On the Performance of Generational and Steady-State MOEA/D in the Multi-Objective 0/1 Knapsack Problem

Saúl Zapotecas-Martínez
Department of Applied Mathematics and Systems
UAM-Cuajimalpa
CDMX, C.P. 05300, MÉXICO
Email: szapotecas@correo.cua.uam.mx

Adriana Menchaca-Méndez
Technologies for Information in Science
ENES, Campus Morelia, UNAM
Morelia, C.P. 58190, Michoacán, MEXICO
Email: amenchaca@enesmorelia.unam.mx

Abstract—**The multi-objective evolutionary algorithm based on decomposition (MOEA/D) has attracted the attention of several investigators working on multi-objective optimization. At each iteration, MOEA/D generates an offspring solution from a parent's neighborhood. The new solution is evaluated, and, according to the decomposition approach, it can replace one or more solutions from the neighborhood, maintaining the population updated. In this sense, MOEA/D can be considered as a steady-state algorithm that maintains updated its population once a new solution is generated. In this work, we investigate the performance of MOEA/D in the multi-objective 0/1 knapsack problem considering a steady-state version and a proposed generational version. We explore the benefits of the generational version proposed in this paper. According to results, we show that the proposed approach can obtain a suitable performance in the multi-objective 0/1 knapsack problem employing between two and eight objective functions. Additionally, we propose a two-stage hybrid algorithm that employs the two different approaches of MOEA/D (i.e., the steady-state and generational versions). Our results reveal that the proposed hybrid approach can outperform the original MOEA/D in the many-objective settings of the 0/1 knapsack problem.**

I. Introduction

During the development of multi-objective optimization techniques, different principles have emerged to deal with the so-called multi-objective optimization problems (MOPs). The use of Pareto optimality and performance indicators were two popular approaches adopted by multi-objective algorithms in the early days of the 2000s. However, as has been discussed by several authors [1], [2], multi-objective algorithms adopting either Pareto optimality or performance indicators become inefficient as their performances decrease as the number of objectives increases. This has motivated the development of new strategies to deal with this type of problem. As a result, a novel stream of multi-objective evolutionary algorithms (MOEAs) based on decomposition has emerged as an alternative to deal with MOPs. Due to its efficiency and effectiveness, the multi-objective evolutionary algorithm based on decomposition [3] (MOEA/D) has attracted the attention of several investigators working on the multi-objective optimization field. MOEA/D relies on the regularity property of continuous MOPs, which establishes that, under some condition, the Pareto set (and Pareto front) of a continuous MOP with M objectives forms an $(M-1)$ dimensional piecewise continuous manifold in the decision space (and the objective space) [3]. In this way, MOEA/D solves several scalarizing (neighboring) subproblems, which are formulated by the same number of weight vectors in a cooperative way. This strategy to solve MOPs has become very useful to deal with complicated MOPs [4], [3], [5]. In the last decade, decomposition-based MOEAs have become an excellent alternative to deal with multi-objective problems. Thus, several variants or improvements of MOEA/D have been developed [4], [6], [5]. On the other hand, swarm-based metaheuristics using decomposition have also been investigated by some researchers [7], [8], [9], [10].

Although MOEA/D has successfully employed in several complicated MOPs [4], [11], its study in discrete problems is still a path to investigate. In this paper, we study the performance of MOEA/D in the multi-objective 0/1 knapsack problem in a many-objective setting (i.e., employing more than three objective functions). Particularly, we focus our investigation on the performance of MOEA/D from the perspective of steady-state and generational algorithms. In our study, we show that a generational version of a decomposition-based MOEA can obtain a good performance in discrete problems. Moreover, we show that by hybridizing the two different approaches (i.e., the steady-state and generational versions), it can be possible to outperform the original MOEA/D. As we will see later on, the proposed approach is able to achieve competitive results when solving multi-objective 0/1 knapsack problems adopting between two and eight objective functions.

The remainder of this paper is organized as follows. Section II introduces basic concepts related to multi-objective optimization and decomposition. Section III presents some studies related to steady-state and generational evolutionary algorithms. In Section IV, we introduce the proposed approach to solve the multi-objective 0/1 knapsack problem. Section V presents the experimental study and the analysis of results. Finally, in Section VI, we provide our conclusions and some possible paths for future research.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

II. BACKGROUND

A. Preliminaries of Multi-objective Optimization

Assuming maximization, a general *multi-objective optimization problem* (MOP) can be stated as:

$$\begin{array}{rlll} \text{maximize:} & \mathbf{F}(\mathbf{x}) & & \\ \text{s.t.} & g_i(\mathbf{x}) \leq 0, & i = 1, \ldots, p & \\ & h_j(\mathbf{x}) = 0, & j = 1, \ldots, q & \\ & \mathbf{x} \in X & & \end{array} \quad (1)$$

where $\mathbf{x} = (x_1, \ldots, x_n)^\intercal$ is an n dimensional vector of decision variables. The vector $\mathbf{F} = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}))^\intercal$ consists of M objective functions f_j's to be maximized. $g_i(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$ represent the p inequality constraints and the q equality constraints, respectively. The set of solutions satisfying the constraints of problem (1) defines the feasible region $\Omega \subset X$. In the case of pseudo-boolean combinatorial problems, the search space is defined by $X = \{0, 1\}^n$.

The following definitions introduce the concepts of interest in multi-objective optimization [12].

Definition *1:* Let $\mathbf{x}, \mathbf{y} \in \Omega$, we say that $\mathbf{x}$ *dominates* $\mathbf{y}$ (denoted by $\mathbf{x} \succ \mathbf{y}$) if and only if: *1)* $f_i(\mathbf{x}) \geq f_i(\mathbf{y})$ for all $i \in \{1, \ldots, M\}$ and *2)* $f_j(\mathbf{x}) > f_j(\mathbf{y})$ for at least one $j \in \{1, \ldots, M\}$.

Definition *2:* Let $\mathbf{x}^\star \in \Omega$, we say that $\mathbf{x}^\star$ is a *Pareto optimal* solution, if there is no other solution $\mathbf{y} \in \Omega$ such that $\mathbf{y} \succ \mathbf{x}^\star$.

Definition *3:* The *Pareto optimal set* PS is defined by: $PS = \{\mathbf{x} \in \Omega | \mathbf{x} \text{ is a Pareto optimal solution}\}$ and its image $PF = \{\mathbf{F}(\mathbf{x}) | \mathbf{x} \in PS\})$ is called *Pareto front* PF.

In the multi-objective optimization, we are interested in finding a finite number of solutions from the Pareto set, maintaining a proper representation of the Pareto front.

B. Decomposing a Multi-objective Optimization Problem

It is well-known [12] that a Pareto optimal solution to the problem (1) is an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the objective functions f_i's ($i \in \{1, \ldots, M\}$). Many scalarizing approaches have been proposed to aggregate the objectives of a MOP. In this study, we adopt the *Modified Tchebycheff* function since it offers a certain advantage over other approaches [13].
Nonetheless, other scalarizing functions can also be easily adopted. See, for example, those methods presented [14], [12].

Modified Tchebycheff approach. The modified Tchebycheff function [13] transforms the vector of objective functions $\mathbf{F}$ into a scalar maximization problem. Assuming maximization problems, the modified Tchebycheff problem is written as:

$$\begin{array}{rl} \text{minimize} & g^{mtch}(\mathbf{x}|\lambda, \mathbf{z}) = \max_{1 \leq j \leq M} \{\frac{1}{\lambda_j} |z_j - f_j(\mathbf{x})|\} \\ s.t. & \mathbf{x} \in \Omega \end{array} \quad (2)$$

where Ω is the feasible region, $\mathbf{z} = (z_1, \ldots, z_k)^\intercal$ is the reference point such that $z_j = \max\{f_j(\mathbf{x}) | \mathbf{x} \in \Omega\}$ for each $i = 1, \ldots, M$, and $\lambda = (\lambda_1, \ldots, \lambda_M)^\intercal$ is a weight vector, i.e., $\lambda_j \geq 0$ for all $j = 1, \ldots, M$ and $\sum_{j=1}^{M} \lambda_j = 1$.

For each Pareto optimal point $\mathbf{x}^\star$, there exists a weight vector λ such that $\mathbf{x}^\star$ is the optimum solution of equation (2) and each optimal solution of equation (2) is a Pareto optimal solution of equation (1). It is possible to obtain an appropriate representation of the Pareto front by solving different scalarizing problems. In such a case, the scalarizing problems are defined by a set of well-distributed weight vectors, which establish the search direction during the optimization process. In this way, an appropriate approximation to the Pareto front can be reached by optimizing a set of scalarizing functions.

C. Multi-Objective Evolutionary Algorithm based on Decomposition

The Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [3] transforms a MOP into several scalarizing subproblems. Considering $\Lambda = \{\lambda^1, \ldots, \lambda^N\}$ as a well-distributed set of weighting coefficient vectors, MOEA/D finds the best solution to each subproblem defined by each weight vector using a scalarizing function. As pointed out before, we employed the *Modified Tchebycheff* scalarizing function, see Equation (2). Since the reference vector $\mathbf{z} = (z_1, \ldots, z_M)^\intercal$ (in Equation (2)) is unknown *a priori*, MOEA/D states each component z_j by the maximum value for each objective f_j found during the search. On the other hand, in the case of a weighting coefficient $\lambda_j = 0$, λ_j is set to $10e{-}4$, for $j = 1, \ldots, M$.

In MOEA/D, a neighborhood (π^i) of a weight vector λ^i is stated as a set of its closest weight vectors in Λ. Therefore, the neighborhood of a weight vector λ^i contains all the indexes of the T closest weight vectors to λ^i. Thus, the reproduction and replacement mechanism of a solution takes place exclusively into the neighborhood of the concerned solution. Throughout the evolutionary process, MOEA/D tries to find the best solution to each subproblem, maintaining a population of N solutions $P = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ where $\mathbf{x}^i \in \Omega$ is the current solution to the i^{th} subproblem.

For the continuous case, there exist some variants of the original MOEA/D introduced by Zhang and Li [3]. In particular, MOEA/D-DE [4] is a variant of MOEA/D, which was specifically designed to solve MOPs with complicated PSs. The main differences between both algorithms (without considering the evolutionary operators) are i) MOEA/D-DE uses a probability (δ) to defined different neighborhoods and; ii) MOEA/D-DE defines a maximum number of solutions (n_r) to replace into the neighborhood. Thus, a generalized version of MOEA/D can be written as shown in Algorithm 1. This version of MOEA/D (assuming maximization problems) is the one adopted in this study.

III. STEADY-STATE AND GENERATIONAL MULTI-OBJECTIVE ALGORITHMS

Since the early 1980s, the performance of the generational and steady-state evolutionary algorithms have been studied [15], [16], [17], [18], [19], [20], [21], [22]. These two types of evolutionary algorithms differ in their replacement strategy. A generational evolutionary algorithm generates several offspring

Algorithm 1: General Framework of MOEA/D

```
Input:
  $N$: A number of subproblems to be decomposed;
  $\Lambda$: A set of weight vectors $\{\lambda^1, \ldots, \lambda^N\}$;
  $T$: The neighborhood size;
  $G_{\max}$: A maximum number of generations;
Output:
  $P$: A final approximation of the Pareto set.
1  $\mathbf{z} = (-\infty, \ldots, -\infty)^\intercal$;
2  Generate a random set of solutions $P = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ in $\Omega$;
3  for $i = 1, \ldots, N$ do
4      $B^i \leftarrow \{i_1, \ldots, i_T\}$, such that: $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the $T$
         closest weight vectors to $\lambda^i$;
5      $z_j \leftarrow \max\{z_j, f_j(\mathbf{x}^i)\}$;          // for $j \in \{1, \ldots, k\}$
6  $g = 0$;
7  while $g < G_{\max}$ do
8      foreach $i \in perm(\{1, \ldots, N\})$ do
9          if $rand() < \delta$ then $\pi^i = perm(B^i)$ ;
10         else $\pi^i = perm(\{1, \ldots, N\})$ ;
11         Generate a trial solution $\mathbf{y}$ by using solutions with
             indices in $\pi^i$;
12         $z_j \leftarrow \max\{z_j, f_j(\mathbf{y})\}$;          // $j \in \{1, \ldots, k\}$
13         $c = 0$;
14         foreach $j \in \pi^i$ do
15             if $g^{mtch}(\mathbf{y}|\lambda^j) < g^{mtch}(\mathbf{x}^j|\lambda^j)$ and $c < n_r$
                 then
16                 $\mathbf{x}^j = \mathbf{y}$;
17                 $c = c + 1$;
18     $g = g + 1$;
19 return $P$;
```

solutions by generation. Generally, the number of children is equal to the number of parents. Thus, the offspring solutions replace the parents ((μ, μ)-selection) or compete between them and survive the best individuals ($(\mu + \mu)$-selection). On the other hand, a steady-state evolutionary algorithm chooses two individuals from the population and generates a new offspring, the new solution is inserted in the population (i.e., updates the current population), and the worst individual is removed. Vavak et al. [19] were interested in solving problems in nonstationary environments. In these problems, the steady-state version of an evolutionary algorithm outperformed the generational version in terms of the convergence speed because the generational version can use outdated fitness values. In [20], the authors dealt with a scheduling problem. In their study, the authors noticed that the generational version of a genetic algorithm could outperform a steady-state genetic algorithm in terms of the convergence and quality of solutions. Jones et al. [21] were interested in the genetic robustness, which is a measure of the average change in fitness of an individual as a result of genetic codification. In their study, the authors used the two peaks problem using both a generational and a steady-state algorithm. They concluded that for this problem, the role of genetic robustness is significantly different in both versions. Therefore, they expect that it occurs the same in other problems. Recently, Jiang et al. [22] proposed to combine the generational and the steady-state versions of an evolutionary algorithm to solve dynamic multi-objective optimization problems. Their idea was to exploit the convergence speed of a steady-state algorithm and the diversity achieved by its generational version.

MOEA/D can be considered as a steady-state evolutionary algorithm because it creates a new individual (from a neighborhood), which can replace one or more solutions (in the neighborhood) according to its the scalarizing function value. In the specialized literature, it is possible to find several studies adopting MOEA/D to solve the multi-objective 0/1 knapsack problem, see the related work presented in [3], [23], [24], [25], [26], [27], [28], [29], [30]. However, all of these works employ a steady-state version of MOEA/D paying particular attention to different components of MOEA/D.

In [23], [25], the authors studied two scalarizing functions and proposed a mechanism to combine them into MOEA/D. Ishibuchi et al. [24] studied the behavior of MOEA/D when it uses large populations. They concluded that the efficiency of MOEA/D is not degraded by increasing the population size. However, a large population can obtain a large number of non-dominated solutions. Tan et al. [26] focused their study on how to generate convex weight vectors in high dimensional objective spaces. Thus, the authors proposed the use of the uniform design to generate the weighting coefficients for MOEA/D in a many-objective setting. Kafafy et al. [27] proposed to hybridize MOEA/D with other metaheuristics. They used an adaptative discrete differential evolution operator (called path-Relinking) to generate promising solutions to the multi-objective problem. Ke et al. [31] proposed to use the ant colony optimization metaheuristic into MOEA/D. In this algorithm, a neighborhood of MOEA/D is seen as an ant colony. Ishibuchi et al. [28] studied different scalarizing functions into MOEA/D when solving the multi-objective 0/1 knapsack problem. Sato [29] proposed the inverted PBI scalarizing approach in MOEA/D to improve the spread of the non-dominated solutions, and the resulting approach was proved using the multi-objective 0/1 knapsack problem. In [30], the authors studied the behavior of different MOEAs, including MOEA/D on the many-objective 0/1 knapsack problem using. According to results, MOEA/D was able to outperform MOEAs based on other principles. More recently, Zapotecas et al. [32], [33] employed geometric operators into MOEA/D to deal with the multi-objective 0/1 knapsack problem in a many-objective setting. As we can find in the specialized literature, investigations related to the performance of steady-state and generational MOEAs based on decomposition is a topic less studied which deserves to be investigated.

In this paper, we study the behavior of MOEA/D and a generational version of MOEA/D, which employs a survival mechanism based on the scalar selection strategy (*cf.* Section IV). In our study, we compare both approaches and propose a new algorithm that combines the generational and the steady-state version of MOEA/D. In the following section, we expose (in detail) the proposed version of each algorithm.

IV. A Generational MOEA/D and a Hybrid Approach

A generational version of MOEA/D can be stated by generating the full pool of offspring solutions, and then, from the parent and offspring population, a new population is obtained. In the following sections, we introduce a generational version of MOEA/D and a hybrid version that adopts the generational and steady-state principles.

A. Generational MOEA/D

A generational version of MOEA/D creates the whole offspring population whose cardinality is the same as the number of parent solutions. In this way, from the parent and offspring populations, a new population is obtained. In this paper, we follow the principles of decomposition employing the scalar selection mechanism in order to obtain the best solutions to each scalarizing function. More precisely, the new population is obtained from the set of parents and children selecting the N best solutions to the N scalarizing subproblems. Assuming maximization problems, Algorithm 2 shows the generational version of MOEA/D, which is studied in this work. In Algorithm 2, once an initial population is created (line 2), the evolution of the generational MOEA/D is carried out by selecting two random solutions from the current population (line 10). These two solutions are crossed and mutated in order to obtained two offspring solutions (line 11). The new solutions are stored in the Q population (line 13). After the pool of offspring solutions is fulled, the scalar selection mechanism is carried as shown in Algorithm 3. As we can see, the best solutions found in T (the union of parents P and children Q) to each scalarizing subproblem is preferred (see line 3 in Algorithm 3). In this work, the generational version of MOEA/D is called MOEA/D-Gen.

B. Hybrid Approach: Combining steady-state and generational MOEA/D

The proposed hybrid approach considers the generational and steady-state principles adopted by traditional evolutionary algorithms. In this way, our proposed hybrid algorithm interacts between a steady-state and a generational MOEA/D using a specific rule. It is possible to define several rules to interact with both algorithms. However, it can be generalized by defining a probabilistic or deterministic event p subjects to a tolerance Ψ. In this way, the probability of employing the generational principle is performed by the probabilistic/deterministic event with a threshold Ψ, otherwise the steady-state principle is carried out. In algorithm 4, we present the general framework of the hybrid algorithm (called here MOEA/D-HGen). As can be seen, in line 8, the rule to interact between both principles is introduced.

V. Experimental Study

A. Multi-Objective 0/1 Knapsack Problem

In order to test the performance of the different version of MOEA/Ds, we adopt one of the most studied NP-hard problems from combinatorial optimization, the knapsack problem in its multi-objective formulation.

Algorithm 2: A generational MOEA/D

```
Input:
  $N$: A number of subproblems to be decomposed;
  $\Lambda$: A set of weight vectors $\{\lambda^1, \ldots, \lambda^N\}$;
  $G_{\max}$: A maximum number of generations;
Output:
  $P$: A final approximation of the Pareto set.
1  $\mathbf{z} = (-\infty, \ldots, -\infty)^\intercal$;
2  Generate a random set of solutions $P = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ in $\Omega$;
3  for $i = 1, \ldots, N$ do
4      $z_j \leftarrow \max\{z_j, f_j(\mathbf{x}^i)\}$;   // for $j \in \{1, \ldots, k\}$
5  $g = 0$;
6  while $g < G_{\max}$ do
7      $i = 0$;
8      $Q = \emptyset$;
9      while $i < N$ do
10         Chose randomly two solutions $\mathbf{x}^1, \mathbf{x}^2 \in P$;
11         Generate two trial solutions $\mathbf{y}^1$ and $\mathbf{y}^2$ from
             crossover and mutation using the parents $\mathbf{x}^1$ and $\mathbf{x}^2$;
12         $z_j \leftarrow \max\{z_j, f_j(\mathbf{y}^1), f_j(\mathbf{y}^2)\}$;   // for $j \in \{1, \ldots, k\}$
13         $Q = Q \cup \{\mathbf{y}^1, \mathbf{y}^2\}$;
14         $i = i + 2$;
15     $P = ScalarSelection(P, Q, \Lambda, \mathbf{z})$;
16     $g = g + 1$;
17 return $P$;
```

Algorithm 3: Scalar Selection

```
Input:
  $P$: The parent population;
  $R$: The offspring population;
  $\Lambda$: The set of weight vectors $\{\lambda^1, \ldots, \lambda^N\}$;
  $\mathbf{z}$: The reference point;
Output:
  $P = \{\mathbf{x}^1 \ldots, \mathbf{x}^N\}$: The updated population.
1  $T = P \cup Q$;
2  for $i \in \{1, \ldots, N\}$ do
3      $\mathbf{x}^i = \arg\min_{\mathbf{y} \in T}\ g^{mtch}(\mathbf{y}|\lambda^i|\mathbf{z})$;
4  return $P = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$;
```

Given a collection of n items and a set of M knapsacks, the multi-objective 0/1 knapsack problem (MO-KNP) seeks a subset of items subject to capacity constraints based on a *weight function* vector $w : [0,1]^n \to \mathbb{N}^M$, while maximizing a *profit function* vector $p : [0,1]^n \to \mathbb{N}^M$. Formally it can be stated as:

$$\begin{array}{rll} \text{maximize:} & f_j(\mathbf{x}) = \sum_{i=1}^{n} p_{ji} \cdot x_i & j \in \{1, \ldots, M\} \\ \text{s.t.} & \sum_{i=1}^{n} w_{ji} \cdot x_i \leqslant c_j & j \in \{1, \ldots, M\} \\ & x_i \in \{0, 1\} & i \in \{1, \ldots, n\} \end{array} \quad (3)$$

where $p_{ji} \in \mathbb{N}$ is the profit of item i on knapsack j, $w_{ji} \in \mathbb{N}$ is the weight of item i on knapsack j, and $c_j \in \mathbb{N}$ is the capacity of knapsack j.

We consider the standard instances employing a random uncorrelated profit and weight integer values taken uniformly

Algorithm 4: General Framework of MOEA/D-HGen

```
Input:
  $N$: A number of subproblems to be decomposed;
  $\Lambda$: A set of weight vectors $\{\lambda^1, \dots, \lambda^N\}$;
  $T$: The neighborhood size;
  $G_{\max}$: A maximum number of generations;
Output:
  $P$: A final approximation of the Pareto set.
1  $\mathbf{z} = (-\infty, \dots, -\infty)^\intercal$;
2  Generate a random set of solutions $P = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ in $\Omega$;
3  for $i = 1, \dots, N$ do
4      $B^i \leftarrow \{i_1, \dots, i_T\}$, such that: $\lambda^{i_1}, \dots, \lambda^{i_T}$ are the $T$ closest weight vectors to $\lambda^i$;
5      $z_j \leftarrow \max\{z_j, f_j(\mathbf{x}^i)\}$;    // for $j \in \{1, \dots, k\}$
6  $g = 0$;
7  while $g < G_{\max}$ do
8      if $(p < \Psi)$// the deterministic/probabilistic event
9      then
10         $Q = \emptyset$;
11         while $i < N$ do
12             Chose randomly two solutions $\mathbf{x}^1, \mathbf{x}^2 \in P$;
13             Generate two trial solutions $\mathbf{y}^1$ and $\mathbf{y}^2$ from crossover and mutation using the parents $\mathbf{x}^1$ and $\mathbf{x}^2$;
14             $z_j \leftarrow \max\{z_j, f_j(\mathbf{y}^1), f_j(\mathbf{y}^2)\}$;
               // $j \in \{1, \dots, k\}$
15             $Q = Q \cup \{\mathbf{y}^1, \mathbf{y}^2\}$;
16             $i = i + 2$;
17         $P = ScalarSelection(P, Q, \Lambda, \mathbf{z})$;
18     else
19         foreach $i \in perm(\{1, \dots, N\})$ do
20             if $rand() < \delta$ then $\pi^i = perm(B^i)$ ;
21             else $\pi^i = perm(\{1, \dots, N\})$ ;
22             Generate a trial solution $\mathbf{y}$ by using solutions with indices in $\pi^i$;
23             $z_j \leftarrow \max\{z_j, f_j(\mathbf{y})\}$;    // $j \in \{1, \dots, k\}$
24             $c = 0$;
25             foreach $j \in \pi^i$ do
26                 if $g^{mtch}(\mathbf{y}|\lambda^j, \mathbf{z}) < g^{mtch}(\mathbf{x}^j|\lambda^j, \mathbf{z})$ and $c < n_r$ then
27                     $\mathbf{x}^j = \mathbf{y}$;
28                     $c = c + 1$;
29     $g = g + 1$;
30 return $P$;
```

from $[10, 100]$. The capacity is defined to half of the total weight of a knapsack for each objective function, i.e. $c_j = \frac{1}{2}\sum_{i=1}^{n} w_{ji}$ for $j = 1, \dots, M$. In these conditions, about 50% of the items are expected to be in the Pareto optimal front.

In our experimental study, we generate random problems of 500 items for each objective space dimension. We consider instances with 2, 3, 4, 5, 6, 7, and 8 objectives[1]. In order to satisfy the constraints of the problem, we adopt a traditional repair mechanism, which guarantees the feasibility of solutions [34]. This mechanism removes items sorted in increasing order of the maximum profit/weight ratio over all knapsacks one at a time until the constraints are satisfied.

[1]The set of instances adopted in our comparative study are available at https://sites.google.com/view/mo-knp/

TABLE I: H values for the two-layered simplex-lattice design

M (objectives)	Layer	Layer configuration	Number of weights
2	1	$H = 99$	100
3	1	$H = 19$	210
4	1	$H = 9$	220
5	1	$H = 6$	210
6	2	$H_1 = 4, H_2 = 3$	182
7	2	$H_1 = 4, H_2 = 2$	238
8	2	$H_1 = 3, H_2 = 2$	156

B. Experimental Setup

We compare the decomposition-based MOEAs, i.e., MOEA/D, MOEA/D-Gen, and MOEA/D-HGen experimentally. The algorithms were evaluated, adopting the one-point crossover and the bit-wise mutation as in the original version of MOEA/D [3]. For a fair comparison, the set of weight vectors for all the algorithms was the same, and it was generated using the Simplex-lattice design [35], as follows. The settings of N (number of weights and population size) and $\Lambda = \{\lambda^1, \dots, \lambda^N\}$ are controlled by a parameter H. More precisely, $\lambda^1, \dots, \lambda^N$ are weight vectors whose component scalar weights λ^i_j ($i = 1, \dots, N$ and $j = 1, \dots, M$) take values in $\left\{\frac{0}{H}, \frac{1}{H}, \dots, \frac{H}{H}\right\}$. In this way, the number of all possible choice of vectors in Λ is given by $N = C^{M-1}_{H+M-1}$, where M is the number of objective functions. As we know, this number increases binomially with the number of objectives. Therefore, this methodology becomes impractical when the number of objectives increases. In this paper, we adopted the two-layered simplex-lattice design [36] to deal with MOPs in high dimensional objective spaces. This strategy uses the simplex-lattice design to generate an outside layer and an inside layer in the weights set. Fig. 1 illustrates the two-layered simplex-lattice design in $\mathbb{R}^3$ when using $H_1 = 2$ for the outside layer and $H_2 = 1$ for the inside layer. In this study, we compare the decomposition-based approaches by using the weights given by the two-layered simplex-lattice design for problems with more than five objectives. Otherwise, a single layer is employed. The complete configuration of H values for different dimensions of the two-layered simplex-lattice design is shown in Table I.

Table II presents the parameter settings used in our experimental study. The parameters for the adopted algorithms are stated as suggested by their respective authors. T is the neighborhood size, δ and n_r are the probability of selecting a determined neighborhood, and the maximum number of replacements in the neighborhood. P_c and P_m are the crossover rate and mutation rate.

As a preliminary investigation, we employ a deterministic event that establishes the two-stage version in the proposed MOEA/D-HGen algorithm. At the first stage, the generational scheme of MOEA/D, which promotes exploration, is employed by considering a period of time (the deterministic event). A

TABLE II: Parameters for MOEA/D, MOEA/D-Gen, and MOEA/D-HGen

Parameter	MOEA/D	MOEA/D-Gen	MOEA/D-HGen
T	20	—	20
δ	0.9	—	0.9
n_r	2	—	2
P_c	1	1	1
P_m	1/n	1/n	1/n
Ψ	—	—	0.75
G_{max}	5,000	5,000	5,000

subsequent exploitation phase is performed by the steady-state MOEA/D. Thus, the parameter p (in Algorithm 4) is related to the period of time in which the exploration phase is performed. In our study, $p = \frac{g}{G_{\max}}$, such that g and $G_{\max}$ are the current generation and the maximum number of generations executed by MOEA/D-HGen. Therefore, the parameter p in Algorithm 4 increases with the number of iterations. Consequently, at the beginning of MOEA/D-HGen, $p = 0$. In our experimental study, the search for all the evolutionary approaches was restricted to perform $G_{max} = 5,000$ generations.

C. Performance Assessment

In order to assess the performance of the algorithms considered in this study, 30 independent runs were performed for each MO-KNP instance. We adopt the hypervolume indicator to assess the proximity and distribution of solutions achieved by the MOEAs. Mathematically the hypervolume indicator can be stated as follows.

Normalized Hypervolume (HV) The hypervolume indicator (HV) was introduced in [37] to assess the performance of MOEAs. This performance indicator is Pareto compliant [38], and quantifies both proximity and distribution of non-dominated solutions along the PF of MOP. The hypervolume corresponds to the non-overlapped volume of all the hypercubes formed by a reference point $\mathbf{r}$ (given by the user) and each solution $\mathbf{a}$ in the PF approximation (A). Hypervolume indicator is mathematically stated as:

$$HV(A) = \mathcal{L}\left(\bigcup_{\mathbf{a}\in A}\{\mathbf{x}|\mathbf{a} \succ \mathbf{x} \succ \mathbf{r}\}\right) \tag{4}$$

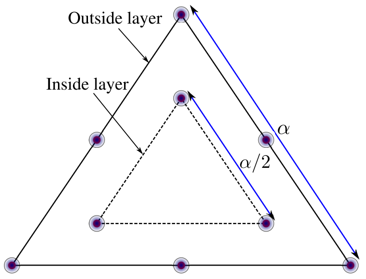


Fig. 1: Illustration of the two-layered simplex-lattice design. The outside layer is stated by $H_1 = 2$ (generating six weights vectors), while the inside layer is set by $H_2 = 1$ (generating three weights vectors)

where $\mathcal{L}$ denotes the Lebesgue measure and $\mathbf{r} \in \mathbb{R}^M$ denotes a reference vector being dominated by all solutions in A.

Therefore, the normalized HV indicator (denoted here as HV_n) is defined by

$$HV_n(A) = \frac{HV(A)}{\Pi_{i=1}^{M}|r_i - u_i|} \tag{5}$$

where $\mathbf{u} = (u_1, \ldots, u_M)^T$ is the known ideal vector and M denotes the number of objectives. Thus, HV_n value is given into the range $[0, 1]$. A high value of this performance indicator means that the set A has a good approximation and distribution along the true PF.

In our study, for a MO-KNP instance, the reference point $\mathbf{r} = (r_1, \ldots, r_M)^\intercal$ and the ideal vector $\mathbf{u} = (u_1, \ldots, u_M)^\intercal$ are stated by $r_j = \min_{\mathbf{x}\in T} f_j(\mathbf{x})$ and $u_j = \max_{\mathbf{x}\in T} f_j(\mathbf{x})$, where T is the set of non-dominated solutions found in the final approximations obtained by the algorithms in all the experiments.

D. Numerical Results and Discussion

Table III summarizes the results achieved by the algorithms in the MO-KNP instances adopting between two and eight objective functions. In each cell, the number on the left is the average indicator value, and the number on the right (in small font size) is the standard deviation from 30 independent experiments for each MO-KNP instance. The best values for each performance indicator and test problem are reported in **boldface**.

a) MOEA/D vs MOEA/D-Gen: In Table III, we can observe that the results obtained by MOEA/D-Gen were highly competitive against those produced by the original MOEA/D when solving the MO-KNP instance adopting up to five objective functions. As can be seen, the performance of the proposed MOEA/D-Gen deteriorated with the increase of objective functions. Note besides that, the performance of MOEA/D-Gen became much better than the original MOEA/D for the two-objective knapsack problem. This performance can be visualized in Fig. 2a. From this figure, we noted that the proposed MOEA/D-Gen achieved a better approximation of solutions towards the extreme portions of the Pareto front. This behavior suggests that maintaining the whole offspring population (as MOEAD/-Gen does it), it is possible to promote diversity to achieve the extremes of the Pareto front. Although this performance was not seen in high dimensional spaces, we argue that an additional mechanism can be implemented to improve the performance of MOEA/D-Gen when employing more objective functions.

b) MOEA/D-HGen vs MOEA/D-Gen: Considering the above discussion, we proposed the hybrid algorithm which tries to interact between the exploration and the exploitation of solutions during the search. From Table III, we can see that MOEA/D-HGen outperformed the generational version of MOEA/D-Gen in all the MO-KNP instances. As we can see in Fig. 2b, the distribution and approximation of solutions along the Pareto front was achieved in a better way by MOEA/D-HGen. Moreover, the achievement towards the extremes

TABLE III: Table of results achieved by MOEA/D, MOEA/D-Gen, and MOEA/D-HGen for the MO-KNP with 2, 3, 4, 5, 6, 7, and 8 objectives using the normalized Hypervolume indicator.

Objectives	MOEA/D	MOEA/D-Gen	MOEA/D-HGen
2	0.6464$\pm$0.016	0.6660$\pm$0.011	**0.6704**$\pm$0.013
3	0.5355$\pm$0.006	0.5205$\pm$0.008	**0.5382**$\pm$0.006
4	0.2853$\pm$0.007	0.2543$\pm$0.010	**0.2868**$\pm$0.008
5	**0.1685**$\pm$0.008	0.1389$\pm$0.007	0.1672$\pm$0.008
6	0.0937$\pm$0.009	0.0768$\pm$0.007	**0.0939**$\pm$0.007
7	0.0338$\pm$0.004	0.0297$\pm$0.004	**0.0346**$\pm$0.004
8	0.0123$\pm$0.002	0.0105$\pm$0.002	**0.0126**$\pm$0.002

portions of the Pareto front was outperformed by MOEA/D-HGen. These results suggest that the hybrid algorithm adopting the steady-state and the generational approaches can outperform the generational version of MOEA/D.

c) MOEA/D-HGen vs MOEA/D: Finally, we discuss the results obtained by MOEA/D-HGen and the original MOEA/D. Table III provides a quantitative assessment of the performance of the algorithms regarding the HV_n performance indicator. That means, the solutions obtained by MOEA/D-HGen achieved a better approximation and distribution along the Pareto front than those produced by MOEA/D in most of the MO-KNP instances. The exception was the five-objective knapsack problem when MOEA/D performed better than MOEA/D-HGen. However, as can be seen, the proposed hybrid algorithm was not significantly overcome. Fig. 2c shows the Pareto front approximations achieved by MOEA/D-HGen and MOEA/D. From this figure, it is possible to observe clearly that the MOEA/D-HGen obtained a better distribution of solutions along the Pareto front. The wide coverage of the Pareto front obtained by MOEA/D-HGen can be seen more clearly when comparing the original MOEA/D and MOEA/D-HGen. This behavior suggests that combining the steady-state and generational principle of evolutionary algorithms can provide better performance than an algorithm based on a single approach. Therefore, we conclude that the proposed hybrid approach is a viable algorithm to deal with the multi-objective 0/1 knapsack problem in high dimensional objective spaces.

VI. Conclusions

In this paper, we investigated different versions of MOEA/D. The original MOEA/D and two modified versions of MOEA/D introduced in this paper. The first modification of MOEA/D is a generational version of MOEA/D. This version creates the whole offspring population with the same number of solutions in the parent population. After that, the offspring and parent populations are joined, and the best N solutions are selected according to the principle of decomposition implicitly introduced in the scalar selection mechanism. We noticed that the generational version of our algorithm was highly competitive, achieving a better diversity towards the extremes portions of the Pareto front than the original MOEA/D. However, such performance deteriorated when the number of objectives increased. This way, we introduced a two-stage hybrid algorithm that employs the two different approaches of MOEA/D studied herein. Although several algorithms can be adopted in our comparative study, to understand in a better way the behavior of the algorithms, we compare the proposed algorithms against MOEA/D. Our results show that a combination between the steady-state and the generation principle adopted by evolutionary algorithms can overcome an algorithm based on a single principle. It is worth noticing that our proposed hybrid algorithm adopts a determinist rule to interact between the steady-state and generational principles. However, we advise that the performance of the proposed hybrid algorithm can be outperformed by tunning this parameter properly. In our ongoing research, we focus our investigation on this parameter. We consider that an appropriate tuning (which can be dynamic) of this parameter can be improved the performance of the proposed MOEA/D-HGen significantly.

Acknowledgment

The second author acknowledges support from PAPIIT project IA105918.

References

[1] K. Bringmann and T. Friedrich, "Approximating the Least Hypervolume Contributor: NP-Hard in General, But Fast in Practice," in *Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009*, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds. Nantes, France: Springer. Lecture Notes in Computer Science Vol. 5467, April 2009, pp. 6–20.

[2] K. Ikeda, H. Kita, and S. Kobayashi, "Failure of pareto-based moeas: does non-dominated really mean near to optimal?" in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 2, 2001, pp. 957–962.

[3] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[4] H. Li and Q. Zhang, "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, April 2009.

[5] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2015.

[6] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of moea/d on cec09 unconstrained mop test instances," in *2009 IEEE Congress on Evolutionary Computation*, May 2009, pp. 203–208.

[7] W. Peng and Q. Zhang, "A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems," in *2008 IEEE International Conference on Granular Computing*, Aug 2008, pp. 534–537.

[8] M. A. Medina, S. Das, C. A. C. Coello, and J. M. Ramírez, "Decomposition-based modern metaheuristic algorithms for multi-objective optimal power flow–a comparative study," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 10–20, 2014.

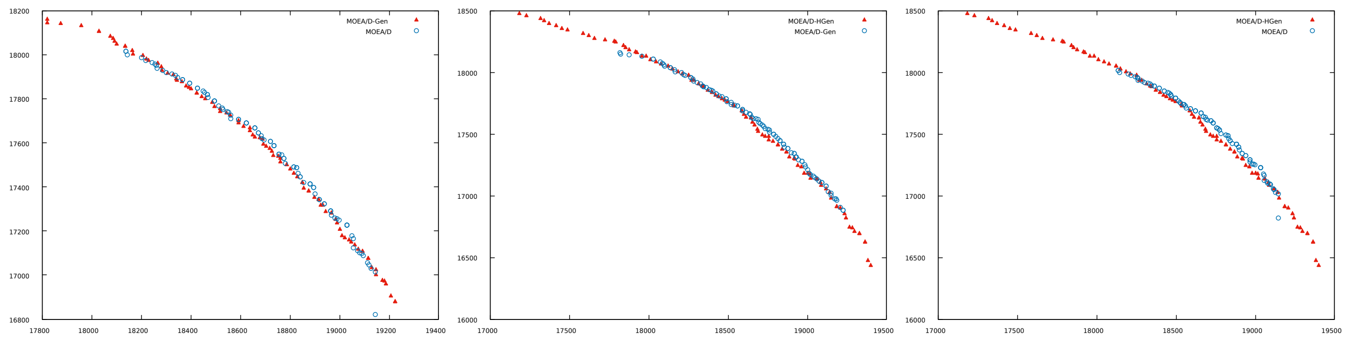
[9] S. Zapotecas-Martínez and C. A. Coello Coello, "A multi-objective particle swarm optimizer based on decomposition," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 69–76.

[10] S. Zapotecas-Martínez, A. García-Nájera, and A. López-Jaimes, "Multi-objective grey wolf optimizer based on decomposition," *Expert Systems with Applications*, vol. 120, pp. 357–371, 2019.

[11] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A Regularity Model-Based Multiobjective Estimation of Distribution Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, February 2008.

[12] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston, Massachuisetts: Kluwer Academic Publishers, 1999.

[13] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "Moead/d with adaptive weight adjustment," *Evol. Comput.*, vol. 22, no. 2, pp. 231–264, 2014.



(a) Pareto front approximations obtained by MOEA/D-Gen and MOEA/D (b) Pareto front approximations obtained by MOEA/D-HGen and MOEA/D-Gen (c) Pareto front approximations obtained by MOEA/D-HGen and MOEA/D.

Fig. 2: Comparison of Pareto fronts approximations achieved by MOEA/D, MOEA/D-Gen, and MOEA/D-HGen. The plots correspond to the best execution produced by the algorithms in the two-objective knapsack problem.

[14] M. Ehrgott, *Multicriteria Optimization*, 2nd ed. Springer, Berlin, June 2005.

[15] J. J. Grefenstette, “Optimization of control parameters for genetic algorithms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 122–128, Jan 1986.

[16] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, “A study of control parameters affecting online performance of genetic algorithms for function optimization,” in *Proceedings of the Third International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 51–60. [Online]. Available: http://dl.acm.org/citation.cfm?id=93126.93145

[17] D. E. Goldberg and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms,” ser. Foundations of Genetic Algorithms, G. J. RAWLINS, Ed. Elsevier, 1991, vol. 1, pp. 69 – 93. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080506845500082

[18] D. Noever and S. Baskaran, “Steady-state vs. generational genetic algorithms: A comparison of time complexity and convergence properties,” Tech. Rep., 07 1992.

[19] F. Vavak and T. C. Fogarty, “Comparison of steady state and generational genetic algorithms for use in nonstationary environments,” in *Proceedings of IEEE International Conference on Evolutionary Computation*, May 1996, pp. 192–195.

[20] K. P. Dahal and J. R. McDonald, “Generational and steady-state genetic algorithms for generator maintenance scheduling problems,” in *Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer Vienna, 1998, pp. 259–263.

[21] J. Jones and T. Soule, “Comparing genetic robustness in generational vs. steady state evolutionary algorithms,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’06. New York, NY, USA: ACM, 2006, pp. 143–150. [Online]. Available: http://doi.acm.org/10.1145/1143997.1144024

[22] S. Jiang and S. Yang, “A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 65–82, Feb 2017.

[23] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, “Adaptation of scalarizing functions in moea/d: An adaptive scalarizing function-based multiobjective evolutionary algorithm,” in *Evolutionary Multi-Criterion Optimization*, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 438–452.

[24] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, “Evolutionary many-objective optimization by nsga-ii and moea/d with large populations,” in *2009 IEEE International Conference on Systems, Man and Cybernetics*, Oct 2009, pp. 1758–1763.

[25] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, “Simultaneous use of different scalarizing functions in moea/d,” in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’10. New York, NY, USA: ACM, 2010, pp. 519–526. [Online]. Available: http://doi.acm.org/10.1145/1830483.1830577

[26] Y. Tan, Y. Jiao, and X. Wang, “Moea/d with uniform design for the multiobjective 0/1 knapsack problem,” in *2011 Seventh International Conference on Computational Intelligence and Security*, Dec 2011, pp. 96–100.

[27] A. Kafafy, A. Bounekkar, and S. Bonnevay, “Hybrid metaheuristics based on moea/d for 0/1 multiobjective knapsack problems: A comparative study,” in *2012 IEEE Congress on Evolutionary Computation*, June 2012, pp. 1–8.

[28] H. Ishibuchi, N. Akedo, and Y. Nojima, “A study on the specification of a scalarizing function in moea/d for many-objective knapsack problems,” in *Learning and Intelligent Optimization*, G. Nicosia and P. Pardalos, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 231–246.

[29] H. Sato, “Inverted pbi in moea/d and its impact on the search performance on multi and many-objective optimization,” in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’14. New York, NY, USA: ACM, 2014, pp. 645–652. [Online]. Available: http://doi.acm.org/10.1145/2576768.2598297

[30] H. Ishibuchi, N. Akedo, and Y. Nojima, “Behavior of Multiobjective Evolutionary Algorithms on Many-Objective Knapsack Problems,” *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 264–283, April 2015.

[31] L. Ke, Q. Zhang, and R. Battiti, “Moea/d-aco: A multiobjective evolutionary algorithm using decomposition and antcolony,” *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1845–1859, Dec 2013.

[32] S. Zapotecas-Martínez, B. Derbel, A. Liefooghe, H. E. Aguirre, and K. Tanaka, “Geometric differential evolution in moea/d: A preliminary study,” in *Advances in Artificial Intelligence and Soft Computing*, G. Sidorov and S. N. Galicia-Haro, Eds. Cham: Springer International Publishing, 2015, pp. 364–376.

[33] S. Zapotecas-Martinez, A. Moraglio, H. E. Aguirre, and K. Tanaka, “Geometric particle swarm optimization for multi-objective optimization using decomposition,” in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 69–76. [Online]. Available: https://doi.org/10.1145/2908812.2908880

[34] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, Nov 1999.

[35] H. Scheffé, “Experiments With Mixtures,” *Journal of the Royal Statistical Society*, vol. Series B (Methodological), 20, no. 2, pp. 344–360, 1958.

[36] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, Aug 2014.

[37] E. Zitzler and L. Thiele, “Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study,” in *Parallel Problem Solving from Nature V*, A. E. Eiben, Ed. Amsterdam: Springer-Verlag, September 1998, pp. 292–301.

[38] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, “Performance Assessment of Multiobjective Optimizers: An Analysis and Review,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, April 2003.