Many-Objective Directed Evolutionary Line Search

Evan J. Hughes*
Department of Informatics and Systems Engineering
Cranfield University
Shrivenham, Swindon, UK, SN6 8LA
e.j.hughes@cranfield.ac.uk

ABSTRACT

Algorithms capable of performing efficient and controllable many objective optimisation are becoming more necessary as the complexity of optimisation problems to be solved increases. This paper describes a new algorithm that combines elements of traditional gradient based optimisation methods along with a powerful many-objective capable search process. The algorithm exploits the directed line search (such as Golden Section Search) procedures found in many single-objective gradient based algorithms in order to both explore and exploit features in the optimisation landscape. The target vector and aggregation methods used in the MSOPS algorithm have been employed to provide effective and controllable many-objective optimisation, especially suited to close interaction with a designer where it is often desired to target specific regions of the Pareto front.

The Many Objective Directed Evolutionary Line Search (MODELS) algorithm is demonstrated on a constrained function with a concave Pareto front in up to 20 dimensions and is shown to outperform existing optimisers, some of which are known to perform well for many-objective problems.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Algorithms

Keywords

Multi-Objective, Many-Objective, Line Search, Golden Section, Objective Boundary Identification

*Dr Evan J. Hughes is a Senior Lecturer in the Sensors Group

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

As the optimisation of large problems in commerce and engineering becomes more widespread, the need for optimisation algorithms that can solve for multiple optimisation criteria simultaneously is increasing. Although excellent algorithms exist for optimising both single and small numbers of objectives [3], issues with the simultaneous optimisation of many criteria (4+) causes a significant number of the existing algorithms to under perform [6, 9].

Evolutionary methods have been developed that exploit approximate gradient information from the objective function, however the question arises as to whether a method can be developed that exploits local gradient information, but for optimising many objectives simultaneously and crucially, allows effective targeting of specific regions of the search space by a designer.

Algorithms for multi-objective optimisation that combine evolutionary methods and local search have been studied [11, 13, 10] however they either rely on calculating the full local gradient, rely on Pareto ranking methods which do not scale well with many objectives or do not allow full control of the search regions by a designer.

In developing an algorithm for many-objective optimisation where a designer can have significant control of the search directions, the behaviour of a single objective 'stochastic gradient optimisation' algorithm has been studied to assess the trade-off between population size, gradient estimation and landscape search. The results have been applied to create an effective algorithm for solving many-objective problems.

Section 2 describes a single objective stochastic search algorithm that can exploit local gradient and optimisation surface structure and forms the basis of the many-objective algorithm. Section 3 describes the many objective algorithm in detail and sections 4 and 5 describe the results of the algorithm performance experiments on problems with up to 20 objectives. Finally section 6 concludes.

2. LINE SEARCH METHODS

2.1 Introduction

In most conventional gradient-based optimisation methods, the algorithms first evaluate the local gradient of the parameter space of the function to be solved at a random starting location in order to identify a search direction ('downhill' for minimisation, 'uphill' for maximisation) and then perform a *line-search* in the identified search direction. The line-search traverses the parameter space, attempting to iden-

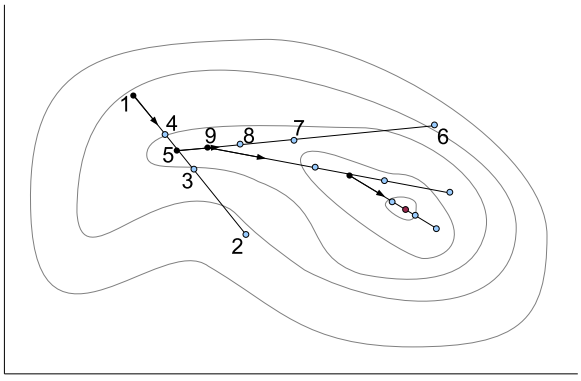

Figure 1: Gradient-based line-search process showing 4 iterations of gradient calculations and line searches, each line search with 4 binary-chop evaluations (17 evaluation points in total). The order of the first 9 evaluations are indicated, with gradients being calculated for evaluation 1, 5, 9 and 13

tify the point on the search line that has the minimum (or maximum) objective value. The process of gradient calculation and line search is repeated, but now from the last identified minima. The process is illustrated in Figure 1.

When gradient information is available, the size and direction of the steps in the line search process may be made adaptive using approaches such as Newton's method [4]. For problems where evolutionary methods are often employed, such as problems with discontinuities in the gradient, a more stochastic line search process may be employed where algorithm robustness is traded against the amount of local landscape structure that is used.

This section first describes the generation of the search direction and then line search methods. The section then describes how constraints are handled in the search process and how the length of line to search is generated. Finally the use of a population of search points and directions is investigated.

2.2 Generation of Search Direction

In classical gradient-based optimisation algorithms, if an analytic description of the gradient is available, then the calculation often requires approximately the same time resources as an objective calculation. If an analytic description of the gradient is not available, then the local gradient may be approximated numerically. For a given parameter vector $\vec{X}$, each parameter element, x_i, is perturbed in turn by a small quantity Δx_i and the objective function(s) evaluated. Given the changes in objective value and the parameter space step sizes taken, the component of the local gradient in parameter i may be approximated as $\nabla f_i(x)$ using (1), where Δx_i is a small value used to create as step in the direction of parameter i only. Therefore for a problem that has N_x parameters to optimise, N_x objective evaluations will be required to establish the first-order approximation of the local gradient of a single parameter vector $\vec{X}$.

$$\nabla f_i(x) \approx \frac{f(x) - f(x + \Delta x_i)}{\Delta x_i} \quad (1)$$

Although taking $\vec{V}$ as being the direction of steepest descent (minimisation) is a good choice for simple unimodal funtions, there are many cases where the optima forms a 'long valley' and the steepest descent approach can be deceived into excessive 'zig-zag' behaviour. The problem can be seen in Figure 1 where the first step is directed along the steepest gradient, however due to the contours of the fitness landscape, the first step is downward and to the right, rather than towards the true optima.

As the proposed search algorithm applies aggregation functions to the many objectives, the problem is transformed into a set of single objective optimisation problems. In general, as long as $\vec{V}$ is directed downhill (for minimisation), the optimisation process may proceed in a satisfactory manner. A question arises though: in a multi-modal environment where the local gradient direction could be reversed relative to the global optima, what is the trade-off between the number of function evaluations used to calculate the gradient compared to other more approximate choices for the direction to search next?

For problems with many decision parameters to optimise, the computational overhead of full gradient calculations can be very large. One approach to reducing computational burden is to generate K random offset vectors $\Delta\vec{X}_i$ and then evaluate the K fitness values from $\vec{X}+\Delta\vec{X}_i, \forall i \in K$, giving K approximations of the local gradient. The direction that provides the steepest search direction can then be chosen for $\vec{V}$ (noting that the direction of $-\vec{V}$ should be used if the gradient indicated that the chosen $\vec{V}$ was uphill if minimising etc.).

An alternative method that has been implemented within the algorithm is to exploit the behaviour of the population in a manner similar to crossover in evolutionary algorithms. For a percentage of the time (80% in the algorithms used for many-objective optimisation), $\vec{V}$ is chosen based on an estimate of the local gradient and for the remaining time, the search is biased towards other members of the population.

The approach is to choose C other population members and generate the difference vectors between the current point of interest and the selected population members. Currently the choice of C members is uniformly at random from the population, but bias could be introduced towards 'fitter' solutions if desired. The distance to each of the C other solutions is calculated and the closest chosen. The direction vector to the chosen population member is normalised to unit length and used to represent the search direction $\vec{V}$. By choosing the closest population member from a random choice of C, if the objective surface is multimodal and the population fragments into different local optima, then $\vec{V}$ is more likely to be representative of good search directions in the niche local to the current solution being explored. A choice of C=4 has been employed to allow a degree of tolerance to multiple optima; the larger C, the less is the effect of the steering behaviour.

2.3 Golden Section Search

A simple approach to the line search is to place the first evaluation point at some distance from the point where the search direction has been calculated. In Figure 1, point 1 is an example of the first random start point. The gradient is calculated and point 2 is placed at some distance away from point 1 in the direction of the search, indicated by the arrow. A third point is now placed somewhere along the line, between points 1 and 2.

Once the first 3 points have been placed, the line is now

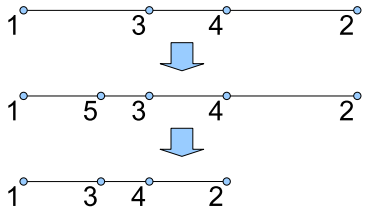

Figure 2: Golden section line search. The end-points 1&2 are placed and then the bisectors 3&4 added. If end point 2 is chosen to be removed, the new point 5 is added, maintaining the golden ratio. The points may now be re-named and the process repeated iteratively

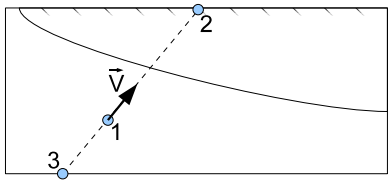

Figure 3: Impact of constraints in parameter space: point 1 is the current solution and $\vec{V}$ is the chosen search direction. Point 2 is the furthest point in the direction of $\vec{V}$ that still satisfies all of the implicit constraints. Point 3 is the last point on the line in the reverse direction of $\vec{V}$ that satisfies the implicit constraints. The hatched region shows an explicit constraint.

divided into two segments. One of the segments is now chosen to be sub-divided further. The process of subdivision can be repeated either until there is little change in the objective values calculated, or a given number of evaluations have been performed. A new search direction can now be generated and the process repeated.

A structured search method is to use a Golden-Section process [11]. The golden section search is based on the golden ratio which is determined by the scale factor of $\tau = (1+\sqrt{5})/2$. The ratio is the solution to $1/x = x-1$ and $\tau \approx 1.618$ and $1/\tau \approx 0.618$. The method operates as shown in Figure 2 by placing the two line search end points (x_1 and x_2) and then inserting two more points, their location based on the golden ratio as described in (2).

$$\begin{aligned} x_3 &= x_2-(x_2-x_1)/\tau \\ x_4 &= x_1+(x_2-x_1)/\tau \end{aligned} \tag{2}$$

The search progresses by identifying which of the end-points (x_1 or x_2) is the worst performing; the worst performing point is then 'removed' and a new interval point added. If end-point 2 was removed the new point would be added between points x_1 and x_3 as indicated by location 5 in Figure 2, x_4 would become the 'new' x_2, x_3 would become the 'new' x_4 and the added point would be labelled as x_3. The ratios between the four points is maintained, even though only a single new point is added.

2.4 Constraint Handling

There are two forms of constraints that can be considered in the search process. The first form are implicit constraints which form the limits of each of the parameters and ultimately define the parameter search volume; the second form of constraints are explicit constraints, generally in the form of inequalities to be satisfied.

The implicit constraints may be handled easily as shown in Figure 3 by projecting the search line in the chosen direction $\vec{V}$ but only until an implicit constraint on the parameter range is violated. Points 2 and 3 in the Figure show the limits on line search in both the forward and reverse direction defined by $\vec{V}$. Figure 3 also indicates an explicit constraint as the hatched region. Often it is not practical to calculate the intersection of the search direction with the explicit constraints and hence they must form part of the optimisation process.

The explicit inequality constraints are handled during the line search process by selecting the line end points for 're-moval' based on (a variant of a penalty based on feasability [2]):

1. If both points are constrained by an inequality, then the point which violates the inequality the least is superior.

2. If one point violates an inequality and the other does not, the non-constrained solution is considered superior

3. If neither point are constrained then the point with the best fitness (lowest for minimisation, highest for maximisation) is considered superior.

2.5 Determining Length of Line Search

When no explicit gradient information is used in order to stop the line search process, it is most convenient to perform each line search for a fixed number of function evaluations. A difficulty arises in that although a search direction $\vec{V}$ can be determined, the distance to search over is unclear. A strategy has been adopted that sets the length of the search line to lie between the minimum distance which corresponds to the start point $\vec{X}$ and the maximum distance as determined where the search line first violates an implicit constraint on a parameter (e.g. point 2 in Figure 3).

In the early iterations (generations) of the algorithm, exploration is desired therefore the line search should investigate solutions ranging from the minimum up to the maximum distance as defined by the implicit constraints. As the algorithm progresses, it is important to restrict the distance of the line search closer to $\vec{X}$ in order to allow the best identified solution to be fine-tuned.

The length of the line to search is first scaled so that $l \in [0, 1]$ corresponds to the range where l=0 represents the location of $\vec{X}$ and l=1 corresponds to the intersection of the search line with the implicit constraint boundary of the parameter space (i.e. point 2 in Figure 3). The furthest point on the line search is then calculated as $\vec{X} + lD_C\vec{V}$ where D_C is the Euclidian distance from the point $\vec{X}$ to the boundary of parameter space in the direction of unit vector $\vec{V}$.

The desired line length l is drawn from a Beta distribution (as a special case of a Dirichlet Distribution [1]) using a ratio of Gamma distributed variables as described in (3), where n is the number of the current algorithm iteration, N_{IT} is the maximum number of iterations and Gamma$(\beta, 1)$ is a

random variable drawn from a Gamma distribution with scale=1 and shape of β.

$$\begin{aligned} \beta = & \quad 2\left(1-\frac{n-1}{N_{IT}}\right) \\ \gamma_1 = & \quad \text{Gamma}(\beta, 1) \\ \gamma_2 = & \quad \text{Gamma}(1, 1) \\ l = & \quad \frac{\gamma_1}{\gamma_1+\gamma_2} \end{aligned} \tag{3}$$

When n=1, the line length l is drawn from a reverse-triangular distribution (probability density rising from zero at l=0 to density of 2 at l=1), leading to many long search lines in the early phases of the algorithm. Half way through the optimisation run (n=$N_{IT}/2$) l is drawn from a uniform distribution. As n increases thereafter, the probability density function that determines l appears with a shape similar to an exponential distribution, leading to shorter line searches and therefore exploitation of the local environment.

2.6 Population Vs. Gradient Vs. Line Search

A simple single-objective optimisation algorithm that exploits the line-search process whilst performing a global optimisation of multi-modal functions is:

1. Generate P random parameter vectors $\vec{x_i}$: the set $Q = \{\vec{x_i}, \forall i \in [1, P]\}$
2. Evaluate the parameter vectors in set Q, $O = \{f(\vec{x_i}), \forall \vec{x_i} \in Q\}$
3. For $j = 1 \ldots P$
   (a) Generate and evaluate K random offset vectors: $z = f(\vec{x_j}+\Delta\vec{X_i}), \forall i \in K$
   (b) Set $\vec{V_j}$ as the unit length vector in the direction $\Delta\vec{X}$ that gave the best value for z (if z is uphill, for minimisation set $\vec{V_j} = -\vec{V_j}$)
   (c) Draw a random line search length l from (3)
   (d) Perform line search in direction of $\vec{V_j}$ starting at point $\vec{x_j}$ for a length of l. Record the location and fitness of all N_L points evaluated as part of line search.
4. From the set of $P\times(K+N_L)$ evaluations that were performed during the gradient estimation and line searches for the population, take the best P and use to replace the current population in set Q and corresponding objective results in set O.
5. loop back to item 3 for N_{IT} iterations until the desired total number of function evaluations have been performed.
6. Return the sets Q as the best identified solutions.

The total number of fitness evaluations is $N_{EV} = P(1 + N_{IT}(K+N_L))$, therefore the question arises as to how the balance between the population size P, number of points for the gradient estimation K and the number of points placed during the line search N_L should be chosen. As the three parameters are varied, the number of search iterations N_{IT} (generations) will vary accordingly.

An experiment was conducted initially to establish the best choice for K vs. N_L, with the population size P, the number of search iterations N_{IT} and the total number of evaluations held fixed. The experiment was conducted using the modified Griewank function [8] over a range of parameter vector sizes from 2 to 100 dimensions. The experiments were also conducted for a range of population sizes and maximum number of function evaluations. The results were consistent in that the best results were obtained when $K \leq N_x$, often with $K = 5$ performing well for up to the 100 dimensional problem tested. For some very rough test problems studied, $K = 0$ (i.e. choosing $\vec{V}$ as a random direction) was superior, maximising N_L the number of points on the line search. The result was interesting in suggesting that it is often better to expend function evaluations as part of the line search, rather than to spend time in finding an apparently good search direction. With a single objective and a random search direction, there is still a 50% probability that $\vec{V}$ will be in a 'downhill' direction and in the early phases of the algorithm, the random search direction can be considered similar to mutation in a classic evolutionary algorithm and acts as an exploration operator.

Generally for non-trivial problems the algorithmically simpler choice of K=0 provided adequate performance and for simplicity of tuning and algorithm comparison K=0 has been selected for use in the many-objective algorithm evaluation.

A second experiment was conducted to study the trade-off between the population size and the number of points used on the line search. For very small numbers of total function evaluations (in the range of $N_{EV} \leq 1000$ function evaluations in total), a small population size of $P \leq 10$ points was found to be beneficial for many problems. For $N_{EV} > 1000$, a population size of $P = 1$ was found to be superior on all functions tested, even where the function is highly multimodal. The use of P=1 and small value for K provides effectively a stochastic version of the classic gradient based algorithms, however the approach forms a very capable global optimiser, rather than just the local optimisation of the classical algorithms.

For the line search, the optimal number of points N_L varied, but was often in the range of 10 to 20 points. For all of the many-objective experiments, 10 points were used for the line search process.

3. MANY OBJECTIVE ALGORITHM

3.1 Introduction

For many-objective optimisation, the high dimensional objective space causes issues with many fitness ranking approaches as not only does the percentage of non-dominated solutions in any given population increase rapidly with increasing dimensionality, but also even large population sizes can only form a very sparse sample of the objective region [9, 6, 12]. The result is often that the optimisation algorithms lose their effectiveness at progressing the solutions towards the true Pareto front.

We can define a many-objective problem as:

$$\text{Find } \vec{X} \text{ that minimises } \vec{F}(\vec{X}) = [f_1(\vec{X}), \ldots, f_{N_{Obj}}(\vec{X})]$$

subject to:

$$\vec{g}(\vec{X}) \leq 0 := [g_q(\vec{X}) \leq 0, \forall q \in [1, N_g]] \tag{4}$$

$$\vec{h}(\vec{X}) = 0 := [h_w(\vec{X}) = 0, \forall w \in [1, N_h]] \tag{5}$$

where $f()$ is one of N_{Obj} objective functions, $g()$ is one of

N_g inequality constraints and $h()$ is one of N_h equality constraints. In the remainder of this paper it is assumed that all equality constraints have been converted to inequality constraints by defining $g(\vec{X}) = h(\vec{X}) - \epsilon$, where ϵ is a small tolerance value.

In the many-objective space, the local gradient is now a matrix of partial derivatives defined by the Jacobian evaluated at point $\vec{x}$.

$$J_{\vec{x}} = \begin{bmatrix} \frac{\partial f_1(\vec{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\vec{x})}{\partial x_{N_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{N_D}(\vec{x})}{\partial x_1} & \cdots & \frac{\partial f_{N_D}(\vec{x})}{\partial x_{N_x}} \end{bmatrix} \quad (6)$$

If an analytic solution exists for $J_{\vec{x}}$ then it can be exploited to generate a search direction $\vec{V}$ by optimising $\vec{V}$ to maximise the gradient of the aggregated metric value, m in a similar manner to the method described in [11]. The aggregated value m is used to form the many-objective process using the relationship in (7) where the function $A(\cdot)$ is the aggregation function in a search direction determined by the designer-supplied vector $\vec{W}$.

$$m = A(J_{\vec{x}}\vec{V}^T + \vec{F}(\vec{x}), \vec{W}) \quad (7)$$

3.2 Algorithmic Approach

To convert the single objective stochastic line search optimiser to solve many-objective problems, methods must be incorporated to:

1. Drive a set of parameter vectors towards the true Pareto Front.

2. Produce a useful sampling of the objective space that can be controlled by a designer and focussed easily into regions of interest.

The strategy that has been taken is based on the MSOPS and MSOPS-II algorithms [5, 7] which use a set of 'target vectors' to direct and control the search process. The target vectors can either be supplied and controlled *a-priori* by a designer, or established on-line during the optimisation run. The distribution of the final solutions in the objective space is determined by the spatial distribution of the target vectors.

For the analysis of the algorithm in this paper, the set of target vectors has been defined for convenience *a-priori* before each optimisation run, rather than being established dynamically during each run. Each target vector is coupled with at least one aggregation function that turns each objective vector into a scalar value for use in the optimisation process. The choice of particular aggregation functions should be made to best suit the characteristics of the problem being optimised.

The algorithm operates by treating each target vector and associated aggregation function as a single-objective optimisation process and tracks the parameter vector $\vec{X}$ that best optimises the aggregated objective values in the target vector direction. The 'population' is now the set formed by the best solutions for each of the target vector/ aggregation function combinations. The key element that distinguishes the approach from a full optimisation of each target vector in turn is that the results from each line search that is performed are used to update **all** of the target vector best solutions, providing the benefit of a population based approach and allowing an estimate of the full Pareto front to be generated in a single run of the optimiser.

3.3 Aggregation Methods

Each of the aggregated optimisations is directed by its own vector of weights, or target vector. Thus the algorithm uses a set of target vectors to search in parallel for solution points that lie on the Pareto front. It is also possible to combine searches in different directions, with different reference points, searches using different aggregation functions, all within a single optimisation run. A key advantage is the algorithm does not rely on Pareto ranking to provide selective pressure.

As aggregation methods (eg. weighted min-max, ϵ-constraint, VADS [5], goal attainment etc.) are very simple to process, the calculation of each of the performance metrics is fast. The different properties of different aggregation methods can be exploited easily and for the algorithm used in this paper, one aggregation method has been chosen to be used in each run of the optimiser, although the algorithm can use many simultaneously. Weighted Min-Max has been selected as it is aggressive at targeting points that lie on the Pareto front.

3.3.1 Weighted Min-Max

The weighted min-max score of N_{obj} objectives is calculated using (8), where w_i is the weight of the i^{th} objective f_i, m is the aggregated fitness value and Z_i is the i^{th} component of a utopia reference point.

$$m = \max_{i=1}^{N_{obj}}(w_i(f_i - Z_i)) \quad (8)$$

Weighted min-max is able to generate points on both convex and concave Pareto fronts. The weight vector corresponds to a point on the Pareto front in the true direction given by the vector $\vec{T} = [1/w_1,\ 1/w_2,\ \ldots]$. The Weighted Min-Max metric is capable of identifying any point that lies on the Pareto front, irrespective of whether the local topology is concave or convex.

Multiple aggregation methods may be used during the same optimisation process such as Vector Angle Distance Scaling (VADS) [5]. The VADS metric is designed specifically for identifying the *Objective Front*, rather than just the Pareto front and complements the weighted Min-Max approach well.

3.4 Algorithm Structure

The structure of the Many-Objective algorithm is similar in nature to the single objective algorithm in section 2.6. There are 4 key differences present however:

1. The size of the working population is determined by the number of target vector/ aggregation function search definitions

2. The size P of initial random population $\vec{X}$ vectors is independent of the size of the working population and may be as small as P=1, a single point if desired.

3. For each line search, the vectors of objective functions are aggregated using the target vector and aggregation function of the current search direction and the aggregated result m is minimised.

4. The search direction $\vec{V}$ is determined either at random for 80% of the time, or biased towards other members of the working population for the remainder.

The outline of the algorithms is as follows:

1. Generate N_T search vectors $\vec{t_i}$:
   the set $T = \{\vec{t_i}, \forall i \in [1, N_T]\}$
2. Generate P random parameter vectors $\vec{x_i}$
   $Q = \{\vec{x_i}, \forall i \in [1, P]\}$
3. Evaluate the parameter vectors in set Q,
   $O = \{\vec{f}(\vec{x_i}), \forall \vec{x_i} \in Q\}$
4. For $j = 1 \ldots N_T$
   (a) For $q = 1 \ldots P$
      i. Aggregate search vector $\vec{t_j}$ with aggregation function $A_j()$ and objective vector $\vec{O_q}$ to form vector of metric values for set Q:
      $m_q = A_j(\vec{O_q}, \vec{t_j})$
   (b) Identify the best performing parameter vector $\vec{x}$ in Q and corresponding objective vector in set O for each target vector and record in set $\mathbf{B}$:
   $\mathbf{B}_j = \arg \min_{\vec{x_i} \in Q, \vec{O_i} \in O} (m_i, \forall i \in [1, P])$
5. Select index j uniformly at random from the range $[1, N_T]$
   (a) Generate unit length vector for the search direction either by
      - biased from $\vec{x_j}$ towards other members of the population, or
      - if K=0 generate $\vec{V_j}$ in a random direction, or
      - Generate and evaluate K random offset vectors: $z = f(\vec{x_j} + \Delta\vec{X_i}), \forall i \in K$ then Set $\vec{V_j}$ as the unit length vector in the direction $\Delta\vec{X}$ that gave the best value for z (if z is uphill, for minimisation set $\vec{V_j} = -\vec{V_j}$)
   (b) Draw a random line search length l from (3),
   (c) Perform line search in direction of $\vec{V_j}$ starting at point $\vec{x_j}$ for a length of l. Record the location and fitness of all N_L points evaluated as part of line search into sets Q and O for the parameter and objective vectors respectively.
6. For $j = 1 \ldots N_T$
   (a) For $q = 1 \ldots N_L$
      i. Aggregate search vector $\vec{t_j}$ with aggregation function $A_j()$ and objective vector $\vec{O_q}$ to form vector of metric values for set O:
      $m_q = A_j(\vec{O_q}, \vec{t_j})$
   (b) Identify the best performing parameter vector $\vec{x}$ in Q and corresponding objective vector in set O for each target vector and record as structure $\mathbf{b}$:
   $\mathbf{b} = \arg \min_{\vec{x_i} \in Q, \vec{O_i} \in O} (m_i, \forall i \in [1, P])$
   (c) If the metric value in structure $\mathbf{b}$ is an improvement on the aggregated metric value currently stored for search vector j, then replace the details in $\mathbf{B}_j$ with the newly identified solution in structure $\mathbf{b}$
   (d) Return the set $\mathbf{B}$ as the best identified solutions.
7. loop back to item 5 for N_{IT} iterations until the desired total number of function evaluations have been performed.

In the version of the algorithm used to generate results for this paper, a target vector is chosen at random at each iteration (line 5 of the algorithm). If desired, the target vectors could be stepped through in turn or selected based on the performance of the aggregated fitness.

For this paper, the target vector set is generated *a-priori* using the approach described in [5], however the automatic adjustment methods that are employed in the MSOPS-II algorithm [7] can be incorporated successfully to remove the requirement for *a-priori* selection of the search vectors.

4. PERFORMANCE COMPARISONS

4.1 Introduction

To determine the behaviour of the MODELS algorithm, a sequence of experiments have been conducted and the results compared to the same experiments conducted using, NSGA-II, MSOPS and Random Search.

An objective function with a continuous concave Pareto front that is defined by a constraint boundary has been chosen to compare the performance of the algorithms as the dimensionality of the objective space increases. The objective function has been selected so that it is conceptually simple to solve and therefore reduces any bias that may be encounterd from the different search operators of the individual algorithms; the experiment is concerned with how the performance scales with the number of objectives, not how well different crossover and mutation operators suit the fitness landscape.

In order to allow the NSGA-II algorithm [3] to converge to a stable solution set for the higher dimensional problems, a population of 100 for 150 generations was employed in NSGA-II and MSOPS. For MODELS and the random search, a corresponding limit of 15000 function evaluations was employed.

For the MODELS algorithm, 100 target vectors were generated, spread uniformly across the objective region and each target vector was assessed using both the Weighted Min-Max and VADS aggregation functions, leading to a total of $N_L = 200$ points in the working population. For the analysis, only the 100 solutions corresponding to the Weighted Min-Max aggregation function were extracted.

The random search algorithm used is as follows:

1. Generate 100 search vectors $\vec{t_i}$
   $T = \{\vec{t_i}, \forall i \in [1, 100]\}$
2. Generate $P = 15000$ random parameter vectors $\vec{x_i}$
   $Q = \{\vec{x_i}, \forall i \in [1, P]\}$
3. Evaluate the parameter vectors in set Q,
   $O = \{\vec{f}(\vec{x_i}), \forall \vec{x_i} \in Q\}$
4. For $j = 1 \ldots 100$
   (a) For $q = 1 \ldots P$
      i. Aggregate search vector $\vec{t_j}$ with aggregation function $A_j()$ and objective vector $\vec{O_q}$ to form vector of metric values for set Q:
      $m_q = A_j(\vec{O_q}, \vec{t_j})$

(b) Identify the best performing parameter vector $\vec{x}$ in Q and corresponding objective vector in set O for each target vector and record in set $\mathbf{B}$:
$\mathbf{B}_j = \arg \min_{\vec{x_i} \in Q, \vec{O_i} \in O} (m_i, \forall i \in [1, P])$

5. Return the set $\mathbf{B}$ as the best identified solutions.

The random search process reports the best 100 identified solutions. The 15000 random points are generated uniformly at random in the parameter space in the same manner as the initial populations in all the optimisation algorithms.

4.2 Objective Function

For the algorithmic comparisons, an objective function vector that has known scalability properties in objective space has been employed. The objective is to minimise each element of the parameter vector $\vec{X}$, subject to a single explicit constraint and implicit constraints on each element of the parameter vector. The set of objective functions form a Pareto front along a constraint boundary comprising the first n-orthant of unit hypershpere (and hence is concave). If the distance from the origin to identified objective vectors are grouped to form histograms, an ideal optimisation result would have all solutions with a distance of unity from the objective space origin.

$$\begin{aligned} f(\vec{X}) = & \quad \vec{X} \\ \text{subject to:} & \\ g(\vec{X}) \geq & \quad \sqrt{\textstyle\sum_{i=1}^{N_{obj}} x_i^2} \\ x_i \in & \quad [0,1] \forall i \end{aligned} \tag{9}$$

5. ALGORITHM PERFORMANCE RESULTS

5.1 Introduction

The four optimisation algorithms have been tested on the hypersphere test function for a range of objective space dimensions from 2 to 20 dimensions. Each algorithm was run for 100 independent trials on each of the different objective space sizes. The MODELS algorithm used a Golden Section line search with $N_L = 10$ points on the line and an initial population of $P = 100$.

5.2 Performance Comparison

All 100 output points from all 100 trials have been gathered and analysed. The prime analysis has been to determine the distance of each output point from the true Pareto front location. As the Pareto front is defined by a constraint boundary that follows the unit hypersphere, the distance of each output solution in objective space from the objective space origin has been calculated; thus an ideal output would have all 100 results in all 100 trials having a distance of unity from the objective space origin.

Figure 4 shows histograms of the distance from the objective space origin of the 100 points output from the 100 trials for the 4 algorithms overlaid. With each algorithm performing 15000 function evaluations, the 2D results show that all of the algorithms are solving the problem very well, with MODELS providing a slight advantage over MSOPS and NSGA-II. All of the evolutionary methods are outperforming the random search as expected, although the random search is still performing very well due to the large number of function evaluations employed. The spread of solutions

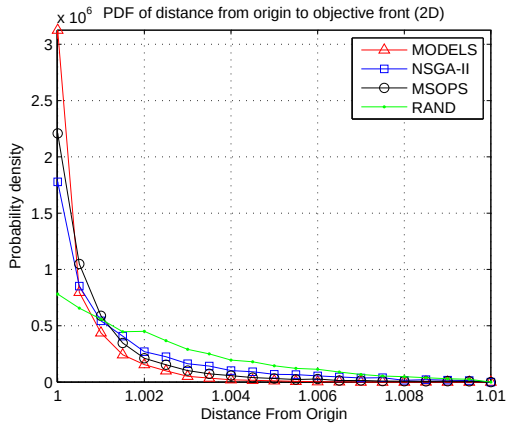

Figure 4: Histograms of results of 100 output vectors from 100 repeated trials of the MODELS, NSGA-II, MSOPS and random search optimisers for 2D objective function

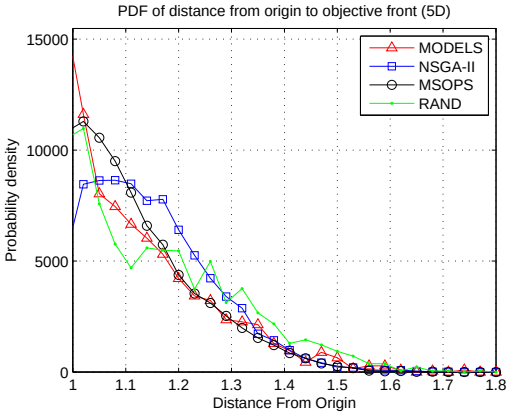

Figure 5: Histograms of results of 100 output vectors from 100 repeated trials of the MODELS, NSGA-II, MSOPS and random search optimisers for 5D objective function

from all of the algorithms has been analysed and all of the methods produce well spread results; the spread of results in MODELS, MSOPS and the random search is determined by the *a-priori* target set as expected.

Figure 5 shows the performance in 5D. All of the algorithms are finding the problem harder and are of an almost comparable performance in the order of MODELS is the most accurate, followed by MSOPS, Random search and finally NSGA-II.

Figure 6 shows the performance in 10 objective dimensions. The performance of the algorithms has degraded further with NSGA-II being significantly worse than the random search. MSOPS is still just outperforming the random search but the MODELS algorithm is producing far superior results to all of the other algorithms with a high proportion of very near Pareto solutions still.

Figure 7 shows the performance in 20 objective dimensions. The performance of the algorithms has degraded further still with NSGA-II still being significantly worse than

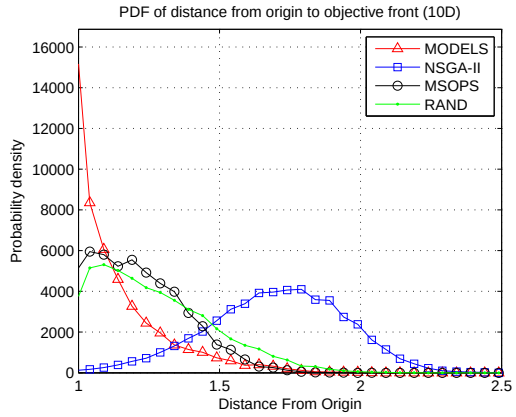

Figure 6: Histograms of results of 100 output vectors from 100 repeated trials of the MODELS, NSGA-II, MSOPS and random search optimisers for 10D objective function

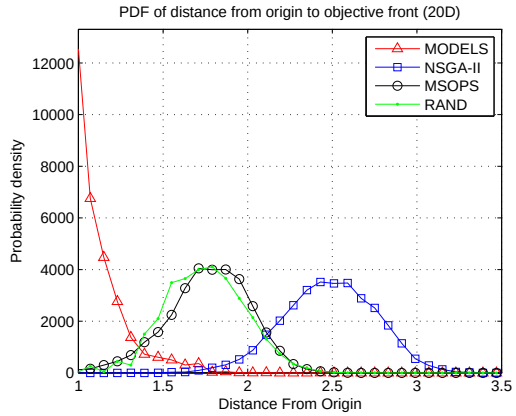

Figure 7: Histograms of results of 100 output vectors from 100 repeated trials of the MODELS, NSGA-II, MSOPS and random search optimisers for 20D objective function

the random search. The performamce of MSOPS is now comparable to the random search but the MODELS algorithm is again producing far superior results to all of the other algorithms with a high proportion of very near Pareto solutions still. Analysis of the 20D results has indicated that the spread of solutions from the MODELS algorithm is beginning to degrade, however in comparison to the output of the other 3 algorithms, the reduction in diversity can be tolerated.

6. CONCLUSIONS

A new algorithm designed specifically to optimise many-objective problems and with mechanisms to exploit partial gradient information within the search landscape has been presented.

Repeated trials have indicated that the performance of the algorithm is comparable to existing multi-objective optimisation approaches in low-dimensional objective spaces, but can significantly outperform existing methods for higher dimensional problems.

Software for the algorithms and files to allow the results to be recreated are at http://code.evanhughes.org

7. REFERENCES

[1] Dirichlet distribution. http://en.wikipedia.org/wiki/Dirichlet_distribution.

[2] C. A. C. Coello. A survey of constraint handling techniques used with evolutionary algorithms. Technical report, Laboratorio Nacional de Informática Avanzada, 1999.

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

[4] I. Griva, S. G. Nash, and A. Sofer. *Linear and Nonlinear Optimization*. Cambridge University Press, 2009.

[5] E. J. Hughes. Multiple single objective pareto sampling. In *Congress on Evolutionary Computation 2003*, pages 2678–2684, Canberra, Australia, Dec. 2003. IEEE.

[6] E. J. Hughes. Evolutionary many-objective optimisation: Many once or one many? In *IEEE Congress on Evolutionary Computation, 2005*, volume 1, pages 222–227. IEEE, Sept. 2005.

[7] E. J. Hughes. MSOPS-II: A general-purpose many-objective optimiser. In *Congress on Evolutionary Computation CEC 2007*, pages 3944–3951, Singapore, Sept. 2007. IEEE.

[8] M. Locatelli. A note on the griewank test function. *Journal of Global Optimization*, 25:169–174, 2003.

[9] R. C. Purshouse. *On the Evolutionary Optimisation of Many Objectives*. PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK, Sept. 2003.

[10] K. Sindhya, A. Sinha, K. Deb, and K. Miettinen. Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In *Proceedings of Congress on Evolutionary Computation*, CEC'09, pages 2919–2926, 2009.

[11] D. Vieira, R. Takahashi, and R. Saldanha. Multicriteria optimization with a multiobjective golden section line search. *Mathematical Programming*, pages 1–31, 2010.

[12] T. Wagner, N. Beume, and B. Naujoks. Pareto-, Aggregation-, and Indicator-based methods in many-objective optimization. In *Evolutionary Multi-Criterion Optimization, EMO 2007*, pages 742–756, Matsushima, Japan, 2007. Springer LNCS 4403.

[13] S. Zapotecas Martínez and C. A. Coello Coello. A proposal to hybridize multi-objective evolutionary algorithms with non-gradient mathematical programming techniques. In *Proceedings of PPSN X*, pages 837–846. Springer-Verlag, 2008.