Reference Point based Archived Many Objective Simulated Annealing

Raunak Sengupta[a], Sriparna Saha[b]

[a]*Department of Electrical Engineering*
[b]*Department of Computer Science and Engineering*
Indian Institute of Technology Patna, India

Abstract

Recent research in optimization theory focuses on developing algorithms for the many objective optimization problems. These problems require careful attention to efficiently handle the increasing number of objectives and also to address the various challenges associated with these types of problems. The present study introduces a new unconstrained many-objective optimization algorithm called reference point based many objective simulated annealing algorithm (RSA). The algorithm incorporates several modules including a reference point based clustering technique to control the size of the archive which stores the solutions, a novel mutation strategy termed as mutation-switching and a new acceptance probability function. Unlike the existing simulated annealing based multiobjective optimization techniques, the current work explores the use of archive-to-archive transition rather than point-to-point transition as done in the traditional simulated annealing based algorithms such as AMOSA. The results in this paper show that archive-to-archive transition yields significantly better results. The performance of RSA is validated and compared with many other state-of-the-art algorithms on a number of unconstrained benchmark problems of DTLZ and WFG series with up to 15 objectives. The experimental results show that RSA outperforms AMOSA, NSGA-III, MOEA/D-PBI and θ-DEA for a majority of the test cases considered.

Keywords: Many-objective optimization, Clustering, Pareto optimality, Simulated Annealing, Mutation switching, Acceptance Probability, Reference Points.

1. Introduction

Multiobjective optimization (MOO) deals with the optimization of more than one objective function simultaneously [1, 2]. The objectives of a MOO problem are usually conflicting. The subset of MOO problems where four or more objectives are involved is called Many-Objective Optimization (MaOO) problems [3], [4], [5] . MaOO forms an important domain of research on optimization problems where MOO algorithms face severe scalability issues. Several practical applications where MaOO algorithms have been used are control systems [6, 7], automotive engine calibration [8], software engineering [9, 10], industrial scheduling [11, 12], land use management [13], etc.

AMOSA (archived multiobjective simulated annealing) [14] which is a MOO algorithm based on simulated annealing was shown to outperform existing evolutionary MOO techniques like NSGA-II [15] and PAES [16] when applied to optimize a large number of objectives. However it suffers from producing a non-uniform Pareto front for certain kinds of functions, especially functions like DTLZ4 which has a biased density of solutions in the search space. Recently, algorithms such as MOEA/D [17], NSGA-III [18] and θ-DEA [19] which are based on genetic algorithms have been developed and shown to perform well for optimization problems with the number of objectives reaching as high as 15. θ-DEA introduced the concept of θ-domination which is based on the concepts of PBI (Penalty based Boundary Intersection) of MOEA/D-PBI, and combined it with non-dominated sorting used in NSGA-III. These algorithms show the effectiveness of reference point

Email addresses: raunaksengupta@gmail.com (Raunak Sengupta), sriparna.saha@gmail.com (Sriparna Saha)

based selection operations. Efforts have also been made to develop many-objective optimization algorithms based on other meta-heuristics such as differential evolution, particle swarm optimization (PSO) etc. but not so much using the concepts of simulated annealing. However, the single objective simulated annealing algorithm is associated with strong mathematical understanding and it is a popular approach for single objective optimization. The convergence proof of SA also exists in the literature [14].

Motivated by the concepts proposed in the recent reference-point based algorithms, in the current study we have developed a new many-objective optimization algorithm. The developed algorithm is named as RSA (reference point based simulated annealing algorithm). Several new concepts are embodied in RSA. The major motivations of developing RSA and the key contributions of this paper are enlisted below :

1. Simulated annealing is a popular single objective optimization technique supported by strong mathematical basis. But there have been very few attempts in recent past on extending SA to solve many-objective optimization problems.
2. Due to the poor diversity of solutions obtained from AMOSA, RSA concentrates strongly on diversity preservation. It has been proven mathematically in Section 3.8 that the diversity preservation operator used in RSA (clustering operation) yields a monotonically increasing diversity throughout the iterations.
3. Owing to the fact that the Archive as used in RSA has a co-existence of the newly added points as well as relatively old points, it often provides improved directions (vectors pointing from old points towards new points) with a certain probability as has been discussed in a later section (Section 3.9). This feature often helps in finding better solutions faster.
4. The concept of archive-to-archive transition has been introduced in this paper and its usage in RSA has been presented along with an experimental analysis showing its advantages. A discussion on this is presented in Section 2.
5. Studies and experiments performed on the acceptance probability function used in the original AMOSA suggest several shortcomings. Hence a new acceptance probability function has been designed and is shown to be more effective.
6. A new mutation strategy referred to as 'Mutation-Switching' has been used. This strategy is algorithm independent and can be used with other algorithms as well.

In order to prove the efficacy of RSA, several benchmark test problems having objectives in the range of 3 to 15 are considered. The results of RSA are compared with those obtained by NSGA-III [18], MOEA/D-PBI [17] and θ-DEA [19] in terms of the corresponding IGD (Inverse Generational Distance) values. RSA has also been compared with other algorithms which include ones which are not reference-point based such as HypE [20], GrEA [21] and AMOSA [14] in terms of the HV(Hypervolume) value. The experimental results clearly reveal the improved convergence rate of RSA as compared to other mentioned algorithms for a majority of the test problems. Further studies have also been done to experimentally show the motivation behind using archive-to-archive movement. The superiority of RSA is further supported by the Friedman test.

The paper is structured as follows. Section 2 discusses the meaning of the terms 'archive-to-archive (A2A)' transition and 'point-to-point (P2P)' transition. The next section elaborates the RSA algorithm in details. It starts with the key-concepts of RSA and eventually discusses all the steps of RSA along with intuition behind the devised mechanism. Section 3 also presents a theoretical analysis of the clustering algorithm to further strengthen the argument behind using it. Section 3 is followed by the results section(Section 4) which presents all the results obtained along with pertaining discussions. Section 4 starts with the features of the test problems and the hyper-parameter settings of all the algorithms. This is followed by comparison of all the algorithms on DTLZ and WFG test suite. Next, section 4 presents a comparison between the performances for A2A and P2P transitions and shows the superiority of A2A transitions. This section also includes a very detailed hyper-parameter sensitivity analysis which shows results for a large number of hyper-parameter settings, points out the kinds of problems not suited to RSA and also presents the robustness of RSA to hyper-parameter settings. The paper finally ends with the conclusion section.

2. Point-to-Point and Archive-to-Archive transitions

In case of single objective simulated annealing, a new solution at time 't' is obtained by applying perturbation operation on the current-solution at time 't-1' . The simulated annealing based multi-objective optimization algorithms developed so far also mimic this step to generate new solutions. A single solution is selected randomly from the current archive and this is participated in generating the new solution. Depending on conditions, the decision regarding the next current point is taken. The new current point is utilized for generating the new solution in the next iteration. This process is repeated for multiple iterations at a given temperature.

However, it must be realized that unlike single objective optimization problems, the goal of multi-objective optimization algorithms is to obtain a Pareto optimal set of solutions. Hence, the search space consists of all possible solution sets. In the cases when an archive is used to store the solutions, an alternative search strategy could be to apply a set mutation or variation operator on the current archive to generate a new archive which might or might not be better. In this paper, the former strategy has been termed as point-to-point (P2P) transition and the later strategy as archive-to-archive (A2A) transition. Studies in this paper show that the A2A strategy yields significantly better results than the P2P strategy. The set variation of the archive has been performed by selecting a random point from the archive and applying perturbation to it. There can be possibly many other ways to perform set variation.

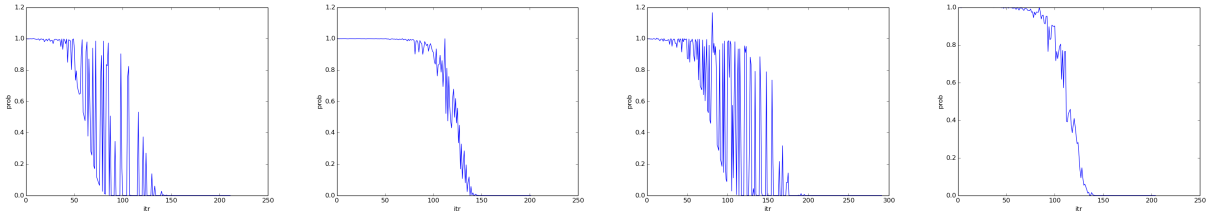
This concept is very similar to the 'Population of sets' concept introduced in [22], where a population of populations/sets are used simultaneously and operations are performed on them to finally end up with a good set of solutions. This is analogous to single objective GAs. However, single objective simulated annealing transits from one point to another rather than maintaining a population. The presented algorithm, RSA similarly jumps from one archive to another archive of solutions in search of a good archive.

3. Reference Point Based Many Objective Simulated Annealing Technique (RSA)

In this section we discuss in detail the different steps of the proposed simulated annealing based many objective optimization technique, namely RSA and the underlying reasons behind the mechanisms.

The key-concepts used in RSA are the following:

1. Clustering Mechanism -
   Similar to AMOSA, RSA also uses an Archive to store the solutions. The size of the Archive is kept limited. Hence a clustering algorithm is necessary to reduce the size when it exceeds some threshold. AMOSA uses single linkage clustering to cluster the points in the objective space. However, this mechanism is not always able to preserve the diversity. Hence, a reference point based clustering technique has been introduced and incorporated into RSA. This mechanism preserves diversity, thus resulting into a monotonically increasing diversity as proven in Section 3.8 .
2. Acceptance Probability Function -
   Experiments and analysis (Section 3.2) suggest that the acceptance probability function used in AMOSA has a lot of shortcomings and often fails to act as a proper acceptance probability function. If one observes the values throughout the iterations, the value of the probability function remains very close to 0.5. Principally, it should start with a very high value (close to one) and gradually decrease to a very low value. A new probability function has been designed and used in RSA which follows this principle (Fig 1).
3. Quantifying Amount of Improvement -
   The amount of domination concept used in AMOSA introduced a method to quantify the amount of improvement a new solution results into. However, it is specific to AMOSA. A new way of quantifying the amount of improvement a particular solution entails if added to the Archive has been proposed in RSA.
4. Archive-to-Archive(A2A) Transition -
   AMOSA and other simulated annealing based MOO algorithms use point-to-point(P2P) transition. RSA however employs A2A transition. This improves the results significantly as shown in Section 4.6. An intuitive reason has been presented in Section 3.10.



(a) (b) (c) (d)

Figure 1: Plots of acceptance probability against iterations (200 equidistant samples from iteration axis) of RSA for 3-objective problems (a) DTLZ1 (b) DTLZ2 (c) DTLZ3 (d) DTLZ4

5. Maintaining a Mixed Archive -
   Unlike AMOSA which maintains a non-dominated archive throughout, RSA accepts solutions which are worse than before thus enabling the algorithm to escape from local optimal fronts.

3.1. Basic Steps of RSA

The algorithm starts with a random archive of solutions. The archive is limited in size having two limits, *HardLimit* and *SoftLimit*. The archive is initialized with some *SoftLimit* number of random solutions. The initialization process is described in detail in Section 3.3 .

For each iteration at a particular temperature, a random solution is chosen from the archive referred to as the $CurrentPt$. This solution undergoes mutation according to the *perturb* function which has been described in Section 3.6 . This mutated solution is referred to as the $NewPt$. This is followed by identification of solutions in the archive which are dominated by the $NewPt$ and the solutions in the archive which dominate the $NewPt$. These solutions are stored in arrays named 'l' and 'k' respectively. Depending on the domination status of the $NewPt$, following cases may be considered :

1. Case 1: If the $CurrentPt$ dominates $NewPt$, the $NewPt$ is added to the archive with a probability p as calculated according to Equation 1.
2. Case 2: If $NewPt$ dominates the $CurrentPt$, two subcases are considered:
   (a) If the size of l is greater than the size of k, the $NewPt$ is added to the archive and the points in the archive dominated by the $NewPt$ are removed from the archive.
   (b) Otherwise, the $NewPt$ is added to the archive with the probability as calculated in Equation 1.

Clustering is performed whenever the number of points in the archive exceeds $SoftLimit$. The process continues for some number of iterations per temperature. The temperature is gradually decreased with some cooling rate. The basic steps of RSA are shown in Algorithm 1.

3.2. New Acceptance Probability Function

Since we are employing A2A transitions, the acceptance probability must be designed to quantify the improvement in the archive as a whole. Let us assume that the solutions in the archive which are dominated by the $NewPt$ are stored in the array l and the solutions in the archive which dominate the $NewPt$ are stored in the array k. The sizes of these arrays are used to calculate the acceptance probability of $NewPt$. The number of solutions in the archive which are dominated by the $NewPt$ can be considered as a measure of improvement that the $NewPt$ introduces in the archive. Similarly, the number of solutions in the archive that dominate $NewPt$ can be considered as a measure of deterioration that the $NewPt$ introduces. The difference between these two values is referred to as the amount of dominance of a new mutated archive over the previous archive, unlike AMOSA where the amount of dominance is calculated by the area between two solutions in the objective space. The main disadvantage of the method used in AMOSA is that the calculation of the area usually results into a value very close to zero. This is because the area is the product of a number of fractions (equal to the number of objective functions). The acceptance probability being a

sigmoidal function therefore produces values very close to 0.5 for all the iterations without much variance. The new probability function for accepting the new solution, $NewPt$, is given below :

$$p = e^{\frac{(l.size - k.size)}{(arch.size * temp)}} \tag{1}$$

Here $arch.size$ is the total number of solutions on the archive and $temp$ is the current value of temperature. In this equation, $(l.size - k.size)$ is the amount of domination which has been normalized by dividing it by the archive size. The equation is therefore similar to the one which is traditionally used in single objective simulated annealing algorithm whose characteristics are well studied.

We observe from Figure 1 that the acceptance probability values for the initial iterations are almost 1. As the iterations continue, the acceptance probability in general starts decreasing. During the end, the acceptance probability drops to 0. During these iterations, the algorithm is trying to exploit rather than explore. We observe that the probability value from the formula might exceed 1. This happens because in the case 2, the value of $(l.size() - k.size())$ might turn out to be positive. In such cases, the probability is taken to be 1.

Algorithm 1 RSA

Input parameters: T_{max} : Starting temperature value, T_{min} : Minimum temperature value, *iter* : number of iterations per temperature, *SoftLimit* : soft limit of the Archive, *HardLimit* : hard limit of the Archive, α : cooling rate.
`arch = Randomly Initialized Archive`
$temp = T_{max}$ ▷ Starting temperature value set
while $temp \geq T_{min}$ **do** ▷ Until the temperature reaches the minimum after cooling
    for `i = 1 :  iter` **do**
        $r \leftarrow random(0, arch.size)$
        $NewPt \leftarrow perturb(CurrentPt)$ ▷ Generate new point
        $l \leftarrow$ pts in $arch$ dominated by $NewPt$
        $k \leftarrow$ pts in $arch$ that dominate $NewPt$
        $prob \leftarrow \exp((l.size - k.size)/(arch.size * temp))$ ▷ Acceptance probability
        if ($CurrentPt$ dominates $NewPt$) **then** ▷ Selection Mechanism
            if $rand(0,1) < prob$ **then**
                $AddToArchive(NewPt)$
                if $arch.size > SoftLimit$ **then**
                    $cluster()$
        else
            if $l.size > k.size$ **then**
                $AddToArchive(NewPt)$
                $RemoveFromArchive(l)$
                if $arch.size > SoftLimit$ **then**
                    $cluster()$
            else
                if $rand(0,1) < prob$ **then**
                    $AddToArchive(NewPt)$
                    if $arch.size > SoftLimit$ **then**
                        $cluster()$
    $temp \leftarrow \alpha * temp$ ▷ Cooling mechanism

3.3. Initialization

The algorithm starts with the construction of reference points on a unit hyperplane of dimension same as the number of objective functions. The points can be chosen according to the requirements of the user.

Reference points may be placed with a higher density in the regions where the user thinks would be necessary. In the absence of any preference, one might place the points in a structured manner as mentioned by Das and Dennis [23]. The algorithm would attempt to retain points nearest to the lines formed by joining these reference points to the origin.

The algorithm also requires to initialize an archive. This initialization step is very similar to that of AMOSA [14]. $\gamma \times SoftLimit$ number of points are randomly initialized where γ is just a multiplying factor.

3.4. Adaptive Normalization

First, the ideal solution of the archive is determined by finding the minimum (we assumed that all the objectives are of minimization type) values for different objectives over all the solutions in the archive. We thus obtain the ideal point $z_{ideal} = (z_1^{min}, z_2^{min}, ..., z_M^{min})$. Thereafter, each point in the archive is translated by subtracting the corresponding co-ordinates of the ideal point from it.

Next we determine the extreme points in each objective by minimizing the corresponding Achievement Scalarizing Function (ASF). These extreme points are used to construct a hyperplane whose intercepts with each of the objective axes are found out. In case of degeneracy or negative intercepts, we simply use the maximum value in each objective over all the translated points in the archive. The objective functions are then normalized by dividing each objective value with the calculated intercept value for all the points in the archive. The pseudo-code for adaptive normalization is given by Algorithm 2.

One might also perform adaptive normalization using corner sort [24] as done in [25].

Algorithm 2 Adaptive_Normalization

M = number of objectives, arch : current archive, arch.size : size of archive, ASF : Achievement Scalarizing Function

for `i = 1 : M` **do**
    $z_i^{min} \leftarrow min_{a \in arch} a_i$
    for $j = 1 : arch.size$ **do**
        $normalized[j]_i \leftarrow (arch[j]_i - z_i^{min})$
for `i = 1 : M` **do**
    $ExtremePts[i] \leftarrow argmin_{j \in arch}(ASF_j)$
$ConstructHyperplane()$
$CalculateIntercepts()$
if (Duplicate Points Exist || Negative Intercepts) **then**
    $intercept_i \leftarrow max_{a \in archive} a_i$
for `i = 1 : M` **do**
    for $j = 1 : arch.size$ **do**
        $normalized[j]_i \leftarrow normalized[j]_i / intercept_i$

3.5. Association Operation

The Association function simply associates each point in the archive with the nearest reference line. The Euclidean metric has been used to find the distance. It creates two arrays - *line_pointer* and *population_spread*. The *line_pointer* array is of the same length as that of the Archive. Each element of this array corresponds to an unique point in the archive and stores the index of the line nearest to the point under consideration as well as the distance of the said point from the line. The *population_spread* array is of the same length as that of the reference-line array and stores the number of associated points corresponding to each line.

3.6. Perturb

A new mutation strategy referred to as *Mutation Switching* has been introduced in this paper and is one of the key components of RSA. Mutation switching involves switching periodically between two or more mutation techniques. This kind of switching between mutation techniques often helps in combining the pros

Algorithm 3 Association

```
arch: current archive
for i = 1 : arch.size do
    a1 ← index(nearestlinetoarch[i])
    a2 ← distance(arch[i], ref_line[a1])
    line_pointer[i] ← [a1, a2]
    population_spread[a1] + +
```

of individual mutation techniques involved. RSA uses a combination of Simulated Binary crossover (SBX) [26] and a differential evolution based mutation. RSA also switches to traditional mutation techniques such as polynomial mutation and Laplacian sampling based mutation. The crossover and mutation techniques are mentioned below :

1. SBX Mutation - In the Simulated Binary Crossover (SBX) scheme, the procedure for generating a new solution from an old solution is as follows - for a parent solution, we first choose another random point in the archive. This is followed by usual SBX crossover between the two solutions which produces two candidate solutions. Out of the two, only the first one is chosen for further processing. The expressions for SBX crossover are as follows :

$$x_i^{1,t+1} = 0.5[(1+\beta_{q_i})x_i^{1,t} + (1-\beta_{q_i})x_i^{2,t}],$$
$$x_i^{2,t+1} = 0.5[(1-\beta_{q_i})x_i^{1,t} + (1+\beta_{q_i})x_i^{2,t}]$$

   where, $x_i^{1,t+1}$ and $x_i^{2,t+1}$ are the two new solutions, $x_i^{1,t}$ and $x_i^{2,t}$ are the two parent solutions and β_{q_i} is sampled from the probability distribution:

$$P(\beta_i) = \left\{ \begin{array}{ll} 0.5(\eta_c+1)\beta_i^{\eta_c}, & \text{if } \beta_i \leq 1 \\ 0.5(\eta_c+1)\frac{1}{\beta_i^{\eta_c+2}}, & \text{otherwise} \end{array} \right\}$$

2. Differential Evolution based mutation - Differential evolution based mutation basically involves generating a new point from three different randomly chosen points from the archive, **a**, **b** and **c** using the usual expression: $\mathbf{z} = \mathbf{a} + F \times (\mathbf{b} - \mathbf{c})$ which is followed by binomial crossover.
3. Polynomial mutation - The newly generated point is given by the expression:

$$y_i^{1,t+1} = x_i^{t+1} + (x_i^U - x_i^L)\delta_i$$

   where, i is the i-th variable in the decision space, t refers to the generation number, x_i^U is the upper bound of the i-th variable, x_i^L is the lower bound of the i-th variable and δ_i is calculated from the polynomial probability distribution:

$$P(\delta) = 0.5(\eta_m+1)(1-|\delta|)^{\eta_m}$$

4. Laplacian Sampling based mutation - In Laplacian sampling based mutation, the parameter to be perturbed is chosen at random and perturbed with a random variable drawn from a Laplacian distribution $p(\epsilon) \propto e^{-\|\sigma\epsilon\|}$, where the scaling factor σ sets the magnitude of the perturbation.

Laplacian mutation alone fails to produce a good Pareto optimal front in case of functions such as DTLZ4 and others which have some biased density of solutions in the search space. Differential evolution based mutation followed by performing crossover between this new point and the current point, performs extremely well in producing a good spread of points for functions such as DTLZ4. However, this mutation technique suffers from the problem of getting stuck at local optima. Use of Laplacian and Polynomial mutation removes the problem of getting stuck at local optimal fronts. Further, performing SBX crossover between the current point and a randomly chosen point from the archive improves the convergence.

Algorithm 4 Perturb

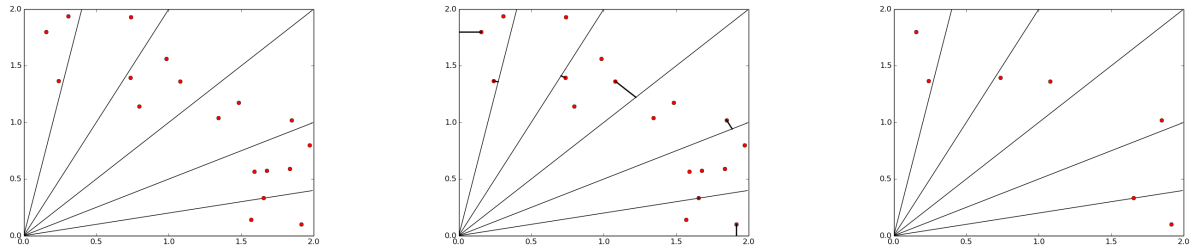
$iter$ = current iteration count at some temperature
tot_iter = total number of iterations per temperature
$switch\_factor$= mutation switching factor
$flag1$, $flag2$ = determine the use of polynomial mutation

if $iter > (tot\_iter \times switch\_factor)$ **then**
    $NewPt \leftarrow differential\_mut(CurrentPt)$
    if $flag2 == 1$ **then**
        $NewPt \leftarrow polynomial\_mut(NewPt)$
    if $rand(0,1) < prob_2$ **then**
        $NewPt \leftarrow laplacian\_mut(NewPt)$
else
    $NewPt \leftarrow SBX\_mut(CurrentPt)$
    if $flag1 == 1$ **then**
        $NewPt \leftarrow polynomial\_mut(NewPt)$
    if $rand(0,1) < prob_1$ **then**
        $NewPt \leftarrow laplacian\_mut(NewPt)$

One solution is to switch between the techniques periodically. Mutation switching uses a parameter - *switch_factor*, which is a number between 0 and 1. This factor defines the fraction of iterations per temperature that will perform SBX mutation. Mutation switching produces an uniform spread of points in the global optimal front. Interestingly this strategy also reduces the number of function evaluations required to converge to the global optimal front by a factor of about 1.2 - 1.5 times when compared to using any one of the mutation techniques. The mutation strategy uses parameters $flag1$, $flag2$, $prob1$ and $prob2$. Setting $flag1$ results into the $NewPt$ obtained from SBX_mut to again pass through $polynomial\_mut$. $prob1$ is the probability of the $NewPt$ passing through real_mutation. $flag2$ and $prob2$ play some similar roles for the case of $differential\_mut$. Experimental results suggest that having small values for $prob1$ or $prob2$ often prevent the algorithm from getting stuck at local optima. However, large values of $prob1$ and $prob2$ result into poor convergence. Different combinations of the hyper-parameters mentioned above are used for different kinds of problems, thus increasing flexibility of the algorithm. The results of a detailed hyper-parameter sensitivity analysis have been presented in Section 4.7 . The Mutation Switching strategy can be seen as an alternative to adaptive variation of hyper-parameters which are difficult to engineer. Adaptive algorithms have to spend a large number of iterations with sub-optimal hyper-parameter settings in order to search the best hyper-parameter setting. For the cases where the landscape of the problem at hand is partially known, it might be more efficient to use the hyper-parameter settings specified in Section 4.7 which presents a detailed hyper-parameter sensitivity analysis.

3.7. Reference Point based Clustering Algorithm

The aim of the reference point based clustering algorithm is to remove points from the archive in a way such that the values of the *population_spread* array derived after association become more uniform. Such removal of points achieves two purposes - removes poor solutions and reduces computational burden. To do this, the index of the *population_spread* array with the highest corresponding value is first found out. This index corresponds to the reference line with the highest number of associated points. Among all the points associated with this reference line, the one which is farthest is removed from the archive. This procedure is continued until the archive size is brought down to the $HardLimit$. The procedure is presented in Algorithm 4. The process has also been demonstrated in Figure 2.



(a) (b) (c)

Figure 2: The association and clustering in 3 steps.(a) Initial Archive (b) Association with the reference lines (c) Archive after elimination of points

Algorithm 5 Cluster

arch: current archive, *cnt*: counter
$Nor \leftarrow Adaptive\_Normalization(arch)$
$line\_pointer, population\_spread \leftarrow Association(Nor)$
while $arch.size > HardLimit$ **do**
    $ind \leftarrow argmax(population\_spread)$
    $max\_pop \leftarrow population\_spread[ind]$
    $max\_d \leftarrow -1$
    $cnt \leftarrow 0$
    for $i = 1 : arch.size$ **do**
        if $line\_pointer[i]_{index} == ind$ **then**
            if $line\_pointer[i]_{dist} > max\_d$ **then**
                $ind\_r = i$
                $max\_d = line\_pointer[i]_{dist}$
                $cnt++$
                if $cnt == max\_pop$ **then**
                    break
    $archive.delete(ind\_r)$
    $Nor.delete(ind\_r)$
    $population\_spread[line\_pointer[ind\_r]_{index}] --$
    $line\_pointer.delete(ind\_r)$

3.8. Proof of Monotonic Improvement in Diversity during Population Reduction

In this section, we categorically prove that the 'population reduction' mechanism in the clustering algorithm does result into a monotonic improvement in diversity, while, the solution selection mechanism might lead to a decrease in diversity at times. We also show that during the later iterations when all the solutions are mostly non-dominated to each other, monotonic improvement in diversity is achieved all at a time. We assume that Adaptive Normalization is not used (i.e., the objective functions have the same scale) to ignore the effect of co-ordinate shifting and scaling. To prove these statements, we use the *D_metric* from [27] to quantify the diversity of population.

$$D\_metric^g = \frac{n}{pop\_size}\sqrt{\sum_{i=1}^{n}(S_i^g - S_{ideal})^2} \tag{2}$$

where, g is the current generation, n is the number of reference lines, pop_size is the number of population members, S_i^g is the number of members associated to the i-th reference line at generation g, $S_{ideal} = \frac{pop\_size}{n}$ and $\sum_{i=1}^{n} S_i^g = pop\_size$.

The *D_metric* captures the uniformity of the spread of population members throughout the region covered by the reference lines. A lower value of *D_metric* would indicate a more uniform distribution and therefore a better diversity. Ideally, the value of *D_metric* should be 0 which occurs when all the reference lines have equal number of associated members with a value equal to S_{ideal}.

For our analysis, we assume $pop\_size = n$. Therefore $S_{ideal} = 1$ and -

$$D\_metric^g = \sqrt{\sum_{i=1}^{n}(S_i^g - 1)^2} \tag{3}$$

where, $\sum_{i=1}^{n} S_i^g = n$ and $S_i^g \in I$.

Now, let us consider the case where after generation g, the evolutionary algorithm that is being executed has resulted into a population such that the m-th reference line has lost an associated member while k-th reference line has added another population member i.e., $S_m^{g+1} = S_m^g - 1$ and $S_k^{g+1} = S_k^g + 1$ (for the equality $\sum_{i=1}^{n} S_i^g = n$ to maintain). We first prove that *D_metric* captures the intuition that a transfer of a point from a reference line with lower number of associated points to a reference line with higher number of associated points results into loss of diversity.

For loss in diversity :

$$\begin{aligned}
& D_metric^{g+1} > D_metric^g \\
\implies & \sqrt{\sum_{i=1}^{n}(S_i^{g+1} - 1)^2} > \sqrt{\sum_{i=1}^{n}(S_i^g - 1)^2} \\
\implies & \sum_{i=1}^{n}(S_i^{g+1} - 1)^2 > \sum_{i=1}^{n}(S_i^g - 1)^2 \\
\implies & (S_m^{g+1} - 1)^2 + (S_k^{g+1} - 1)^2 > (S_m^g - 1)^2 + (S_k^g - 1)^2 \\
\implies & (S_m^g - 2)^2 + (S_k^g)^2 > (S_m^g - 1)^2 + (S_k^g - 1)^2 \\
\implies & S_m^g < S_k^g + 1
\end{aligned}$$

From this we note two points :

1. Transition of a point from any reference line with lower associated points to a reference line with higher associated points leads to an increase in the value of *D_metric*, thus resulting into a loss in diversity.
2. Removal of a point from reference line with $S_m = 1$ never leads to improvement.

Now, in RSA, the clustering operation is performed only when the population size of the Archive exceeds $SoftLimit$. After the clustering and reduction of population, the population size is restored back to $HardLimit$. We choose the value of $HardLimit$ to be equal to the number of reference lines, n. The clustering operation has two phases - the $AssociationPhase$ and the $PopulationReductionPhase$.

AssociationPhase -
This phase involves associating each point with a reference line. This phase does not remove solutions or change position of any solution and thus does not contribute to variations in diversity.

PopulationReductionPhase -
The population reduction mechanism finds the reference lines with the highest associated number of population members and removes the farthest member. To prove that this mechanism yields a monotonic improvement in diversity, we consider two cases -

1. A member was removed from some reference line l. Therefore,

$$D\_metric_1^{g+1} = \sqrt{\sum_{i=1, i\neq l}^{n} (S_i^g - 1)^2 + (S_l^g - 2)^2} \tag{4}$$

2. A member was removed from a different reference line k. Therefore,

$$D\_metric_2^{g+1} = \sqrt{\sum_{i=1, i\neq k}^{n} (S_i^g - 1)^2 + (S_k^g - 2)^2} \tag{5}$$

For $D\_metric_1^{g+1}$ to be better than $D\_metric_2^{g+1}$:

$$\begin{aligned} & D\_metric_1^{g+1} < D\_metric_2^{g+1} \\ \implies & (S_l^g - 2)^2 + (S_k^g - 1)^2 < (S_l^g - 1)^2 + (S_k^g - 2)^2 \\ \implies & S_l^g > S_k^g \end{aligned} \tag{6}$$

Since, the population removal mechanism finds reference lines with highest value of S_i^g, it always satisfies the above inequality (6) and thus leads to the maximum possible decrease in the value of D_metric. It also yields an improvement in diversity after each iteration of population member removal.

Also, during the Population Reduction phase, the inequality $\sum_{i=1}^{n} S_i > n$ is maintained. As soon as this inequality stops holding true, the population reduction phase ends. Let us assume that during this phase, there is a reference line r with $S_r = 1$. To maintain the inequality, there must exist at least one reference line, p such that $S_p > 1$. Therefore, the mechanism while removing population members, never chooses reference lines with number of associated members equal to 1 and consequently never renders any reference line empty which would otherwise worsen the diversity. Hence, it has been proven that the employed clustering mechanism always yields a monotonic improvement in diversity and never renders a reference line with a single associated member empty.

However, the selection mechanism of RSA might contribute to a deterioration of the diversity of solutions with a certain probability. In between two consecutive calls to the clustering operation, it might so happen that some of the parent solutions get removed from those reference lines which have only one associated point. This happens when Case 2(a) as mentioned in Section 3.1 gets satisfied i.e., $NewPt$ dominates $CurrentPt$ and the number of solutions it dominates in the archive is greater than the number of ones which dominate it. However, in the later iterations, the solutions are mostly non-dominated in which case the condition leading to Case 2(a) will not be satisfied. In such cases, all of the old points are retained for the clustering operation and in the worst case scenario, there will be no new point to associate to the empty reference line. Thus, in the later iterations, a monotonic improvement in the diversity can be expected.

3.9. How Archives Inherently Facilitate Finding Improvement Directions

An archive maintains a co-existence of newly added solutions as well as the parent solutions. The mating scheme for finding the mates while perturbing a solution assigns equal probability to all the other solutions in the archive. With a certain probability, a solution from the previous generation but associated to the same or neighboring reference lines may be chosen. The direction vector from the old solution to the new solution in the decision space will point towards a direction where the probability of obtaining a better solution has a higher probability than other random directions. This case is similar to the situation in [28]. However, no explicit mechanism has been devised to take advantage of this. This probability is automatically embedded in the use of archive as in RSA.

3.10. Why A2A Transition ?

The aim of a single objective optimization algorithm is to produce a single optimum solution which is either the maximum or the minimum of the given objective function. Hence a single objective simulated annealing algorithm uses a P2P transition for obvious reasons. However, majority of the literature proposes the usage of P2P transition in multiobjective simulated annealing algorithms as well. [29, 30, 14] The aim of a MOO algorithm is to produce a set/archive of points which approximates the Pareto optimal front as precisely as possible. Hence an A2A transition should yield better results. Mutation of an archive can simply be defined as the perturbation of one of the points in the archive selected at random. This perturbed/mutated archive would result into a new archive which is either better or worse than the old archive. In the former case, we start our next iteration from the new archive. In the latter case, we start our next iteration with the new archive only with a certain probability calculated according to the probability expression mentioned in Equation 1. Hence, we keep perturbing the archive until we get a satisfactory archive.

We also observe that for a single-objective optimization case, the aim is to produce an archive containing only one point. However, the mutation of such an archive is equivalent to the mutation of a single point. Therefore, P2P transition is a special case of the A2A transition. The advantages of using an A2A transition over P2P transition is illustrated experimentally in the results section 4.6 .

3.11. Computational Complexity of RSA

In this section we have analyzed the computational complexity of the proposed approach, RSA. Let the following be the list of symbols : M $\rightarrow$ number of objectives, HL $\rightarrow$ value of Hard Limit, SL $\rightarrow$ value of Soft Limit, tot_iter $\rightarrow$: total number of function evaluations.
Procedure to check domination status between $CurrentPt$ and $NewPt$ is $O(M)$
Procedure to check domination status between $NewPt$ and the $Archive$ is $O(M \times SL)$
The following operations from 1 to 6 are sub-operations of the $Cluster$ operation .

1. Identification of ideal points : $O(M \times SL)$
2. Identification of extreme points : $O(M^2 \times SL)$
3. Determination of intercepts : $O(M^3)$
4. Normalization of the Archive : $O(M \times HL)$
5. Association of members of the Archive : $O(SL \times HL \times M)$
6. Deletion of members from archive : $O((SL - HL) \times (SL + HL))$

Therefore, the complexity of one iteration of RSA becomes
$M + (M \times SL) + \frac{1}{(SL-HL)}((M \times SL) + (M^2 \times SL) + (M^3) + (SL \times HL \times M) + ((SL - HL) \times (SL + HL)))$
$\Rightarrow O(M \times SL)$

Overall complexity of the algorithm is $O(M \times SL \times tot\_iter)$ where tot_iter is the total number of function evaluations of RSA calculated using the values of $Tmax$, $Tmin$ and $iter$ as mentioned in Section 4.3 .

4. Results and Discussion

In this section, we present a comparison of RSA with other state of the art algorithms on DTLZ1 - DTLZ4 test functions from the DTLZ test suite [31] and WFG1 - WFG9 test functions from WFG test suite [32]. For each DTLZ test function, the number of objectives is set as $M \in \{3, 5, 8, 10, 15\}$. For each WFG test function, the number of objectives is set as $M \in \{3, 5, 8, 10\}$. According to the recommendations of Ref. [31], the number of decision variables for a particular DTLZ test function is set as $n = M + r - 1$, where $r = 5$ for DTLZ1 and $r = 10$ for DTLZ2, DTLZ3 and DTLZ4. Different specifications have been used in the literature for WFG test functions. This study uses specifications as used in the papers [19, 33, 34]. The number of decision variables is set as $n = 24$. The position-related variable $k = M - 1$ and the distance-related variable, $l = n - k$. WFG2 and WFG3 functions require the distance-related variable to be an even number. Hence for WFG2 and WFG3 test functions with M = 8 and M = 10, we set the value of n as 23.

This paper uses two comparison metrics, IGD (Inverse Generational Distance) value [35] for comparison among reference-point based algorithms and Hypervolume value [36] for a more general comparison. The approach towards calculating HV is same as that in Ref. [19] and has also been elaborated later. We also perform a comparison between RSA and RSA* (RSA with P2P transition instead of A2A) which demonstrates the superiority of archive-to-archive transition strategy. We use spherical co-ordinate plots [37] for visualizing the final Pareto fronts. The algorithms involved in the comparisons can be classified into two types - reference-point based and non reference-point based. Reference-point based algorithms include RSA, MOEA/D-PBI [17], $\theta - DEA$ [19], NSGA-III [18] and dMOPSO [38]. Non reference-point based algorithms include GrEA [21], HypE [20] and AMOSA [14].

4.1. Features of the Test Problems

In practical life, the problems that need to be optimized can have varied landscapes. The possible features of a problem at hand have been classified into five different types in the literature [32]:

1. *Geometry* : Shape of final Pareto front can be Convex, Concave, Linear, Mixed, Degenerate
2. *ParameterDependencies* : Separable Objectives, Non-Separable Objectives
3. *Bias* : Presence of a bias in the mapping from decision space to objective space
4. *ManytoOnemappings* : Pareto one-to-one , Pareto many-to-one , Flat regions, Isolated Optima
5. *Modality* : Uni-modal, Multi-Modal (Presence of Local Optimal Fronts)

The features of the various WFG and DTLZ problems that have been considered in the study are given in Table 1 .

4.2. Comparison Metrics

The IGD metric can provide a combined information about the convergence and diversity of the obtained solutions. In reference point based algorithms, the targeted points are calculated by finding the point of intersection of the provided reference lines with the true Pareto optimal surface. These targeted points z_i are computed and named as Z_{eff}. For any algorithm, we obtain the final archive of points in the objective space and call them the set **A**. Now, we compute the IGD metric as the average Euclidean distance of points in set Z_{eff} with their nearest members of all points in set A as -

$$IGD(A, Z_{eff}) = \frac{1}{|Z_{eff}|} \sum_{i=1}^{|Z_{eff}|} min_{j=1}^{|A|} d(z_i, a_j)$$

where, $d(z_i, a_j)$ is the Euclidean distance between z_i and a_j. A smaller value of IGD results into the conclusion that the obtained points are closer to their associated reference points and are thus better. If say a part of the obtained Pareto front is missing, this will result into a higher value of $d(z_i, a_j)$ for the empty reference lines and thus will increase the value of the resultant IGD metric.

Table 1: TEST PROBLEM FEATURES

Problem	**Features**
DTLZ1	Linear, Multi-Modal
DTLZ2	Concave
DTLZ3	Concave, Multi-Modal
DTLZ4	Concave, Biased
WFG1	Mixed, Biased, Scaled
WFG2	Convex, Disconnected, Multi-Modal, Scaled, Non-Separable
WFG3	Linear, Degenerate, Scaled, Non-Separable
WFG4	Concave, Multi-modal, Scaled
WFG5	Concave, Deceptive, Scaled
WFG6	Concave, Scaled, Non-Separable
WFG7	Concave, Biased, Scaled
WFG8	Concave, Biased, Scaled, Non-Separable
WFG9	Concave, Biased, Multi-Modal, Deceptive, Scaled, Non-Separable

Table 2: PARAMETER SETTINGS OF RSA FOR DIFFERENT TEST PROBLEMS

Problem	**flag1**	**flag2**	**switch**	**prob1**	**prob2**
DTLZ1	false	true	0.75	0.1	0.0
DTLZ2	false	false	0.75	0.0	0.1
DTLZ3	true	false	0.75	0.1	0.0
DTLZ4	false	false	0.75	0.0	0.0
WFG1	false	false	0.75	0.0	0.0
WFG2	false	false	0.5	0.1	0.2
WFG3	false	true	0.25	0.2	0.1
WFG4	false	true	1.0	0.0	0.1
WFG5	false	false	0.5	0.0	0.0
WFG6	false	false	1.0	0.0	0.2
WFG7	false	false	0.75	0.0	0.1
WFG8	false	true	0.75	0.2	0.0
WFG9	false	true	0.75	0.0	0.2

However, the IGD metric is not applicable for evaluation of algorithms not based on reference points such as HypE, GrEA etc. For comparing such algorithms with RSA, the Hypervolume indicator has been used. It has been shown in [36] that the Hypervolume indicator is strictly Pareto compliant. Let $r = (r_1, r_2, ..., r_m)$ be a reference point in the objective space which is dominated by any point in the obtained archive. To make sure this happens, we set r to $1.1 * z_{nad}$. Since the objective functions might have different scales, we normalize the objective values of the points in the archive and the reference point, r using the ideal point z^* (which is a 0 vector) and the nadir point z^{nad} of the exact Pareto front. The hypervolume is calculated using the expression -

$$HV(A, r) = volume(\cup_{f \in A}[f_1, r_1] \times ... \times [f_m, r_m])$$

The number of objectives for the problems for which the HV has been calculated do not exceed 10. Hence, the HV values have been evaluated exactly without any approximations.

4.3. Parameter Settings for Algorithms

The parameters for other comparing algorithms are set as suggested in [19],[39],[21],[18] and [17]. The parameter settings for all the algorithms have also been categorically specified below :

1. *ReproductionOperatorParameters* : The values of η_c and η_m for RSA, NSGA-III and θ-DEA are set as 30 and 20 respectively. For the other algorithms using SBX crossover and polynomial mutation, η_c and η_m are set as 20 and 20, respectively. RSA additionally uses a differential evolution based mutation and Laplacian mutation. We set F = 0.5 and CR = 0.2 for differential mutation and r_mut = 0.25 for Laplacian mutation. For algorithms using SBX crossover, the crossover probability is set as 1.0 and the mutation probability is set as 1/n where n is the input dimension. The size of $SoftLimit$ is usually taken to be 2-3 times that of $HardLimit$ (Values mentioned in Table 3) . The rest of the parameter settings for RSA are different for different test functions and given in Table 2 .
2. *PenaltyParameter* : The algorithms θ-DEA, MOEA/D-PBI and dMOPSO require a penalty parameter θ. For the comparisons performed, θ is set to a value of 5.

Table 3: SETTING OF DIFFERENT PARAMETER VALUES

No. of Objectives	Divisions (Outer, Inner)	No. of reference lines	NSGA-III population size	HardLimit	SoftLimit
3	12,0	91	92	91	300
5	6,0	210	212	210	500
8	3,2	156	156	156	400
10	3,2	275	276	275	600
15	2,1	135	136	135	400

3. $NeighborhoodSize$: The neighborhood size for MOEA/D-PBI is set as 20.
4. $SamplingSize$: HypE approximates the hypervolume using Monte Carlo approximation. The Monte Carlo sampling size is set to 10,000 as given in [39].
5. $GridDivisions$: The grid divisions of GrEA for different test functions are according to the guidelines in [21].
6. $NumberofFunctionEvaluations$: The simulated annealing based algorithms, RSA and AMOSA do not have generations unlike the other algorithms. The number of function evaluations for RSA and AMOSA are defined by the values of T_{max}, T_{min}, α(cooling rate) and $iter$. The number of function evaluations for simulated annealing based algorithms is given by the expression :

$$tot\_iter = iter \times \lfloor \log_{\alpha}(\frac{T_{min}}{T_{max}}) \rfloor$$

For RSA, we have set the values of T_{max}, T_{min} and α as 100, 0.000001 and 0.99 respectively. For AMOSA the values are set as 100, 0.000001 and 0.8 as used in the original implementation . The value of itr has been chosen such that the number of function evaluations is as close as possible to that of the other algorithms. The population size and the number of divisions for reference point based algorithms are the same as used in the paper [19]. The number of generations are also listed in the comparison tables. The corresponding value of $iter$ for RSA and AMOSA for each of the cases is obtained by simply solving the equation :

$$pop\_size \times num\_of\_gen = iter \times \lfloor \log_{\alpha}(\frac{T_{min}}{T_{max}}) \rfloor$$

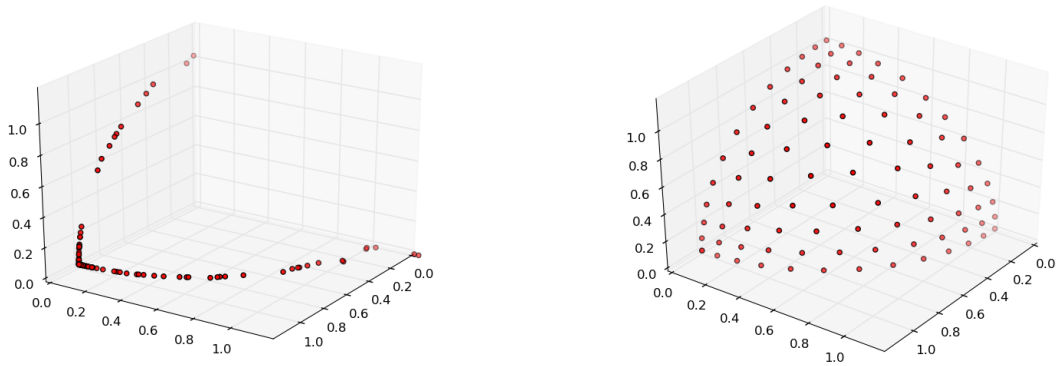
The $iter$ values used for RSA corresponding to DTLZ problems have been explicitly mentioned in Tables 5, 6 and that for WFG test problems have been mentioned in Table 4 .
7. The final values of the parameters - $flag1$, $flag2$, $prob1$, $prob2$ and $switch$ are given in Table 2 . Even though the number of parameters seems to be quite high, their appropriate values can be selected after consulting small ranges. Even though the table lists different values for the switch factor, a typical value of 0.75 works well for most cases. The values of $flag1$ and $flag2$ can be either true or false. It can be observed that the value of $flag1$ has been kept false for most of the cases. The values of $prob1$ and $prob2$ vary between 0.0 and 0.2. A higher value results in deterioration of performance. The different parameters for different problems show the importance of mutation operators and also give hints about the nature of the problems. A detailed sensitivity analysis of these hyper-parameters has been performed in Section 4.7 .

4.4. Comparison on DTLZ problems

In this section we compare the algorithms on normalized DTLZ1 - DTLZ4 problems. The first comparison given in Table 5 and Table 6 involves algorithms all of which are reference point based. Hence we use IGD

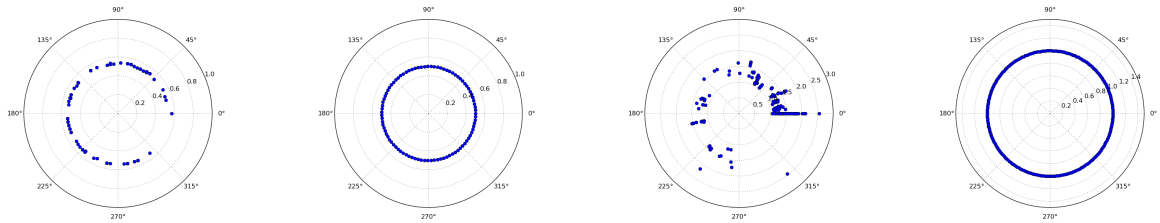
Table 4: Iter and Tot_iter VALUES FOR RSA CORRESPONDING TO WFG FUNCTIONS

No. of Objectives	**Iter(WFG)**	**Tot_iter(WFG)**
3	20	36640
5	86	157552
8	128	234496
10	300	503800



(a) (b)

Figure 3: Final Pareto fronts produced for DTLZ4 by (a) AMOSA (b) RSA



(a) (b) (c) (d)

Figure 4: Final Pareto fronts produced for 3-objective DTLZ1 by (a) AMOSA (b) RSA; and final Pareto fronts produced for 5-objective DTLZ2 by (c) AMOSA (d) RSA

values for the first comparison. The second comparison given in Table 7 involves algorithms which are not reference point based. We use the hypervolume metric for comparing such algorithms.

The challenge posed by the DTLZ1 problem is convergence to the global Pareto optimal front. It has a linear PF given by $\sum_{i=1}^{M} f_i = 0.5$ but with $(11^k - 1)$ local optimal fronts. We observe that the performance of RSA on DTLZ1 is significantly superior than all the other algorithms namely NSGA-III, MOEA/D-PBI and θ-DEA. This shows that RSA does not suffer from getting stuck at local optima when the hyper-parameters are correctly chosen as given in Table 2 . AMOSA on the other hand is not able to reach the global PF for higher number of objectives and thus reports a hypervolume value of 0. The combinations (SBX + Laplacian) and (Diff + Polynomial) turn out to be extremely good for problems such as DTLZ1 which is linear and multi-modal.

DTLZ2 is a simple uni-modal test problem which has a concave PF. From Table 5 , we see that the performance of MOEA/D-PBI is quite better than the other algorithms. RSA mostly outperforms the other two algorithms. This time RSA does not use polynomial mutation at all. It only performs Laplacian mutation 10% of the times after differential mutation. The spherical co-ordinate plots comparing RSA and AMOSA are given in Figure 4.

DTLZ3 poses the challenge of overcoming the local optimal fronts just like DTLZ1. It has $(3^k - 1)$ local fronts, all parallel to each other. Like DTLZ2, DTLZ3 has a spherical PF of radius 1 as well. The performance of the algorithms on the DTLZ3 problem is distributed. We observe that for higher number of objectives, i.e., 8 and above, MOEA/D-PBI suffers from getting stuck at local optimal fronts as can be seen by the large magnitudes of the worst IGD values. RSA does not suffer from any such kind of problems and produces consistent results. The obtained IGD values of RSA, θ-DEA and NSGA-III are comparable. However the corresponding average HV value for RSA is higher than that of the other algorithms. This can be explained by the fact that the worst IGD values of RSA are in general lower than those of the other algorithms as can be seen from Tables 5 and 6 .

The DTLZ4 problem introduces the challenge of maintaining diversity among the final set of obtained points due to the existence of bias in the mapping of DTLZ4 from decision space to the objective space. We observe that MOEA/D-PBI performs extremely poorly for all the cases of DTLZ4 as compared to the other algorithms. This is probably because MOEA/D-PBI involves mating constraints. From Figure 3, we also observe the poor performance of AMOSA on DTLZ4. This shows the importance of considering the whole population when performing mutation/mating for problems like DTLZ4 which have a biased density of solutions. RSA clearly outperforms all the other algorithms when evaluated on the DTLZ4 problems. It is interesting to note that for such functions, it is a better choice to not use Laplacian or Polynomial mutation. The use of only population based mutation operators suffices to obtain good results.

4.5. Comparison on WFG Problems

The WFG test suite introduces a methodical way of introducing various complexities and features into different test functions that can be used to test an algorithm. Table 8 presents a comparison of RSA with the other state of the art algorithms on the WFG test suite with parameter settings as mentioned before.

Table 5: BEST, MEDIAN, AND WORST IGD VALUES FOR RSA, MOEA/D-PBI, θ-DEA AND NSGA-III ON M-OBJECTIVE DTLZ 1 - 2 PROBLEMS WHERE M IS VARIED OVER THE SET {3,5,8,10,15}. BEST PERFORMANCES ARE SHOWN IN BOLD.

Problem	M	MaxGen	Iter	Tot_iter	RSA	MOEA/D-PBI	$\theta - DEA$	NSGA-III
DTLZ1					$\mathbf{8.522 \times 10^{-5}}$	4.095×10^{-4}	5.655×10^{-4}	4.880×10^{-4}
	3	400	20	36640	$\mathbf{1.727 \times 10^{-4}}$	1.495×10^{-3}	1.307×10^{-3}	1.308×10^{-3}
					$\mathbf{2.513 \times 10^{-3}}$	4.743×10^{-3}	9.449×10^{-3}	4.880×10^{-3}
					$\mathbf{8.055 \times 10^{-5}}$	3.179×10^{-4}	4.432×10^{-4}	5.116×10^{-4}
	5	600	69	126408	$\mathbf{9.831 \times 10^{-5}}$	6.372×10^{-4}	7.328×10^{-4}	9.799×10^{-4}
					$\mathbf{1.372 \times 10^{-4}}$	1.635×10^{-3}	2.138×10^{-3}	1.979×10^{-3}
					$\mathbf{1.686 \times 10^{-3}}$	3.914×10^{-3}	1.982×10^{-3}	2.044×10^{-3}
	8	750	64	117248	$\mathbf{1.987 \times 10^{-3}}$	6.106×10^{-3}	2.704×10^{-3}	3.979×10^{-3}
					$\mathbf{2.312 \times 10^{-3}}$	8.537×10^{-3}	4.620×10^{-3}	8.721×10^{-3}
					$\mathbf{1.155 \times 10^{-3}}$	3.872×10^{-3}	2.099×10^{-3}	2.215×10^{-3}
	10	1000	150	274800	$\mathbf{1.324 \times 10^{-3}}$	5.073×10^{-3}	2.448×10^{-3}	3.462×10^{-3}
					$\mathbf{2.126 \times 10^{-3}}$	6.130×10^{-3}	3.935×10^{-3}	6.869×10^{-3}
					$\mathbf{2.747 \times 10^{-3}}$	1.236×10^{-2}	2.442×10^{-3}	2.649×10^{-3}
	15	1500	111	203352	$\mathbf{3.167 \times 10^{-3}}$	1.431×10^{-2}	8.152×10^{-3}	5.063×10^{-3}
					$\mathbf{4.825 \times 10^{-3}}$	1.692×10^{-2}	2.236×10^{-1}	1.123×10^{-2}
DTLZ2					6.932×10^{-4}	$\mathbf{5.432 \times 10^{-4}}$	1.042×10^{-3}	1.262×10^{-3}
	3	250	13	23816	8.892×10^{-4}	$\mathbf{6.406 \times 10^{-4}}$	1.569×10^{-3}	1.357×10^{-3}
					1.082×10^{-3}	$\mathbf{8.006 \times 10^{-4}}$	5.497×10^{-3}	2.114×10^{-3}
					2.293×10^{-3}	$\mathbf{1.219 \times 10^{-3}}$	2.720×10^{-3}	4.254×10^{-3}
	5	350	40	73280	2.945×10^{-3}	$\mathbf{1.437 \times 10^{-3}}$	3.252×10^{-3}	4.982×10^{-3}
					3.428×10^{-3}	$\mathbf{1.727 \times 10^{-3}}$	5.333×10^{-3}	5.863×10^{-3}
					9.816×10^{-3}	$\mathbf{3.097 \times 10^{-3}}$	7.786×10^{-3}	1.371×10^{-2}
	8	500	43	78776	1.494×10^{-2}	$\mathbf{3.763 \times 10^{-3}}$	8.990×10^{-3}	1.571×10^{-2}
					1.722×10^{-2}	$\mathbf{5.198 \times 10^{-3}}$	1.140×10^{-2}	1.811×10^{-2}
					6.545×10^{-3}	$\mathbf{2.474 \times 10^{-3}}$	7.558×10^{-3}	1.350×10^{-2}
	10	750	112	205184	7.880×10^{-3}	$\mathbf{2.778 \times 10^{-3}}$	8.809×10^{-3}	1.528×10^{-2}
					9.139×10^{-3}	$\mathbf{3.235 \times 10^{-3}}$	1.020×10^{-2}	1.697×10^{-2}
					9.792×10^{-3}	$\mathbf{5.254 \times 10^{-3}}$	8.819×10^{-3}	1.360×10^{-2}
	15	1000	74	135568	1.084×10^{-2}	$\mathbf{6.005 \times 10^{-3}}$	1.133×10^{-2}	1.726×10^{-2}
					1.274×10^{-2}	$\mathbf{9.409 \times 10^{-3}}$	1.484×10^{-2}	2.114×10^{-2}

Table 6: BEST, MEDIAN, AND WORST IGD VALUES FOR RSA, MOEA/D-PBI, θ-DEA AND NSGA-III ON M-OBJECTIVE DTLZ 3 - 4 PROBLEMS WHERE M IS VARIED OVER THE SET {3,5,8,10,15}. BEST PERFORMANCES ARE SHOWN IN BOLD.

Problem	M	MaxGen	Iter	Tot_iter	RSA	MOEA/D-PBI	$\theta - DEA$	NSGA-III
DTLZ3					$\mathbf{3.683 \times 10^{-4}}$	9.773×10^{-4}	1.343×10^{-3}	9.751×10^{-4}
	3	1000	50	91600	$\mathbf{5.795 \times 10^{-4}}$	3.426×10^{-3}	3.541×10^{-3}	4.007×10^{-3}
					$\mathbf{1.443 \times 10^{-3}}$	9.113×10^{-3}	5.528×10^{-3}	6.665×10^{-3}
					2.064×10^{-3}	$\mathbf{1.129 \times 10^{-3}}$	1.982×10^{-3}	3.086×10^{-3}
	5	1000	115	210680	2.883×10^{-3}	$\mathbf{2.213 \times 10^{-3}}$	4.272×10^{-3}	5.960×10^{-3}
					$\mathbf{3.867 \times 10^{-3}}$	6.147×10^{-3}	1.911×10^{-2}	1.196×10^{-2}
					2.095×10^{-2}	6.459×10^{-3}	$\mathbf{8.769 \times 10^{-3}}$	1.244×10^{-2}
	8	1000	85	155720	2.724×10^{-2}	1.948×10^{-2}	$\mathbf{1.535 \times 10^{-2}}$	2.375×10^{-2}
					6.898×10^{-2}	1.123	$\mathbf{3.826 \times 10^{-2}}$	9.649×10^{-2}
					7.209×10^{-3}	$\mathbf{2.791 \times 10^{-3}}$	5.970×10^{-3}	8.849×10^{-3}
	10	1500	225	412200	9.230×10^{-3}	$\mathbf{4.319 \times 10^{-3}}$	7.244×10^{-3}	1.188×10^{-2}
					$\mathbf{1.416 \times 10^{-2}}$	1.010	2.323×10^{-2}	2.083×10^{-2}
					1.110×10^{-2}	$\mathbf{4.360 \times 10^{-3}}$	9.834×10^{-3}	1.401×10^{-2}
	15	2000	148	271136	1.802×10^{-2}	$\mathbf{1.664 \times 10^{-2}}$	1.917×10^{-2}	2.145×10^{-2}
					7.692×10^{-2}	1.220	6.210×10^{-1}	$\mathbf{4.195 \times 10^{-2}}$
DTLZ4					$\mathbf{9.987 \times 10^{-5}}$	2.929×10^{-1}	1.866×10^{-4}	2.915×10^{-4}
	3	600	30	54960	$\mathbf{1.250 \times 10^{-4}}$	4.280×10^{-1}	2.506×10^{-4}	5.970×10^{-4}
					$\mathbf{2.613 \times 10^{-4}}$	5.234×10^{-1}	5.320×10^{-1}	4.286×10^{-1}
					$\mathbf{1.758 \times 10^{-4}}$	1.080×10^{-1}	2.616×10^{-4}	9.849×10^{-4}
	5	1000	115	210680	$\mathbf{2.093 \times 10^{-4}}$	5.787×10^{-1}	3.790×10^{-4}	1.255×10^{-3}
					$\mathbf{3.851 \times 10\text{-}4}$	7.348×10^{-1}	4.114×10^{-4}	1.721×10^{-3}
					$\mathbf{1.516 \times 10^{-3}}$	5.298×10^{-1}	2.780×10^{-3}	5.079×10^{-3}
	8	1250	106	194192	$\mathbf{1.942 \times 10^{-3}}$	8.816×10^{-1}	3.098×10^{-3}	7.054×10^{-3}
					$\mathbf{2.474 \times 10^{-3}}$	9.723×10^{-1}	3.569×10^{-3}	6.051×10^{-1}
					$\mathbf{6.490 \times 10^{-4}}$	3.966×10^{-1}	2.746×10^{-3}	5.694×10^{-3}
	10	2000	300	549600	$\mathbf{8.312 \times 10^{-4}}$	9.203×10^{-1}	3.341×10^{-3}	6.337×10^{-3}
					$\mathbf{1.108 \times 10^{-3}}$	1.077	3.914×10^{-3}	1.076×10^{-1}
					$\mathbf{9.775 \times 10^{-4}}$	5.890×10^{-1}	4.143×10^{-3}	7.110×10^{-3}
	15	3000	221	404872	$\mathbf{1.446 \times 10^{-3}}$	1.133	5.904×10^{-3}	3.431×10^{-1}
					$\mathbf{1.759 \times 10^{-3}}$	1.219	7.680×10^{-3}	1.073

Table 7: MEAN HV VALUES FOR RSA AND OTHER STATE OF THE ART ALGORITHMS. BEST PERFORMANCES ARE SHOWN IN BOLD. HERE M DENOTES THE NUMBER OF OBJECTIVE FUNCTIONS.

Problem	M	RSA	MOEA/D-PBI	θ-DEA	NSGA-III	GrEA	HypE	dMOPSO	AMOSA
DTLZ1	3	**1.120313**	1.116679	1.116137	1.117600	1.072987	1.117483	1.074976	1.108762
	5	**1.578100**	1.577632	1.576983	1.577027	1.509905	1.466936	1.482412	0.000000
	8	**2.138472**	2.136337	2.137924	2.137837	2.105894	1.999087	1.824428	0.000000
	10	**2.592921**	2.592233	2.592719	2.592792	2.566547	2.520526	2.317805	0.000000
DTLZ2	3	0.745160	0.744137	0.743778	0.743523	0.740172	**0.753191**	0.712523	0.748851
	5	**1.310423**	1.307343	1.306928	1.303638	1.304274	0.756079	1.239853	0.889425
	8	1.981071	1.978216	1.977904	1.969096	**1.989406**	0.892917	1.816420	1.078543
	10	**2.515577**	2.515040	2.514259	2.508717	2.515566	1.174930	2.428399	0.406965
DTLZ3	3	0.744855	0.736044	0.736938	0.737407	0.678608	**0.750325**	0.665529	0.735126
	5	**1.308800**	1.303168	1.303987	1.301481	1.135837	0.740621	1.252229	1.014567
	8	**1.979989**	1.251873	1.968943	1.954336	1.622438	0.881516	1.428208	1.268642
	10	**2.523104**	2.406221	2.512662	2.508879	2.306975	1.175350	2.107556	0.812540
DTLZ4	3	**0.744847**	0.406020	0.729265	0.744634	0.573359	0.549999	0.677459	0.685146
	5	**1.309178**	1.205512	1.308945	1.308698	1.307539	1.014145	1.203429	1.036436
	8	1.980854	1.826489	1.980779	1.980236	**1.981321**	0.925370	1.829561	1.156142
	10	**2.515523**	2.423727	2.515436	2.515172	2.514960	1.235517	2.438748	1.488669

WFG1 investigates an algorithm's ability to obtain the PF of a function with flat/polynomial bias and mixed geometry. We see that RSA performs rather poorly for this test function and outperforms only dMOPSO. MOEA/D-PBI gives a high performance for the 5-objective case. However its performance relatively decreases with an increase in the number of objectives. $\theta - DEA$ and GrEA provide a more consistent set of results. The exact source of difficulty leading to the poor performance of RSA has been investigated in a later section.

WFG2 introduces the challenge of obtaining a disconnected convex set of fronts. We observe that since the problem is multi-modal, RSA requires Laplacian mutation operators for the best performance. HypE gives the best performance for the 3-objective case. RSA outperforms the other algorithms for 5 and 8 objectives.

WFG3 has a linear and degenerate Pareto front. We observe that the performance of RSA for 3 objectives is relatively poor. However for higher number of objectives, RSA gives competitive results. For 10 objectives, RSA gives the second highest hypervolume value and is next only to HypE.

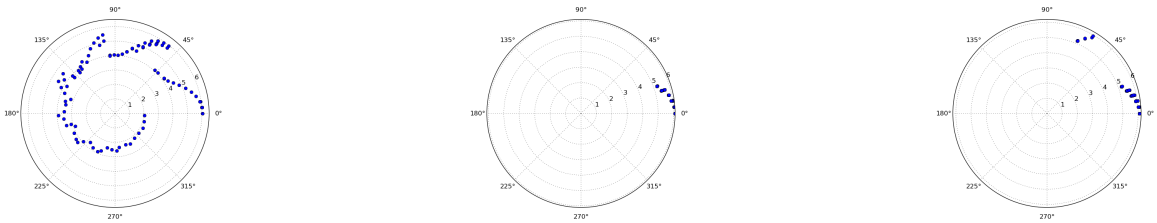
The problems WFG4-WFG9 have the same PF geometry which is a part of a hyper-ellipse with radii r_i = 2i where $i \in 1, 2, ..., M$. WFG4 is featured by the local optimal fronts. RSA uses both Laplacian and Polynomial mutation for this problem. From the HV values, we observe very competitive results by RSA. RSA is next only to the best for all the cases of WFG4.

We see the superiority of RSA for the functions WGF5 - WFG8. RSA performs significantly better than the other algorithms for these functions. RSA is able to cope with the deceptive nature of WFG5 and bias present in WFG7 and WFG8 test functions and clearly outperforms the other algorithms. RSA produces good results for the WFG9 functions as well. The effectiveness of RSA becomes more apparent from Figure 5. We shall also see in a later section that RSA is very robust in terms of hyper-parameter settings for these problems

Finally from the results, we conclude that RSA might be a worthy alternative for certain kinds of problems. It is clearly able to cope with various problem features such as multi-modality, deceptive nature, non-separability and scaled functions which often prove challenging to other algorithms.

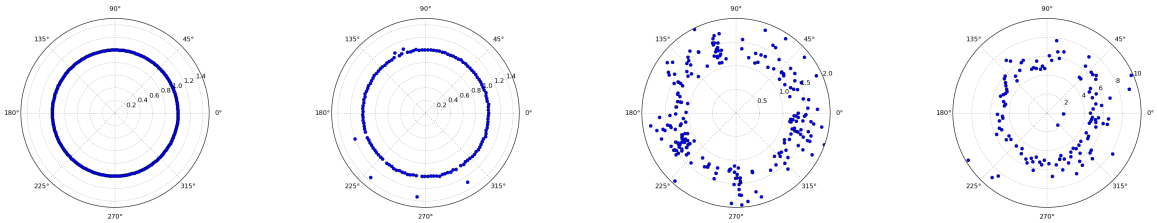
Table 8: MEAN HV VALUES FOR RSA AND OTHER STATE OF THE ART ALGORITHMS FOR WFG FUNCTIONS. BEST PERFORMANCES ARE SHOWN IN BOLD. HERE M DENOTES THE NUMBER OF OBJECTIVES.

Problem	M	MaxGen	RSA	MOEA/D-PBI	θ-DEA	NSGA-III	dMOPSO	GrEA	HypE
WFG1	3	400	0.673171	0.657143	0.704526	0.669729	0.403170	0.846287	**0.976181**
	5	750	0.727470	**1.349888**	1.138794	0.859552	0.461233	1.268898	0.911020
	8	1500	0.705425	1.755326	**1.875997**	1.424963	0.484046	1.769013	1.536599
	10	2000	0.505525	1.799394	2.364268	2.249535	0.536340	**2.365107**	2.268813
WFG2	3	400	1.231475	1.111085	1.227226	1.226956	1.125810	1.226099	**1.244737**
	5	750	**1.602088**	1.520168	1.597188	1.598410	1.478517	1.570086	1.535704
	8	1500	**2.135485**	2.016854	2.124411	2.136525	1.971067	2.102930	2.084336
	10	2000	2.565232	2.459026	2.578311	**2.588104**	2.406484	2.570389	2.556327
WFG3	3	400	0.736561	0.757034	0.814962	0.819758	0.774135	0.834432	**0.847567**
	5	750	0.976683	0.906075	**1.028412**	1.013941	0.957250	1.013341	0.977617
	8	1500	1.123956	0.770754	1.147203	1.221543	1.093482	1.263233	**1.351959**
	10	2000	1.600964	0.524917	1.573090	1.567908	1.004506	1.577058	**1.720463**
WFG4	3	400	0.737022	0.685079	0.729664	0.728892	0.643591	0.717433	**0.750714**
	5	750	**1.289076**	1.161435	1.286861	1.285072	1.074986	1.271279	0.855942
	8	1500	1.958817	1.188847	**1.964648**	1.962156	1.078243	1.933492	1.137827
	10	2000	2.502320	1.432285	**2.504065**	2.502319	1.330296	2.481299	1.557451
WFG5	3	400	**0.714891**	0.656189	0.687005	0.687220	0.633971	0.669193	0.698642
	5	750	**1.239427**	1.120619	1.222746	1.222480	1.049378	1.219312	0.893813
	8	1500	**1.861308**	1.279934	1.850361	1.850281	0.671722	1.862278	1.183477
	10	2000	**2.353588**	1.541144	2.346521	2.346581	0.303028	2.335886	1.659310
WFG6	3	400	**0.716621**	0.654956	0.690060	0.685939	0.657493	0.677130	0.708633
	5	750	**1.238401**	1.041593	1.223099	1.219001	1.116645	1.224094	0.441287
	8	1500	**1.865392**	0.698152	1.841974	1.843340	1.087488	1.858232	0.475371
	10	2000	**2.345172**	0.811370	2.333417	2.326666	1.189267	2.331650	0.627106
WFG7	3	400	0.735007	0.619531	0.731157	0.729030	0.589746	0.721095	**0.752768**
	5	750	**1.298770**	1.073783	1.295864	1.291999	0.992021	1.298471	0.525322
	8	1500	1.970387	0.813288	1.973601	1.971529	0.986483	**1.992683**	0.521515
	10	2000	**2.507527**	0.950840	2.508710	2.507511	1.154313	2.503361	1.234357
WFG8	3	400	**0.704703**	0.633207	0.666959	0.665932	0.491854	0.656139	0.686264
	5	750	**1.220695**	0.968246	1.183904	1.182260	0.836739	1.173895	0.340591
	8	1500	**1.792107**	0.326124	1.768213	1.759882	0.195763	1.733031	0.693903
	10	2000	**2.301930**	0.255629	2.297054	2.280276	0.237683	2.252147	1.014398
WFG9	3	400	0.715329	0.564868	0.680306	0.670081	0.652229	0.688081	**0.733841**
	5	750	1.230264	1.028928	1.224104	1.212266	1.031762	**1.238784**	1.053354
	8	1500	1.835850	0.882226	1.842840	1.803989	1.017887	**1.860060**	1.526635
	10	2000	**2.368015**	1.095281	2.364149	2.326700	1.125591	2.343906	2.044502



(a) (b) (c)

Figure 5: Radial coordinate plots of WFG6 for (a) RSA (b) NSGA-III (c) θ-DEA



(a) (b) (c) (d)

Figure 6: Final Pareto optimal fronts produced for DTLZ3 by RSA (a) 10-objective (b) 15-objective and RSA* for (c) 10-objective and (d) 15-objective

*4.6. Comparison between RSA and RSA**

RSA* is the version of RSA which adopts P2P transition instead of aA2A transition. For P2P transition, the $CurrentPt_n$ in an iteration n is obtained from the iteration $n-1$ as either the $CurrentPt_{n-1}$, or $NewPt_{n-1}$ depending on the domination status of this $NewPt_{n-1}$. This process is unlike that of the original RSA where the $CurrentPt_n$ is chosen randomly from the *Archive* at each iteration and much similar to the previous simulated annealing based multi-objective optimization algorithms that have been developed.

The results in this section show the clear superiority of the A2A transition strategy and thus stand as a testimony to the theory and logic behind using this strategy as mentioned in Section 3.10 . As observed from Table 9, RSA has a much higher convergence rate than RSA* for all of the standard DTLZ test problems. The fact that P2P transition is not suited for multi-objective simulated annealing algorithms is emphasized by the results obtained for the DTLZ3 test problems. The existence of a convergence proof for single objective simulated annealing algorithm often makes it an attractive alternative to other metahueristics in the single objective domain. However, in the many-objective domain, we see that adopting the same P2P transition strategy leads to getting stuck at local optimal fronts of the DTLZ3 test problems with objectives 5 or higher. The difference in the final obtained Pareto fronts is more apparent from the Radial Co-ordinate plots in Figure 6. We observe from the figures that for the same number of function evaluations, RSA reaches the global Pareto front which corresponds to a value of the radius equal to one. However, in case of RSA*, the majority of the archive solutions were not able to overcome the local fronts.

4.7. Sensitivity Analysis of Hyper-parameters

This section presents a detailed hyper-parameter sensitivity analysis on the WFG Test suite (all problems with 3-objectives). The WFG test suite has been particularly chosen due to the wide variety of features included in the test problems and the ease in changing the hardness of a problem by simply changing the WFG function parameters. There are several parameter tuning tools available that can be used to evaluate the performance limits of RSA such as irace [40] and ParamILS [41]. However, we have performed the sensitivity analysis manually because the number of parameters are not too high.

Table 9: MEDIAN IGD VALUES FOR RSA AND RSA* ON M-OBJECTIVE DTLZ1-4 PROBLEMS WHERE M IS VARIED OVER THE SET {3,5,8,10,15}. BEST RESULTS ARE SHOWN IN BOLD. RSA* IS THE VERSION OF RSA WITH P2P TRANSITION INSTEAD OF A2A TRANSITION.

Problem	m	RSA	RSA*
DTLZ1	3	$\mathbf{1.727 \times 10^{-4}}$	3.920×10^{-4}
	5	$\mathbf{9.831 \times 10^{-5}}$	1.328×10^{-3}
	8	$\mathbf{1.987 \times 10^{-3}}$	1.076×10^{-2}
	10	$\mathbf{1.324 \times 10^{-3}}$	6.696×10^{-3}
	15	$\mathbf{3.167 \times 10^{-3}}$	1.197×10^{-2}
DTLZ2	3	$\mathbf{8.892 \times 10^{-4}}$	2.081×10^{-3}
	5	$\mathbf{2.945 \times 10^{-3}}$	1.099×10^{-2}
	8	$\mathbf{1.494 \times 10^{-2}}$	3.857×10^{-2}
	10	$\mathbf{7.880 \times 10^{-3}}$	3.099×10^{-2}
	15	$\mathbf{1.084 \times 10^{-2}}$	3.123×10^{-2}
DTLZ3	3	$\mathbf{5.795 \times 10^{-4}}$	2.129×10^{-2}
	5	$\mathbf{2.883 \times 10^{-3}}$	0.650
	8	$\mathbf{2.724 \times 10^{-2}}$	2.502
	10	$\mathbf{9.230 \times 10^{-3}}$	0.189
	15	$\mathbf{1.802 \times 10^{-2}}$	0.726
DTLZ4	3	$\mathbf{1.250 \times 10^{-4}}$	2.442×10^{-4}
	5	$\mathbf{2.093 \times 10^{-4}}$	8.965×10^{-4}
	8	$\mathbf{1.942 \times 10^{-3}}$	6.980×10^{-3}
	10	$\mathbf{8.312 \times 10^{-4}}$	4.828×10^{-3}
	15	$\mathbf{1.446 \times 10^{-3}}$	3.892×10^{-3}

To perform the analysis, we have considered the following hyper-parameter values :

1. $flag1$: { 0 , 1 }
2. $flag2$: { 0 , 1 }
3. $switch$: { 0.0 , 0.25 , 0.5 , 0.75 , 1.0 }
4. $prob1$: { 0.0 , 0.1 , 0.2 }
5. $prob2$: { 0.0 , 0.1 , 0.2 }

Average Hyper-Volume value of 5 evaluations has been used to evaluate the performance of each of all the possible combinations of the hyper-parameters. There are in total 900 combinations. For each of the test functions, we have presented the top 2 parameter settings, the 2 worst parameter settings, their corresponding HV values and the difference between the best and the worst HV values (this captures the robustness of RSA to hyper-parameter settings) in the Tables 10 and 11 . In the tables, the hyper-parameter setting is mentioned in the format { flag1 , flag2 , switch , prob1 , prob2 } :

From these hyper-parameter sensitivity tables, we make the following notes :

1. RSA performs extremely well for the functions WFG4-WFG9 and WFG2. It can also be seen that RSA is quite robust to the parameter settings for these functions as can be inferred from the small magnitudes of the corresponding difference between the best and worst HV values.
2. The performance of RSA is rather poor for WFG1 and WFG3. RSA is not very robust to the parameter settings for these functions either as can be inferred from the high variance between best and worst HV values.
3. Although HypE and GrEA yield relatively better results for the WFG1 function, none of the algorithms seems to perform very good for this function.

It is necessary to find the exact sources of the difficulties introduced by WFG1 and WFG3 that lead to the poor performance of RSA. It would also be insightful to look at the performance of RSA if the difficulty introduced by multi-modal and deceptive landscapes were varied. To perform this study, we run RSA on custom WFG functions derived from WFG1, WFG4 and WFG5. The observed results in this direction are elaborated below :

1. We first construct a function with the same features as WFG1 with only the flat bias feature missing. We run RSA on this function for 5 evaluations with the same hyper-parameter settings which yield the best result for WFG1 and obtain an average HV value equal to 0.244454 .
2. We construct a function with the same features as that of WFG1 but with the PF shape set as concave. Running RSA on this function for 5 evaluations with the same hyper-parameter settings as above yields an average HV value of 0.159226 .
3. We construct a function with the same features as that of WFG1 but with the value of the argument α of the function b_poly set to 0.2 . This reduces the bias towards 0 as compared to the original WFG1 . After running RSA on this function 5 times with the same hyper-parameter settings as above, we obtain an average HV value of 1.093730 .
4. We construct a function with the same features as that of WFG1 but with the value of the argument α of the function b_poly set to 0.8 . This reduces the bias towards 0 even further. After running RSA on this function 5 times with the same hyper-parameter settings as above, we obtain an average HV value of 1.234598 .
5. We construct a function with the same features as that of WFG1 but with the value of the argument α of the function b_poly set to 50 . This increases the bias largely towards 1. After running RSA on this function 5 times with the same hyper-parameter settings as above, we obtain an average HV value of 1.258144 .
6. Now, we construct a function with the same features as WFG4 but with the value of the argument A of the function s_multi set to 50. This increases the number of hills thus increasing the difficulty of the problem. After running RSA on this function 5 times with the same hyper-parameter settings which yield the best performance for WFG4, we obtain an average HV value of 0.727816 .

Table 10: RESULTS OF HYPER-PARAMETER SENSITIVITY STUDY OF RSA FOR WFG1-7 TEST PROBLEMS

Problem	Hyper-parameter setting	HV value	Rank
WFG1	{ 0 , 0 , 0.75 , 0.0 , 0.0 }	0.673171	1
	{ 0 , 0 , 0.75 , 0.1 , 0.0 }	0.671986	2
	{ 0 , 1 , 1.0 , 0.2 , 0.0 }	0.225654	900
	{ 0 , 1 , 1.0 , 0.2 , 0.1 }	0.231007	899
	Robustness(Best HV - Worst HV)	0.447517	
WFG2	{ 0 , 0 , 0.5 , 0.1 , 0.2 }	1.231475	1
	{ 0 , 0 , 0.5 , 0.2 , 0.1 }	1.230503	2
	{ 0 , 0 , 1.0 , 0.0 , 0.2 }	1.173164	900
	{ 0 , 0 , 1.0 , 0.0 , 0.1 }	1.180775	899
	Robustness(Best HV - Worst HV)	0.058311	
WFG3	{ 0 , 1 , 0.25 , 0.2 , 0.1 }	0.736561	1
	{ 0 , 1 , 0.5 , 0.0 , 0.2 }	0.731205	2
	{ 0 , 0 , 1.0 , 0.2 , 0.0 }	0.607383	900
	{ 0 , 1 , 1.0 , 0.2 , 0.1}	0.613021	899
	Robustness(Best HV - Worst HV)	0.129178	
WFG4	{ 0 , 1 , 1.0 , 0.0 , 0.1 }	0.737022	1
	{ 0 , 1 , 1.0 , 0.0 , 0.2 }	0.736076	2
	{ 0 , 1 , 0.0 , 0.0 , 0.1 }	0.709138	900
	{ 1 , 1 , 0.0 , 0.0 , 0.2 }	0.709154	899
	Robustness(Best HV - Worst HV)	0.027884	
WFG5	{ 0 , 0 , 0.5 , 0.0 , 0.0 }	0.714891	1
	{ 0 , 1 , 0.75 , 0.0 , 0.2 }	0.714745	2
	{ 1 , 1 , 0.0 , 0.1 , 0.2 }	0.693960	900
	{ 0 , 1 , 0.0 , 0.1 , 0.1}	0.694782	899
	Robustness(Best HV - Worst HV)	0.020931	
WFG6	{ 0 , 0 , 1.0 , 0.0 , 0.2 }	0.716621	1
	{ 0 , 0 , 0.75 , 0.1 , 0.0 }	0.716577	2
	{ 1 , 1 , 0.0 , 0.2 , 0.1 }	0.661648	900
	{ 0 , 1 , 0.0 , 0.1 , 0.1 }	0.663010	899
	Robustness(Best HV - Worst HV)	0.054973	
WFG7	{ 0 , 0 , 0.75 , 0.0 , 0.1 }	0.735588	1
	{ 0 , 0 , 0.75 , 0.2 , 0.1 }	0.735464	2
	{ 1 , 1 , 0.0 , 0.0 , 0.0 }	0.712160	900
	{ 1 , 1 , 0.0 , 0.1 , 0.2 }	0.712796	899
	Robustness(Best HV - Worst HV)	0.023428	

Table 11: HYPER-PARAMETER SENSITIVITY RESULTS FOR WFG8 AND WFG9

Problem	Hyper-parameter setting	HV value	Rank
WFG8	{ 0 , 1 , 0.75 , 0.2 , 0.0 }	0.704703	1
	{ 0 , 1 , 0.75 , 0.0 , 0.0 }	0.704656	2
	{ 0 , 1 , 1.0 , 0.2 , 0.0 }	0.687036	900
	{ 0 , 1 , 1.0 , 0.2 , 0.1 }	0.687172	899
	Robustness(Best HV - Worst HV)	0.017667	
WFG9	{ 0 , 1 , 0.75 , 0.0 , 0.2 }	0.715191	1
	{ 1 , 1 , 0.75 , 0.2 , 0.0 }	0.713142	2
	{ 0 , 0 , 0.0 , 0.0 , 0.2 }	0.676903	900
	{ 0 , 1 , 0.0 , 0.0 , 0.1 }	0.683935	899
	Robustness(Best HV - Worst HV)	0.038288	

7. We construct a function with the same features as WFG4 but with the value of the argument B of the function *s_multi* set to 5. This increases the size of the hills thus increasing the difficulty of the problem. After running RSA on this function 5 times with the same hyper-parameter settings which yield the best performance for WFG4, we obtain an average HV value of 0.728614 .
8. We construct a function with the same features as WFG4 but with the value of the argument B of the function *s_multi* set to 20. This decreases the number of hills thus decreasing the difficulty of the problem. After running RSA on this function 5 times with the same hyper-parameter settings which yield the best performance for WFG4, we obtain an average HV value of 0.737998 .
9. Next, we construct a function with the same feature as WFG5 with the value of the argument B of the function *s_decept* set to 0.0005. This reduces the size of the aperture of the well that leads to the global minimum, thus increasing the difficulty of the problem. We run RSA on this function for 5 evaluations with the same hyper-parameter settings which yield the best result for WFG5 and obtain an average HV value equal to 0.712284 .

From the above study, it can be easily inferred that problems which have a large polynomial bias towards 0 pose challenges to RSA. As we increase the value of α, the obtained HV value starts to increase as well. However, a high bias towards 1 does not pose as great a challenge. This can also be observed from the obtained results of RSA corresponding to the DTLZ4 function. Removing the flat bias from WFG1 or changing its shape to concave does not improve the obtained HV value, thus indicating that these are not the sources of difficulty. We also observe that even though RSA performs good on WFG6, it performs poorly on WFG3. The only difference between the two functions is the shape of the Pareto Front. However, RSA performs good on DTLZ1 which has a linear PF. Thus, degenerate functions are also not very suited for optimization using RSA. Conversely, if running RSA on a problem with unknown landscape features yields a poor HV value, it might be indicative of the presence of either polynomial bias with low value of α or degeneracy in the unknown function. From the observations 6,7,8,9 in the above list, it can be seen that RSA continues to perform well with the same hyper-parameter settings even after the difficulty has been changed.

All of these observations can be used as a heuristic to choose the right hyper-parameters for a particular kind of problem when using RSA for optimization. This experimental analysis also calls for a much detailed theoretical study to find the reason behind the observed performances.

4.8. Time Analysis

In this section, we compare RSA and NSGA-III in terms of the time taken to optimize the DTLZ problems for approximately same number of function evaluations (the number of function evaluations for different test

Table 12: TIMES (in seconds) TAKEN BY RSA AND NSGA-III FOR DTLZ PROBLEMS

Problem	M	Time taken by RSA(s)	Time taken by NSGA-III(s)	Total evaluations by RSA	Total evaluations by NSGA-III
DTLZ1	3	4.25	1.23	36640	36800
	5	57.25	21.65	126408	127200
	8	62.31	21.59	117248	117000
	10	313.21	95.83	274800	275000
DTLZ2	3	3.51	1.12	23816	23000
	5	36.95	15.32	73280	74200
	8	46.49	14.71	78776	78000
	10	265.62	85.42	205184	207000
DTLZ3	3	13.00	4.85	91600	92000
	5	103.28	32.22	210680	212000
	8	83.93	25.20	155720	156000
	10	516.02	92.12	412200	414000
DTLZ4	3	9.53	3.51	54960	55200
	5	116.39	41.71	210680	212000
	8	117.65	40.43	194192	195000
	10	650.29	229.31	549600	552000

problems are also mentioned in Tables 5 and 6). RSA and NSGA-III are both implemented using C++ (code for NSGA-III obtained from http://web.ntnu.edu.tw/ tcchiang/publications/nsga3cpp/nsga3cpp.htm). The codes are executed on a machine with 4th-gen Intel Core i7 CPU (2.5GHz) and 8GB DDRL3 RAM. The execution times taken by both the approaches for solving DTLZ1-4 problems with number of objectives varying from 3 to 10 are given in Table 12 .

It can be observed, that RSA takes more time per evaluation than NSGA-III at the cost of a better performance (one must also keep in mind that timings are also dependent on coding practices). In other words, RSA performs more processing while evaluating a new solution. However, one must also note that RSA performs better for comparatively lesser number of function evaluations (the number of function evaluations for RSA are lesser than that of NSGA-III for most of the cases). As the stated problems (DTLZ1-4) are simple in terms of objective functions, the process time (time taken to execute internal operations specific to an algorithm) dominates objective function calculation time. Thus, the execution time of RSA is higher than that of NSGA-III. For expensive optimization problems such as aircraft design [42], crash worthiness optimization [43], water network design [44] and optimization in robotics using simulations [45], RSA would prove much useful as in these cases, the total time taken for optimization is dominated by the time taken for the function evaluations. The time spent on rest of the operations will be negligible and thus, the number of function evaluations required to obtain the same performance is of utmost importance. RSA outperforms NSGA-III in this respect. Moreover, a parallel version of RSA can also aid in reducing the execution times. The comparison of a solution with all the members of archive can be carried out parallely to reduce the execution time.

4.9. Friedman Test

In order to statistically validate RSA, Friedman Test has been performed. All the algorithms in Table 8 have been used over all the cases of DTLZ and WFG test functions. Table 13 shows the total ranks of different algorithms which are obtained by summing the individual ranks of the algorithms for each of the test problem (DTLZ and WFG). We use the following formula for calculating the value of χ_r^2 :

Table 13: TOTAL RANKS OF DIFFERENT ALGORITHMS

RSA	MOEA/D-PBI	θ-DEA	NSGA-III	GrEA	HypE	dMOPSO
118	278	142	173	179	245	321

$$\chi_r^2 = \frac{12}{nk(k+1)} \sum R^2 - 3n(k+1)$$

Here n is the number of test problems and k is the number of algorithms.There are in total 52 cases and 7 algorithms. Therefore, k = 7 and n = 52. From the chi-square table, we observe that for k = 7 and $\alpha = 0.05$, the obtained χ_r^2 must be greater than 12.5916 for the rejection of the null hypothesis with a confidence level of 95 % stating that all the algorithms have similar performance. Putting the values, we obtain a value of $\chi_r^2 = 138.29$ thus rejecting the null hypothesis.

5. Conclusion and Future Scope

This paper presents a new simulated annealing based many objective optimization algorithm, namely RSA. The use of Archive-to-Archive movement instead of point-to-point movement for handling the problems with multiple objectives is introduced in RSA. The results in the last section experimentally show the improvement in the performance that is obtained by using this strategy. This paper also presents a strategy which has been named as mutation switching. Mutation switching involves periodically switching between different types of reproduction operators. Mutation switching helps in taking advantages of different kinds of reproduction operators and can be adjusted according to the features of the problem at hand. Some settings help in overcoming the local optimal fronts while others help in reaching a higher convergence. A very detailed hyper-parameter sensitivity analysis has been performed to guide the users and also to demonstrate the role that reproduction operators play in obtaining the solutions. This paper also presents a theoretical analysis of the clustering algorithm that has been used in RSA. To demonstrate the advantages of RSA, we have performed comparisons with various other state-of-the-art algorithms over standard DTLZ and WFG test suites. The results clearly suggest that RSA is a strong competitor for various kinds of problems.

Since the concept of mutation switching is independent of the algorithm, it should be worthwhile to research the effects of combining this concept with other algorithms. Another domain of interest while developing a simulated annealing based algorithm is different types of cooling strategies that can be adopted. Various kinds of cooling strategies have been proposed in the literature [46]. Experimenting with these cooling strategies or developing a new and more effective one for many objective optimization might be a promising research area. In future, we shall try to extend our algorithm to solve constrained many-objective optimization problems and apply RSA for solving various real life problems. We shall also try to make the operations involved in RSA more computationally efficient and develop a parallel implementation of the algorithm in an attempt to improve the speed of RSA. Due to the very few existing theoretical work pertaining to MaOO algorithms, a prospective direction might be a detailed theoretical analysis of the algorithm and finding out the exact reason behind its observed performances. Using mutation switching introduces a large number of parameters that need to be set by the user. This problem can be countered by trying to develop adaptive methods that enable the algorithm to set these parameters by itself.

References

[1] K. Deb, Multi-objective optimization using evolutionary algorithms, volume 16, John Wiley & Sons, 2001.

[2] C. A. C. Coello, D. A. Van Veldhuizen, G. B. Lamont, Evolutionary algorithms for solving multi-objective problems, volume 242, Springer, 2002.

[3] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation 1 (2011) 32–49.

[4] C. Lücken, B. Barán, C. Brizuela, A survey on multi-objective evolutionary algorithms for many-objective problems, Comput. Optim. Appl. 58 (2014) 707–756.
[5] B. Li, J. Li, K. Tang, X. Yao, Many-objective evolutionary algorithms: A survey, ACM Comput. Surv. 48 (2015) 13:1–13:35.
[6] P. J. Fleming, R. C. Purshouse, R. J. Lygoe, Many-Objective Optimization: An Engineering Design Perspective, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 14–32. URL: https://doi.org/10.1007/978-3-540-31880-4_2. doi:10.1007/978-3-540-31880-4_2.
[7] J. G. Herrero, A. Berlanga, J. M. M. López, Effective evolutionary algorithms for many-specifications attainment: Application to air traffic control tracking filters, Trans. Evol. Comp 13 (2009) 151–168.
[8] R. J. Lygoe, M. Cary, P. J. Fleming, A Real-World Application of a Many-Objective Optimisation Complexity Reduction Process, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 641–655. URL: https://doi.org/10.1007/978-3-642-37140-0_48. doi:10.1007/978-3-642-37140-0_48.
[9] K. Praditwong, M. Harman, X. Yao, Software module clustering as a multi-objective search problem, IEEE Transactions on Software Engineering 37 (2011) 264–282.
[10] A. S. Sayyad, T. Menzies, H. Ammar, On the value of user preferences in search-based software engineering: A case study in software product lines, in: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, IEEE Press, Piscataway, NJ, USA, 2013, pp. 492–501. URL: http://dl.acm.org/citation.cfm?id=2486788.2486853.
[11] A. Sülflow, N. Drechsler, R. Drechsler, Robust Multi-Objective Optimization in High Dimensional Spaces, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 715–726. URL: https://doi.org/10.1007/978-3-540-70928-2_54. doi:10.1007/978-3-540-70928-2_54.
[12] Y. Yuan, H. Xu, Multiobjective flexible job shop scheduling using memetic algorithms, IEEE Transactions on Automation Science and Engineering 12 (2015) 336–353.
[13] O. Chikumbo, E. Goodman, K. Deb, Approximating a multi-dimensional pareto front for a land use management problem: A modified moea with an epigenetic silencing metaphor, in: 2012 IEEE Congress on Evolutionary Computation, 2012, pp. 1–9. doi:10.1109/CEC.2012.6256170.
[14] S. Bandyopadhyay, S. Saha, U. Maulik, K. Deb, A simulated annealing-based multiobjective optimization algorithm: AMOSA, IEEE Trans. Evolutionary Computation 12 (2008) 269–283.
[15] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii, Lecture notes in computer science 1917 (2000) 849–858.
[16] J. Knowles, D. Corne, The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation, in: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 1, IEEE, 1999.
[17] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evolutionary Computation 11 (2007) 712–731.
[18] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints, Evolutionary Computation, IEEE Transactions on 18 (2014) 577–601.
[19] Y. Yuan, H. Xu, B. Wang, X. Yao, A new dominance relation-based evolutionary algorithm for many-objective optimization, IEEE Trans. Evolutionary Computation 20 (2016) 16–37.
[20] J. Bader, E. Zitzler, Hype: An algorithm for fast hypervolume-based many-objective optimization, Evolutionary Computation 19 (2011) 45–76.
[21] S. Yang, M. Li, X. Liu, J. Zheng, A grid-based evolutionary algorithm for many-objective optimization, IEEE Trans. Evolutionary Computation 17 (2013) 721–736.
[22] J. Bader, D. Brockhoff, S. Welten, E. Zitzler, On Using Populations of Sets in Multiobjective Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 140–154. URL: https://doi.org/10.1007/978-3-642-01020-0_15. doi:10.1007/978-3-642-01020-0_15.
[23] I. Das, J. E. Dennis, Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems, SIAM Journal on Optimization 8 (1998) 631–657.
[24] H. Wang, X. Yao, Corner sort for pareto-based many-objective optimization, IEEE Trans. Cybernetics 44 (2014) 92–102.
[25] M. Asafuddoula, T. Ray, R. A. Sarker, A decomposition-based evolutionary algorithm for many objective optimization, IEEE Trans. Evolutionary Computation 19 (2015) 445–460.
[26] K. Deb, R. B. Agrawal, Simulated binary crossover for continuous search space, Complex Systems 9 (1994) 1–34.
[27] R. Sengupta, M. Pal, S. Saha, S. Bandyopadhyay, Population dynamics indicators for evolutionary many-objective optimization, in: Progress in Intelligent Computing Techniques: Theory, Practice, and Applications, Springer, 2017, pp. 1–10.
[28] X. Qiu, J. X. Xu, K. C. Tan, H. A. Abbass, Adaptive cross-generation differential evolution operators for multiobjective optimization, IEEE Transactions on Evolutionary Computation 20 (2016) 232–244.
[29] B. Suman, P. Kumar, A survey of simulated annealing as a tool for single and multiobjective optimization, Journal of the Operational Research Society 57 (2006) 1143–1160.
[30] K. I. Smith, R. M. Everson, J. E. Fieldsend, Dominance measures for multi-objective simulated annealing, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2004, 19-23 June 2004, Portland, OR, USA, 2004, pp. 23–30. URL: http://dx.doi.org/10.1109/CEC.2004.1330833. doi:10.1109/CEC.2004.1330833.
[31] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multiobjective optimization, Evolutionary Multiobjective Optimization (2005) 105–145.
[32] S. Huband, P. Hingston, L. Barone, R. L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evolutionary Computation 10 (2006) 477–506.

[33] H. Ishibuchi, Y. Setoguchi, H. Masuda, Y. Nojima, Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes, IEEE Trans. Evolutionary Computation 21 (2017) 169–190.
[34] R. H. Gómez, C. A. C. Coello, MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013, 2013, pp. 2488–2495. URL: http://dx.doi.org/10.1109/CEC.2013.6557868. doi:10.1109/CEC.2013.6557868.
[35] D. A. V. Veldhuizen, G. B. Lamont, Multiobjective evolutionary algorithm research: A history and analysis, 1998.
[36] A. Auger, J. Bader, D. Brockhoff, E. Zitzler, Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications, Theor. Comput. Sci. 425 (2012) 75–103.
[37] Z. He, G. G. Yen, Visualization and performance metric in many-objective optimization, IEEE Trans. Evolutionary Computation 20 (2016) 386–402.
[38] W. Peng, Q. Zhang, A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems, in: The 2008 IEEE International Conference on Granular Computing, GrC 2008, Hangzhou, China, 26-28 August 2008, 2008, pp. 534–537. URL: http://dx.doi.org/10.1109/GRC.2008.4664724. doi:10.1109/GRC.2008.4664724.
[39] J. Bader, E. Zitzler, A Hypervolume-Based Optimizer for High-Dimensional Objective Spaces, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 35–54. URL: http://dx.doi.org/10.1007/978-3-642-10354-4_3. doi:10.1007/978-3-642-10354-4_3.
[40] M. Lpez-Ibez, J. Dubois-Lacoste, L. P. Cceres, M. Birattari, T. Sttzle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43 – 58.
[41] F. Hutter, T. Stützle, K. Leyton-Brown, H. H. Hoos, Paramils: An automatic algorithm configuration framework, CoRR abs/1401.3492 (2014).
[42] R. A. Shah, P. M. Reed, T. W. Simpson, Many-Objective Evolutionary Optimisation and Visual Analytics for Product Family Design, Springer London, London, 2011, pp. 137–159. URL: https://doi.org/10.1007/978-0-85729-652-8_4. doi:10.1007/978-0-85729-652-8_4.
[43] H. Fang, M. Rais-Rohani, Z. Liu, M. Horstemeyer, A comparative study of metamodeling methods for multiobjective crashworthiness optimization, Computers & Structures 83 (2005) 2121 – 2136.
[44] F. di Pierro, S.-T. Khu, D. Savi, L. Berardi, Efficient multi-objective optimal design of water distribution networks on a budget of simulations using hybrid algorithms, Environmental Modelling & Software 24 (2009) 202 – 213.
[45] M. Tesch, J. Schneider, H. Choset, Expensive multiobjective optimization for robotics, in: 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 973–980. doi:10.1109/ICRA.2013.6630691.
[46] Y. Nourani, B. Andresen, A comparison of simulated annealing cooling strategies, Journal of Physics A: Mathematical and General 31 (1998) 8373.