sParEGO – A Hybrid Optimization Algorithm for Expensive Uncertain Multi-Objective Optimization Problems

João A. Duro[1], Robin C. Purshouse[1], Shaul Salomon[1,2], Daniel C. Oara[1], Visakan Kadirkamanathan[1], and Peter J. Fleming[1]

[1] University of Sheffield, UK {j.a.duro,r.purshouse}@sheffield.ac.uk
[2] ORT Braude College of Engineering, Israel, shaulsal@braude.ac.il

Abstract. Evaluations of candidate solutions to real-world problems are often expensive to compute, are characterised by uncertainties arising from multiple sources, and involve simultaneous consideration of multiple conflicting objectives. Here, the task of an optimizer is to find a set of solutions that offer alternative robust trade-offs between objectives, where robustness comprises some user-defined measure of the ability of a solution to retain high performance in the presence of uncertainties. Typically, understanding the robustness of a solution requires multiple evaluations of performance under different uncertain conditions – but such an approach is infeasible for expensive problems with a limited evaluation budget. To overcome this issue, a new hybrid optimization algorithm for expensive uncertain multi-objective optimization problems is proposed. The algorithm – sParEGO – uses a novel uncertainty quantification approach to assess the robustness of a candidate design without having to rely on expensive sampling techniques. Hypotheses on the relative performance of the algorithm compared to an existing method for deterministic problems are tested using two benchmark problems, and provide preliminary indication that sParEGO is an effective technique for identifying robust trade-off surfaces.

Keywords: Expensive optimization, surrogate-based optimization, robust optimization, multi-objective optimization

1 Introduction

The ability of simulations to predict the performance of a candidate design is constantly increasing. While some simulations can produce high-fidelity outputs relatively quickly, a typical mesh-based simulation can run for several hours, and even days. Even if a design team has access to supercomputing resources, the extensive run-time still implies that perhaps only a few hundred candidate designs can be explored using high-fidelity modelling resources. Unfortunately, conventional multi-objective optimization algorithms implemented in commercial packages typically require tens of thousands of function evaluations to converge on a high quality solution [1]. Therefore, the search for a promising design using expensive evaluation functions on a limited computational budget poses a great challenge.

To exacerbate this problem, optimizing for a robust solution is itself a computationally demanding task. In order to gain confidence over the robustness of a solution to uncertainties, the statistical properties of the expected solution's performance must be quantified. In a world where the complexity of the high-fidelity models essentially produces a black-box mapping of inputs to outputs, such statistical properties would typically be found through repeated evaluation of the same solution using those high-fidelity models. However, repeatedly sampling a single candidate design is computationally expensive.

To address the above, we propose a framework for expensive uncertain multi-objective optimization problems (MOPs). The key aims are to: (i) exploit expensive, black-box evaluation function for a candidate design; (ii) account for multiple sources of uncertainty, such as fidelity of evaluation functions and manufacturing tolerances; and (iii) provide an understanding of the risk and opportunity trade-offs between candidate designs with respect to a given robustness metric. To achieve this the framework leverages ParEGO [2], an algorithm for multi-objective optimization, which has been demonstrated to provide good results for optimization runs limited to a small number of function evaluations. ParEGO itself is a multi-objective extension to Jones et al.'s [3] seminal efficient global optimization (EGO) algorithm for single-objective problems. The main limitation of ParEGO is that it has not been designed to handle problems featuring uncertainty (although there is some evidence that it can perform favourably in noisy environments [4]). Therefore a fundamental part of the framework is how ParEGO can be extended to consider evaluation functions as samples of random variates. We refer to this new algorithm as *stochastic ParEGO* or *sParEGO*.

In the remainder of this paper, first the robustness metric used is described in Section 2, and the proposed framework is presented in Section 3. The hypotheses on the relative performance of the algorithm are introduced in Section 4. The experimental settings and findings are in Section 5. The paper concludes with Section 6.

2 Threshold-based robustness metric

A general single-objective robust optimisation problem can be formulated as:

$$\min_{\mathbf{x}\in\Omega} S = \mathbf{f}(\mathbf{x}, \mathbf{U}). \tag{1}$$

Here, $\mathbf{x} = [x_1, \ldots, x_{n_x}]$ is a vector of n_x decision variables in a feasible domain Ω, $\mathbf{U}$ is a vector of random variables that includes all the uncertainties associated with the optimisation problem. These uncertainties may be an outcome of manufacturing tolerances, a noisy environment, evaluation inaccuracies etc. A single scenario of the variate $\mathbf{U}$ is denoted as $\mathbf{u}$. Since uncertainties are involved, the scalar objective S is also a random variate, where every scenario of the uncertainties, $\mathbf{u}$, is associated with an objective value s.

In a robust optimisation scheme, the random objective value is replaced with a robustness criterion, denoted by the indicator $I[S]$. Several criteria are commonly used in the literature, which can be broadly categorised into three main approaches:

1. **Worst-Case Scenario.** The worst objective vector, considering a bounded domain in the neighbourhood of the nominal values of the uncertain variables.
2. **Aggregated Value.** An integral measure of robustness that amalgamates the possible values of the uncertain variables (e.g. mean value or variance).
3. **Threshold Probability.** The probability for the objective function to be better than a defined threshold.

In our framework the third approach, suggested by Beyer and Sendhof [5], is used. A threshold q is considered as a satisficing performance for the objective value s. When s is uncertain, denoted by the random variable S, the probability for S to satisfy the threshold level can be seen as a confidence level c. For a minimization problem this can be written as:

$$c(S, q) = \Pr(S < q). \tag{2}$$

A robustness indicator used in this paper is based on minimization of the threshold q for a pre-defined confidence level c, meaning that the confidence in the resulting performance can be specified (e.g. by a decision-maker).

A stochastic unconstrained multi-objective optimization problem (MOP), which is the focus of this study, can be formulated as:

$$\min_{\mathbf{x} \in \Omega} \mathbf{Z} = \mathbf{f}(\mathbf{x}, \mathbf{U}). \tag{3}$$

where $\mathbf{Z}$ is a multivariate random vector of n_z performance criteria, and $\mathbf{f}$ is a set of functions mapping from decision-space to objective-space. Due to uncertainties over the problem parameters or the mapping functions themselves, every evaluation of the same decision vector may result in a different realisation of the objective vector $\mathbf{z} = [z_1, \ldots, z_{n_z}]$.

3 The Framework of the sParEGO Algorithm

sParEGO is a surrogate-based multi-objective optimization algorithm for dealing with stochastic MOPs. The algorithm shares many similarities with ParEGO including the ability to approximate expensive MOPs over a realistically small number of function evaluations. The main idea is that the uncertain distribution in objective space of every candidate solution is not quantified through uncertainty quantification methods (e.g. Monte Carlo sampling). Instead, every solution is evaluated once, and the distribution is approximated based on the performance of nearby solutions. A pseudo-code of sParEGO is presented in Algorithm 1 and a general description of its working principles is as follows.

The decision variables and objectives are normalised to non-dimensional units in the following manner:

$$\tilde{x}_i = (x_i - x_i^l)/(x_i^u - x_i^l),\ i = 1, \dots, n_x, \tag{4}$$
$$\tilde{z}_j = (z_j - z_j^*)/(z_j^n - z_j^*),\ j = 1, \dots, n_z, \tag{5}$$

where x_i^u and x_i^l are the upper and lower boundaries of the i^{th} decision variable, z_j^n and z_j^* are the j^{th} components of the estimated nadir and ideal vectors, and the tilde accent represents a normalised, dimensionaless variable. The normalised values are used for all operations within the algorithm. Before a candidate design is evaluated, it is re-scaled to the natural dimensions.

sParEGO decomposes the overall MOP into a number of single-objective problems by using a set of (reference) direction vectors to guide the search towards different regions of the Pareto front[3]. The set of all direction vectors is denoted by $\mathcal{D}$ (Line 1). The direction vectors are picked (one at the time) based on their sequence in the set $\mathcal{D}$. Once all direction vectors have been traversed by the optimizer the vectors in the set $\mathcal{D}$ are shuffled (Line 5). This prevents any bias that might arise due to repeatedly using the same sequence of direction vectors during the entire optimization process.

The procedure used to generate the initial set of solutions ($\mathcal{X}$) is described in Section 3.2 (Line 2). Following this, all solutions in the set $\mathcal{X}$ are evaluated and their performance is stored in the set $\mathcal{Z}$ (Line 3). The ideal and nadir vectors are then updated (Line 7). A scalar fitness value is obtained for each solution by using a scalarising function as mentioned in Section 3.1 (Line 8). The robustness indicator values of the solutions are estimated and stored in the set $\mathcal{I}$ (Line 9), and these are used to construct a surrogate model (Line 10). A search procedure is then conducted over the model to find a solution $\mathbf{x}^{\text{new}}$ that optimizes the given robustness indicator based on the concept of expected improvement (Line 11). A new solution $\mathbf{x}^{\text{pert}}$ is generated by applying a perturbation to $\mathbf{x}^{\text{new}}$ (Line 12). This ensures that all generated solutions have at least one nearby solution. The new solutions are added to $\mathcal{X}$ (Line 13) and, once evaluated, their performance is stored in $\mathcal{Z}$ (Line 14). The algorithm goes back to Line 4 and the procedure repeats itself until a stopping criteria is satisfied.

The robustness indicator values of the solutions are estimated based on the procedure in Line 17. The first step is to identify, for each solution, all the nearby solutions. For this, we define the concept of neighbourhood and consider that two solutions are neighbours if their distance in normalised decision-space is within a user-defined neighbourhood distance δ (Line 19). The statistical properties of the performance of a solution is approximated from the other neighbouring solutions (Line 23). Finally, the robustness indicator values of the solutions are estimated for a given robustness criterion I (Line 24).

[3] More details about the decomposition strategy are provided in Section 3.1.

Algorithm 1 sParEGO Pseudo-code

```
    Parameters: initial set size $n_{\text{init}}$, surrogate model maximum set size $n_{\max}$, maximum distance
    between newly generated solutions $\delta_{\text{pert}}$, robustness criterion $I$, neighbourhood distance $\delta$
 1: $\mathcal{D} \leftarrow$ set of all reference direction vectors                ▷ Equation 6 (Section 3.1)
 2: $\mathcal{X} \leftarrow$ generate initial set of solutions using $n_{\text{init}}$ and $\delta_{\text{pert}}$      ▷ Section 3.2
 3: $\mathcal{Z} \leftarrow \mathbf{f}(\mathcal{X})$                                        ▷ evaluate the initial set
 4: while stopping criteria not satisfied do
 5:     Shuffle the set $\mathcal{D}$
 6:     for all $\mathbf{d} \in \mathcal{D}$ do
 7:         update ideal and nadir vectors
 8:         $\mathcal{S} \leftarrow$ calculate scalar fitness value of all solutions      ▷ Equation 7 (Section 3.1)
 9:         $\mathcal{I} \leftarrow$ RobustnessApproximation($\mathcal{X}, \mathcal{S}, \delta$)            ▷ Sections 3.3 and 3.4
10:         $model \leftarrow$ fit a Surrogate model to the indicator values $\mathcal{I}$ using $n_{\max}$   ▷ Section 3.5
11:         $\mathbf{x}^{\text{new}} \leftarrow$ maximize the expected improvement based on $model$
12:         $\mathbf{x}^{\text{pert}} \leftarrow$ add a neighbour to $\mathbf{x}^{\text{new}}$ using $\delta_{\text{pert}}$               ▷ Section 3.5
13:         $\mathcal{X} \leftarrow \mathcal{X} \cup \{\mathbf{x}^{\text{new}}, \mathbf{x}^{\text{pert}}\}$
14:         $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{\mathbf{f}(\mathbf{x}^{\text{new}}), \mathbf{f}(\mathbf{x}^{\text{pert}})\}$                  ▷ evaluate the new solutions
15:     end for
16: end while
17: procedure RobustnessApproximation($\mathcal{X}, \mathcal{S}, \delta$)
18:     for all $\mathbf{x}_i \in \mathcal{X}$ do
19:         update the neighbourhood $\mathcal{N}(\mathbf{x}_i)$ for a given $\delta$        ▷ Equation 10 (Section 3.3)
20:     end for
21:     $\mathcal{I} \leftarrow \emptyset$
22:     for all $\mathbf{x}_i \in \mathcal{X}$ do
23:         approximate the distribution of $S_i$                          ▷ Section 3.3
24:         calculate robustness indicator $I[S_i]$                        ▷ Section 3.4
25:         $\mathcal{I} \leftarrow \mathcal{I} \cup I[S_i]$
26:     end for
27: end procedure
```

3.1 Decomposition

A decomposition-based algorithm decomposes the MOP into a number of single-objective problems, each approaching the global trade-off surface from a different direction. The i^{th} sub-problem is associated with a reference direction vector $\mathbf{d}^i$ which is taken from the set $\mathcal{D}$. The set is constructed by using a Simplex Lattice design:

$$\mathcal{D} = \left\{ \mathbf{d} = [d_1, \ldots, d_{n_z}] \mid \sum_{j=1}^{n_z} d_j = 1 \wedge d_j \in \left\{ \frac{0}{h}, \frac{1}{h}, \ldots, \frac{h}{h} \right\} \text{for all } j \right\}, \tag{6}$$

where h is a parameter that defines the number of divisions for each objective.

Each sub-problem assigns a scalar fitness value to each solution. This is achieved by using a scalarising function $f(\mathbf{z}, \mathbf{w})$ that maps an objective vector $\mathbf{z}$ into a scalar value according to a vector of weights $\mathbf{w} = [w_1, \ldots, w_{n_z}]$. The scalarising function used is the weighted Tchebycheff, which is given by:

$$s = \max_{1 \leq i \leq n_z} \{w_i z_i\}. \tag{7}$$

For a given direction vector $\mathbf{d}$ there is a corresponding weighting vector that minimizes the scalarising function [6]. The optimal weighting vector $\mathbf{w}$ for the scalarising function in (7) is defined as:

$$w_i = t_i \Big/ \sum_{i=1}^{n_z} t_i, \quad \text{where } \; t_i = (d_i + \epsilon)^{-1}, \quad i = 1, \ldots, n_z, \tag{8}$$

where ϵ is a small number to prevent division by zero, and the normalisation enforces the weighting vector's elements to sum up to one.

3.2 Initialisation

In sParEGO, the robustness assessment of a candidate design relies on the determination of its statistical properties, which in turn depends on the information of the neighbouring solutions. Hence, in order to support the robustness assessment from the beginning of the optimization process, for any solution in the inital set there is at least one nearby solution in desision-space. Let n_{init} denote the size of the initial set, then the procedure is as follows:

1. To provide a good coverage, a space-filling design technique (Latin Hypercube sampling) is used to generate a fraction of the total n_{init}. We suggest this fraction to be a quarter.
2. For every existing solution in $\mathcal{X}$, another solution is generated by applying a random perturbation where the upper bound is within a hypersphere with a radius of δ_{pert} which is smaller than δ.
3. The rest of the solutions are generated by randomly selecting an existing solution from $\mathcal{X}$ and applying a perturbation as in the previous step. This step is repeated until the number of solutions in $\mathcal{X}$ is equal to n_{init}.

The second step enforces that every solution has at least one neighbour. The third step seeds the initial population with neighbourhoods of different sizes.

3.3 Uncertainty quantification

The most important difference between sParEGO and ParEGO is that the former assumes that the outcome of an evaluation function is a realization of a random variate. Therefore, the scalarised function value cannot be used directly to construct the surrogate model, and a utility indicator value is used instead. For every direction vector, the surrogate model is constructed to search for a design that will optimize a given robustness indicator (described in Section 3.4). The guiding principle is to avoid having to repeatedly sample every candidate design to assess its statistical properties in objective-space. Instead, these properties (specifically, measures of central tendency and dispersion) are approximated from the available information of other candidate design evaluations.

Approximation of the central tendency The stochasticity of the problem might originate from a variety of sources, including variations in decision-space.

For this type of uncertainty, two designs with similar nominal values can be identical when realised. Therefore, the performance of a candidate design should be calculated from the performance of neighbouring designs as well. Two solutions $\mathbf{x}_i$ and $\mathbf{x}_j$ are considered as neighbours if their Euclidean distance in normalised decision-space is smaller than or equal to δ, that is:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \delta. \tag{9}$$

For a solution $\mathbf{x}_i$ with a scalar fitness given by s_i, the statistical properties of the scalar fitness are approximated from the neighbouring solutions as follows: First, the neighbourhood $\mathcal{N}(\mathbf{x}_i)$ of the solution is defined[4]:

$$\mathcal{N}(\mathbf{x}_i) = \left\{\mathbf{x}_j \in \mathcal{X} \mid \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \delta\right\}. \tag{10}$$

Next, the approximated mean function value, μ_s, is derived from the neighbourhood. Members that are closer to $\mathbf{x}_i$ are given a larger weight, denoted as v in Equation (11), in approximating its properties. Since the weight of most solutions in the neighbourhood is smaller than 1, the overall "neighbourhood size" ς_i is smaller than $|\mathcal{N}(\mathbf{x}_i)|$:

$$v_j = \frac{\delta - \|\mathbf{x}_i - \mathbf{x}_j\|_2}{\delta}, \ \forall \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i), \tag{11}$$

$$\varsigma_i = \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} v_j, \tag{12}$$

$$\mu_{s,i} = \frac{1}{\varsigma_i} \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} v_j s_j, \tag{13}$$

where $\mu_{s,i}$ is the approximated mean of the scalar fitness function for $\mathbf{x}_i$.

Approximation of the dispersion Once the expected mean is known, the expected value for the variance is calculated:

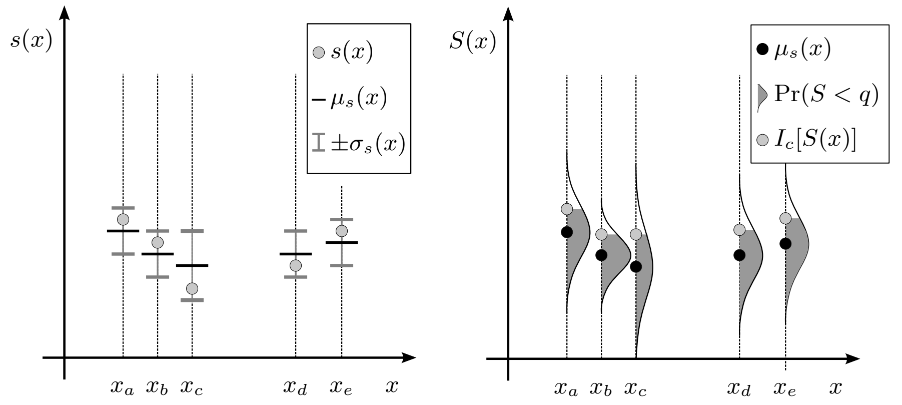
$$\sigma^2_{s,i} = \frac{1}{\varsigma_i} \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} v_j (s_j - \mu_{s,i})^2. \tag{14}$$

An example is shown in Figure 1(a) for an optimization problem with a single decision variable where 5 solutions are divided into two neighbourhoods. The mean and variance of the scalar fitness function is estimated based on their scalar fitness values and their decision-space distance within the neighbourhood.

3.4 Estimating the robustness indicator value

Once the statistical properties of the scalar fitness function have been estimated, the random variable $S(\mathbf{x})$ is assumed to follow a normal distribution with the

[4] Note that according to (10), $\mathbf{x}_i$ is included in the neighbourhood $\mathcal{N}(\mathbf{x}_i)$.



(a) Estimation of the mean value and the variance from the neighbourhood

(b) Assuming a normal distribution for S according to μ_s and σ_s. The shaded area represents a confidence of 80%.

Fig. 1: Approximation of the statistical properties, and estimation of robustness indicator $I_c[\bullet]$.

estimated mean and variance. The robustness indicator is calculated for this distribution with respect to a desired confidence level c (assuming $c \in [0, 100]$). The indicator, denoted $I_c[S]$, is then equal to the c^{th} percentile of the normal distribution with mean μ_s (Equation 13) and variance σ_s^2 (Equation 14).

An example for $I_c[S]$ is given in Figure 1(b) where $I_c[S]$ value corresponds to the 80th percentile of the normal distribution. Following this, the indicator value $I_c[S]$ is considered as the solution's fitness at the current iteration.

3.5 Fitting a Surrogate Model to the Fitness

Now that every solution is associated with a scalar fitness value based on the robust indicator, the algorithm proceeds in a similar fashion to EGO and ParEGO [3, 7]. A surrogate model is fitted to the fitness values, and the expected improvement function is constructed from the model. Above a certain size (approximately 50 solutions), the surrogate model becomes prohibitively expensive to construct. When the number of evaluated solutions exceed this size, a subset of size $n_{\max}$ is chosen according to Algorithm 2.

The first step in Algorithm 2 is to select $n_{\max}/2$ solutions from the population set $\mathcal{X}$ with the best robustness indicator value, and to add these to the set $\mathcal{X}'$ (Line 1). The next step is to select from the remaining solutions those that are closer to the current direction vector $\mathbf{d}$. For this, the normalised objective vectors are projected to the $n_z - 1$ simplex (Line 4). The Euclidean norm between the vectors $\hat{\mathbf{z}}(\mathbf{x})$ and $\mathbf{d}$ is given by the operator $\mathit{\Delta}$ (Line 5). Finally, the $n_{\max}/2$ solutions from $\mathcal{X}''$ with the smallest $\mathit{\Delta}$ distance are added to $\mathcal{X}'$ (Line 7).

To use the expected improvement function we need to estimate its variance ($\hat{\sigma}^2$). For this, we use the density of the solutions in decision-space by knowing that the variance has an inverse correlation to the density of the solutions. A

Algorithm 2 Choosing a Subset to Construct the Surrogate Model

```
Require: population set $\mathcal{X}$, subset size $n_{\max}$, current direction vector $\mathbf{d}$
Ensure: a subset $\mathcal{X}'$ of size $n_{\max}$
1: $\mathcal{X}' \leftarrow n_{\max}/2$ solutions from $\mathcal{X}$ with the best robustness indicator value
2: $\mathcal{X}'' \leftarrow \mathcal{X} \setminus \mathcal{X}'$
3: for all $\mathbf{x} \in \mathcal{X}''$ do
4:     $\hat{\mathbf{z}}(\mathbf{x}) \leftarrow$ project $\mathbf{z}(\mathbf{x})$ to the $n_z - 1$ simplex, implying that $\hat{\mathbf{z}}(\mathbf{x}) = \mathbf{z}(\mathbf{x}) \left\|\mathbf{z}(\mathbf{x})\right\|_1^{-1}$
5:     $\Delta(\mathbf{x}, \mathbf{d}) \leftarrow \left\|\hat{\mathbf{z}}(\mathbf{x}) - \mathbf{d}\right\|_2$
6: end for
7: $\mathcal{X}' \leftarrow \mathcal{X}' \cup n_{\max}/2$ solutions from $\mathcal{X}''$ with the smallest $\Delta$ distance
```

suitable way to estimate the density at a given point $\mathbf{x}$ is to use a non-parametric statistical approach, and in this case we use a kernel density model given by:

$$p(\mathbf{x}) = \frac{1}{n_{\max}} \sum_{\mathbf{x}_i \in \mathcal{X}'} \frac{1}{(2\pi h_b^2)^{n_x/2}} \exp\left\{ -\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2h_b^2} \right\}, \tag{15}$$

where h_b is the bandwidth. We suggest setting the bandwidth to be equal to one hundredth of the mean span of all solutions, that is:

$$h_b = \frac{1}{100 \times n_{\max}} \sum_{\mathbf{x}_i \in \mathcal{X}'} (\max(\mathbf{x}_i) - \min(\mathbf{x}_i)). \tag{16}$$

Based on experimental results we have observed that the kernel density model can be very sensitive to any changes in the density, thus we have used a smoothing function (in this case the arctan function), and the estimated variance at $\mathbf{x}$ is:

$$\hat{\sigma}^2(\mathbf{x}) = \left(\frac{1}{\pi/2} \arctan\left(\frac{1}{p(\mathbf{x})} \right) \right)^2. \tag{17}$$

After fitting a surrogate model to the solutions from $\mathcal{X}'$, the next task is to find the solution that maximizes the expected improvement function. For this, any suitable off-the-shelf single-objective optimizer can be used, and we have chosen ACROMUSE [8]. The identified solution, denoted by $\mathbf{x}^{\text{new}}$, is added to the population together with a neighbouring solution $\mathbf{x}^{\text{pert}}$, generated using the same perturbation as that described in Section 3.2.

4 Hypothesis testing

We employ a hypothesis testing approach to study the performance of sParEGO compared with ParEGO in dealing with MOPs on a limited computational budget. We postulate two hypotheses, each relating to anticipated pathological behaviour of one of the algorithms:

1. If the problem is deterministic, and the region close to the Pareto front is highly multi-modal, sParEGO will incorrectly interpret the multimodality as stochasticity, and converge on seemingly 'robust' solutions that are actually non-optimal. However ParEGO's convergence will be unaffected.

2. If the problem is highly stochastic, and the region close to the Pareto front is smooth, ParEGO will identify seemingly high-performance solutions that are actually non-robust. However sParEGO's convergence will be unaffected.

To test these hypotheses, we use two variants of the WFG4 problem [9]. Both variants have two objectives and five decision variables. The first two decision variables are position parameters and the last three are distance parameters. For the first problem, namely P1, we have modified WFG4 to increase the density and the number of local optima in the periphery of the global optimum. This simulates the effect that stochasticity can have when approaching the Pareto-optimal Front (PF). The second problem, namely P2, is characterised by having a more smooth landscape with no local minima surrounding the global optimum, and stochasticity is added by the toolkit from [10].

The modification applied to WFG4 is as follows. The original formulation of WFG4 applies a transformation to each input parameter (y) given by:

$$\begin{aligned} \text{s\_multi}(y,a,b,c) &= \left(1+\cos(r_2)+b(r_1)^2\right)/(b+2), \\ r_1 &= |y-c|/(\lfloor c-y \rfloor + c), \\ r_2 &= (4a+2)\pi(0.5-0.5r_1), \end{aligned} \tag{18}$$

where a controls the number of minima, b controls the magnitude of the "hill sizes" of the multi-modality, and c is the value for which y is mapped to zero. The number of minima increases up-to $2a+1$ which includes the global optimum at c. We propose a modification to Equation 18 as follows:

$$\begin{aligned} \text{s\_multi}^*(y,a,b,c,d,e) &= (1+\cos(r_2 r_3)+b|r_1|^e)/(b+2), \\ r_3 &= (1-|r_1|)^{2d}, \end{aligned} \tag{19}$$

where d controls the density of the hills around the optimum, and e specifies the polynomial order of the base curve. The effect of these parameters is shown in Figure 2, in that: the density of hills around the optimum increases with an increase in d as shown in Figure 2(a), and; the proximity of local minima from the value zero decreases with an increase in e as shown in Figure 2(b).

The toolkit from [10] is used here to transform the objective vectors of WFG4 into random vectors. The parameters have been chosen to ensure that uncertainty increases towards more optimal regions. The uncertainty also decreases up to a point when moving away from the Pareto region, and then starts increasing again for regions that are further away from the PF. The perturbation is applied to the objective vector by using only its radial component, implying that the perturbation radius is set to zero. This means that an objective vector $\mathbf{z}$ is perturbed only along one direction, defined by the $n_z - 1$ simplex and given by $\hat{\mathbf{z}} = \mathbf{z}/\sum z_i$, for $i = 1, \ldots, n_z$. In practice, instead of using the deterministic value of the distance term in WFG4, we consider it as a random variate with a uniform distribution. The lower bound of the distribution is situated at the deterministic value from the test problem, and the upper bound increases as solutions approach the Pareto region. As a result, for a given robustness criterion

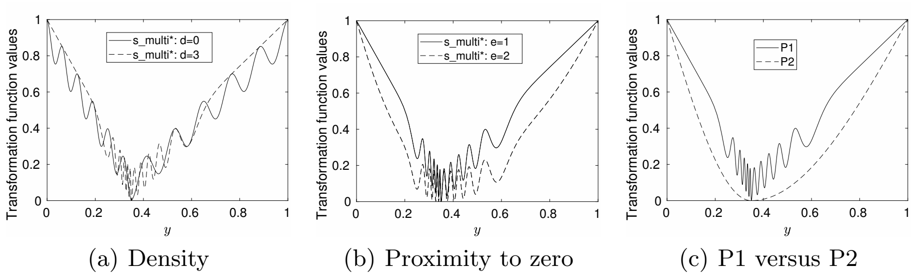


Fig. 2: Transformation function in WFG4 as a function of the input parameter (y). For (a) and (b) the parameters are $a = 5$, $b = 10$, and $c = 0.35$. The difference between P1 and P2 is shown in (c). Moreover, $e = 1$ for (a) and $d = 3$ for (b).

(say, the worst-case scenario as mentioned in Section 2), the worst performance of the Pareto-optimal solutions can be worse than that of some of the non-Pareto-optimal solutions. This gives rise to the term *Robust Pareto-optimal Set* (RPS), which is defined as the set of solutions with the best performance with respect to the given robustness indicator.

Following the above, we have chosen $c = 0.35$ for all test instances. The remaining parameters are: $a = 5$, $b = 10$, $d = 3$, and $e = 1$ for P1; and $a = 0$, $b = 8$, $d = 0$, and $e = 2$ for P2. The transformation function values for these settings are shown in Figure 2(c). Moreover, the confidence level of the robustness indicator $I_c[S]$ is set to 90% (i.e., the 90th percentile of the normal distribution). The PF for P1 corresponds to a quadrant with extremes of 2 and 4 for objectives f_1 and f_2, respectively, and it is shown in Figure 3(a). The PF has been obtained by uniformly generating points along the quadrant. Figure 3(b) shows the performance of the RPS with respect to the given robustness indicator. The RPS has been obtained by using an enumeration where the uniform distribution over the distance term of WFG4 has been replaced by the 90th percentile of the same distribution.

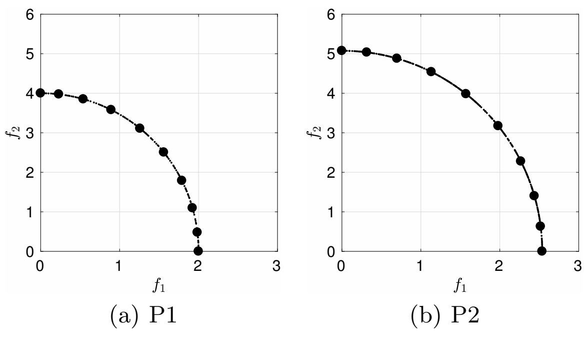


Fig. 3: Pareto-optimal front for P1 in (a), and (b) shows the performance of the RPS with respect to the given robustness indicator for P2.

5 Experimental results

5.1 Experimental settings

For both ParEGO and sParEGO algorithms the number of direction vectors is set to 10. Other common parameters are: $n_{init} = 10$, $n_{max} = 50$ and the optimization budget is set to 5000 function evaluations. For sParEGO: $\delta = 0.1\sqrt{n_x}$ and $\delta_{\text{pert}} = \delta/2$. Inverted Generational Distance (IGD) [6] is used to measure the quality of the obtained sets by the optimizers. The 10 solutions that are marked with a filled circle in Figure 3 are used as the reference sets for IGD, and these solutions correspond to the best optimal solutions for the chosen direction vectors.

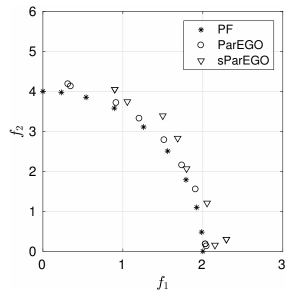
The optimizers report only one solution per direction vector, implying that only 10 solutions are identified at the end of the optimization process. For each direction vector the solution with the minimum scalarised fitness value is chosen. For this, ParEGO uses the scalarised fitness values determined directly by Equation 7, while sParEGO uses the fitness attributed by the robustness indicator.

5.2 Findings

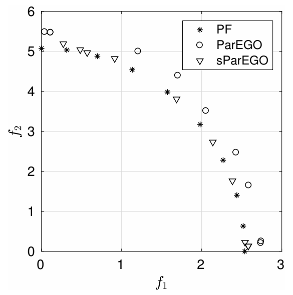
This section presents the experimental results for problems P1 and P2. The results shown in Figure 4 provide both a visual and a analytical assessment of the quality of the solutions obtained by the optimizers in terms of their convergence to and diversity across the PF. The objective vectors for P2 have been determined by evaluating 100 times each decision vector, and the performance of each objective is equal to the 90^{th} percentile of its marginal distribution.

For P1, ParEGO's approximation to the PF is slightly better than for sParEGO as shown in Figures 4(a) and 4(c). This indicates that the multi-modality in P1, close to the vinicity of the PF, is interpreted by sParEGO as a region of high uncertainty. The performance of the solutions with respect to the robustness indicator in this region is captured as being poor according to the statistical inferences made by sParEGO. Hence, most sParEGO solutions are just outside the region where the magnitude of the hill sizes of the multi-modality become relatively large. Nevertheless, it is expected for sParEGO to improve its convergence to the PF with more function evaluations, since the statistical assessment made about the true performance of the solutions that are on the PF is also expected to improve.

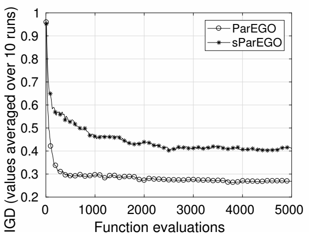
For P2, sParEGO approximation to the PF obtained with respect to the robustness indicator is better than ParEGO as shown in Figures 4(b) and 4(d). The convergence of ParEGO deteriorates along the optimization run as shown in Figure 4(d), while the convergence of sParEGO improves. This indicates that the selection criterion used by ParEGO that promotes solutions with a better nominal performance, leads to a deterioration in the convergence towards the solutions that satisfy the robustness criterion. On the other hand, the uncertainty quantification approach used by sParEGO that is used to estimate the true robustness of the solutions is found to be a better approach in dealing with the task of finding the robust solutions.



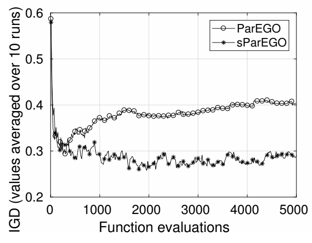
(a) P1: Single run (median IGD)



(b) P2: Single run (median IGD)



(c) P1: IGD



(d) P2: IGD

Fig. 4: Comparison of ParEGO and sParEGO for P1 and P2.

6 Conclusion

This paper has proposed a new multi-objective optimization algorithm for dealing with expensive uncertain MOPs, namely sParEGO. The comparative analysis with the existing algorithm ParEGO has demonstrated that the statistical inferences made by sParEGO are better equipped to assess the robustness of the candidate solutions for a given robustness criterion. However, we have also shown that the existence of a not-so-well-behaved problem landscape can mislead the uncertainty quantification approach when (as is necessary) working on a limited budget of evaluations. Given this, the assumptions made within the framework described in this paper, and their associated risks, are as follows:

1. *The landscape is well-behaved (i.e. smooth, continuous).* The uncertainty distributions are approximated according to available information for other candidate solutions. The underlying assumption for approximating in this way is that similar solutions have similar performance. If the functions are highly ragged and discontinuous, the surrogate models cannot accurately predict their behaviour.
2. *The problem dimensionality is small to medium.* The search is conducted on a surrogate model fitted to the existing evaluated solutions. The surrogate model used in this framework typically produces good estimates for problems with up to 20 design variables.

3. *The maximum distance between solutions to be considered as neighbours, specified by δ, affects the variance of solutions and the convergence rate.* For smooth and continuous functions, a tight neighbourhood is likely to result in smaller variance, but also uses less information from other solutions, which reduces the convergence rate.

Further benchmarking of sParEGO is now needed to confirm its capabilities across a wider set of problem instances. This includes conducting a comparative analysis with other multi-objective robust optimization algorithms, such as those described in the survey in [11]. Other future research directions include: how to approximate the statistical inferences of isolated solutions; how to incorporate constraints; and incorporation of alternative robustness criteria. The source code of the sParEGO algorithm, as well as the test problems, is found within the Liger software, which can be downloaded from the following repository: https://github.com/ligerdev/liger.

Acknowledgements: This work was supported by Jaguar Land Rover and the UK-EPSRC grant EP/L025760/1 as part of the jointly funded Programme for Simulation Innovation. The open-source version of Liger was also supported by the Advanced Propulsion Centre UK (grant J14921). The authors acknowledge Joshua Knowles for useful discussions during the design process of sParEGO, and thank all reviewers for their insightful comments and suggestions.

References

1. Zhou, A., et al.: Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm Evol Comput **1**(1) (2011) 32–49
2. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE T Evolut Comput **10**(1) (2006) 50–66
3. Jones, D., et al.: Efficient global optimization of expensive black-box functions. J Global Optim **13**(4) (1998) 455–492
4. Knowles, J., et al.: Noisy multiobjective optimization on a budget of 250 evaluations. In: Proc. EMO 2009. (2009) 36–50
5. Beyer, H.G., et al.: Robust optimization - a comprehensive survey. Comput Method Appl M **196**(33-34) (2007) 3190–3218
6. Van Veldhuizen, D., et al.: On measuring multiobjective evolutionary algorithm performance. In: Proc. CEC 2000. Volume 1. (2000) 204–211
7. Knowles, J., et al.: Multiobjective optimization on a budget of 250 evaluations. In: Proc. EMO 2005. (2005) 176–190
8. Ginley, B.M., et al.: Maintaining Healthy Population Diversity Using Adaptive Crossover, Mutation, and Selection. IEEE T Evolut Comput **15**(5) (2011) 692–714
9. Huband, S., et al.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE T Evolut Comput **10**(5) (2006) 477–506
10. Salomon, S., et al.: A Toolkit for Generating Scalable Stochastic Multiobjective Test Problems. In: Proc. GECCO 2016. (2016)
11. Jin, Y., et al.: Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on Evolutionary Computation **9**(3) (June 2005) 303–317