Multi-Objective Simulation Optimisation with Noise

Aliya Jangabylova, Duleabom An, Juan Ungredda, Michael Luya
and Utkarsh Sharma
CDT in Mathematics of Real World Systems
Department of Mathematics
University of Warwick, U.K.

June 11, 2018

Abstract

The study is aimed at developing methods for tackling noise in a budget constrained, multi-objective simulation based optimsation environment. Using benchmark results from a tool designed for deterministic multi-objective problems, we implemented methods to tackle noise (e.g. in social sciences or epidemiology). We attempted to outperform the benchmark NSGA-II, using Evolutionary Algorithms along with the Rolling Tide Method, a random forest surrogate and Archived Multi-Objective Simulated Annealing (AMOSA). Within a budget of 4000 simulations and resampling each individual solution 100 times, the three methods were found to show promising results. The Rolling Tide Evolutionary Algorithm in particular outperformed the benchmark results even without the use of a random forest surrogate model. The performance of evolutionary methods is attributed to the fact that a greater proportion of budget is spent on promising results. Separately, AMOSA was shown to provide marginally better results than the benchmark, thus we incorporated the rolling tide method to deal with noise. To conclude statistical tests were performed and the hypervolume metric was used to measure performance against the benchmark.

Contents

1 Introduction

Optimisation has been a key field of research, because of its wide ranging applications in multiple domains. However, a large number of these applications are still theoretical in nature, meaning it is not possible to immediately apply optimisation frameworks to real world data, primarily because of presence of noise, and to an extent, because real world systems tend to be characterised by the presence of more than one objective.

Real world systems exhibit noise in measurement, often in sensory devices used to measure environment variables, or in estimation, as with the supply chain domain, which exhibits demand uncertainty. Noise makes it difficult to assess the fitness of any solution, as what might seem a promising trial solution could actually be a very poor one when resampled. Consider an objective function $f_{observed}(X) = f_{actual}(X) + \psi$, where ψ is uncertainty embedded in the domain specific problem. Presence of this noise could yield different results on each simulation run. However, using an otherwise inferior solution as a superior one could have meaningful impact. For example, it could impact a commercial organisation's viability, a hospital's budget being spent on suboptimal tasks, or social effectiveness of complex governmental policies being compromised.

Compounding the issue of noise is frequently a challenge to be confronted when dealing with optimisation problems. Therefore, we have to consider methods which can deal with the trade-off between multiple objectives. Budget constraints are also introduced in order to reduce the computational complexity of our simulations.

2 Problem Statement

2.1 Company Description

ESTECO is an independent software provider, highly specialised in numerical optimisation, bringing enterprise-wide solutions for design optimisation, simulation data management and automation to over 300 international organizations. ESTECO's main software product is modeFRONTIER, one of the leading platforms for multiobjective engineering design optimisation. It uses metaheuristics for optimisation, and capably links with a wide range of simulation tools to evaluate candidate solutions.

2.2 Objective

Given the challenge posed by ESTECO, the group project will develop strategies to handle noisy multi-objective simulation optimisation problems under a constraint budget of 4000 evaluations[1].

2.3 Supply Chain Domain

As a base problem, ESTECO provided a Supply Chain Management problem in the Chemical Process Industry. Specifically, the multi-objective problem is designed such

[1]The target was suggested by ESTECO.

that there are several plant sites, each with their own processing lines, corresponding capabilities, and categories of products it can service, further constrained by demand amounts and changeover costs. The demand from the customer is the primary source of noise, and assumed to be normally distributed. Other non-technical assumptions made are: (i) all plant sites can serve all sales regions, (ii) transportation is assumed to be always available, and (iii) transportation cost is directly proportional to the distance. Our objectives are: (i) to minimise total costs and (ii) to maximise customer service level. The customer service level is defined as the percentage of demand which can be serviced from existing stock. Decision variables are production amounts for each processing centre and supply amounts from each warehouse to each sales region.

The base code for which multiple objectives were optimised is from [1], using [2]. We converted the base MATLAB code written by [Wu] [3] to Python.

Herewith, the two objectives to be optimised are defined, as in [1] and restated here for ease of reference:

1. Cost: It is comprised of production cost, transportation cost, and inventory holding cost. We try to find the lowest total cost, thus it should be minimised.

2. Customer Service level: It is mainly dependent on whether the demand is met. To increase the figure, we should satisfy the demand in every time period by both insourcing and outsourcing. We aim to maximise it.

Note, whilst the focus is on supply chain domain, the simulation model used would work as a black-box solution, i.e. simply by providing the value of the objective functions.

3 Concepts

3.1 Multiobjective Optimisation Problem

A multi-objective optimisation problem has a number of objective functions and constraints that feasible solutions must satisfy. A minimisation problem is converted to maximisation problem in general by multiplying (-) sign. In the following, we state the multiobjective optimisation problem in its general form [4]:

$$\begin{aligned} Maximise\{\, z_1 = f_1(x), z_2 = f_2(x), ..., z_q = f_q(x) \,\} \\ s.t.\, g_i(x) \leq 0, i = 1, 2, ..., m\ . \end{aligned} \tag{1}$$

Here, $z_k = f_k(\mathbf{x}), k = 1, \ldots, g$ are linear or nonlinear q objective functions that depend on n decision variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. In Pareto optimisation, we define decision and objective spaces, S and Z, respectively. $Z = \{z \in \mathbb{R}^q | z = f_1(x), z = f_2(x), \ldots, z_q = f_q(x), x \in S\}$. Both S and Z are used to illustrate the feasible region. Since the objective functions in Z conflict each other, there is usually no single solution that is best in all objectives, but improving one objective means deteriorating the other objective [4].

3.2 Pareto Optimal Solution

In order to represent the set of best solutions, we provide the following definitions:

- **Domination**: A solution $\mathbf{x}^{(1)}$ dominates another solution $\mathbf{x}^{(2)}$, if and only if: i) The solution $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all objectives. ii) The solution $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective.
- **Non-dominated set**: Among a set of solutions **P**, the non-dominated subset of solutions **P'** are those that are not dominated by any member of the set **P**.
- **Globally Pareto-optimal set**: The non-dominated set of the entire feasible search space S is the globally Pareto-optimal set.
- **Locally Pareto-optimal set**: If for every member $\mathbf{x}$ in a set $\underline{\mathbf{P}}$ there exists no solution $\mathbf{y}$ (in the neighborhood of $\mathbf{x}$ such that $||\mathbf{y} - \mathbf{x}||_\infty \leq \epsilon$, where ϵ is a small positive number) dominating any member of the set $\underline{\mathbf{P}}$, then the solutions belonging to the set $\underline{\mathbf{P}}$ constitute a locally Pareto-optimal set.

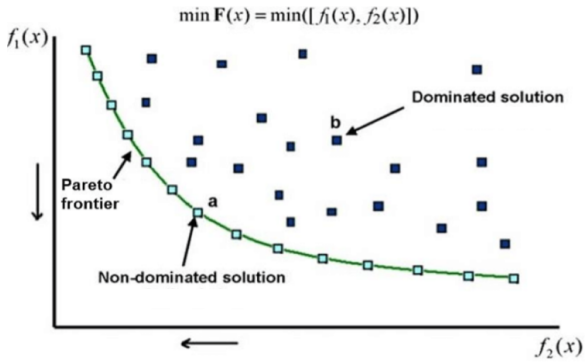


Figure 1: Pareto Optimal Solution [5].

4 Methods

4.1 Motivation on used methods

The methods implemented in this report present certain properties that make them candidates for the current problem. Specifically, RTEA presents a structure that can be applied to ESTECO's packages to improve their results without having to implement other optimisation tools not based in evolutionary methods.

AMOSA is convenient for deterministic problems, since the algorithm converges to the true Pareto front, but the performance deteriorates with increasing levels of noise. We have attempted to combat this using RTEA's re-sampling strategy. Also we considered using surrogate models to improve RTEA.

Algorithm	**Motivation**	**Drawback**
Rolling-tide Evolutionary Algorithm (RTEA)	• Only resamples elite solutions • Easy to implement • Deals effectively with noise	• The resampling parameter k increases the number of parameters by 1
Archive Multi-Objective Simulated Annealing (AMOSA)	• Converges to the Pareto optimal solution in deterministic problems • Fast running time • Low computational complexity	• Slow convergence time • Is not designed for solving stochastic problems • Sensitive to different parameter settings
RTEA + Surrogate Model	• Increases convergence rate	• Inaccurate surrogate model may mislead RTEA to poor candidates • Increases the number of parameters, depending on the type of surrogate model.
RTEA + AMOSA	• Deal with noise from design and objective space	• Slow convergence time

Table 1: Motivations and drawbacks of evaluated algorithms.

The following subsections give the necessary background for understanding the aforementioned algorithms. Specifically, sections **4.2** and **4.3** give background on evolutionary algorithms and RTEA, sections **4.4** and **4.5** introduce AMOSA and **4.6** and **4.7** explain surrogate models and how they are applied to RTEA.

4.2 Fundamentals of Evolutionary Algorithm

An Evolutionary Algorithm (EA) is a heuristic algorithm derived from the theory of evolution. The principle idea of an EA is that if individuals that meet certain selection criteria reproduce, the population will converge to individuals that best satisfy the selection criteria. If an inferior offspring is added, the population can explore other search areas. These principles follow the basic rule of natural selection as suggested in Darwin [6]. Here is an outline for a simple genetic algorithm:

Algorithm 1: Basic EA scheme

1 Initialise the population x^i, i= 1,...,l.
2 Calculate the values of the objective function $f(x^i)$, i= 1,...,l.
3 **while** *break criteria is not satisfied* **do**
4 | Select two individuals $\mathbf{x}^a$ and $\mathbf{x}^b$ according to a fitness measure.
5 | Recombine $\mathbf{x}^a$ and $\mathbf{x}^b$ to generate a new vector **x'**
6 | Generate a mutation/variation: **x'** $\Rightarrow$ **x"**
7 | Evaluate the object function: $f(\boldsymbol{x}")$
8 | Store the new individual in the population.
9 **end**

In the following, we explain some operations made in Algorithm 1:

- **Selection:** Reproduction operators pass superior solutions to the next generation, discarding inferior solutions in a population, while keeping the population size constant. The best solutions are usually selected to be passed to crossover operator.

- **Crossover:** In the crossover operator, two parent solutions are chosen and some portion of the solutions are exchanged to create new solutions (children or offspring). Even though crossover does not always provide a better solution than parent solutions, the chance of creating better solutions is better than random. Among many methods, we used the two-point crossover, shown in Figure 2. Once two cross sites are randomly selected, one string is divided into three substrings. Then the middle substring between two sites is exchanged.

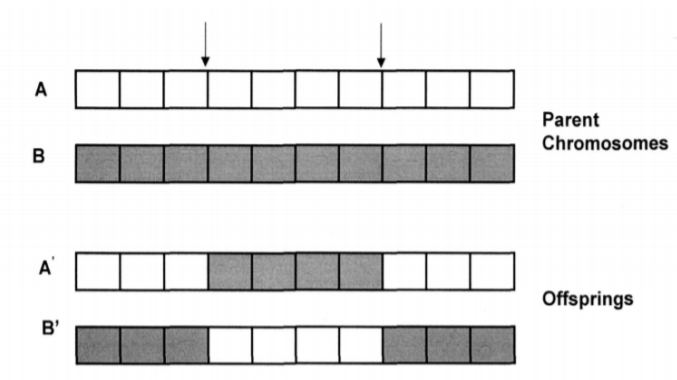


Figure 2: Two-point Crossover [7].

- **Mutation:** Mutation is designed to add diversity to the population and prevent the algorithm from converging to a local minimum. The mutation operator we have used randomly chooses an element of the string and changes it using a uniform distribution bounded by the minimum and maximum allowed.

EAs have many parameters, such as population size and probabilities of occurrence for genetic operations. Usually, a relatively small population size, high crossover probabilities and low mutation probabilities inversely proportional to the chromosome size are recommended [8].

Also, an EA can feature elitism, which is an elite-preserving operator that favours the elites of a population, directly carrying them over to the next generation. Consequentially, a good solution found early on in the run will never be lost unless a better solution is discovered.

One major property of EA heuristics is that the search does not begin with one possible solution, but with a whole population of possible solutions. Additionally, the individuals of the population can exchange solution attributes between them. EAs are also particularly well suited to a set of solutions in the case of multiple objectives. Thus, they are more resistant to premature convergence towards local optima in multimodal search spaces.

4.3 Rolling Tide Evolutionary Algorithm

In this project we use a variant of EA called the Rolling Tide Evolutionary Algorithm (RTEA). This method enhances accuracy of elite solutions (non-dominated solutions) whilst not spending a lot of the computational budget for re-evaluating all samples contained in the archive. We use the following terminology to further explain the algorithm:

- X: A set of feasible solutions.
- x: A single solution from X.
- $Y(x)$: The set of evaluations corresponding to a single solution x.

- $est(Y(x))$: Estimated value of $Y(x)$. For this work, the estimation will be calculated as the non weighted average of the evaluations. Decisions on whether to add a solution x to A are based on the estimated noise-free objective values $est(Y(x))$, estimated from a number of samples $Y(x) = \{y_i(x)\}_{i=1}^{n}$.
- A: An elite archive of non-dominated solutions in X so far. Estimated Pareto Front.
- $F(x)$: The set of evaluations corresponding to a single solution x where $x \in A$.
- x': The mutated child.
- x'': The member of the elite archive with the fewest number of function evaluations.
- r: A number of random locations to initial sample.
- v,u: Two parents from A.

RTEA is outlined in Algorithm 2. On lines 1–2, r design vectors are sampled uniformly and each of these is evaluated a single time. We denote by Y the set of sets comprising the objective evaluations; each element of Y is the set of objective evaluations corresponding to a single solution, x. After initialisation, each element of Y has a single element corresponding to the single evaluation of a member of X. The non-dominated elite archive is initialised from initial evaluations (line 3), and extracted from X, meaning that A is the estimated Pareto set, and X contains the solutions evaluated by the algorithm that are not in A. Also, for each member of X, a single solution which dominates it is recorded; this dominating element may be in X or in A.

After initialisation, the algorithm continues in an optimisation loop. During each iteration the algorithm proceeds by assessing a new solution in design space (as long as the exclusive refinement phase of the algorithm has not been reached). Two solutions are selected from the estimated Pareto set of the solutions evaluated so far (line 7). These solutions are then crossed over with probability of crossover, to create a single child (line 9). This child, x, (or clone of an individual parent if no crossover occurs) is mutated, creating x' (line 13).

After its generation, x' is evaluated once and if it is not dominated by the current estimated Pareto set (based on comparing y' with $est(Y(x))$ for $x \in A$), it is added to the archive (lines 16 and 17), and any elements in A which are dominated by x' (based on their $est(Y(x))$) are returned to X with x' recorded as their single tracked dominator. If, however, x' is dominated by the estimated Pareto front, then a single member of A is chosen as the single dominator for x' (line 19), from the members of A which dominate x'.

In the next step, the member of the elite archive with the fewest number of function evaluations, x'' , is selected and reevaluated an additional time (lines 23, 25, and 26). Prior to reevaluation, all solutions in X that have the selected archive member x'' recorded as their single dominator are extracted (line 23); this set is denoted by $X_{x''}$.

Algorithm 2: Rolling-tide evolutionary algorithm [9]

Require: r
Require: g
Require: P_{cross}
Require: k
Require: z

X := initial_samples(r)
Y := evaluate_samples(X)
$\{A, F\}$:= get_front(X, Y)
t :=r
while $t < g$ **do**
  if $t < (1 - z)g$ **then**
    $\{v$,u$\}$:= sample_leading_edge($A, est(F)$)
    if uniform_rand() $< P_{cross}$ **then**
      x := crossover(v, u)
    else
      x :=v
    end if
    x' := $vary$(x)
    y' := evaluate(x')
    t :=t+1
    if est(F) $\not\prec$ y(x') **then**
      $\{A, F, X, Y\}$:= update_front($A, F, x', \{y'\}, X, Y$)
    else
      track_dominator($A, F, x', \{y'\}$)
    end if
  end if
  for k iterations **do**
    x'' := argmin$\{$ $|Y(x)|$ for $x \in A$ $\}$
    $\{$ X''_x, Y''_x $\}$:= remove_tracking($x'', Y(x''), X, Y$)
    y'' := evaluate(x'')
    $Y(x'')$:= $Y(x'') \bigcup \{y''\}$
    $t : t + 1$
    $A := A \backslash \{x''\}$
    $F := F \backslash \{Y(x'')\}$
    $\{A, F, X, Y\}$:= update_front($A, F, x'', Y(x''), X, Y$)
    for each x and $Y(x)$ pair in X''_x and Y''_x **do**
      $\{A, F, X, Y\}$:= update_front($A, F, x, Y(x), X, Y$)
    end for
  end for
end while

Lines 28–30 ensure that if the reevaluation of x'' means that the solution should no longer be in A, it is removed from A and placed in X. Likewise, lines 31 and 32 ensure that if solutions are no longer dominated by an element of A, they are transferred into A. On line 32, elements that are not inserted into A remain in X and a dominating member of $A \cup X_{x''}$ is set as their single tracked dominator.

Algorithm 3: Pseudocode for the Adopt Algorithm [9]

1: Initial setting for Adopt Algorithm
2: **while** $t <$g **do**
3: MAIN Algorithm to adopt
4: **for** k iterations **do**
5: $x'' :=$ argmin$\{\ |Y(x)|$ for $x \in A\ \}$
6: $\{\ X''_x, Y''_x\ \} :=$ remove_tracking$(x'', Y(x''), X, Y)$
7: $y'' :=$ evaluate(x'')
8: $Y(x'') := Y(x'') \bigcup \{y''\}$
9: $t : t + 1$
10: $A := A \backslash \{x''\}$
11: $F := F \backslash \{Y(x'')\}$
12: $\{A, F, X, Y\} :=$ update_front$(A, F, x'', Y(x''), X, Y)$
13: **for** each x and $Y(x)$ pair in X''_x and Y''_x **do**
14: $\{A, F, X, Y\} :=$ update_front$(A, F, x, Y(x), X, Y)$
15: **end for**
16: **end for**
17: **end while**

RTEA is applicable to many other algorithms. As it is described in Adopt Algorithm 3, we can use Rolling-tide part of RTEA after running adopt algorithm and gaining information from it. We expect to resample best solutions and improve estimation by doing so. In this project, we merge RTEA with AMOSA.

4.4 Simulated Annealing

This family of algorithms is particularly useful when trying to find the global minimum of a single-objective function that is multimodal. This succeeds in finding global minima due to a combination of two very important reasons. Firstly, whenever solutions are perturbed, the perturbations are made via random amounts and secondly, we allow for some risk in our algorithm by accepting solutions that are dominated with a given probability. This differs from greedy algorithms, which only take in solutions immediately better than current solution, and is less likely to fall prey to converging to a local minimum that is not global.

It has been proven theoretically that if the temperature cools logarithmically, the Boltzmann Annealing algorithm converges to the true optimal solution very slowly but almost surely [10].

Annealing schedules are what unqiuely define Simulated Annealing algorithms. These are comprised of four central components [2]:

1. An **initial temperature**.
2. A **final temperature**.
3. A **cooling schedule** which determines how the temperature decreases over time.
4. An **iteration schedule** determining how the number of iterations will increase with decreasing temperature.

When tackling multi-objective problems, additional conditions are required to make the algorithm unique. In the next section, we present a specific example of multi-objective simulated annealing.

4.5 Archived Multi-Objective Simulated Annealing (AMOSA)

The overarching problem that ESTECO has presented is one which is multi-objective rather than single-objective, so it is important to generalise the above algorithm to a multi-objective setting. The algorithm that we will introduce is 'Archived Multi-Objective Simulated Annealing' (AMOSA) for short [11]. The AMOSA algorithm can be used in order to estimate the Pareto fronts of an optimisation problem.

As the name of the algorithm suggests, an archive of points is stored and altered over time. We first generate an archive of points and choose a random point from this archive, calling this our 'current point' or *currpt*. Then, we add a random perturbation to this point, which we call our 'perturbed point' or *newpt*. The pseudocode below then provides us with an algorithm that presents three different cases depending on the status of domination between these two points.

For the algorithm below, it is also important to note that the absolute difference in domination between two points, p_1 and p_2, is calculated using the following formula,

$$\Delta E_{p_1,p_2} = \prod_i \frac{\|E(p_1)[i] - E(p_2)[i]\|}{R_i}$$

where R_i is the range of outputs in the ith objective function we are considering given our current archive of points (this means the values for R_i change as our archive of points change). Finally, we define two additional parameters in the below pseudocode, SL and HL. We define SL to be the maximum size to which the archive can be filled before clustering is used to reduce the size of the archive to HL and we define HL as the maximum number of points in the archive that the AMOSA algorithm returns.

As with the single-objective case, one major advantage of using the AMOSA algorithm is that it has been proven to succeed in converging to the optimum Pareto front, and this algorithm produces a set of non-dominated points provided the algorithm runs slowly

[2] Acceptance probabilities can also be uniquely defined.

enough and long enough. This also leads to one of its greatest disadvantages - there is usually a large amount of time taken to converge to a solution which is considered a good enough approximation to the true optima, and that these solutions take significantly longer to find than evolutionary algorithms and surrogate models do.

The parameters that are provided with the AMOSA algorithm are as follows:

- HL: the largest number of solutions which can be returned by the algorithm.
- SL: the largest number of solutions which can be determined upon termination.
- T_{init}: our initial temperature.
- T_{min}: the temperature threshold used to terminate the algorithm.
- $iter$: the number of iterations used at each temperature.

We begin by initialising a dataset called $Archive$ with a maximum size of SL points. Then we choose an initial temperature value $T = T_{init}$ and a lower threshold temperature of T_{min}. We then randomly select a solution from $Archive$, $currpt$, and we randomly generate a point from our domain of interest and call it $newpt$. These two points are changed and updated with repeated iterations of the AMOSA algorithm found below.

The AMOSA algorithm initialises the archive by first generating $\gamma \times SL$ solutions, where γ is an integer that is greater than 1 (in all of our AMOSA algorithms the number we choose is 2). Then this archive is reduced to its subset of non-dominant solutions and clustering is used to reduce the number of solutions to size HL.

The clustering technique that the AMOSA algorithm uses is called single-linkage clustering, which is provided within the 'scipy' package in Python. This works by first placing each solution into its own cluster. Then we calculate all the minimum inter-cluster distances. The minimum inter-cluster distance is equal to the minimum distance between any one point from one cluster to any other point from the other cluster. After these distances are calculated, the two clusters responsible for the smallest inter-cluster distance are merged together to form new clusters. This process then repeats iteratively until the desired number of clusters is reached.

Once the clustering process has finished, representative solutions from each cluster are chosen to construct the new archive. The representative solution of any cluster is the solution with the smallest averaged minimum inter-cluster distance. This results in our set of evaluations becoming more uniformly distributed across our estimate for the Pareto front, allowing us to obtain a better understanding of its shape.

AMOSA also uses an exponential cooling scheme, meaning that the temperature decreases by a multiplicative factor of α such that $T_{i+1} = \alpha \times T_i$. The paper does not propose a specific iteration scheme, only that the initial number of iterations should be proportional to SL.

Algorithm 4: The AMOSA Algorithm for Minimisation.

while $T > T_{min}$ **do**
  $i = 0$
  for $i < iter$ **do**
    if *currpt dominates newpt* **then**
      k = The number of remaining points in *Archive* that dominate *newpt*.
      $\Delta E_{avg} = \frac{\sum_{i=1}^{k} \Delta E_{i,newpt} + \Delta E_{currpt,newpt}}{k+1}$
      $p = \frac{1}{1+e^{-\Delta E_{avg} \times T}}$, r = Random number between 0 and 1.
      if $r < p$ **then**
        $currpt = newpt$
      end
    else if *currpt and newpt are non-dominating each other* **then**
      if *newpt is dominated by* $k \geq 1$ *points in Archive* **then**
        $\Delta E_{avg} = \frac{\sum_{i=1}^{k} \Delta E_{i,newpt}}{k}$
        $p = \frac{1}{1+e^{-\Delta E_{avg} \times T}}$, r = Random number between 0 and 1.
        if $r < p$ **then**
          $currpt = newpt$
        end
      else if *All of the points in Archive dominate newpt* **then**
        $currpt = newpt$ Append *newpt* to *Archive*.
        if $len(Archive) > SL$ **then**
          Cluster the points in *Archive* into HL clusters.
        end
      else if *newpt dominates* $k \geq 1$ *points in Archive* **then**
        $currpt = newpt$. Append *newpt* to *Archive* and remove all k dominated points from *Archive*.
    else if *newpt dominates currpt* **then**
      if *newpt is dominated by* $k \geq 1$ *points in Archive* **then**
        ΔE_{min} = The minimum absolute difference between $E(newpt)$ and any of the other k dominating points.
        $p = \frac{1}{1+e^{\Delta E_{min} \times T}}$, r = Random number between 0 and 1.
        if $r < p$ **then**
          $currpt$ = point which is responsible for the minimum absolute difference in domination.
        else
          $currpt = newpt$
        end
      else if *All of the points in Archive dominate newpt* **then**
        Remove *newpt* from Archive if *newpt* is in *Archive*.
        $currpt = newpt$
        if $len(Archive) > SL$ **then**
          Cluster the points in *Archive* into HL clusters.
        end
      else if *newpt dominates* $k \geq 1$ *points in Archive* **then**
        $currpt = newpt$ and append *newpt* to *Archive*.
        Remove all k dominated points from *Archive*.
    end
    $i = i + 1$
  end
  $T = \alpha \times T$
end

if $len(Archive) > SL$ **then**
  Cluster the points in $Archive$ into HL clusters.
end
Return Archive.

4.6 Surrogate Modeling

In daily life individuals try to save time by making predictions based on their assumptions. For example, to predict an approximate time of performing a certain task or to anticipate the right amount of food they need to stock up for a week. Without noticing this, in their minds people construct surrogate models using the experience from the past. In engineering design the idea is very similar, in essence surrogate models make valuable predictions based on a restricted amount of information and assumptions. For most surrogate models, the main assumption is that the engineering function is continuous [12].

Surrogate-based optimisation is an optimisation methodology that takes advantage of surrogate modeling techniques to speed up a search for local or global optima. The main idea behind the surrogates is that they mimic the behaviour of expensive simulations to approximate its output. They can also be used to improve the design efficiency and to filter a numerical noise. However, surrogates are of special interest for computationally expensive objective functions as they do not require any simulations. To approximate expensive analysis code, different models can be used such as polynomial functions, neural networks, support vector machines, radial basis functions, decision trees and others. These models learn from the population of sampled data that is observed by running a computationally expensive function [13].

Specifically note that we want the most accurate prediction in the optimum region. The most effective and commonly used technique for estimating error is *cross-validation*. The main idea is to randomly split the training data into k equal subsets. During each turn, one of the subsets is removed from the complete training set and the surrogate model is fitted to the remaining set. The removed data is used as a validation set and at each stage is predicted by the model. Once all subsets are used as the validation set, then n predictions $(\hat{y}^{(1)}, \hat{y}^{(2)}, ..., \hat{y}^{(n)})$ are calculated, where n is the number of observed data in the validation set. The RMSE estimator is used for model selection and surrogate model parameter estimation [12]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2}. \tag{2}$$

4.7 Evolutionary Algorithm combined with Surrogate Model

Evolutionary algorithms need many evaluations to find an acceptable solution. A surrogate response surface is an approximation of the expensive simulation. Each simulation is represented as $y = f(x)$ and the surrogate is $\hat{y} = \hat{f}(x)$, such that $\hat{f}(x) = f(x)+e(x)$, where $e(x)$ represents an approximation error [14]. To mirror the results of EA, the surrogate model does not require any information of internal characteristics of the simulation.

The main criteria for choosing a surrogate model are accuracy, time complexity and number of parameters to be optimised. Usually for multi-objective optimisation problems, nonlinear surrogate models are used, such as Support Vector Machines (SVM) or Artificial Neural Networks (ANN). Even though their accuracies are considerably high, SVM require tuning many parameters such as their kernel functions and their hyperparameters, regularisation parameters and slack variables, otherwise, it is sensitive to overfitting. Regarding the ANN, the main drawback is a complexity of constructing a network that needs to be performed manually and accurately tested. It also needs a lot of data to learn properly, and its architectures need to be fine-tuned to get a good performance. In addition, for considerably expensive objective functions with high dimensionality the training time of SVM and ANN can be large [15]. This paper uses a *random forest* model which is a good compromise between the low training time and high performance [16]. The random forest is an ensemble model that collects different predictions from independent decision trees to combine into a final decision by the majority of votes. Hence, it reduces the overfitting significantly by using the bagging method to resample a subpopulation for each decision tree. It also can be suited for different types and sizes of datasets, meaning that overfitting can be controlled by controlling the depth of a tree and by making a proper pruning at some nodes that can be updated automatically [15].

Common techniques used to exploit the maximum amount of information from a surrogate model combined with EAs are:

1. *surrogate-informed operators* such as crossover or mutation that can benefit from getting more information about f
2. *pre-selection* that is used to re-evaluate a new offspring
3. *direct optimisation of the surrogate model* - if $\hat{f}$ is a sufficiently good approximation of f, then it is used *in lieu* of f, but the number of generations by $\hat{f}$ should be controlled to prevent divergence when the model is imprecise [17].

In our algorithm we used the *pre-selection* method to screen every new offspring and identify the most promising candidate. The surrogate is trained on a population of dominated and non-dominated sets, so that it has a wide spread of data points that reduces the bias and ensures its diversity. Once the mutated child is generated, the random forest evaluates its output and the predicted value is then sorted along with the initial population to check its non-dominance. If the offspring is in the Pareto front then it goes further for expensive evaluation. Otherwise, the algorithm generates a new one for some certain number of iterations until it gets the non-dominant offspring. In RTEA every generated point is used for simulation despite of its quality. By including the surrogate,

we reduce the number of simulations on poor individuals given the restricted budget. Moreover, by disregarding poor or "lucky" individuals, it helps RTEA to approach the true Pareto front.

Algorithm 5: Pre-selection part in surrogate-based RTA

```
X = generate population
Y = evaluate samples
A, F = get dominated and non-dominated sets of (X,Y)
surrogate = build surrogate
while t < budget do
    train surrogate on (A,F)
    i = 0
    for i < iterations do
        RTEA: crossover and mutation
        x = mutated offspring
        predict x = surrogate predict x
        if predict x is in pareto front then
            break
        else
            continue
        end
    end
    evaluate x
    RTEA: reevaluation of best individuals
end
```

If the surrogates are not constructed properly, the error can mislead the RTEA to spread weak individuals, meaning that a poor offspring may be chosen in the next generation and the promising ones may be excluded [17]. To evaluate the accuracy of the random forest, we referred to [18], where they study the variability of predictions and estimate the standard deviations based on the *infinitesimal jackknife* (IJ). The Jackknife procedure is used to reduce the bias and estimate the variance of estimates. The main idea of jackknife is to exclude one observation and compute the estimate based on the remaining observations. However, IJ gives a smaller weight on one observation and obtains an asymptotic result as the weight approaches zero. The variance is estimated as follows:

$$\hat{V}_{IJ-U}^{B} = \hat{V}_{IJ}^{B} - \frac{n}{B^2}\sum(t_b^*(x) - \bar{t^*}(x))^2 \tag{3}$$

$$\hat{V}_{IJ}^{B} = \sum_{i=1}^{n} \hat{Cov_i}^2 \tag{4}$$

$$\hat{Cov_i} = \frac{\sum_b (N_{bi}^* - 1)(t_b^*(x) - \bar{t^*}(x))}{B}, \tag{5}$$

where $\hat{V}_{IJ}^{B}$ is IJ, $\hat{V}_{IJ-U}^{B}$ is a bias-corrected IJ, B is a finite number of bootstrap replicates, N_{bi}^* is the number of times the i^{th} observation occurs in the bootstrap sample b, $t_b^*(x) =$

$t(x; Z^*_{b1}, ..., Z^*_{bn})$ and the Z^*_{bi} are elements from training set in the b^{th}bootstrap sample, for more information refer to [18].

If the priority of a surrogate is a faster rate of convergence, then the surrogate can be used with real simulations, where it will be the responsibility of a 'helper' to find promising fitness values. If a surrogate user wants to reduce the execution time of simulation evaluations, then a well constructed surrogate can be used instead of a real simulation. In this case, a surrogate can replace a real simulation function for mutated offspring estimation as it requires small computational resources. For more challenging problems with sparse data, the usage of multiple alternative surrogates together can give different local optima. Thus, the capability of the global search can be increased at the checking phase. Moreover, it may provide estimations with lower variances by constructing a weighted averaged model [19]. All the above mentioned approaches are flexible and can be combined depending on the problem structure. As our initial aim was to find a Pareto front within a limited budget whilst the execution time was unconstrained, we used a surrogate as an additional 'free' simulation function.

We compared the execution time of RTEA and RTEA+Surrogate algorithms for k= 1 in order to observe whether adding a surrogate takes longer and the extent to which it does so. Each algorithm was executed 20 times. The mean time for RTEA is 548.494 seconds whilst its coefficient of variation 6.63% and for RTEA+Surrogate 605.568 seconds with the coefficient variance equal to 20.44% (processor 2.3 GHz Intel Core i5 was used). The deviation for the second algorithm is comparably wider because of the stochastic environment - in a defined range surrogates do not allow for 'poor' or 'lucky' individuals to go further. Sometimes the promising offspring can be found in one iteration, and in other cases it may take longer to break a loop.

4.8 Quality Measure for Noisy Optimisation

One widely used indicator to measure the quality of a solution set is hypervolume indicator. The hypervolume measures the volume of the dominated portion of the objective space (as shown in Fig 3). Geometrically speaking, the hypervolume indicator measures the volume of the dominated space of all solutions contained in a solution set [20].

5 Parameter Setting

5.1 AMOSA without resampling strategy

In every execution of the AMOSA algorithm we have set $HL = SL = 40$, $\alpha = 0.9$, $\beta = 1.005$, $T_{init} = 1$, $T_{min} = 0.9^{80}$, $iter = 40$. HL and SL have been set to the same values in order to return a front with as many solutions as possible whilst still showing good convergence [3]. The AMOSA paper suggests that the cooling rate α should be set

[3]Even though both HL and SL are set to the same value, clustering is still relevant in these simulations, for a new solution can always be appended, causing the archive to be of length 41 in certain situations. Differences in these parameters may need to be explored in the future through parameter tuning as discussed in the section on future work

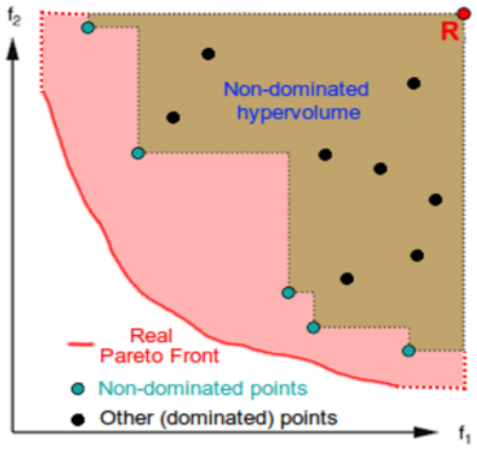


Figure 3: Hypervolume indicator [21].

between 0.5 and 0.99. The cooling rate is chosen in order to balance our efforts between exploration and exploitation of the parameter space. It also mentions that *iter* should be proportional to SL, in order for the system to reach a stationary distribution at each temperature. Additionally, an exponential iteration scheme is chosen, not only due to ease of application, but also because it allows for more refined searches by granting us with a smaller amount of exploration at higher temperatures and a larger amount of exploration at lower temperatures. We are then able to conduct the majority of our searches at lower temperatures.

The iteration scheme was chosen such that the number of iterations strictly increases with every decrease in temperature given our budget constraint. Starting with an initial value for *iter*, use $iter_{i+1} = \beta \times iter_i$ to calculate the number of iterations at every temperature, where $iter_i$ is the number of iterations at temperature T_i, in the situation where any of these values are non-integers. The floor function was used in order to allow for more exploration at lower temperatures provided that $\beta >= 1$ and is close to 1. We set $\beta = 1.005$.

The initial temperature T_{init} was set to 1 in all of our problems primarily for the sake of simplicity. Theoretically, it is better to set T_{init} in accordance with the form of the acceptance probability, allowing the first few solutions to have approximately a probability of 50% of being as the new current point.

5.2 Setting k for RTEA and AMOSA with resampling strategy

In RTEA, Fieldsend, Everson [9] suggest that for large noise levels, smaller k performs better initially on hypervolume, but is then rapidly overtaken by the RTEA algorithms using larger k. However, even with k=1 resamples per iteration of the algorithm, RTEA performs competitively with other state-of-the-art approaches.

In AMOSA we used the same parameter $k = 1$ to compare with the benchmark but we studied the behaviour of its convergence using the Hypervolume measure with k from 1 to 5 in the sensitivity analysis.

5.3 Surrogate model for RTEA

Random forests were trained, for variance analysis, on the group of mutated offspring to estimate its performance. Specifically, a mutated population of size 400 is generated, then every point is reevaluated 50 times considering the noisy environment and averaged over it to get a data label for supervised learning. Next, the reevaluated dataset was split into training and test sets with the test size of 0.25. Finally, IJ variance estimation is applied to get the standard deviations and depict them on the plot.

Figure 4 represents error bar plots which are the results from applying [18]'s method to the random forest. The dots are the fitness values of real simulation function versus the surrogate model and every point represents 50 simulation runs. The error bars estimate the sampling variance of the surrogate, indicating how much the prediction may change if it is trained on a new dataset. The error bars that do not cross the observation-versus-prediction diagonal line suggest that there exist some residual noise which cannot be explained by the surrogate model based on the available predictor features. Every error bar is +/- 1 standard deviation and following the empirical rule "68-95-99.7" of normal distribution 68.27% of values lie within one standard deviation [22]. As can be seen from the errorbar Figure 4, there is a clear linear dependency and the surrogate approximates the output of expensive simulation quite well. The RMSE value for cost function is 0.001 and for customer service level is 0.007.

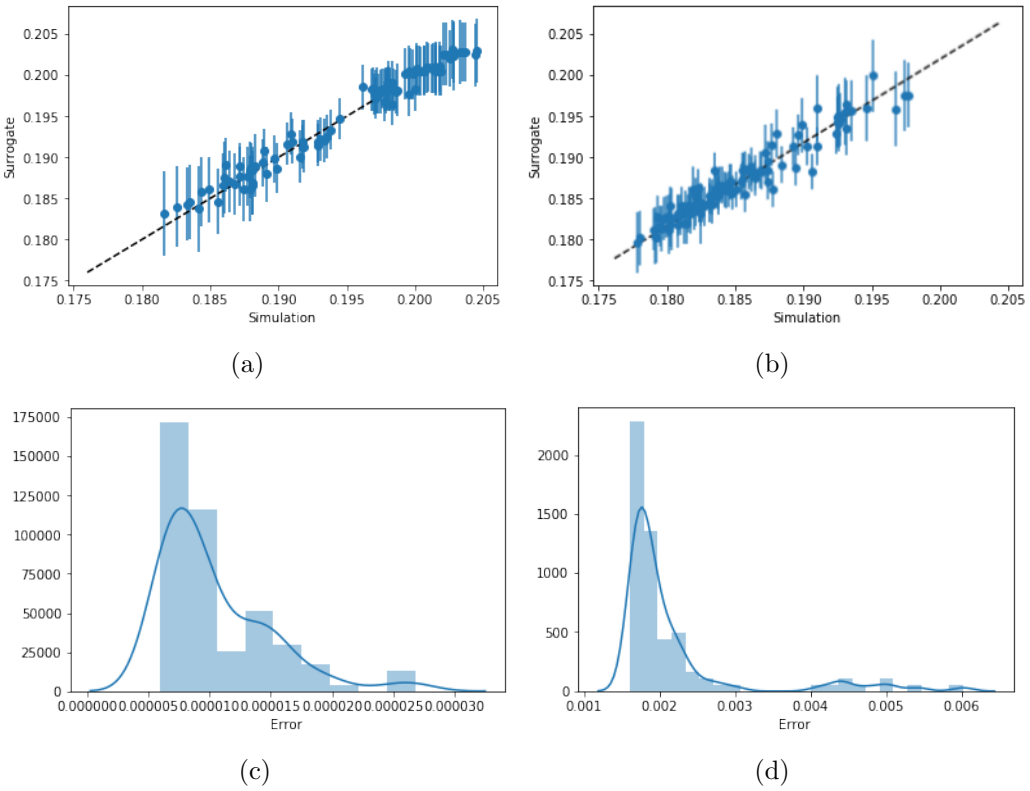


Figure 4: Random forest predictions on two objective functions. The model was tested on 100 candidates. The top left figure represents (a) cost function values, the right is (b) customer service level, (c) and (d) are their corresponding distributions of errors.

To find the best parameters for random forest we used GridSearch with cross-validation package in Python. However, as our initial purpose was to generalise the algorithm for different real-world problems, we tuned only the main two parameters which are the number of trees (estimators) and the maximum depth of a forest (pruning) that can be updated automatically depending on the generated population.

6 Results and Discussion

6.1 Comparison against the benchmark

In this section, we compare the performance of the proposed algorithms with the current benchmark NSGA-II. The results were obtained using 11 replications of the algorithm and each replication consists of 4000 evaluations of the Supply Chain simulation model.

The first column in Figure 5 shows the resulting non-dominated sets of the implemented algorithms. In the second column are boxplots generated from 11 replications of the algorithm using the hypervolume metric. In both columns we are considering two cases after the algorithm finishes the 4000 iterations.

1. The set of non-dominated points obtained are re-evaluated in the Supply Chain model 100 times.

2. The results are not re-evaluated in the Supply Chain model.

In all of our results, the reference point chosen for the hypervolume metric is $(1, -0.1)$ after changing the sign of the customer service level (this is done to make our problem a minimisation problem). Then increasing hypervolume indicates convergence to the Pareto front.

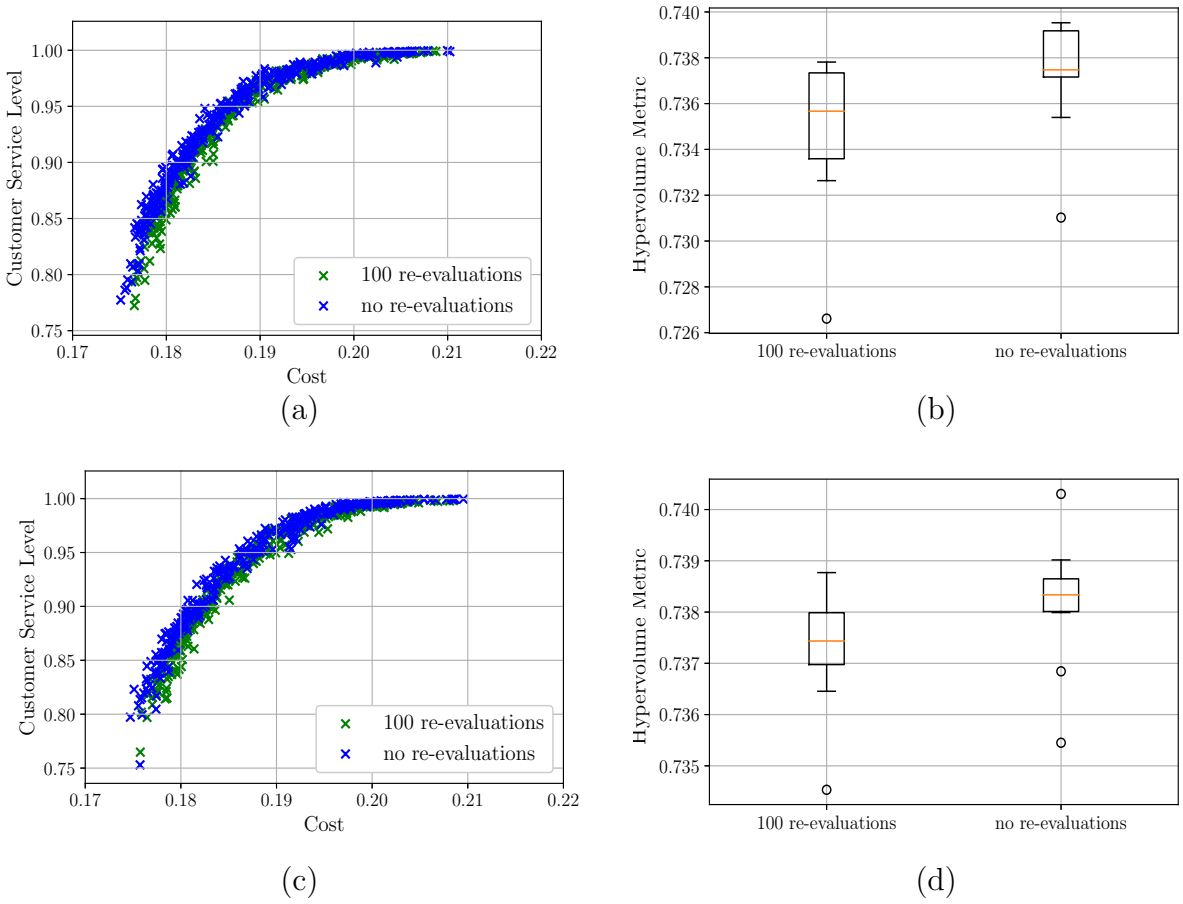


Figure 5: First column: non-dominated Pareto front for (a) RTEA(k=1), (c) RTEA(k=1) + Surrogate with the Benchmark using 11 realizations for each algorithm. Second column: Boxplot of 11 realizations of the hypervolume metric for (b) RTEA(k=1), and (d) RTEA(k=1) + Surrogate

In the case of RTEA (a) and its extension using a surrogate model (c), the quality of its estimations (see Figure 5. (b) and (c)) does not change significantly once the set of solutions $\mathbf{x}$ has been resampled. This difference may be decreased by incrementing the parameter k in the algorithm.

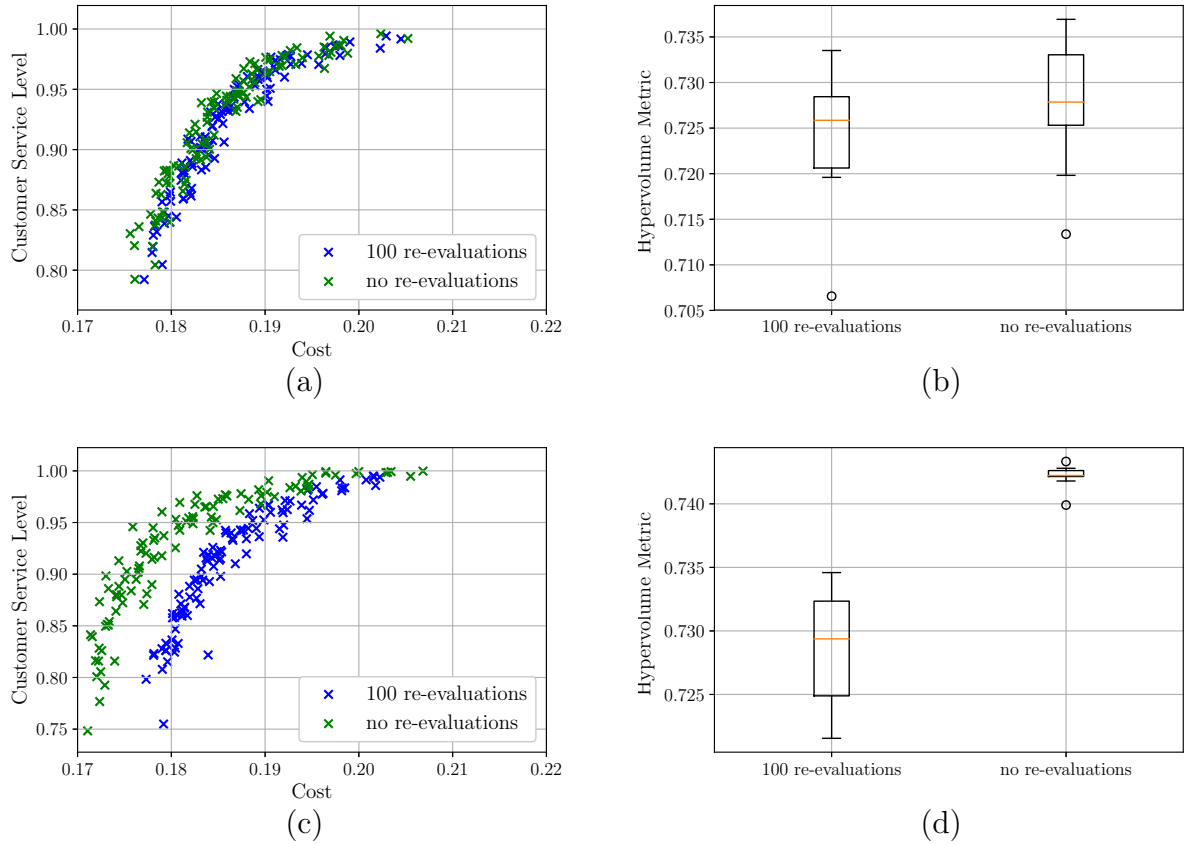


Figure 6: First column: non-dominated Pareto front for (a) AMOSA(k=1) and (c) AMOSA(k=0) compared with the Benchmark using 11 replications for each algorithm. Second column: Boxplot of 11 replications of the hypervolume metric for (b) AMOSA(k=1) and (d) AMOSA(k=0)

As expected, using AMOSA without re-sampling (AMOSA k=0) (c) translates to a decrease of the hypervolume metric after reevaluating the solutions. In contrast, where AMOSA uses RTEA's re-sampling strategy (AMOSA k=1) (a), the quality of the solutions does not change significantly after re-evaluation them which confirms that RTEA's re-sampling strategy can be implemented to other algorithms and it effectively improves the estimation of the non-dominated set.

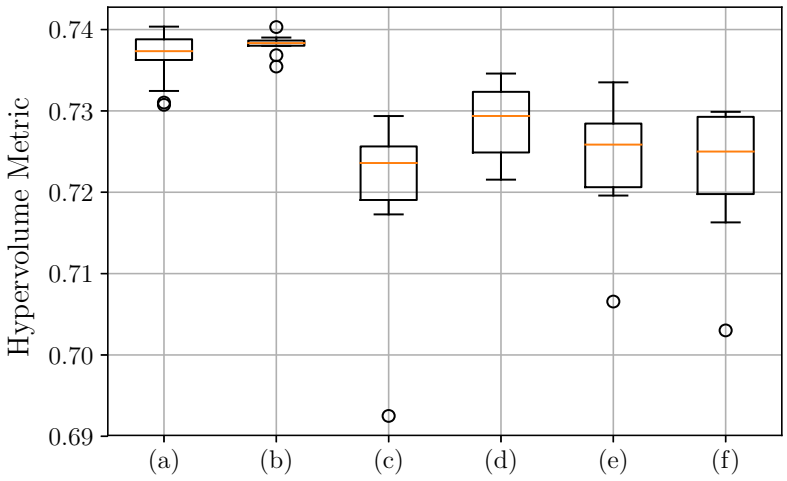


Figure 7: Boxplot of the Hypervolume metric respect to (a) RTEA (k=1), (b) RTEA(k=1) + Surrogate Model, (c) RTEA (k=0), (d) AMOSA (k=0), (e) AMOSA (k=1) and (f) the Benchmark. Each boxplot is the result of 11 realisations of the algorithm, where each point is reevaluated 100 times.

Compared with the current benchmark Figure 7 (f), it is noticeable how RTEA Figure 7 (a) and RTEA with a surrogate 7 (b) are the two algorithms that better perform among the evaluated options. Adding a surrogate model to RTEA does not translate in an improvement of the final non-dominated front. Nevertheless, the increase in performance relies on the re-sampling strategy inside the algorithm as can be seen in Figure 7 (c) where the algorithm without re-sampling decrease its performance on hypervolume metric. This suggests that a re-sampling strategy is an important part of the algorithm that improves the quality of the non-dominated solutions.

There are two reasons for why a resampling strategy works: i) any comparison inside the algorithm now is done using estimates with a reduced variance which makes more difficult for lucky solutions to be accepted and ii) the final non-dominated set has already been re-sampled in the algorithm so any effect of reduced quality once the solutions are re-sampled again is decreased.

For AMOSA algorithm, it is not possible to confirm that RTEA's re-sampling strategy improves the solutions with respect to AMOSA without re-sampling. Also, compared with the benchmark, it is not possible to conclude that the algorithm produces a better Pareto front.

6.2 Convergence Curve Comparison

In this section each algorithm is run 20 times in order to calculate the convergence curves using hypervolume as a quality measure (see Figure 8). This determines the differences in convergence rate to the non-dominated sets reached by the algorithms.

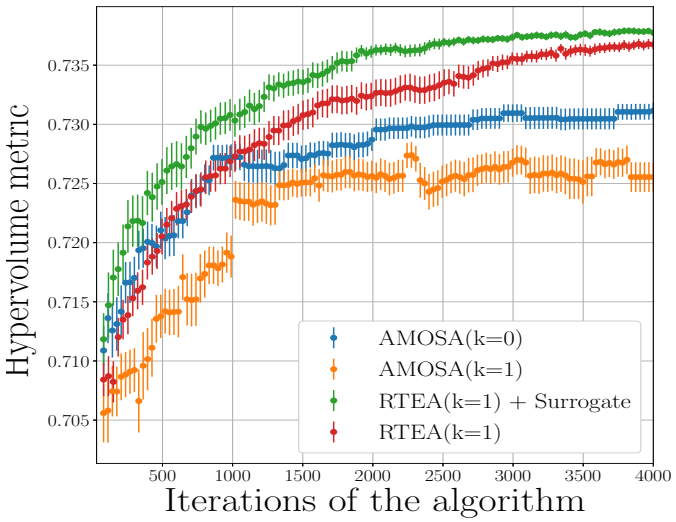


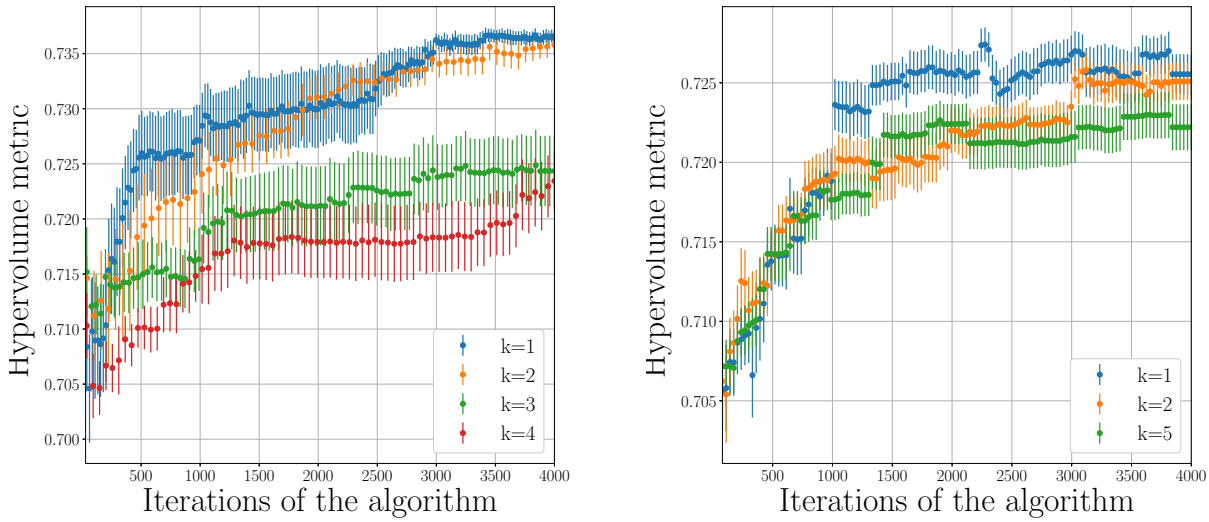
Figure 8: Convergence curve using Hypervolume metric. Each boxplot is the result of 20 realisations of the algorithm.

Compared with AMOSA, RTEA presents a faster convergence rate to a non-dominated front where the addition of a surrogate model improves the rate. In contrast, using resampling in AMOSA reflects a detrimental effect on the final solutions since the algorithm is not exploring new locations effectively by the expense of improving the current solutions.

6.3 Sensitivity Analysis

The number of resamples of archive members taken at each iteration k is inversely proportional to the number of evaluations to RTEA for searching new locations (given a limited number of total function evaluations). But the archived solutions will have estimated objective values of lower accuracy. Here, we perform a sensitivity analysis of RTEA with various numbers of resamples taken at each iteration.

Greater values of k decrease the convergence rate of the algorithm (see RTEA hypervolume Figure 9.(a) which confirms $k = 1$ or $k = 2$ as suitable parameters for this problem). In the case of AMOSA with a resampling strategy, increasing the parameter k (Figure 9 (b)) also has detrimental effects on the convergence curve for the current noise levels.



(a) (b)

Figure 9: (a) RTEA hypervolume metric for each iteration for k=1,2,3,4. (b) AMOSA hypervolume metric for each iteration for k=1,2,5. Each boxplot is the result of 20 realisations of the algorithm.

It is important to outline that the optimal parameter k varies depending on the problem and variability.

7 Conclusion

In conclusion, a survey of existing methods was conducted, where we presented the advantages and disadvantages of each method. Whilst some methods were eliminated (see Appendix, Other Methods), we eventually settled on two broad categories of tools to tackle the problem at hand - evolutionary algorithms and simulated annealing. To deal with the noisiness of the supply chain environment, a rolling tide evolutionary algorithm (RTEA) was implemented along with Archived Multi-Objective Simulated Annealing algorithm (AMOSA).

RTEA effectively outperformed the current benchmark in the hypervolume metric. To further improve the model, the random forest method was used to increase the convergence rate. However, using a surrogate model presents the drawback of increasing the number of parameters that must be tuned.

The AMOSA algorithm without resampling strategy presented the same problem as ESTECO's benchmark. Reevaluations of estimations of non-dominated solutions present a decrease in the actual value, meaning that the algorithm favors 'lucky' solutions. Given this behavior, we adapted RTEA's resampling strategy to the algorithm structure where we could improve the estimations of the Pareto front. However, an increase in accuracy did not translate to a better Pareto front, since the performance is dependent on the base

algorithm.

Furthermore, RTEA's resampling strategy can be generalised to other algorithms which only increase the number of parameters by 1, being a simple approach to improve the accuracy of a base deterministic algorithm. In addition, this parameter can be initially set as $k = 1$ and increased manually, producing more accurate results at the expense of exploration. Nonetheless, setting the parameter as $k = 0$ will tend to increment lucky solutions and low accuracy on the non-dominated front.

8 Future Work

8.1 More Benchmark Problems

The analysis made in this report corresponds to only one benchmark problem, which might limit the generalisation of the results. We suggest to use different problems with different levels of noise.

8.2 Objective Function Estimation

Currently, RTEA and AMOSA were implemented using the non-weighted mean to estimate the value for each decision vector $\mathbf{x}$ in order to compare and sort inside the structure of the algorithms. Interesting extensions could be made using the median as another choice of estimation and the use of confidence intervals in order to express the uncertainty of the estimation.

8.3 Parameter Tuning Strategies

There are multiple recommendations suggested for how parameters should be tuned for optimisation problems, however, these recommendations may vary based on the problem. Within AMOSA, the same can be said for the cooling schedule and iteration schedule. Parameters and annealing schedules were implemented within the AMOSA algorithm based on the experimental success, therefore there should be further investigation into alternative schedules. There are also many parameter searching algorithms that can be included in AMOSA and RTEA, such as the simplex method, which could in theory be used to provide us with better starting conditions for our code for finding global minima and maxima [23].

8.4 Resampling Methods

Only static re-sampling was implemented within AMOSA and RTEA. Ideally, if samples of a particular solution provide us with a small standard deviation, then we require very few samples to estimate the solution over different random seeds. The ideal amount of resampling needed would therefore vary between solutions. Dynamic sequential re-sampling aims to reduce the standard error, being the standard deviation of the sample, divided by the square root of the sample size. Multiple samples of a solution are taken until

the standard error reaches below a certain threshold level and the optimisation algorithm would then continue as usual. There are many other resampling strategies which exist and further investigations should be made into which of these strategies are best for our algorithms [24].

8.5 Surrogate Models

In this work we only used Random Forest as a surrogate model. One possible extension would be to implement different surrogate models and compare their performance. In addition, surrogates can be explored in more depth using different populations along with experiment design (DOE) tools. Also, it would be helpful to determine an optimal population size for surrogates, considering that it is the main and only criteria for surrogate models to learn insight of expensive simulation functions.

References

[1] M. Kulkarni, "Network Design Problems are refined by Genetic Algorithms," 7 2012. http://simopt.org/wiki/images/6/62/Supply.pdf.

[2] P. Rakshit, A. Konar, and S. Das, "Noisy Evolutionary Optimization Algorithms–a Comprehensive Survey," *Swarm and Evolutionary Computation*, vol. 33, pp. 18–45, 2017.

[3] J. Wu, "Supplychain," 01 2012.

[4] Y. Jo.JB, M.Gen, "Network Design Problems are refined by Genetic Algorithms," (Seoul, Korea, Republic of), pp. 1282–1289, Korean Institute of Industrial Engineers, Korean Institute of Industrial Engineers, 5 2012.

[5] M. Somma, "Example of a Pareto Frontier for a Multi-Objective Optimization Problem with two Objective Functions," 2016. https://www.researchgate.netOnline; accessed 17 May 2018.

[6] C. Darwin, *On the Origin of Species, 1859*. Routledge, 2004.

[7] Shodhganga, *Development of Genetic Algorithms for Drilling Optimization Using RSM Based Burr Size Models*, ch. 6, pp. 110–115. Shodhganga, 2015. http://shodhganga.inflibnet.ac.in/bitstream/10603/80603/16/16_chapter206.pdf.

[8] K. Deb, "Multi-Objective Optimization," in *Search Methodologies*, pp. 403–449, Springer, 2014.

[9] J. E. Fieldsend and R. M. Everson, "The Rolling Tide Evolutionary Algorithm: A Multiobjective Optimizer for Noisy Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 103–117, 2015.

[10] Y. Nourani and B. Andresen, "A Comparison of Simulated Annealing Cooling Strategies," *Journal of Physics A: Mathematical and General*, vol. 31, no. 41, p. 8373, 1998.

[11] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A Simulated Annealing-Based Multiobjective Optimization algorithm: AMOSA," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269–283, 2008.

[12] A. I. Forrester and A. J. Keane, "Recent Advances in Surrogate-Based Optimization," *Progress in Aerospace Sciences*, vol. 45, no. 1-3, pp. 50–79, 2009.

[13] Z.-H. Han and K.-S. Zhang, "Surrogate-Based Optimization," in *Real-World Applications of Genetic Algorithms*, InTech, 2012.

[14] A. Syberfeldt, H. Grimm, A. Ng, and R. I. John, "A Parallel Surrogate-Assisted Multi-Objective Evolutionary Algorithm for Computationally Expensive Optimization Problems," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 3177–3184, IEEE, 2008.

[15] I. Nitze, U. Schulthess, and H. Asche, "Comparison of Machine Learning Algorithms Random Forest, Artificial Neural Network and Support Vector Machine to Maximum Likelihood for Supervised Crop Type Classification," *Proc. of the 4th GEOBIA*, pp. 7–9, 2012.

[16] D. Verbeeck, F. Maes, K. De Grave, and H. Blockeel, "Multi-Objective Optimization with Surrogate Trees," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 679–686, ACM, 2013.

[17] I. Loshchilov, *Surrogate-Assisted Evolutionary Algorithms*. PhD thesis, Université Paris Sud-Paris XI; Institut National de Recherche en Informatique et en Automatique-INRIA, 2013.

[18] S. Wager, T. Hastie, and B. Efron, "Confidence Intervals for Random Forests: The Jackknife and the Infinitesimal Jackknife," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1625–1651, 2014.

[19] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-Based Analysis and Optimization," *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[20] T. Friedrich, C. Horoba, and F. Neumann, "Multiplicative Approximations and the Hypervolume Indicator," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pp. 571–578, ACM, 2009.

[21] D. Gaudrie, R. Le Riche, V. Picheny, B. Enaux, and V. Herbert, "Targeting Well-Balanced Solutions in Multi-Objective Bayesian Optimization under a Restricted Budget," in *PGMO Days 2017*, 2017.

[22] X. Shen and S. Agrawal, "Kernel Density Estimation for an Anomaly Based Intrusion Detection System.," 01 2006.

[23] M.-W. Park and Y.-D. Kim, "A Systematic Procedure for Setting Parameters in Simulated Annealing Algorithms," *Computers & Operations Research*, vol. 25, no. 3, pp. 207–217, 1998.

[24] F. Siegmund, A. H. Ng, and K. Deb, "Hybrid Dynamic Resampling Algorithms for Evolutionary Multi-objective Optimization of Invariant-Noise Problems," in *European Conference on the Applications of Evolutionary Computation*, pp. 311–326, Springer, 2016.

[25] M. Gendreau and J.-Y. Potvin, "Handbook of Metaheuristics," vol. 2, pp. 29–36, 2010.

Appendix

A. Fundamentals of Simulated Annealing

Within chemistry, annealing is the process where metal alloys are first heated to very high temperatures and then slowly cooled until they have reached a molecular state with a minimum amount of energy. The probability of such materials being found in a specific physical state can be modeled using statistical mechanics, where we stochastically predict the state of a physical system instead of predicting deterministically. A significant probability distribution pertaining to these processes is the Boltzmann distribution [25], which provides us with the probability P_α of a physical system being in state α with energy E_α at absolute temperature T. More specifically,

$$P_\alpha = \frac{e^{-\frac{E_\alpha}{k_B T}}}{Z(T)} \tag{6}$$

where $Z(T) = \sum_\beta e^{-\frac{E_\beta}{k_B T}}$ is the partition function and k_B is the Boltzmann constant [25].

There are several properties pertaining to this distribution that become relevant when considering Simulated Annealing. States with a lot of energy have a very small probability of acceptance and our probabilities approximate to a uniform distribution. Therefore, we are describing a physical system where every state has an equal chance of being observed. However, when the temperature approaches zero, the only states with non-zero probability of occurrence are the ones with minimal energy.

In single-objective problems, we wish to minimise a function $\mathbf{f}(x)$ given some initial constraints on the point x. We begin by randomly choosing such a point called *current point*. This point is then randomly perturbed producing a new point called *perturbed point* which also satisfies our constraints. Then we find the 'energy difference'[4],

$$\Delta\mathbf{f} = \mathbf{f}(\textit{perturbed point}) - \mathbf{f}(\textit{current point}) \tag{7}$$

between the two points. In the context of minimisation problems, if $\Delta\mathbf{f} \geq 0$, we accept the perturbed point as our new current point for the next iteration. Otherwise, we randomly choose a floating point number between 0 and 1 called r and we see whether or not $r < p$. The formula for p varies depending on which version of Simulated Annealing that is used. In this case, we are provided with a formula for p that is known as Boltzmann Annealing.

$$p = e^{-\frac{\Delta\mathbf{f}}{T}}. \tag{8}$$

If $r > p$, then our current point is unchanged going into the next iteration of the algorithm, else it becomes equal the perturbed point. Finally, we slowly lower the temperature before repeating the process again. The algorithm is terminated when a temperature lowers below a certain threshold. The pseudocode for this algorithm is provided below in algorithm 6.

[4]In the pseudocode for the single-objective case we label our function as E rather than $\mathbf{f}$.

Algorithm 6: Simulated Annealing for Single-Objective Minimisation.

Initialize a dataset.
Choose a high value of T and a low value of T_{min}.
current point = Randomly select a point from your dataset.
while $T > T_{min}$ **do**
    perturbed point = Random perturbation of current point.
    ΔE = E(perturbed point) - E(current point)
    if $\Delta E > 0$ **then**
        r = Random number between 0 and 1.
        $p = e^{-\frac{\Delta E}{T}}$
        if $r < p$ **then**
            current point = perturbed point
        else
            Keep the current point as the current point.
        end
    else
        current point = perturbed point.
    end
    $T = T \times alpha$.
end
Return current point.