A Multimodal Approach for Evolutionary Multi-objective Optimization: MEMO

Cem C. Tutum and **Kalyanmoy Deb**
Department of Electrical and Computer Engineering
Michigan State University, East Lansing, MI 48824, USA
Email: tutum@msu.edu, kdeb@egr.msu.edu
URL: http://www.egr.msu.edu/~kdeb/

COIN Report Number 2014018

Abstract

Most evolutionary multi-objective optimization (EMO) methods use domination and niche-preserving principles in their selection operation to find a set of Pareto-optimal solutions in a single simulation run. However, classical generative multi-criterion optimization methods repeatedly solve a parameterized single-objective problem to achieve the same. Due to lack of parallelism in the classical generative methods, they have been reported to be slow compared to efficient EMO methods. In this paper, we use a specific scalarization method, but instead of repetitive independent applications, we formulate a multimodal scalarization of the multi-objective or many-objective optimization problem and develop a niche-based evolutionary algorithm (MEMO) to find multiple Pareto-optimal solutions in a single simulation run. Proof-of-principle results on two to 10-objective unconstrained and constrained problems using our proposed multimodal approach are able to find hundreds of Pareto-optimal solutions. The proposed MEMO approach is also compared with state-of-the-art evolutionary multi/many-objective optimization methods. MEMO is then applied to a number of engineering design problems. Results are promising and provide a new direction for solving multi- and many-objective optimization problems.

1 Introduction

The success in solving multi-objective optimization problems using evolutionary algorithms (EAs) comes from a balance of three aspects in their selection operation: (i) emphasis of non-dominated solutions in a population, (ii) emphasis of less-crowded solutions in a population and (iii) emphasis of elites in a population (Deb, 2001; Coello et al., 2002). The parallel search ability introduced by the population approach and recombination operator of an EMO algorithm aided by the above properties of its selection operator cause it to move towards the Pareto-optimal set with an adequate spread and computational speed. On the other hand, classical generative multi-objective optimization methods base their search by solving a series of parameterized single-objective problem serially (Miettinen, 1999; Chankong and Haimes, 1983a). One criticism of the generative approaches is their lack of parallelism which makes them computationally expensive compared to EMO methods (Shukla and Deb, 2005).

When a finite set of Pareto-optimal solutions is the target, the multi-objective optimization problem can be considered as a single-objective *multimodal* problem in which each Pareto-optimal solution is targeted as an independent optimal solution. Although the idea is intriguing, what we need is a suitable scalarization method that will allow us to consider such a mapping. In this paper, we use the achievement scalarization function (ASF) method (Wierzbicki, 1980) and construct a multi-modal problem for a multi-objective optimization problem. We then suggest an efficient niching-based evolutionary algorithm (Goldberg and Richardson, 1987; Deb and Goldberg, 1989) for finding multiple optimal solutions in a single simulation run. The resulting multimodal EA (we call MEMO here) is shown to solve two to 10-objective optimization problems for finding hundreds of Pareto-optimal solutions as efficiently as the state-of-the-art EMO methodologies. These proof-of-principle results are encouraging and suggest immediate extension of the approach to solve more complex multi/many-objective optimization problems. The multimodal idea of solving multi-objective optimization problem

is new and has potential to be used with various other forms of scalarization methods. The ability of solving multi-objective optimization problems using a single-objective multimodal optimization problem can eventually lead us to develop a unified optimization approach that may solve different types of optimization problems in a single unified manner.

In the remainder of the paper, we give a brief summary of niching-based evolutionary multimodal optimization methods for single-objective optimization in Section 2. Section 3.1 discusses classical generative multi-criterion optimization methods. In Section 4.2, we present the proposed MEMO algorithm. Results on multi and many-objective optimization problems are shown and compared with NSGA-II (Deb et al., 2002) and recently proposed NSGA-III (Deb and Jain, 2014; Jain and Deb, 2014) in Section 5. Finally, conclusions and extensions to this study are discussed in Section 6.

2 Multimodal Optimization

Multimodal function optimization problems are identified as single-objective optimization problems of having multiple local minima together with a single global optima or multiple global optima having the same objective function values. However, the goal is, unlike in common approach of finding only the best performing solution, to find *all* possible minima (or as many as possible) which are known as basins of attraction. There are two practical benefits of this approach. First, having known the set of high performing solutions would allow a designer or a practitioner to switch between them in the presence of changes in the working environment or considering manufacturing limits. Second, finding this diverse option of optimal solutions may help users unveil a design pattern or philosophy that exists among these multiple relatively good solutions as in the concept of knowledge discovery out of Pareto-optimal set in multi-objective optimization problems, namely innovization (Deb and Srinivasan, 2006). A commonly used engineering application is to design a structure having the minimum weight and the maximum stability (or in other words, maximum fundamental natural frequency). This structural optimization problem leads multimodal solutions because the increase in the first mode frequency is followed by the decrease in the other modes (Czyz and Lukasiewicz, 1995). Protein structure prediction problem (Wong et al., 2010), conceptual laser pulse designs (Shir et al., 2007) and optimal design of induction motors for electric vehicles (Dilettoso and Salerno, 2006) can be given as other real-world applications of the multimodal optimization.

The population based parallel search capability and diversity preserving techniques gives evolutionary algorithms a great advantage for finding multiple optimal solutions in a single run as in the case of solving multi-objective optimization problems. Since classical derivative-based optimization algorithms use a single point for their search, they have to be executed with different initial guesses multiple times hoping to end up with different solutions. The selection operator in traditional EAs works in a way to find a single, most likely the global, optimum solution. Therefore, there is a need to simultaneously preserve multiple optimal solutions through out the consecutive generations in the EA framework. For this purpose, niching, a generic term for diversity preserving techniques, is implemented in a variety of forms, i.e. Cavicchio's preselection idea (Cavicchio, 1970), fitness sharing (Goldberg and Richardson, 1987; Yin and Germay, 1993b), crowding (Mahfoud, 1992; Mengsheol and Goldberg, 1999), clearing (Petrowski, 1996), restricted tournament selection (Harik, 1997), clustering (Yin and Germay, 1993a; Streichert et al., 2003) and conservation of species (Petrowski, 1997) as well as Deb and Saha's bi-objective optimization approach (Deb and Saha, 2012) and others (Ursem, 1999). Most of these methods require one or more niching parameter which must be set right to form appropriate niches around each optimum. Recent renewed interests in multimodal EAs (Yang, 2009; de Castro and Zuben, 2002; Preuss et al., 2007; Barrera and Coello, 2009; Bessaou et al., 2000; Im et al., 2004; Lee et al., 1999; Li et al., 2002; Li, 2005; Rönkkönen et al., 2008; Parrott and Li, 2006; Zhang and Li, 2005; Shir and Bäck, 2005) are some efforts in this direction and support further research in evolutionary multimodal optimization.

These mechanisms mentioned above are, in general, designed to create and maintain diverse subpopulations within the global population. Niching algorithms are categorized into four groups in the review by (Das et al., 2011) based on how the niches are located: i) Sequential Niching (SN), ii) Parallel Niching (PN), iii) Quasi-sequential niching, iv) Hierarchical niching. PN method follows a parallel hill-climbing approach which is similar to a binary search technique, whereas SN method works similar to GAs, but running offline iteratively and keeping track of each optimum a predefined niche-radius away from each other.

3 Multi-objective Optimization

As the name indicates, multi-objective optimization simultaneously deals with multiple conflicting objective functions, and as similar to multimodal optimization, a set of optimal solutions are found as a result. These optimum solutions are considered to be equally important if there is no preference or bias information beforehand and called as Pareto-optimal or non-dominated solutions. In order to have a clear idea of trade-offs, it is always better to have as many non-dominated solutions as possible along the entire region of the front. Therefore, two goals exist in the solution of a multi-objective optimization problem: i) convergence to the true Pareto-optimal set, and ii) maintain a diverse set of Pareto-optimal solutions. The first aim is valid for all kinds of optimization problems, whereas the latter is unique to multi-objective optimization and to multimodal optimization studies.

There are two broad types of algorithms used to solve multi-objective optimization problems: classical point-by-point algorithms and evolutionary population-based algorithms. Classical methods use a single solution for their search and this requires multiple runs of the algorithm to find a set of Pareto-optimal solutions. However, population-based algorithms can find and store multiple Pareto-optimal solutions in a single simulation. It has been shown elsewhere (Shukla and Deb, 2005) that the population approaches allow a parallel search capability which makes them attractive for finding multiple solutions in a computationally quick manner.

3.1 Classical Generative Methods for Multi-objective Problem Solving

Multi and many-objective optimization problems give rise to a set of Pareto-optimal solutions, thereby making them more difficult to be solved than single-objective optimization problems. While efficient EMO methods (Srinivas and Deb, 1994; Fonseca and Fleming, 1993; Horn et al., 1994) were developed since early nineties to find multiple Pareto-optimal solutions in a single simulation run, classical generative methods were suggested since early seventies (Chankong and Haimes, 1983a; Steuer, 1986). In the generative principle (Miettinen, 1999; Chankong and Haimes, 1983b), a multi-objective optimization problem is scalarized to a single-objective function by using one or more parameters. Weighted-sum approach use relative weights for objective functions; epsilon-constraint approach uses a vector of ϵ-values for converting objective functions into constraints; Tchebyshev method use a weight vector for forming the resulting objective function. The idea is then to solve the parameterized single-objective optimization problem with associated constraints repeatedly for different parameter values one at a time. The above scalarization methods, if solved to their optimality, are guaranteed to converge to a Pareto-optimal solution every time (Miettinen, 1999). However, some scalarization methods are not capable of finding certain Pareto-optimal solutions in a problem no matter what parameter values are chosen (Ehrgott, 2000). These methods (such as weighted-sum or Lp-norm (for $p \neq \infty$) methods) are relatively unpopular and methods capable of finding each and every Pareto-optimal solution for certain combination of parameter values, such as epsilon-constraint and Tchebyshev method are popular. Here, we use one such popular scalarization method that also guarantees to find any Pareto-optimal solution.

3.1.1 Achievement Scalarizing Function (ASF) Method

The achievement scalarizing (ASF) method was suggested elsewhere (Wierzbicki, 1980) to solve multi-objective optimization problems. The method requires one reference point $\mathbf{Z} = (z_1, z_2, \ldots, z_M)^T$ and one weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_M)^T$. Both these vectors are associated with the objective space. The reference point is any point on the objective space, with or without any corresponding $\mathbf{x}$-vector. Usually, $\mathbf{Z}$ is chosen as an aspiration point and is often an unattainable point.

The scenario is depicted for a hypothetical two-objective problem in Figure 1. The objective space is shown by the shaded region. For any variable vector $\mathbf{x}$, the corresponding objective vector $\mathbf{f}$ can be computed and is shown in the figure. For the supplied $\mathbf{w}$-vector (representing inverse of the relative preference of objectives), the following ASF function is minimized to find a Pareto-optimal solution:

$$\begin{array}{ll} \text{Minimize} & \text{ASF}(\mathbf{x}) = \max_{j=1}^{M} \left(\frac{f_i(\mathbf{X}) - z_i}{w_i} \right), \\ \text{subject to} & \mathbf{x} \in \mathbf{X}. \end{array} \tag{1}$$

Since our proposed approach is based on ASF function, we discuss how the above formulation can result in a Pareto-optimal solution. For the illustrated objective vector $\mathbf{F} = (f_1, f_2)^T$, the two components

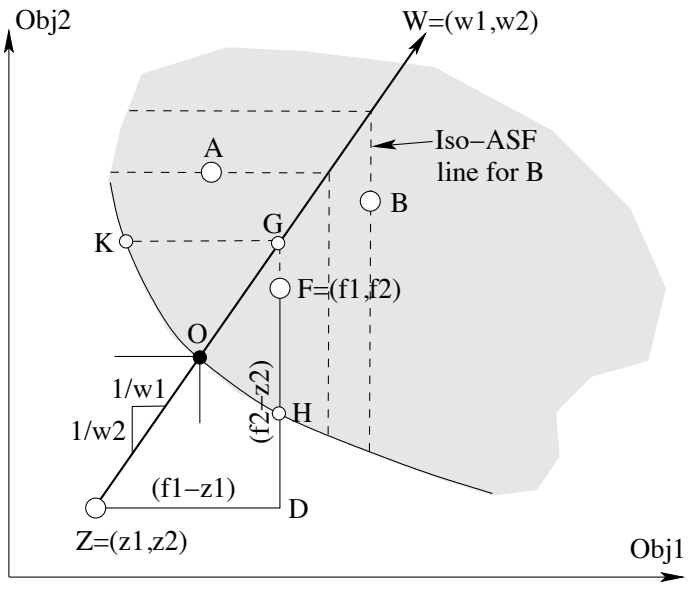

Figure 1: The achievement scalarizing function (ASF) approach is illustrated.

within brackets are $(f_1(\mathbf{x}) - z_1)/w_1$ and $(f_2(\mathbf{x}) - z_2)/w_2$. If the point **F** lies on the weight vector ($\mathbf{Z}\vec{\mathbf{W}}$), that is, the point **G**, the above two components will be identical and the ASF function is equal to any of the two terms (say, it is equal to the first term, ASF(**G**) $= (f_1(\mathbf{x}) - z_1)/w_1$). However, since the objective vector **F** lies below the weight vector, the maximum of the two components will be ASF(**F**) $= (f_1(\mathbf{x}) - z_1)/w_1$. This is identical to ASF(**G**). In fact, for all objective vectors on the line **GH**, the ASF value will be identical to the that at **F**. The same argument can be made with any point on the line **GK**. For such objective vectors, the ASF function will come from the second term and the corresponding ASF function value is identical to the second term at **G**. As argued earlier, the first and second components at **G** are identical, hence it is clear that all objective vectors from **H** to **G** and then to **K** will have identical ASF value. Such 90-degree horizontal-vertical lines meeting at the weight vector can then be called as the iso-ASF line.

The figure also shows two other objective vectors (**A** and **B**) and their respective iso-ASF lines. Clearly, the ASF value for **F** is smaller than that of **A**, which is smaller than ASF value of **B**. Thus, as the iso-ASF line comes closer to the reference point **Z**, the ASF value gets smaller. Since the problem in Equation 1 is a minimization problem, the above logic will result in point **O** as the optimal solution. The reason is as follows. Since the ASF value at **O** is the minimum possible, there cannot be any other feasible objective vector that intersects the weight vector below point **O**. Such points would have come from the region below the right-angled lines shown at **O**. This then means that there does not exist any point that *dominates* point **O**. This then means that point **O** is an efficient point.

The above explanation proves that if Equation 1 is minimized for any **Z** and **w**(having non-negative components and at least one $w_i > 0$), the resulting point is an efficient point. It turns out that the minimum point to the above ASF problem can be a *weakly* efficient point (Miettinen, 1999). To ensure a strictly efficient point, the following augmented scalarizing function (AASF) is suggested:

$$\begin{aligned} \text{Min.} \quad & \text{AASF}(\mathbf{x}) = \max_{j=1}^{M} \left(\frac{f_i(\mathbf{x}) - z_i}{w_i} \right) \\ & + \rho \sum_{i=1}^{M} \left(\frac{f_i(\mathbf{x}) - z_i}{w_i} \right), \\ \text{s.t.} \quad & \mathbf{x} \in \mathbf{X}. \end{aligned} \tag{2}$$

A small value ($\approx 10^{-4}$) is suggested for the parameter ρ. The augmented term makes the iso-AASF lines inclined as shown in Figure 2 and avoids finding weakly efficient points. The weight vector intersects the weakly efficient front at **A**, but the AASF value at this point is not the minimum possible value. The point **O** has a smaller ASF value as the corresponding iso-AASF lines intersect the weight vector closer to the reference point **Z**. The use of AASF for a scenario depicted in Figure 1 still guarantees to

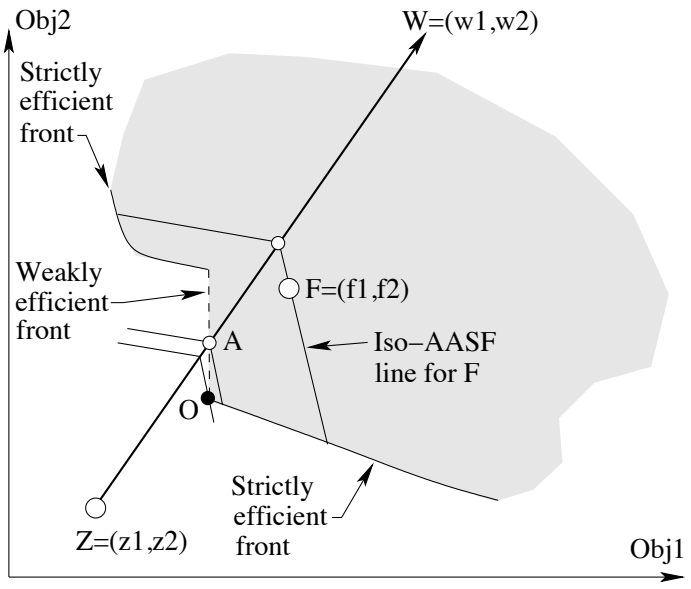

Figure 2: The augmented achievement scalarizing function (AASF) approach is illustrated.

find the identical efficient solution **O**, however for avoids finding weakly efficient solutions.

4 Proposed Multimodal Approach to EMO

In our proposed approach for solving multi-objective optimization problems, we plan to use the augmented achievement scalarizing function (AASF) approach, which requires a reference point $\mathbf{Z}$ and a weight vector $\mathbf{w}$ (discussed in Section 3.1.1). For one combination of $\mathbf{Z}$ and $\mathbf{w}$, we can find a single Pareto-optimal solution by minimizing the corresponding ASF or AASF problem. Thus, if we specify multiple reference points ($\mathbf{Z}^{(k)}$, $k = 1, 2, \ldots, K$) in combination with multiple $\mathbf{w}$-vectors, we can obtain multiple Pareto-optimal solutions each corresponding a single combination, provided we can formulate an appropriate multimodal ASF function from multiple reference points. This principle is the main crux of our proposed multi-modal approach for solving multi-objective optimization problems.

Although above idea is novel, there are two questions that need answering:

1. First, how do we specify a widely-distributed set of reference points on a generic M-dimensional objective space which will automatically generate a set of diverse Pareto-optimal solutions?

2. Second, how do we modify an evolutionary optimization method to find multiple optimal solutions having different ASF values?

We handle these two questions in the following two subsections.

4.1 Formulating a Multi-objective Problem as a Multimodal Problem

In the recent past, decomposition-based methods have been proposed to solve multi- and many-objective optimization problems (Zhang and Li, 2007; Deb and Jain, 2014). In both these methods, Das and Dennis's (Das and Dennis, 1998) strategy for specifying a set of reference directions or reference points were suggested. Our approach here is similar to that in NSGA-III, hence we briefly describe that approach here. On a M-dimensional linear hyper-plane making equal angle to all objective axes and intersecting each axis at one, we specify $K = \binom{(M+p-1)}{p}$ structured points, where p is one less than the number of points along each edge of the hyper-plane. Each reference point $\mathbf{Z}^{(k)} = \left(z_1^{(k)}, z_2^{(k)}, \ldots, z_M^{(k)}\right)^T$ would then satisfy the following condition: $\sum_{i=1}^{M} z_i^{(k)} = 1$. Figure 3 shows distribution of 15 such reference points on a $M = 3$-objective problem with $p = 5$.

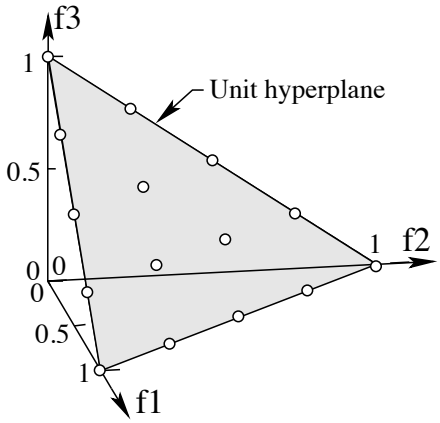

Figure 3: A diverse set of reference points is created using Das and Dennis's strategy for a three-objective problem with $p = 5$.

For each population member $\mathbf{x}$, we can calculate the ASF value ($\text{ASF}(\mathbf{x}, \mathbf{Z}^{(k)})$ with respect to each reference point $\mathbf{Z}^{(k)}$. Thereafter, we can assign the minimum ASF value over all reference points as the fitness to the population member:

$$\text{Fitness}(\mathbf{x}) = \min_{k=1}^{K} \text{ASF}(\mathbf{x}, \mathbf{Z}^{(k)}). \tag{3}$$

This way, a separate minimum will be created at a specific efficient point corresponding to each reference point. We shall illustrate the multimodal ASF through a sketch, but before that, we discuss another important matter.

The formulation of the ASF problem require an automatically defined weight vector $\mathbf{w}$ for each reference point. The resulting multimodal ASF would then depend on the chosen weight vector. One idea would be to use a uniform weight vector, meaning a constant $w_i = 1/\sqrt{M}$ for all i. However, such a weight vector would emphasize the extreme efficient vectors more often than the intermediate ones. We suggest the following combination of reference points and weight vectors for our purpose. Instead of using different reference points and a single weight vector, we suggest using a single reference point and different weight vectors. First, we rename diverse reference points created using Das and Dennis's approach as *pivotal* points, $p_i^{(k)} = z_i^{(k)}$ for all $k = 1, 2, \ldots, K$. Then, an utopian point is declared as the only reference point: $z_i = f_i^* - \epsilon_i$, where f_i^* is the minimum objective value of the i-th objective and ϵ_i is a small positive number (0.01 is used throughout in this paper). Thereafter, each weight vector corresponding to each pivotal point is calculated as follows:

$$w_i^{(k)} = p_i^{(k)} - z_i, \quad \text{for all } k = 1, 2, \ldots, K. \tag{4}$$

The weight vector thus created can be normalized to convert it into a unit vector. Due to the use of an utopian vector as $\mathbf{Z}$, the above weight values are always strictly positive, thereby satisfying the non-negativity requirement of weights for ASF scalarization process of solving multi-objective optimization problems. Since the reference point $\mathbf{Z}$ can be any vector including an utopian vector, the above set-up of multiple ASF problems should result in a multimodal landscape with a set of discrete minima on the efficient front.

Figure 4 shows the contour plot of multimodal ASF for a two-objective ZDT2 problem for $p = 7$. The figure indicates that the ASF has eight different minimum solutions on the efficient front. An appropriately designed niched evolutionary algorithm should now find these eight efficient solutions in a single simulation run. We discuss such an evolutionary optimization algorithm in the next subsection.

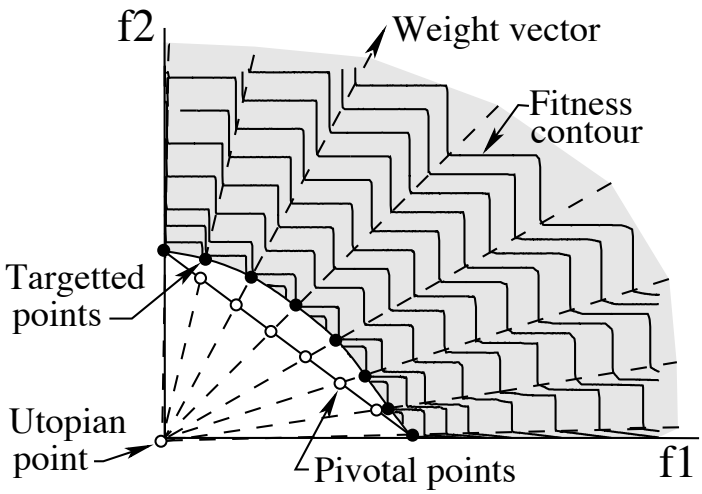

Figure 4: The multimodal ASF having eight minima is illustrated.

4.2 Developing a Niched Evolutionary Algorithm: MEMO

The basic framework of the proposed multi- or many-objective MEMO algorithm is given in Algorithm 1. Like in other guided many-objective optimization algorithms (Deb and Jain, 2014; Zhang and Li, 2007), the diversity among population members in MEMO is maintained by supplying a set of well-spread reference points. These points can either be arranged in a structured manner or based on user preferences. In this paper we use a systematic approach (Das and Dennis, 1998) for the construction of the reference set as explained in Section 4.1. Each reference point is expected to produce a niche of subpopulation, which we call a cluster here. The basic principle is to preserve individuals from different clusters. Another advantage of MEMO is that it does not require non-dominated sorting of solutions into different fronts. Only the first front needs to be identified in some particular type of problems for the sake of normalization. It should be noted that the particular way of handling different cluster members in the selection operation requires an extra cost in terms of function evaluations, but it is affordable.

Algorithm 1 MEMO procedure.

1: $t = 0$, initialize P_t of size N and H reference points $Z^{1,2,\ldots,H}$
2: $Archieve\_and\_Normalize(P_t)$ # **Procedure 1**
3: $[MinAASF, ClusterID] \leftarrow FitnessAssignment(P_t)$ # **Procedure 2**
4: **for** $t = 1$ **to** t_{max} **do**
5: $Q_t \leftarrow Selection(P_t)$ # **Procedure 3**
6: $Q_t \leftarrow Crossover(Q_t)$ # SBX (Deb and Agrawal, 1995)
7: $Q_t \leftarrow Mutation(Q_t)$ # Polynomial Mutation (Deb, 2001)
8: $[MinAASF, ClusterID] \leftarrow FitnessAssignment(Q_t)$ # **Procedure 2**
9: $P_{t+1} \leftarrow Merge\_and\_Reduce(P_t, Q_t)$ # **Procedure 4**
10: **end for**
11: Declare non-dominated solutions of $P_{t_{\max}+1}$ as optimized solutions

4.2.1 Procedure 1: *Archieve_and_Normalize( )*

First, the objective values need to be normalized in order to handle scaled multi-objective optimization problems where each objective function produces different magnitude of values. This is usually the case in real-world problems. Archiving the extreme points and the ensuing normalization procedure are identical to those used in NSGA-III procedure (Deb and Jain, 2014). Initially, the reference points ($\mathbf{Z}^{(k)}$, $k = 1, 2, \ldots, H$) are created on a unit hyper-plane such that each of them satisfies $\sum_{i=1}^{M} z_i^{(k)} = 1$.

However, in subsequent generations, they are updated as follows. At each generation t, the population-best solution $\mathbf{x}^{b,t}$ in terms of AASF value is identified and reference points are updated for the next generation as follows:

$$z_i^{(k,t+1)} = z_i^{(k,0)} \sum_{j=1}^{M} \bar{f}_j(\mathbf{x}^{b,t}), \quad i = 1, 2, \ldots, M \text{ and } k = 1, 2, \ldots, H. \tag{5}$$

In the above equation, $\bar{f}_j(\mathbf{x}^{b,t})$ is the normalized i-th objective function value at $\mathbf{x}^{b,t}$. This makes reference points come close to the non-dominated solutions at every generation. Thus, the reference points lie on a hyper-plane but move with the population to make a meaningful evaluation of the AASF value.

4.2.2 Procedure 2: *FitnessAssignment*( )

As mentioned earlier, in a similar philosophy with indicator based EMO algorithms or classical methods, MEMO converts a multi-objective optimization problem into a single-objective multimodal optimization problem by using the ASF concept that is introduced in Section 3.1.1. Minimum AASF computation (Equation 2) is not only used for the fitness assignment, but also for the cluster identification to help the new niching concept that is applied in the Selection and Merge_and_Reduce operators (**Procedure 2** and **4**).

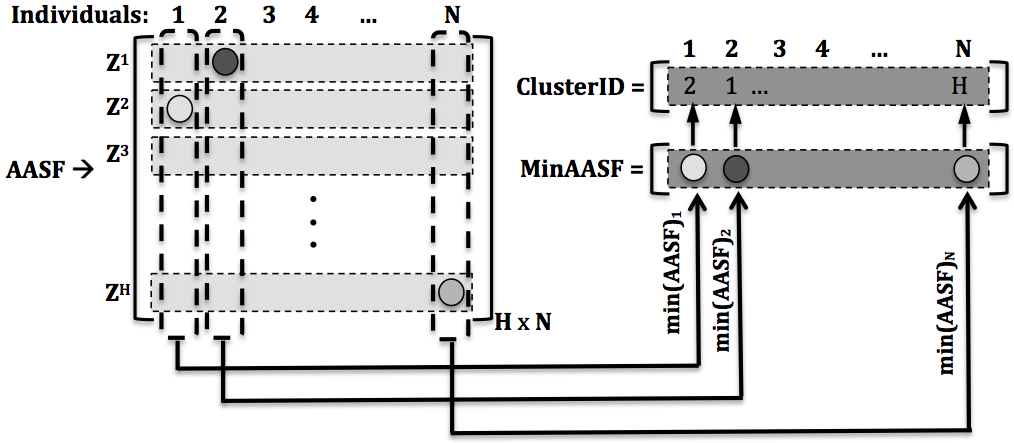

Figure 5: Fitness and cluster identification (reference point association) assignment scheme used in **Procedure 2** (*FitnessAssignment*())

As the first step, for each individual $\mathbf{x}$, AASF calculation with respect to each reference point $\mathbf{Z^{(k)}}$ should be performed. This is illustrated in Figure 5 as a vertical rectangle drawn with a dash line covering a column of the AASF matrix. Then the minimum value of each column (Equation 2) is assigned as the fitness value of that individual in $MinAASF$ array and the corresponding row index of this minimum value, which refers to the identification of the reference point to be associated, is recorded in the $ClusterID$ array. For instance, considering the representative example in Figure 5, the minimum AASF value of the first individual corresponds to the second element in the first column, therefore 2, the row index, is put in $ClusterID$ array as the first element and the minimum value is similarly stored in the $MinAASF$ array. In the initial generations, depending on the complexity of the problem, some individuals of the population might be associated to the same cluster and therefore some of the clusters may be empty as illustrated in Figure 6 as having third and seventh clusters being empty while first, second, fifth and sixth clusters having more than one individuals associated. If the optimization problem has constraints ($n_{cons} \neq 0$), there are two situations to consider for the fitness assignment: i) In case of having both feasible and infeasible solutions ($N_{feasible} > 0$), the minimum AASF and the constraint violation (CV) values, respectively, are given as the fitness value. ii) If there are no feasible solutions ($N_{feasible} = 0$), all solutions get their CV values as their fitness values. Independent of the feasibility check, cluster identification is performed in usual way.

Procedure 2 *FitnessAssignment*(P_t).

Initialize matrix $AASF_{H \times N}$
for $i = 1$ **to** $i = H$ **do**
  for $j = 1$ **to** $j = N$ **do**
    $AASF(i, j) \leftarrow compute\_AASF(\mathbf{Z}^i, P_t)$ # Equation 2
  end for
end for
if $N_{feasible} > 0$ **then**
  Index $feasible$ and $infeasible$ solutions
  for $i = 1$ **to** $i = N_{feasible}$ **do**
    $[Fitness_{(i,feasible)}, ClusterID_i] \leftarrow minimize(AASF(rows_{(1:H)}, column_{(i,feasible)}))$
  end for
  for $i = 1$ **to** $i = N_{infeasible}$ **do**
    $Fitness_{(i,infeasible)} \leftarrow CV_i$
    $[\sim, ClusterID_i] \leftarrow minimize(AASF(rows_{(1:H)}, column_{(i,infeasible)}))$
  end for
else if $N_{feasible} = 0$ **then**
  for $i = 1$ **to** $i = N$ **do**
    $Fitness_i \leftarrow CV_i$
    $[\sim, ClusterID_i] \leftarrow minimize(AASF(rows_{(1:H)}, column_i))$
  end for
end if
return $[MinAASF, ClusterID]$

4.2.3 Procedure 3: *Selection*( )

Binary tournament selection is used for the reproduction process. As aforementioned, the selection operator helps to preserve different cluster members throughout the generations. At each tournament, individuals from the same cluster are shuffled first and then compared pairwise. In an unconstrained optimization problem, two solutions are compared with respect to their AASF values ($MinAASF$) and the one with the smaller value is selected. If the problem has constraints (that is, $n_{cons} \neq 0$), three pairwise comparisons are performed: i) If both solutions are feasible, the decision is given as similar to the unconstrained case, ii) If both solutions are infeasible, then the one with the smaller constraint violation (CV) value is selected, iii) Finally, the feasible solution is preferred in case of comparing it with an infeasible solution. However, if there is only one solution in a cluster, that solution is automatically copied to Q_t. Limiting these pairwise comparisons between two solutions from the same cluster brings an extra cost of having slightly a bigger offspring population as compared to fixed-size EMO algorithms. However skipping the non-dominated sorting operation in MEMO or at least reducing the computational load by searching for only the first front and having the capability of simultaneously handling many-objective functions is an important trade-off. This may matter only in computationally heavy real-world optimization problems. But in all our simulations, we compare our proposed MEMO method with existing multi- and many-objective algorithms in terms of overall function evaluations.

Procedure 3 $Selection(P_t)$

```
1: tour_max ← 2
2: for k = 1 to k = tour_max do
3:    for cluster = 1 to cluster = H do
4:       tour_k ← [ ]
5:       tour_k ← shuffle(cluster_individual_indices)
6:       N_cluster ← size(tour_k)
7:       if N_cluster = 1 then
8:          Q_t ← P_t(tour_k[1])
9:       else if  then
10:         for j = 1 to j = N_cluster do
11:            if P_t(tour_k[j]) and P_t(tour_k[j + 1]) are feasible? then
12:               Q_t ← minimum(Fitness(tour_k[j], tour_k[j + 1]))
13:            else if P_t(tour_k[j]) and P_t(tour_k[j + 1])) are infeasible? then
14:               Q_t ← minimum(CV(tour_k[j], tour_k[j + 1]))
15:            else
16:               Q_t ← feasible(P_t(tour_k[j], tour_k[j + 1]))   # Feasible one is picked
17:            end if
18:         end for
19:      end if
20:   end for
21: end for
22: return  Q_t
```

Procedure 4 $Merge\_and\_Reduce(P_t, Q_t)$.

```
1: R_t ← P_t ∪ Q_t
2: Archieve_and_Normalize(R_t)
3: [MinAASF, ClusterID] ← FitnessAssignment(R_t)
4: clusters ← { }                                   # Niching (see Fig.6) starts
5: for i = 1 to i = H do
6:    j ← 1, 2, ..., size(clusters_i)
7:    clusters_i ← sort(clusters_i{Fitness_j})      # Local sorting
8: end for
9: counter ← 1
10: N_non−empty ← size(clusters_i ≠ 0)
11: while size(P_t+1) ≤ N do
12:    for k = 1 to k = N_non−empty do
13:       P_t+1 ← clusters_k{individual_counter}
14:    end for
15:    counter ← counter + 1
16: end while                                       # Niching ends
17: return  P_t+1
```

4.2.4 Procedure 4: $Merge\_and\_Reduce(\,)$

$Merge\_and\_Reduce(\,)$ is another important *niching* operation of the proposed MEMO algorithm which also ensures elitism. First, at generation t, parent P_t and offspring Q_t populations are combined to create a new population R_t of which size can be bigger than $2N$ where N is the size of P_t. However the size of the next parent population P_{t+1} obtained at the end of this procedure is reduced to N. After normalizing and evaluating the combined population R_t, we apply the niching procedure which is composed of two steps. In the first step, all clusters are filled with the individuals having the right $ClusterID$ value which was determined at $FitnessAssignment$ procedure (Section 4.2.2). In the same step, members of the same cluster are sorted with respect to their fitness values, that is, either $MinAASF$ or CV value, depending on their feasibility status. This is represented in Figure 6 with

circles of different grey scales being sorted from smaller radius (better fitness) to larger radius (worse fitness). In the second step, best solutions of each cluster (i.e., those encircled with dashed line and indicated as $counter = 1$) are put into P_{t+1} and this procedure is repeated for the next best solutions of all clusters ($counter = 2, 3, ...$) until $N_{fit} = H$ solutions are stored in P_{t+1}. In the early generations, some of the clusters might be empty but all of them will eventually be filled with at least one solution as the algorithm converges and produces better spread solutions. At the end, each individual of the final population is likely to be the optimum solution with respect to a specific reference point. These points together will then form the efficient set for the given multi-objective optimization problem.

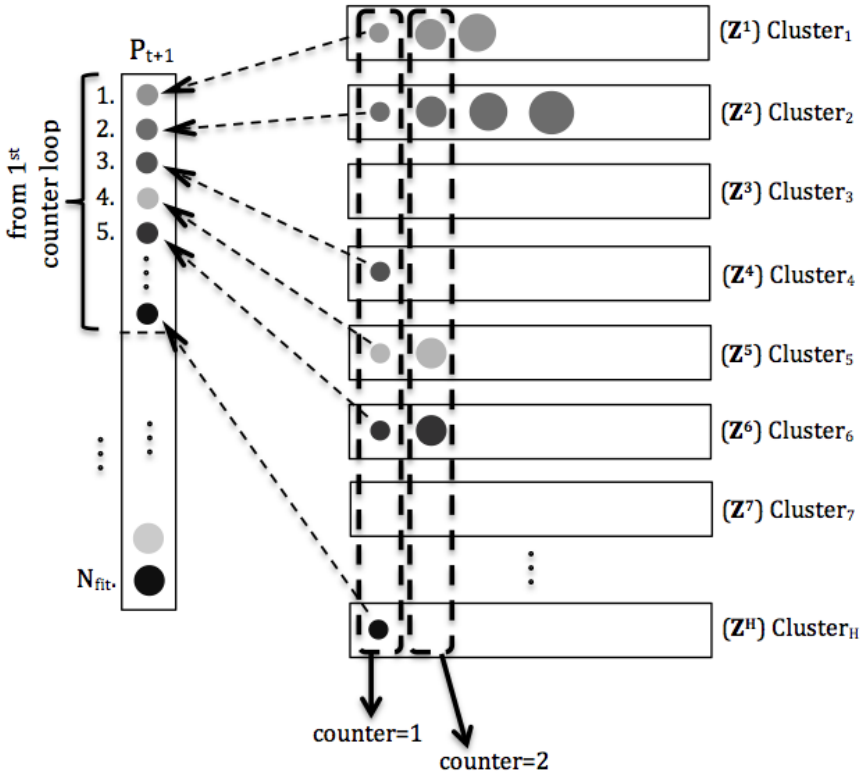

Figure 6: Niching scheme used in **Procedure 4** ($Merge\_and\_Reduce()$).

Overall, the MEMO procedure uses a predefined set of reference points to establish a niching within its population, but uses a single-objective AASF function described in Equation 2 to identify a suitable niche (or cluster) and evaluate each population member. In the following subsection, we estimate the computational complexity of one generation of MEMO procedure.

4.2.5 Computational Complexity of One Generation of MEMO

As mentioned before, Procedure 1, that is, $Archive\_and\_Normalize()$ in MEMO, is identical to Algorithm 2 in NSGA-III (Deb and Jain, 2014). However, the non-dominated sorting procedure, which is applied (before normalization) to assign each member its fitness (rank) and diversity (crowding distance) information, is not required in MEMO. The procedure in MEMO requires identification of non-dominated solutions from a population of size N, requiring $O(Nlog^{M-2}N)$ computations (Kung et al., 1975). An update of $H = N$ reference points (Equation 5) requires $O(N)$ computations. AASF matrix computation in $FitnessAssignment()$ requires calculation of AASF for every population member for every reference direction, thereby requiring HN or $O(N^2)$ computations. Other computations in this routine have smaller complexity. The $Selection()$ operator implements the niching idea and avoids comparison of population members among different clusters, therefore, the worst-case scenario happens when each cluster has a single associated population member. This results in creating an offspring population Q_t of size $2N$ (instead of N), but since the selection operator compares a pair of solutions at

a time in a systematic manner, this routine requires $O(N)$ computations. It will be interesting to investigate as a future study if restricting the size of Q_t to N by a more careful and systematic comparison of parent population P_t would affect the performance of MEMO. The $Merge\_and\_Reduce()$ does a few computationally expensive operations, particularly since the combined population R_t can be of size $3N$ as a worst-case scenario. First, it calls $Archive\_and\_Normalize()$ which requires $O((3N)log^{M-2}(3N))$ computations at most. Thereafter, the $FitnessAssignment()$ routine requires $H(3N)$ or $O(N^2)$ computations. The other sorting operations in this routine require lesser computational efforts. Thus, the overall computational complexity of one generation of MEMO, in its worst case, is $O(N^2)$.

5 Results and Discussions

In this section, we present the simulation results of MEMO algorithm on two to 10-objective optimization problems. The results of two-objective unconstrained and constrained optimization problems are compared with NSGA-II (Deb et al., 2002), whereas the rest of the results are compared with those of NSGA-III (Deb and Jain, 2014; Jain and Deb, 2014), since our method has a similar framework where a set of user-supplied reference points or weight vectors are used for guidance purposes. As a performance metric, we have chosen the hyper-volume (HV) metric (Zitzler and Thiele, 1998; Zitzler et al., 2003) for two-objective problems, which measures the volume of the dominated portion of the objective space, and in literature is also used as a selection pressure in indicator based EMO algorithms (Beurne et al., 2007; Igel et al., 2007; Bader and Zitzler, 2011). For problems with higher number of objectives, the inverse generational distance (IGD) metric (Zhang et al., 2008; Veldhuizen and Lamont, 1998), which provides information about the convergence and diversity of the obtained solutions, is used. The exact set of efficient solutions Z are known for the test problems and it is used for the computation of the IGD metric, as the average Euclidean distance of points in Z with their nearest members of all points in the efficient set obtained by MEMO. The smaller the IGD value indicated better performance whereas it is opposite for the HV metric. For each problem, algorithms are executed 20 times with different initial populations. Best, median, and worst HV and IGD performance values evaluated using the population members from the final generation are reported. Table 1 shows the number of chosen reference points (H) for different number of objectives (M) for a scalable test problem. The population size (N) of MEMO is identical to H, whereas the population size (N') of NSGA-III is equal to or higher than H ensuring being the smallest multiple of four. For two-objective problems, population size of NSGA-II is kept the same as in MEMO.

Table 1: The number of objectives and reference points for the test problems and the population size of MEMO and NSGA-III algorithms

Number of objectives (M)	Number of reference points (H)	MEMO population size (N)	NSGA-II/NSGA-III population size (N')
2	32	32	32
3	91	91	92
5	210	210	212
8	156	156	156
10	275	275	276

For two-objective problems, we have used $p = 31$ which results in $K = \binom{(2+31-1)}{31}$ or 32 reference points. We have used $p = 12$ for three-objective problems ($H = 91$) and $p = 6$ for five-objective problems ($H = 210$). For problems having more than five objectives, to have at least one intermediate reference point, $p \geq M$ results in a large number of (namely, $K = \binom{(8+8-1)}{8} = 5{,}040$ reference points for $M = 8$ and $p = 8$). To avoid this explosion, two layers of reference points with small values of p are used. For instance, for the eight-objective problem, $p = 3$ is chosen at the outer layer ($K = \binom{(8+3-1)}{3}$ $= 120$) and $p = 2$ is chosen for the inner layer ($K = \binom{(8+2-1)}{2} = 36$) resulting in total of 156 reference points. Ten-objective problems are prepared similarly, with $p = 3$ and $p = 2$ in outer and inner layers, respectively ($H = 220+55 = 275$). Table 2 presents other MEMO, NSGA-II, and NSGA-III parameters

used in this study.

Table 2: Parameter values used for MEMO and NSGA-II/NSGA-III where n is the number of variables.

Parameters	MEMO	NSGA-II/NSGA-III
SBX probability (Deb and Agrawal, 1995), p_c	1	1
Polynomial mutation probability (Deb, 2001), p_m	1/n	1/n
η_c	30	30
η_m	20	20

5.1 Two-objective Problems

First, we present results for two-objective problems.

5.1.1 Unconstrained Problems

We start testing MEMO with well-known two-objective, 30-variable unconstrained problems ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. As mentioned in Table 1, 32 reference points are used, therefore the population size is set accordingly. Since ZDT3 problem has a discontinuous efficient front, some of the reference points do not have a corresponding population or cluster member, thus to obtain a well distributed front, the number of reference points is set to 50 for this case. HV values are computed keeping the reference point set to [1.1, 1.1] for ZDT1, ZDT2, ZDT3 and ZDT4 problems and [1.05, 0.9] for the ZDT6 problem. Best, median and worst results are shown in Table 3 together with those obtained with NSGA-II. Though the results are very close, MEMO seems to perform slightly better.

Table 3: Best, median and worst hypervolume values for two-objective ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 problems using MEMO and NSGA-III algorithms.

Problem	$t_{\max}$		MEMO	NSGA-II
ZDT1	200	Best	**8.555 x** 10^{-1}	8.482 x 10^{-1}
		Median	**8.481 x** 10^{-1}	8.365 x 10^{-1}
		Worst	**8.534 x** 10^{-1}	8.298 x 10^{-1}
ZDT2	200	Best	5.218 x 10^{-1}	**5.250 x** 10^{-1}
		Median	5.172 x 10^{-1}	**5.229 x** 10^{-1}
		Worst	5.096 x 10^{-1}	**5.198 x** 10^{-1}
ZDT3	200	Best	**1.037 x** 10^{0}	1.034 x 10^{0}
		Median	**1.035 x** 10^{0}	1.029 x 10^{0}
		Worst	**1.034 x** 10^{0}	1.007 x 10^{0}
ZDT4	500	Best	**8.601 x** 10^{-1}	8.551 x 10^{-1}
		Median	**8.568 x** 10^{-1}	8.492 x 10^{-1}
		Worst	8.382 x 10^{-1}	**8.397 x** 10^{-1}
ZDT6	300	Best	**2.839 x** 10^{-1}	2.776 x 10^{-1}
		Median	**2.821 x** 10^{-1}	2.723 x 10^{-1}
		Worst	**2.781 x** 10^{-1}	2.671 x 10^{-1}

Figures 7-10 show the distribution of efficient front and the corresponding reference set for ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 problems. The performance of ZDT4 is similar to ZDT1 and therefore not shown here. Recall that the originally supplied reference points gets updated according to the current location of the non-dominated population members. In these figures, the reference points at the end of final generation ($t_{\max}$) are shown.

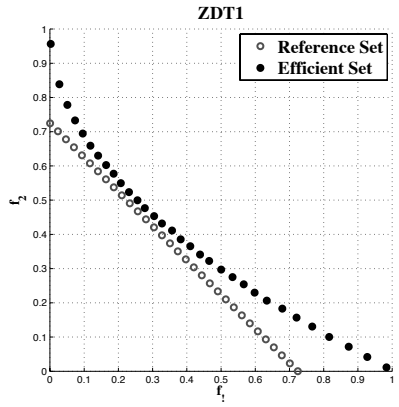

Figure 7: MEMO solutions for ZDT1 problem.

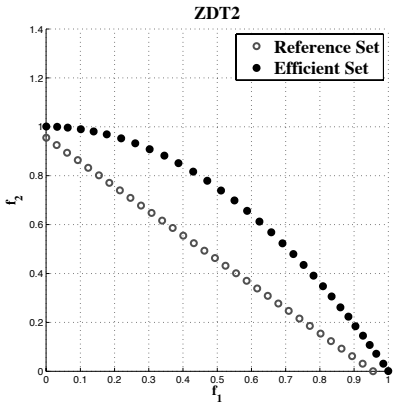

Figure 8: MEMO solutions for ZDT2 problem.

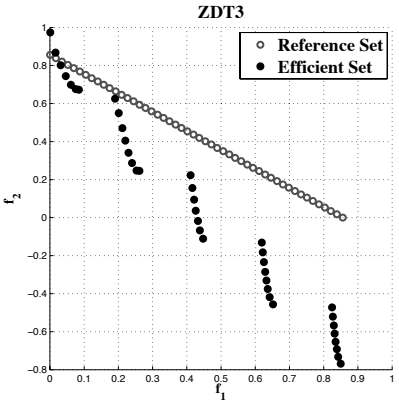

Figure 9: MEMO solutions for ZDT3 problem.

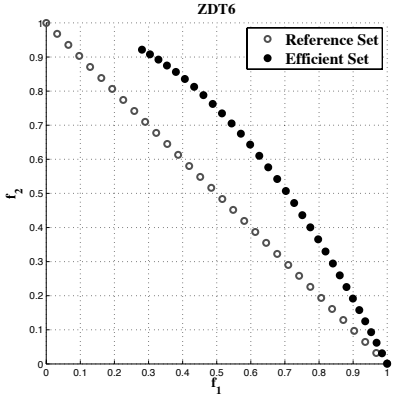

Figure 10: MEMO solutions for ZDT6 problem.

It is interesting to note that for ZDT3 and ZDT6, not all reference points will develop a cluster, as they have no supporting efficient point, but our proposed MEMO procedure is able to work on these problems equally well.

5.1.2 Constrained Problems

Next, we test the constraint handling capability of the proposed algorithm with two-objective and two-variable BNH (Binh and Korn, 1997), SRN (Srinivas and Deb, 1994) and TNK (Tanaka, 1995) test problems as well as an engineering optimization problem originated in the pultrusion process which is an efficient composite manufacturing process (Tutum et al., 2014). The reference sets are not shown in Figures 11 to 13, because the magnitudes of the objective values are at very different order as compared to [0,1] range as of the reference set. Algorithm settings are done with respect to Table 1 except for the TNK and pultrusion problem. Since TNK problem has a discontinuous efficient front, 50 uniformly distributed points are used for the reference set. The pultrusion problem is presented in detail at Section 5.1.3.

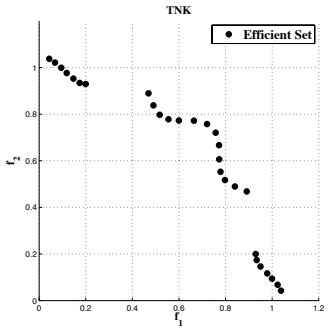

Figure 11: MEMO solutions on TNK problem.

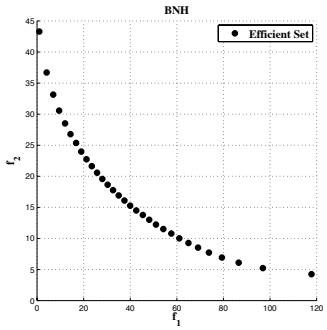

Figure 12: MEMO solutions on BNH problem.

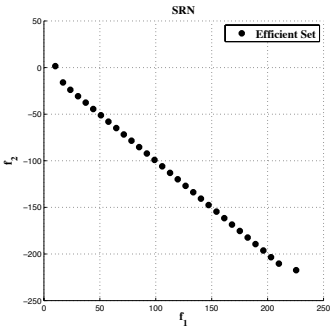

Figure 13: MEMO solutions on SRN problem.

The HV values obtained with MEMO and NSGA-II have been shown in Table 4. The results are very close, but MEMO seems to perform slightly better for this set of problems.

Table 4: Best, median and worst hypervolume values for two-objective BNH, SRN and TNK problems using MEMO and NSGA-II algorithms.

Problem	$t_{\max}$		MEMO	NSGA-II
BNH	300	Best	**6.150 x** 10^3	6.139 x 10^3
		Median	**6.146 x** 10^3	6.121 x 10^3
		Worst	**6.136 x** 10^3	6.073 x 10^3
SRN	300	Best	**4.190 x** 10^4	4.165 x 10^4
		Median	**4.187 x** 10^4	4.147 x 10^4
		Worst	**4.175 x** 10^4	4.136 x 10^4
TNK	300	Best	6.440 x 10^{-1}	**6.541 x** 10^{-1}
		Median	6.415 x 10^{-1}	**6.518 x** 10^{-1}
		Worst	6.328 x 10^{-1}	**6.494 x** 10^{-1}

5.1.3 Engineering Problem: Pultrusion Process

The engineering problem for the two-objective constrained optimization case is chosen from a composite manufacturing process application called pultrusion. The 3D steady-state approach for the thermo-chemical analysis of the process had previously been proposed elsewhere (Baran et al., 2013). The temperature and the degree of cure fields are obtained using the CV/FD (Control Volume/Finite Difference) method. Only one heater is used on the die and seven design variables (i.e., T-heater temperature, u_{pull}-pulling speed, w_{die}-die width, h_{die}-die height, L_{die}-die length, L-heater length, L_{init}-initial cooling length) are chosen which are considered to have major effect on the performance of the process. The optimization problem was described in detail in the original work (Tutum et al., 2014). The first objective is chosen as maximization of the production rate (that is, the pulling speed, u_{pull}) and the second objective is minimization of total energy consumption (TEC) which was formulated by authors as the multiplication of the area and the temperature of the heater. First objective is equivalently converted to the minimization problem by multiplying it with -1. There are three constraints to be satisfied. The first constraint is related to geometrical configuration of the heating die. The second constraint controls the maximum temperature in the composite part which ensures that it is below $240°C$. The third constraint ensures that the average cure degree (that is, degree of solidification of the composite part) measured at the end of the die should be higher than 0.9 (1.0 refers to fully solidified or cured state of the composite material and values close to 0.0 indicates the liquid state of the resin-fiber mix). The problem formulation is given in Equation 6. Each process simulation (or function evaluation) takes approximately 15 seconds on a quad-core CPU having 2.7 GHz and 16 GB of RAM. Keeping this

computational cost in mind we have used $p = 49$ which results in 50 reference points and therefore 50 individuals in the population of the MEMO algorithm. The maximum number of generations is set to $t_{\max} = 100$. Overall computation time needed for one run of MEMO was recorded to be approximately 26 hours.

$$\begin{array}{ll} \text{Maximize} & f(\mathbf{x}) = u_{pull}, \\ \text{Minimize} & f(\mathbf{x}) = TEC = Area_{heater}T = (w_{die}L)T, \\ \\ \text{subject to} & g_1(\mathbf{x}) = L_{die} \geq L_{init} + L, \\ & g_2(\mathbf{x}) = T_{max} < 240°C, \\ & g_3(\mathbf{x}) = \alpha_{avg,exit} > 0.9, \\ & g_4(\mathbf{x}) = 60 \leq L_{init} \leq 240\, mm, \\ & g_5(\mathbf{x}) = 60 \leq L \leq 360\, mm, \\ & g_6(\mathbf{x}) = 150 \leq T \leq 250°C, \\ & g_7(\mathbf{x}) = 600 \leq L_{die} \leq 1000\, mm, \\ & g_8(\mathbf{x}) = 120 \leq u_{pull} \leq 200\, mm/min, \\ & g_{9-10}(\mathbf{x}) = 45.4 \leq w_{die}, h_{die} \leq 200\, mm. \end{array} \tag{6}$$

The resulting efficient front (16 Pareto solutions), which is uniformly distributed between the limits of the pulling speed (first objective), is shown in Figure 14. Other 34 reference points did not result in any supporting efficient solution. Since in such real-world problems, no knowledge of the extent of efficient front is known beforehand, this is often an issue. The original NSGA-III study suggested a relocation plan for moving the reference points close to the observed efficient points adaptively (Jain and Deb, 2014). Similar ideas can be incorporated in MEMO, particularly for solving real-world problems. Nevertheless, these results are consistent with those obtained with a direct application of NSGA-II presented in the original study (Tutum et al., 2014).

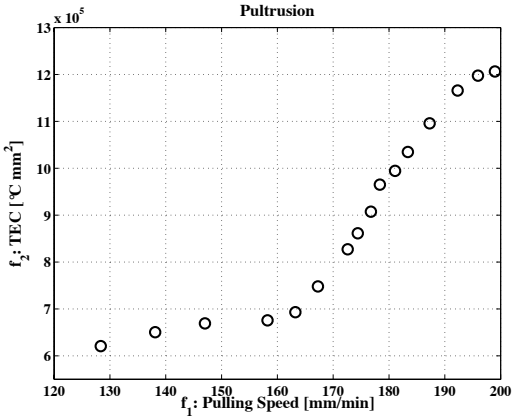

Figure 14: MEMO solutions for the pultrusion problem.

5.2 Three-objective Problems

Next, we apply our proposed MEMO procedure to three-objective problems.

5.2.1 Unconstrained Problems

First, we solve three-objective DTLZ1 and DTLZ2 problems, thereafter, we consider the scaled DTLZ1 and DTLZ2 as well as the convex DTLZ2 for unconstrained three-objective numerical test problems (Deb et al., 2005). The performance of MEMO is also investigated by testing it on an engineering problem called as – Crashworthiness (Liao et al., 2008) and results are compared with those of NSGA-III.

5.2.2 Normalized Test Problems

As a set of three-objective normalized test problems, DTLZ1 and DTLZ2 are used where the magnitudes of objective function values (f_i's) in their corresponding efficient fronts are similar. DTLZ problems are scalable regarding the number of objectives functions (M) as well as the number of variables, $n = M + k - 1$, where $k = 5$ for DTLZ1 and $k = 10$ for DTLZ2. Figure 15 shows the efficient front for DTLZ1. The obtained points are shown in black circles together with the final set of reference points marked in gray circles. The efficient front for three-objective DTLZ1 lies in $f_i \in [0, 0.5]$. Figure 16

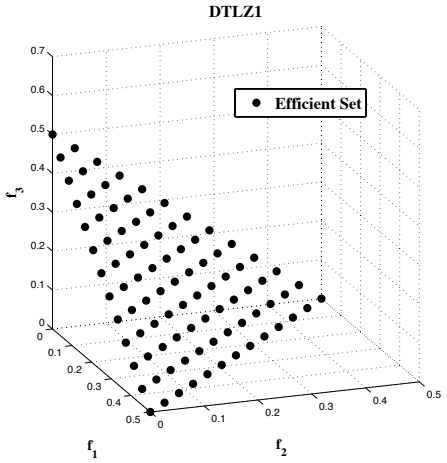

Figure 15: MEMO solutions are shown for DTLZ1 problem.

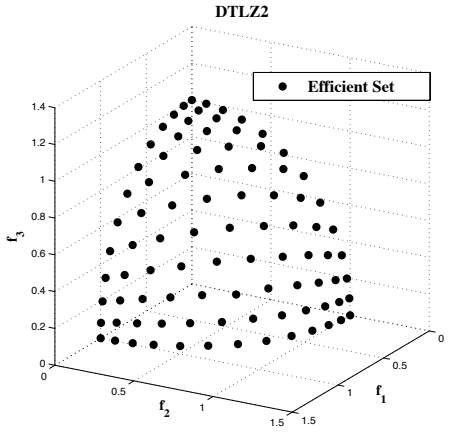

Figure 16: MEMO solutions are shown for DTLZ2 problem.

shows the efficient front for DTLZ2. Both the non-convex efficient front and the linear reference set lie in $f_i \in [0, 1]$ for DTLZ2, touching each other only at the corners, as shown in the figure. Both figures demonstrate MEMO's ability to find a uniformly distributed set of points on the respective efficient fronts. Table 5 summarizes the performance of two algorithms (MEMO and NSGA-III) using the IGD metric. MEMO seems to work better for three-objective DTLZ1 problem, whereas NSGA-III works slightly better for DTLZ2 over 20 runs.

Table 5: Best, median and worst IGD values obtained for MEMO and NSGA-III on three-objective DTLZ1 and DTLZ2 problems. Best results are shown in bold.

Problem	M	$t_{\max}$		MEMO	NSGA-III
DTLZ1	3	400	Best	**1.413 x** 10^{-4}	4.880 x 10^{-4}
			Median	**2.360 x** 10^{-4}	1.308 x 10^{-3}
			Worst	**2.008 x** 10^{-3}	4.880 x 10^{-3}
DTLZ2	3	250	Best	**1.045 x** 10^{-3}	1.262 x 10^{-3}
			Median	1.538 x 10^{-3}	**1.357 x** 10^{-3}
			Worst	2.311 x 10^{-3}	**2.114 x** 10^{-3}

5.2.3 Scaled DTLZ Test Problems

Real-world optimization problems usually involve objective values having different order of magnitudes. For example, a well-known structural design of a simply loaded beam problem has objectives and constraints related to stresses, displacements, natural frequencies, or weight of the structure where each of these may have values varying between 10^{-3} (namely, displacement constraint in terms of mm) to 10^6 (namely, stress constraint in terms of MPa). To investigate an algorithm's performance on such problems having differently scaled objective values, we consider scaled-version of DTLZ1 and DTLZ2 problems. In this section, only three objectives are considered. The scaling factor is 10^i where original

f_i's (f_1, f_2 and f_3) are multiplied with 10^0, 10^1 and 10^2, respectively. MEMO and NSGA-III still use the same methodology (Das and Dennis, 1998) to create the reference set. Normalization in **Procedure 1** enables the use of this reference set without any modification for the scaled problems. Figures 17 and 18 show the uniform distribution of the efficient sets which have different orders of magnitudes at each axis for scaled DTLZ1 and DTLZ2, respectively. Table 6 summarizes the IGD performance metric results and MEMO seems to have performed better than NSGA-III in general.

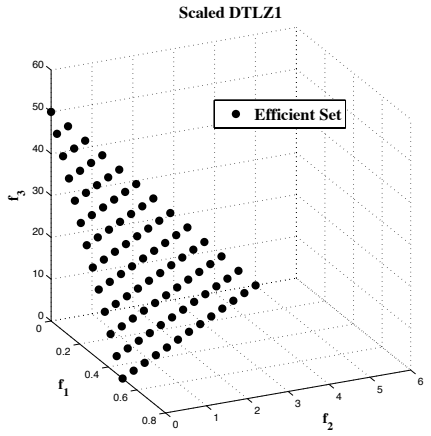

Figure 17: MEMO solutions are shown for scaled DTLZ1 problem.

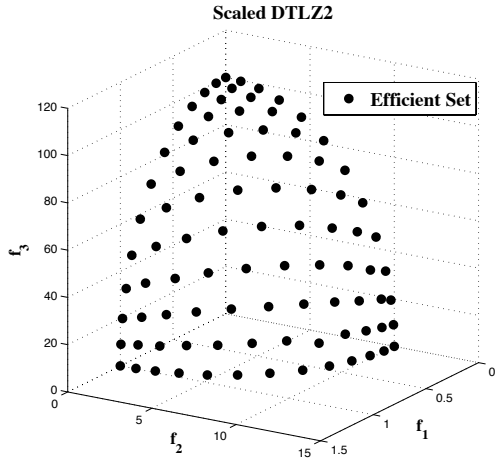

Figure 18: MEMO solutions are shown for scaled DTLZ2 problem.

Table 6: Best, median and worst IGD values for three-objective scaled DTLZ1 and DTLZ2 problems using MEMO and NSGA-III algorithms. A scaling factor of 10^i, i=1, 2 and 3 is used.

Problem	M	Scaling factor	$t_{\max}$		MEMO	NSGA-III
Scaled DTLZ1	3	10^i	400	Best	**3.441 x** 10^{-4}	3.853 x 10^{-4}
				Median	**6.696 x** 10^{-4}	1.214 x 10^{-3}
				Worst	**2.071 x** 10^{-3}	1.103 x 10^{-2}
Scaled DTLZ2	3	10^i	250	Best	**6.983 x** 10^{-4}	1.347 x 10^{-3}
				Median	**1.334 x** 10^{-3}	2.069 x 10^{-3}
				Worst	5.346 x 10^{-3}	**5.284 x** 10^{-3}

5.2.4 Convex Test Problem

DTLZ1 problem has a linear efficient front whereas DTLZ2 has a concave efficient front. To investigate the performance of MEMO on a convex efficient front, (Deb and Jain, 2014) suggested a new problem based on DTLZ2 problem. Figure 19 shows the obtained points on a three-objective convex efficient front problem with $H = 91$ reference points. Although the reference points are uniformly placed on a normalized hyper-plane, the efficient set is not uniformly distributed which creates a problem for the multi-objective optimization algorithms.

Table 7 presents the IGD metric results for both algorithms. NSGA-III performs slightly better than MEMO in this problem, but the results are comparable.

5.2.5 Engineering problem: Crashworthiness Design

This engineering problem is taken from (Liao et al., 2008) and it evaluates some of the major safety requirements of crashworthiness design by using a "full frontal crash" and an "offset-frontal crash"

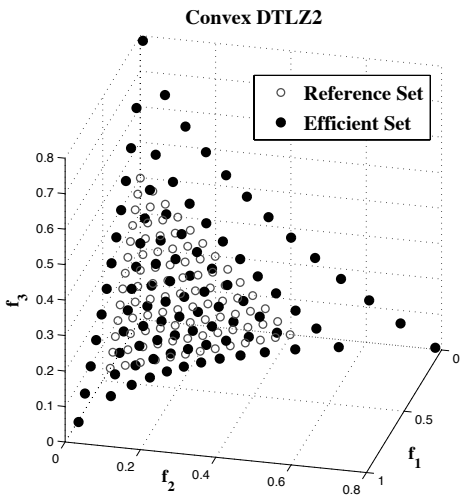

Figure 19: MEMO solutions and reference points for convex DTLZ2 problem.

Table 7: Best, median and worst IGD values for three-objective Convex DTLZ2 problem using MEMO and NSGA-III algorithms.

Problem	M	$t_{\max}$		MEMO	NSGA-III
Convex DTLZ2	3	250	Best	**2.144 x** 10^{-3}	2.603 x 10^{-3}
			Median	4.702 x 10^{-3}	**4.404 x** 10^{-3}
			Worst	9.063 x 10^{-3}	**8.055 x** 10^{-3}

test simulations. For this purpose, the mass of the structure, the integration of collision acceleration between t_1=0.05 s and t_2=0.07 s in the full frontal crash, and finally the toe board intrusion in the offset-frontal crash are considered to be three objectives (each to be minimized) as function of five reinforced members around the frontal structure. Analytical formulations for these objective functions, which were approximated by a surrogate methodology in the original work (Liao et al., 2008), are used in the current paper instead of computationally expensive transient finite element simulations.

For this problem, $p = 16$ is chosen which results in $K = \binom{(3+16-1)}{16}$ or 153 reference points, thus the population size of MEMO and NSGA-III are 153 and 156, respectively. Both algorithms are run for 200 generations. Figure 20 shows the obtained points with red '+' markers. The non-uniform scaling of objectives is clear from the figure. The efficient front is composed of 60 points though 153 reference points are used. This indicates that the rest of the reference points does not have any corresponding Pareto-optimal solutions, in other words a discontinuous Pareto-optimal set exists as a solution to this crashworthiness design problem.

The closeness of MEMO and NSGA-III solutions to the true efficient front is investigated by comparing those obtained with the classical generative method (using **fmincon**, indicated by black '+' markers). The IGD values are shown in Table 8. NSGA-III seems to perform slightly better than NSGA-III in this problem. Overall, since these values are relatively small, we can conclude that the optimal front obtained by these two algorithms are close to the classically optimized one. This problem shows that MEMO is able to find a uniform set of representative solutions for an engineering problem having several difficulties such as scaling of different objectives and a discontinuous Pareto set.

5.2.6 Constrained Problems

All the above problems did not have any constraints. Now, we apply MEMO procedure to several constrained multi- and many-objective optimization problems.

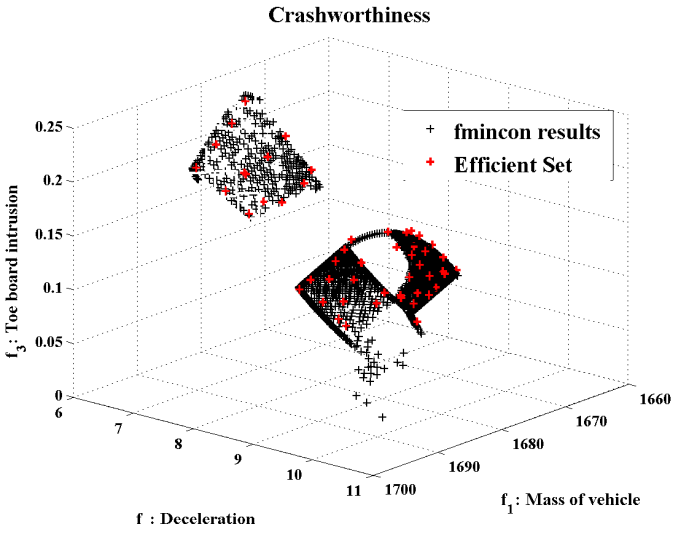

Figure 20: MEMO solutions on crashworthiness problem.

Table 8: Best, median and worst IGD values for three-objective Crashworthiness problem using MEMO and NSGA-III algorithms.

Problem	$t_{\max}$		MEMO	NSGA-III
CRASHWORTHINESS	200	Best	7.220 x 10^{-2}	**1.9 x** 10^{-3}
		Median	7.735 x 10^{-2}	**2.2 x** 10^{-3}
		Worst	7.838 x 10^{-2}	**2.6 x** 10^{-3}

5.2.7 Normalized Test Problems

In this study, two types of constrained problems are examined for three-to-ten objectives and these are classified as type-1 and type-3 problems in the original work (Jain and Deb, 2014). In type-1problems, the original Pareto-optimal set is identical to the one in the unconstrained version. There is an infeasible barrier in approaching the Pareto-optimal front in the first type of the problems. This is achieved by adding a constraint to the original problem (Jain and Deb, 2014). The barrier provides infeasible regions in the objective space that an algorithm must learn to overcome, thereby providing a difficulty in converging to the true Pareto-optimal front. DTLZ1 and DTLZ3 problems are modified according to this principle as similar to the previous study (Jain and Deb, 2014). For the type 1 constrained DTLZ1, the objective functions are kept the same as they were in the original DTLZ1 problem, while the following constraint is now added:

$$c(\mathbf{x}) = 1 - \frac{f_M(\mathbf{x})}{0.6} - \sum_{i=1}^{M-1} \frac{f_i(\mathbf{x})}{0.5} \geq 0. \tag{7}$$

In all simulations, we use $k = 5$ variables for the original g-function (Deb et al., 2005), thereby making a total of $(M + 4)$ variables to the M-objective C1-DTLZ1 problem. In the case of C1-DTLZ3 problem, a band of infeasible space is introduced adjacent to the efficient front (Jain and Deb, 2014). Again, the objective functions are kept the same as in original DTLZ3 problem (Deb et al., 2005), while the following constraint is added:

$$c(\mathbf{x}) = \left(\sum_{i=1}^{M} f_i(\mathbf{x})^2 - 16\right)\left(\sum_{i=1}^{M} f_i(\mathbf{x})^2 - r^2\right) \geq 0, \tag{8}$$

where, $r = \{9, 12.5, 12.5, 15\}$ is the radius of the hyper-sphere for $M = \{3, 5, 8, 10\}$. For C1-DTLZ3, we use $k = 10$, so that total number of variables are $(M + 9)$ in a M-objective problem. The results

for three-objective constrained DTLZ1 and DTLZ3 are presented below in Table 9. In cases where algorithm got stuck in local Pareto-optimal front the corresponding IGD value is not shown; instead the number of successful runs out of 20 are shown in brackets.

Table 9: Best, median and worst IGD values for three-objective C1-DTLZ1, C1-DTLZ3, C3-DTLZ1 and C3-DTLZ4 problems using MEMO and NSGA-III algorithms. For C1-DTLZ3 problem, 9 and 13 runs out of 20 runs came close to efficient fronts for MEMO and NSGA-III, respectively.

Problem	M	$t_{\max}$		MEMO	NSGA-III
C1-DTLZ1	3	500	Best	**9.532 x 10^{-4}**	1.229 x 10^{-3}
			Median	**1.018 x 10^{-3}**	4.932 x 10^{-3}
			Worst	**8.304 x 10^{-3}**	2.256 x 10^{-2}
C1-DTLZ3	3	1000	Best	2.705 x 10^{-3}	**8.649 x 10^{-4}**
			Median	3.834 x 10^{-2}	**8.139 x 10^{-3}**
			Worst	(9)	**(13)**
C3-DTLZ1	3	750	Best	**4.663 x 10^{-3}**	5.221 x 10^{-3}
			Median	**7.444 x 10^{-3}**	9.120 x 10^{-3}
			Worst	**1.972 x 10^{-2}**	2.058 x 10^{-2}
C3-DTLZ4	3	750	Best	**8.701 x 10^{-3}**	1.862 x 10^{-2}
			Median	**2.001 x 10^{-2}**	2.456 x 10^{-2}
			Worst	**3.602 x 10^{-2}**	5.586 x 10^{-1}

Next, we consider two type-III problems considered in the NSGA-III study (Jain and Deb, 2014). In these problems, M linear constraints are added to DTLZ1 and M quadratic constraints are added to DTLZ4 problem to construct C3-DTLZ1 and X3-DTLZ4 problems, respectively. The efficient surfaces for these problems come from some portions of the active constraint surfaces. In the case of C3-DTLZ1 problem, objective functions are same as in the original formulation (Jain and Deb, 2014), however, following M linear constraints are added:

$$c(\mathbf{x}) = \sum_{i=1, i\neq j}^{M} f_j(\mathbf{x}) + \frac{f_i(\mathbf{x})}{0.5} - 1 \geq 0, \ \ \forall j = 1, 2, ..., M. \tag{9}$$

As similar to C1-DTLZ1 problem, $k = 5$ is used in the original g-function, thereby making a total of $(M + 4)$ variables. For C3-DTLZ4 problem, the following M quadratic constraints are added:

$$c(\mathbf{x}) = \frac{f_j^2(\mathbf{x})}{4} + \sum_{i=1, i\neq j}^{M} f_i^2(\mathbf{x}) - 1 \geq 0, \ \ \forall j = 1, 2, ..., M. \tag{10}$$

DTLZ4 problem causes difficulty in creating solutions in certain parts of the objective space. For this problem, we have used $n = M + 4$ variables. Figures 21 and 22 show the distribution of MEMO points on three-objective version of DTLZ1 and DTLZ4 problems. In these problems, the original efficient front is infeasible and the new efficient front lies on several constraint boundaries, thereby making these problems difficult to solve. A uniform distribution of points on the entire efficient front indicates the ability of MEMO to handle such complex problems. Table 9 summarizes the results for C3-DTLZ1 and C3-DTLZ4 test problems having three objectives and three constraints.

5.2.8 Engineering Problem: Car Side Impact Problem

This engineering problem has three objectives: i) minimization of the weight of the car, ii) minimization of the pubic force experienced by a passenger, iii) minimization of the average velocity of the V-Pillar responsible for withstanding the impact load. All the three objectives are conflicting, therefore, a three-dimensional efficient front is expected. There are ten constraints involving limitations on a variety of response parameters and there are eleven design variables. Mathematical formulation for the problem is given in the work by (Jain and Deb, 2014).

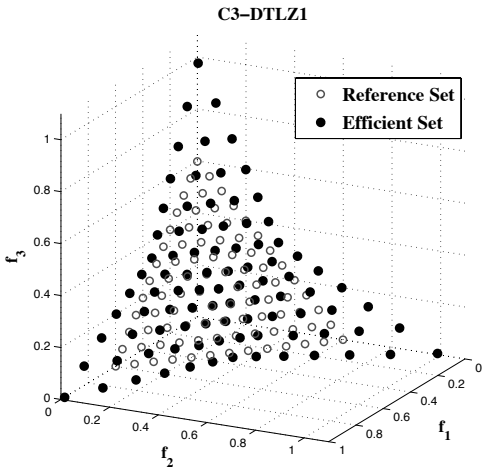

Figure 21: MEMO solutions and reference points are shown for constrained C3-DTLZ1 problem.

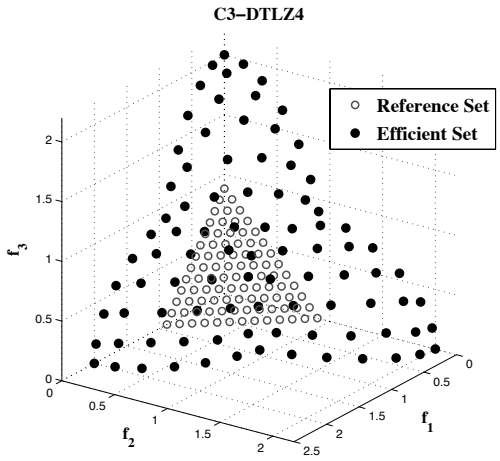

Figure 22: MEMO solutions and reference points are shown for constrained C3-DTLZ4 problem.

Similar to the crashworthiness design problem presented in section 5.2.5, $p = 16$ is chosen resulting in 153 uniformly distributed reference points. MEMO is applied with the population size of 153 for 500 generations. 95 unique solutions corresponding to 95 reference points are found. These points are shown with red solid circles in Figure 23. The rest of the reference points does not have a corresponding feasible solution. These results are next tested against a classical generative method (`fmincon()`). For this purpose, 6,216 reference points are created (by taking $p = 110$).

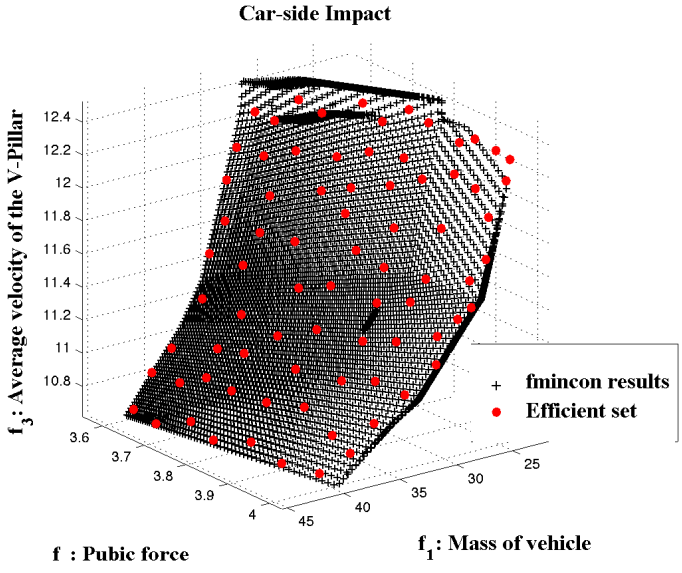

Figure 23: MEMO solutions for the car side-impact problem.

Table 10 shows the IGD metric computed by comparing the solutions obtained by classical generative method (`fmincon()`) with those obtained by MEMO algorithm. NSGA-III performs better, however the relatively small IGD values indicate that MEMO is also able to find a well distributed and converged set of Pareto solutions, as shown in Figure 23.

Table 10: Best, median and worst IGD values for three-objective car side-impact problem using MEMO and NSGA-III algorithms.

Problem	$t_{\max}$		MEMO	NSGA-III
CAR-SIDE IMPACT	500	Best	7.090 x 10^{-2}	**9.8 x** 10^{-4}
		Median	8.970 x 10^{-2}	**1.1 x** 10^{-4}
		Worst	9.890 x 10^{-2}	**1.3 x** 10^{-3}

5.3 Many-objective Problems

Having being successful in finding as many as around 100 different Pareto-optimal solutions using a multimodal approach, we are now ready to test MEMO for many-objective optimization problems for finding 200+ optima simultaneously.

5.3.1 Unconstrained Problems

First, we apply MEMO algorithm to many-objective version of unconstrained DTLZ1 and DTLZ2 problems. Table 11 presents IGD values for 5, 8 and 10-objective problems. Although NSGA-III's performance is slightly better, MEMO also manages to find small IGD metric values. Figures 24 and 25

Table 11: Best, median and worst IGD values obtained for MEMO and NSGA-III on M-objective DTLZ1 and DTLZ2 problems. Best performance is shown in bold.

Problem	M	$t_{\max}$		MEMO	NSGA-III
DTLZ1	5	600	Best	**3.703 x** 10^{-4}	5.116 x 10^{-4}
			Median	1.735 x 10^{-3}	**9.799 x** 10^{-4}
			Worst	3.707 x 10^{-3}	**1.979 x** 10^{-3}
	8	750	Best	5.509 x 10^{-3}	**2.044 x** 10^{-3}
			Median	7.414 x 10^{-3}	**3.979 x** 10^{-3}
			Worst	**8.683 x** 10^{-3}	8.721 x 10^{-3}
	10	1000	Best	7.206 x 10^{-3}	**2.215 x** 10^{-3}
			Median	9.244 x 10^{-3}	**3.462 x** 10^{-3}
			Worst	1.201 x 10^{-2}	**6.869 x** 10^{-3}
DTLZ2	5	350	Best	**3.584 x** 10^{-3}	4.254 x 10^{-3}
			Median	**4.499 x** 10^{-3}	4.982 x 10^{-3}
			Worst	**5.561 x** 10^{-3}	5.862 x 10^{-3}
	8	500	Best	4.379 x 10^{-2}	**1.371 x** 10^{-2}
			Median	5.872 x 10^{-2}	**1.571 x** 10^{-2}
			Worst	6.915 x 10^{-2}	**1.811 x** 10^{-2}
	10	750	Best	5.401 x 10^{-2}	**1.350 x** 10^{-2}
			Median	6.093 x 10^{-2}	**1.528 x** 10^{-2}
			Worst	6.701 x 10^{-2}	**1.697 x** 10^{-2}

show the parallel coordinate plots for 10-objective DTLZ1 and DTLZ2 problems, respectively, solved using MEMO. The trade-off and a broad distribution of objective values across all 10 objectives are clear from these two figures.

Table 12 presents MEMO's performance on scaled version of many-objective DTLZ1 and DTLZ2 problems. It is clear from the table that MEMO's performance is comparable to NSGA-III's performance on these problems.

Finally, we present MEMO results on five, eight, and 10-objective convex DTLZ2 problem in Table 13. Once again, comparable results to NSGA-III procedure are found for the proposed MEMO procedure.

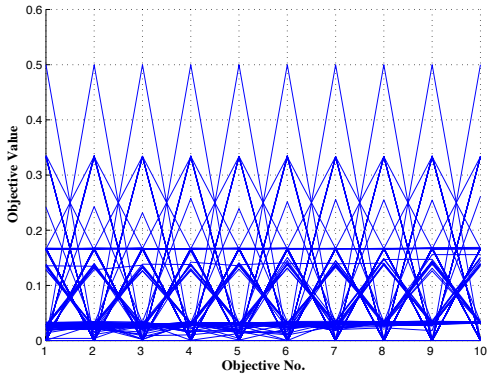

Figure 24: MEMO solutions are shown for scaled 10-objective DTLZ1 problem.

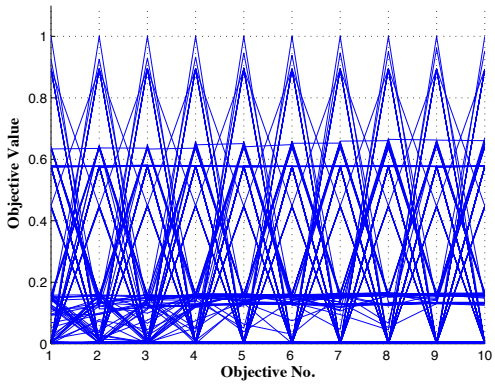

Figure 25: MEMO solutions are shown for scaled 10-objective DTLZ2 problem.

5.3.2 Constrained Problems

Next, we apply MEMO on constrained many-objective DTLZ1 and DTLZ3 problems proposed in the NSGA-III study (Jain and Deb, 2014). Although MEMO's performance is slightly worse than NSGA-III in these problems, the ability of MEMO procedure to produce a very similar performance to a truly many-objective optimizer is interesting.

Table 14 presents the best, median and worst IGD metric values for five, eight and 10-objective constrained C1-DTLZ1 and C1-DTLZ3 problems for solutions obtained using MEMO and NSGA-III. Identical parameter values are used for both algorithms. Similar performances can be observed from the table.

Next, we apply MEMO to constrained C3-DTLZ1 and C3-DTLZ4 problems for which all Pareto-optimal solutions lie on the different constraint boundaries, thereby making the problems difficult to solve. Table 14 shows the IGD metric values of obtained solutions for MEMO on five, eight and 10-objective problems and compares them with NSGA-III. Similar performances can be observed. It is interesting that a multimodal single-objective approach (MEMO) is able to find hundreds of optimal (in this case, Pareto-optimal) solutions simultaneously with an identical number of function evaluations to that in a state-of-the-art many-objective evolutionary optimization algorithm (NSGA-III). Figures 26 and 27 show 10-objective parallel coordinate plots of C3-DTLZ1 and C3-DTLZ4 problems. Figures indicate a good distribution of trade-off solutions.

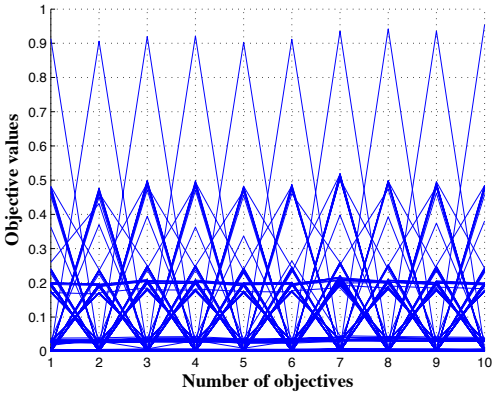

Figure 26: MEMO solutions are shown for constrained 10-objective C3-DTLZ1 problem.

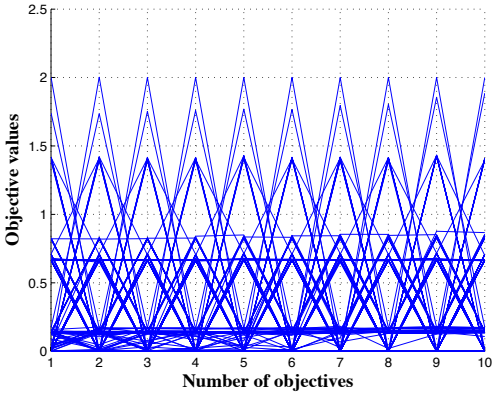

Figure 27: MEMO solutions are shown for constrained 10-objective C3-DTLZ4 problem.

Table 12: Best, median and worst IGD values for M-objective scaled DTLZ1 and DTLZ2 problems using MEMO and NSGA-III algorithms.

Problem	M	Scaling factor	$t_{\max}$		MEMO	NSGA-III
Scaled DTLZ1	5	10^i	600	Best	**6.405 x** 10^{-4}	1.099 x 10^{-3}
				Median	**1.155 x** 10^{-3}	2.500 x 10^{-3}
				Worst	**3.647 x** 10^{-3}	3.921 x 10^{-2}
	8	3^i	750	Best	1.183 x 10^{-2}	**4.659 x** 10^{-3}
				Median	5.449 x 10^{-2}	**1.051 x** 10^{-2}
				Worst	**1.097 x** 10^{-1}	1.167 x 10^{-1}
	10	2^i	1000	Best	3.912 x 10^{-2}	**3.403 x** 10^{-3}
				Median	7.133 x 10^{-2}	**5.577 x** 10^{-3}
				Worst	1.328 x 10^{-1}	**3.617 x** 10^{-2}
Scaled DTLZ2	5	10^i	350	Best	**9.603 x** 10^{-3}	1.005 x 10^{-2}
				Median	3.188 x 10^{-2}	2.564 x 10^{-2}
				Worst	**6.417 x** 10^{-2}	8.430 x 10^{-2}
	8	3^i	500	Best	5.482 x 10^{-2}	**1.582 x** 10^{-2}
				Median	8.901 x 10^{-2}	**1.788 x** 10^{-2}
				Worst	1.083 x 10^{-1}	**2.089 x** 10^{-2}
	10	3^i	750	Best	7.794 x 10^{-2}	**2.113 x** 10^{-2}
				Median	2.320 x 10^{-1}	**3.334 x** 10^{-2}
				Worst	5.224 x 10^{-1}	**2.095 x** 10^{-1}

Table 13: Best, median and worst IGD values for M-objective convex DTLZ2 problem using MEMO and NSGA-III algorithms.

Problem	M	$t_{\max}$		MEMO	NSGA-III
Convex DTLZ2	5	750	Best	**6.992 x** 10^{-3}	7.950 x 10^{-3}
			Median	2.474 x 10^{-2}	**1.341 x** 10^{-2}
			Worst	3.429 x 10^{-2}	**1.917 x** 10^{-2}
	8	2000	Best	**2.183 x** 10^{-2}	2.225 x 10^{-2}
			Median	4.199 x 10^{-2}	**2.986 x** 10^{-2}
			Worst	5.229 x 10^{-2}	**4.234 x** 10^{-2}
	10	4000	Best	**3.766 x** 10^{-2}	7.389 x 10^{-2}
			Median	**5.414 x** 10^{-2}	9.126 x 10^{-2}
			Worst	**6.576 x** 10^{-2}	1.051 x 10^{-1}

Table 14: Best, median and worst IGD values for M-objective Constrained DTLZ problems using MEMO and NSGA-III algorithms. Number of successful runs out of 20 runs are shown in brackets for some cases.

Problem	M	$t_{\max}$		MEMO	NSGA-III
C1-DTLZ1	5	600	Best	3.237×10^{-3}	$\mathbf{2.380 \times 10^{-3}}$
			Median	$\mathbf{4.122 \times 10^{-3}}$	4.347×10^{-3}
			Worst	1.309×10^{-2}	$\mathbf{1.024 \times 10^{-2}}$
	8	800	Best	6.475×10^{-3}	$\mathbf{4.843 \times 10^{-3}}$
			Median	3.101×10^{-2}	$\mathbf{1.361 \times 10^{-2}}$
			Worst	6.583×10^{-2}	$\mathbf{4.140 \times 10^{-2}}$
	10	1000	Best	7.042×10^{-3}	$\mathbf{3.042 \times 10^{-3}}$
			Median	1.007×10^{-2}	$\mathbf{6.358 \times 10^{-3}}$
			Worst	4.296×10^{-2}	$\mathbf{2.762 \times 10^{-2}}$
C1-DTLZ3	5	1500	Best	5.319×10^{-3}	$\mathbf{1.028 \times 10^{-3}}$
			Median	8.408×10^{-2}	$\mathbf{5.101 \times 10^{-2}}$
			Worst	(8)	**(15)**
	8	2500	Best	6.776×10^{-3}	$\mathbf{1.656 \times 10^{-3}}$
			Median	5.094×10^{-2}	$\mathbf{1.196 \times 10^{-2}}$
			Worst	(4)	**(14)**
	10	3500	Best	1.739×10^{-2}	$\mathbf{2.437 \times 10^{-3}}$
			Median	8.370×10^{-2}	$\mathbf{1.445 \times 10^{-2}}$
			Worst	(2)	**(18)**
C3-DTLZ1	5	1250	Best	$\mathbf{5.660 \times 10^{-3}}$	1.130×10^{-2}
			Median	$\mathbf{1.491 \times 10^{-2}}$	1.964×10^{-2}
			Worst	$\mathbf{2.180 \times 10^{-2}}$	4.745×10^{-2}
	8	2000	Best	4.655×10^{-2}	$\mathbf{1.243 \times 10^{-2}}$
			Median	6.043×10^{-2}	$\mathbf{2.104 \times 10^{-2}}$
			Worst	$\mathbf{7.878 \times 10^{-2}}$	8.196×10^{-2}
	10	3000	Best	5.551×10^{-2}	$\mathbf{8.450 \times 10^{-3}}$
			Median	6.059×10^{-2}	$\mathbf{1.509 \times 10^{-2}}$
			Worst	7.037×10^{-2}	$\mathbf{3.753 \times 10^{-2}}$
C3-DTLZ4	5	1250	Best	$\mathbf{1.084 \times 10^{-2}}$	3.247×10^{-2}
			Median	$\mathbf{1.560 \times 10^{-2}}$	3.854×10^{-2}
			Worst	$\mathbf{2.308 \times 10^{-2}}$	3.466×10^{-2}
	8	2000	Best	6.707×10^{-2}	$\mathbf{5.558 \times 10^{-2}}$
			Median	5.524×10^{-1}	$\mathbf{2.646 \times 10^{-1}}$
			Worst	9.322×10^{-1}	$\mathbf{8.886 \times 10^{-1}}$
	10	3000	Best	6.983×10^{-2}	$\mathbf{4.247 \times 10^{-2}}$
			Median	7.237×10^{-2}	$\mathbf{5.927 \times 10^{-2}}$
			Worst	9.951×10^{-1}	$\mathbf{9.092 \times 10^{-1}}$

6 Conclusions

In this paper, we have suggested a procedure for converting a multi/many-objective optimization problem into a suitable multimodal scalarized single-objective problem in which every Pareto-optimal solution of the original multi-objective problem becomes an independent optimum to the transformed multimodal problem. Thereafter, we have developed a niching-based multimodal evolutionary algorithm to find multiple (as many as 275) optimal solutions simultaneously. Finally, we have developed a constraint handling methodology that is well suited to the niching strategy to solve constrained multi- and many-objective optimization problems. Results on two to 10-objective test problems and on several practical design problems from automotive and manufacturing industries have been compared with other state-of-the-art EMO methodologies and comparable results have been reported. The ability of a single-objective multimodal approach to find hundreds of Pareto-optimal solutions in a single simulation using an evolutionary algorithm reliably and consistently on many constrained and unconstrained problems remains as a hallmark achievement of this study.

The conversion procedure is interesting and should open further viable avenues for using other scalarization methods for finding multiple Pareto-optimal solutions. On a similar spirit, other evolutionary multimodal optimization methods can also be tried and this may motivate researchers to pay further attention to multimodal optimization. The current implementation of MEMO does not restrict the size of offspring population to parent population size. Restricting its size and investigating its effect on MEMO would be an interesting future study. However, the ability of a multimodal yet single-objective optimization algorithm to reliably find multiple and in the tune of hundreds of multiple optimal (in this case, Pareto-optimal) solutions in handling multi- and many-objective optimization problems demonstrated in this study should allow researchers to develop a unified single-objective optimization method that is generic enough to solve different types of optimization problems.

Acknowledgments

The authors acknowledge the support provided by the Department of Electrical and Computer Engineering, Michigan State University for executing this study.

References

Bader, J. and Zitzler, E. (2011). HypE: An algorithm for fast hypervolume-based many-objective optimization. *MIT Press Evolutionary Computation Journal*, 19(1):45–76.

Baran, I., Hattel, J. H., and Tutum, C. C. (2013). Thermo-chemical modelling strategies for the pultrusion process. *Applied Composite Materials*, 20:1247–1263.

Barrera, J. and Coello, C. A. C. (2009). *A Review of Particle Swarm Optimization Methods Used for Multimodal Optimization*, pages 9–37. Berlin: Springer.

Bessaou, M., Petrowski, A., and Siarry, P. (2000). Island model cooperating with speciation for multimodal optimization. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN-6)*, pages 437–446. Springer-Verlag.

Beurne, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

Binh and Korn, U. (1997). MOBES: A multi-objective evolution strategy for constrained optimization problems. In *Third International Conference on Genetic Algorithms*, pages 176–182.

Cavicchio, D. (1970). *Adapting search using simulated evolution*. PhD thesis, Univ. of Michigan, Ann Arbor.

Chankong, V. and Haimes, Y. Y. (1983a). *Multiobjective Decision Making Theory and Methodology*. New York: North-Holland.

Chankong, V. and Haimes, Y. Y. (1983b). *Multiobjective Decision Making Theory and Methodology*. New York: North-Holland.

Coello, C. A. C., VanVeldhuizen, D. A., and Lamont, G. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. MA: Kluwer.

Czyz, J. A. and Lukasiewicz, S. A. (1995). Multimodal optimization of structures with frequency constraints. *AIAA Journal*, 33(8):1496–1502.

Das, I. and Dennis, J. (1998). Normal-Boundary Intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal of Optimization*, 8(3):631–657.

Das, S., Maity, S., Qu, B.-Y., and Suganthan, P. (2011). Real-parameter evolutionary multimodal optimization a survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88.

de Castro, L. N. and Zuben, F. J. V. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. UK: Wiley.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In *Third International Conference on Genetic Algorithms*, pages 42–50.

Deb, K. and Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601.

Deb, K. and Saha, A. (2012). Multimodal optimization using a bi-objective evolutionary algorithm. *MIT Press Evolutionary Computation Journal*, 20(1):27–62.

Deb, K. and Srinivasan, A. (2006). Innovization: Innovating design principles through optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2006*, page 16291636.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). *Scalable test problems for evolutionary multi-objective optimization*, pages 105–145. Springer-Verlag London.

Dilettoso, E. and Salerno, N. (2006). A self-adaptive niching genetic algorithm for multimodal optimization of electromagnetic devices. *IEEE Transactions on Magnetics*, 42(4):1203–1206.

Ehrgott, M. (2000). *Multicriteria Optimization*. Berlin: Springer.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423. San Mateo, CA: Morgan Kaufmann.

Goldberg, D. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *2nd International Conference on Genetic Algorithms*, pages 41–49.

Harik, G. (1997). Finding multi-modal solutions using restricted tournament selection. In *6th International Conference on Genetic Algorithms, ICGA-95*, pages 24–31.

Horn, J., Nafploitis, N., and Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multi-objective optimization. In *First IEEE Conference on Evolutionary Computation*, pages 82–87.

Igel, C., Hansen, N., and Roth, S. (2007). Covariance Matric Adaptation for multi-objective optimization. *MIT Press Evolutionary Computation Journal*, 15(1):1–28.

Im, C., Kim, H., Jung, H., and Choi, K. (2004). A novel algorithm for multimodal function optimization based on evolution strategy. *IEEE Transactions on Magnetics*, 40(2):1224–1227.

Jain, H. and Deb, K. (2014). An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):601–622.

Kung, H. T., Luccio, F., and Preparata, F. P. (1975). On finding the maxima of a set of vectors. *Journal of the Association for Computing Machinery*, 22(4):469–476.

Lee, C., Cho, D., and Jung, H. (1999). Niching genetic algorithm with restricted competition selection for multimodal function optimization. *IEEE Transactions on Magnetics*, 35(3).

Li, J.-P., Balazs, M. E., Parks, G. T., and Clarkson, P. J. (2002). A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3):207–234.

Li, X. (2005). Efficient differential evolution using speciation for multimodal function optimization. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO-2005)*, pages 873–880.

Liao, X., Li, Q., Zhang, W., and Yang, X. (2008). Multiobjective optimization for crash safety design of vehicle using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35:561–569.

Mahfoud, S. (1992). Crowding and preselection revisited. In *ParallelProblemSolving from Nature, PPSN 1992*, pages 27–37.

Mengsheol, O. and Goldberg, D. E. (1999). Probabilistic crowding: deterministic crowding with probabilistic replacement. In *Proceedings of genetic and Evolutionary computation conference (GECCO-1999)*, pages 409–416.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer.

Parrott, D. and Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, 10(4):440–458.

Petrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In *3rd IEEE International Conference on Evolutionary Computation*, pages 789–803.

Petrowski, A. (1997). An efficient hierarchical clustering technique for speciation. Technical report, Institute National des Telecommunications, Evry, France.

Preuss, M., Rudolph, G., and Tumakaka, F. (2007). Solving multimodal problems via multiobjective techniques with Application to phase equilibrium detection. In *Proceedings of the Congress on Evolutionary Computation (CEC2007)*. Piscataway, NJ: IEEE Press.

Rönkkönen, J., Li, X., Kyrki, V., and Lampinen, J. (2008). A generator for multimodal test functions with multiple global optima. In *Proceedings of the Seventh International Conference on Simulated Evolution and Learning (SEAL-08)*, pages 239–248.

Shir, O. and Bäck, T. (2005). Niching in evolution strategies. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO-2005)*, pages 915–916.

Shir, O. M., Emmerich, M., Bck, T., and Vrakking, M. J. J. (2007). Conceptual designs in laser pulse shaping obtained by niching in evolution strategies. In *Evolutionary Methods for Design, Optimization and Control*.

Shukla, P. and Deb, K. (2005). Comparing classical generating methods with an evolutionary multi-objective optimization method. In *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO-2005)*, pages 311–325.

Srinivas, N. and Deb, K. (1994). Multi-objective optimization using non-dominated sorting in genetic algorithms. *Structural and Multidisciplinary Optimization*, 2:221–248.

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation and Application*. New York: Wiley.

Streichert, F., Stein, G., Ulmer, H., and Zell, A. (2003). A clustering based niching EA for multimodal search spaces. In *Proceedings of the International Conference Evolution Artificielle*, pages 293–304. LNCS 2936.

Tanaka, M. (1995). GA-based decision support system for multi-criteria optimization. In *The International Conference on on Systems, Man and Cybernetics*, pages 1556–1561.

Tutum, C. C., Baran, I., and Deb, K. (2014). Optimum design of pultrusion process via evolutionary multi-objective optimization. *Int. J. Adv. Manuf. Technol.*, 72:1205–1217.

Ursem, R. (1999). Multinational evolutionary algorithms. In *Proceedings of Congress of Evolutionary Computation (CEC-99)*.

Veldhuizen, D. V. and Lamont, G. B. (1998). Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Dayton, OH.

Wierzbicki, A. P. (1980). The use of reference objectives in multiobjective optimization. In Fandel, G. and Gal, T., editors, *Multiple Criteria Decision Making Theory and Applications*, pages 468–486. Berlin: Springer-Verlag.

Wong, K.-C., Leung, K.-S., and Wong, M.-H. (2010). Protein structure prediction on a lattice model via multimodal optimization techniques. In *Genetic and Evolutionary Computation Conference, GECCO 2010*, pages 155–162.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In *Stochastic Algorithms: Foundations and Applications, SAGA-2009*, pages 169–178. Lecture Notes in Computer Sciences, 5792.

Yin, X. and Germay, N. (1993a). *A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization*, pages 450–457. Springer Verlag.

Yin, X. and Germay, N. (1993b). A fast genetic algorithm with sharing scheme using clustering analysis methods in multimodal function optimization. In *Proceedings of International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 450–457.

Zhang, J. and Li, K. (2005). A sequential niching technique for particle swarm optimization. In *Proceedings of 2005 international conference on intelligent computing (ICIC-2005)*, pages 390–399.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zhang, Q., Zhou, A., Zhao, S. Z., Suganthan, P. N., Liu, W., and Tiwari, S. (2008). Multiobjective optimization test instances for the CEC-2009 special session and competition. Technical report, Nanyang Technological University, Singapore.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms - A comparative case study. In *5th parallel Problem Solving From Nature*, pages 292–301.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.