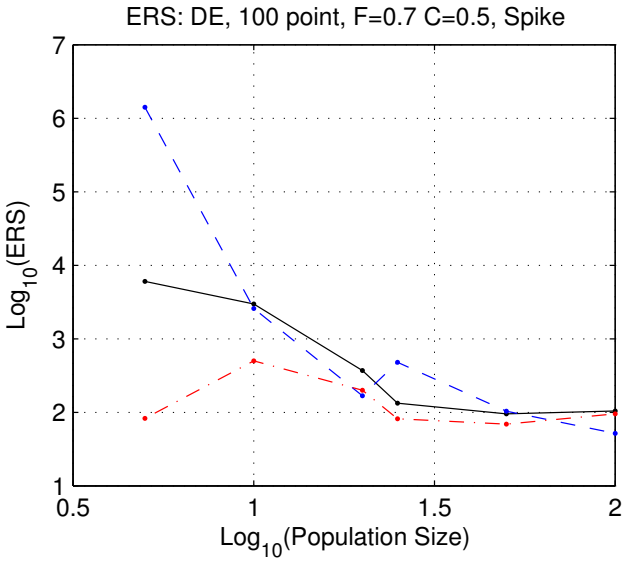

Fig. 6. ERS metric for DE algorithm with 100 evaluations and 20 trials. Solid line is N_{Median}, dashed line is N_{Best} and dash-dot is N_{Worst}

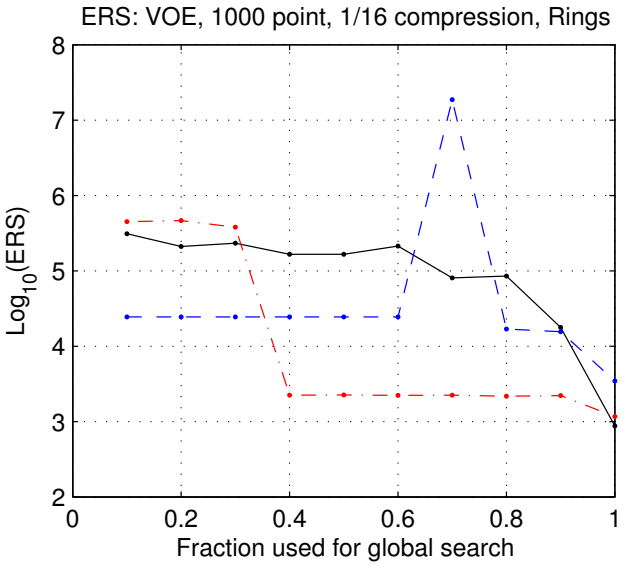

Fig. 7. ERS metric for VOE algorithm with 1000 evaluations and 20 trials. Solid line is N_{Median}, dashed line is N_{Best} and dash-dot is N_{Worst}

Figure 6 shows the performance of Differential Evolution. A population size of 10 with 10 generations gave very acceptable results with a random search of around 4000 points being needed to match the performance. The results are as anticipated as DE is self-adaptive and does require a decent population size in order to be most effective. The comparison of N_{Worst} and N_{Median} indicate that the performance of DE is slightly erratic on this test function.

B. Test function 2 - 'rings'

As the function is difficult, 1000 objective evaluations for each trial proved necessary for both optimisers to identify good solutions. With only 100 runs available, neither algorithm proved any better than a random search and demonstrates that it is important to test the baseline performance of each algorithm. With the 1000 points, the VOE algorithm managed to identify the true global optima once out of a total of 200 runs (DE failed to identify the true global). Figure 7 shows the ERS metrics for the VOE algorithm and figure 8 shows the ERS metrics for the DE algorithm.

It is clear from the VOE plot in figure 7 that a wide variation in the amount of global search made little difference to the median performance, with 10% global search being a good compromise (approximately $300,000$ evaluations would be required of a random search to match the performance of VOE with 1000 evaluations). The line for N_{Worst} is very satisfactory and as it is better than N_{Median} showing that the VOE algorithm tends to provide good answers repeatedly on this problem.

Figure 8 shows the performance of Differential Evolution. A population size of 20 with 50 generations gave acceptable results with a random search of around 6000 points being needed to match the performance of the actual 1000 evaluations. The performance of N_{Worst} and N_{Median} indicate that the performance of DE is again mildly erratic on this

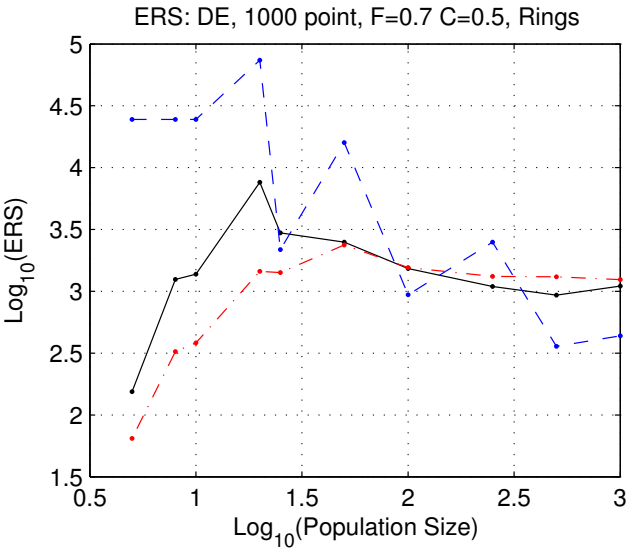

Fig. 8. ERS metric for DE algorithm with 1000 evaluations and 20 trials. Solid line is N_{Median}, dashed line is N_{Best} and dash-dot is N_{Worst}

test function, but it is clear that for many combinations of population size and number of generations, DE is not significantly better than a simple random search.

C. Test function 3 - Huygen benchmark

As the test function is accessed via the web, 100 objective evaluations for each trial proved to be a sensible upper limit. Figure 9 shows the ERS metrics for the VOE algorithm and figure 10 shows the ERS metrics for the DE algorithm.

It is clear from the VOE plot in figure 9 that the algorithm is again quite robust to the amount of global search, with 10% or 20% global search again being a good compromise (approximately $32,000$ evaluations would be required of a random search to match the performance of VOE with 100

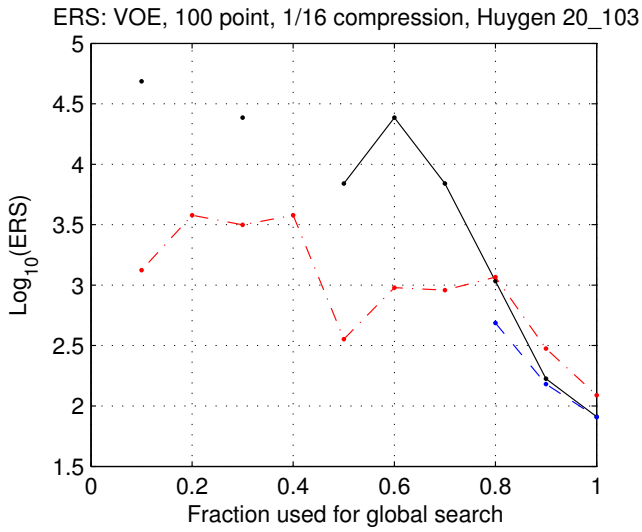

Fig. 9. ERS metric for VOE algorithm with 100 evaluations and 10 trials. Solid line is N_{Median}, dashed line is N_{Best} and dash-dot is N_{Worst}. Some points could not be evaluated due to the optimisation results being better than the best held in the CDF.

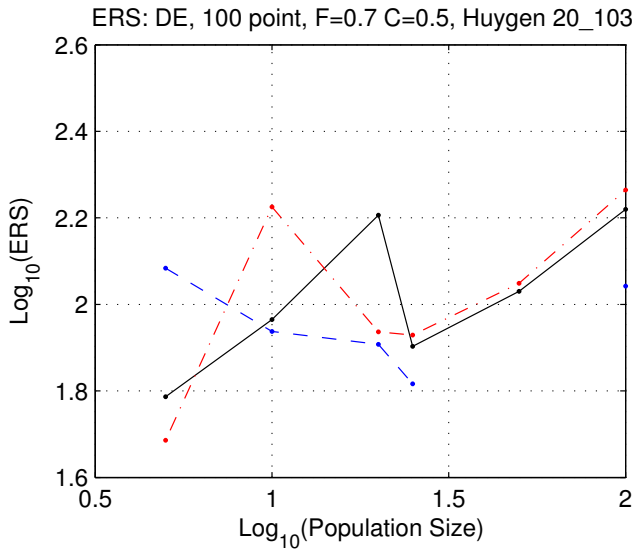

Fig. 10. ERS metric for DE algorithm with 1000 evaluations and 10 trials. Solid line is N_{Median}, dashed line is N_{Best} and dash-dot is N_{Worst}

evaluations). The lines for N_{Best} and N_{Worst} are satisfactory but are both generally lower than N_{Median} showing that the VOE algorithm with 100 points is more erratic than a random search with $32,000$ points would be. With $35,000$ samples being used to create the cumulative probability density function, the N_{Best} line and N_{Median} lines have been clipped for low ratios of global search.

Figure 10 shows the performance of Differential Evolution. A population size of 20 with 5 generations gave results with a random search of around 160 points being needed to match the performance of the actual 100 evaluations. The performance of N_{Worst} and N_{Best} generally match N_{Median} showing reasonably robust behaviour, although it is not significantly better than a random search (the results are all within the confidence intervals expected from a 100 point random search). Generally we can see from N_{Median} that DE is not suited to operation on this test function with only 100 points. When 1000 points are used, DE is much more effective (yet not as effective as VOE with 1000 points).

VII. Conclusions

This paper has introduced a *fundamental* metric with strong mathematical foundations for algorithm benchmarking that allows the performance of different algorithms to be quantified and compared directly and analytically. The metric reveals not only the raw performance of the optimiser, but also the reliability of the solutions the optimiser is likely to generate.

The paper has applied the concept of the idealised search heuristic to create a powerful new optimisation algorithm that is designed to provide reliable performance for problems where very few objective calculations can be performed. The VOE algorithm allows full independent control over the exploration and exploitation phases of the search, yet requires minimal tuning. The research demonstrates how the new algorithm exploits the information from all the evaluations performed to give much more structure to the location of trial points when compared to a typical evolutionary approach.

VIII. Acknowledgements

I would like to thank Cara MacNish, School of Computer Science & Software Engineering, The University of Western Australia for her help with the Huygens benchmark data sets.

References

[1] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001. ISBN 0-471-87339-X.

[2] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, 9(1):3–12, 2005.

[3] Cara MacNish. Huygens search and optimisation benchmarking suite. http://karri.csse.uwa.edu.au/cara/huygens/cec2006.php. Last accessed 17/1/2006.

[4] Evan. J. Hughes. Multi-objective binary search optimisation. In *Second International Conference on Evolutionary Multi-Criterion Optimisation, EMO'03*, pages 102–117, Faro, Portugal, 8-11 April 2003. Springer LNCS.

[5] Chongshan Zhang and Khaled Rasheed. Improving GA search reliability using maximal hyper-rectangle analysis. In *The Genetic and Evolutionary Computation Conference (GECCO'2005)*, pages 1185 – 1192, Washington DC, USA, 2005. ACM Press.

[6] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1993. ISBN 0-521-44592-2.

[7] Franz Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surveys*, 23:345–405, 1991.

[8] Malcolm Sambridge. Geophysical inversion with a neighbourhood algorithm – I. Searching a parameter space. *International Journal of Geophysics*, 138:479–494, 1999.

[9] Jonathan Richard Shewchuk. Updating and constructing constrained delaunay and constrained regular triangulations by flips. In *Nineteenth Annual Symposium on Computational Geometry*, pages 181–190, San Diego, California, June 2003. Association for Computing Machinery.

[10] Rainer Storn and Kenneth Price. Differential Evolution- a simple and effective adaptive scheme for global optimization over continuous spaces. http://http.ICSI.Berkeley.edu/∼storn/code.html. last accessed Jan 2006.