

European Journal of Operational Research

A Reference Set Based Many-Objective Co-evolutionary Algorithm with an Application to the Knapsack Problem

--Manuscript Draft--

Manuscript Number:	EJOR-D-20-01764
Article Type:	Theory and Methodology Paper
Section/Category:	(S) Evolutionary computations
Keywords:	Evolutionary computations; Multiple objective programming; Combinatorial optimization; Many-objective; Co-evolution
Corresponding Author:	Mert Sahinkoc, Ph. D. Bogazici Universitesi Istanbul, Please Select TURKEY
First Author:	Mert Sahinkoc, Ph. D.
Order of Authors:	Mert Sahinkoc, Ph. D. Ümit Bilge, Prof.
Abstract:	<p>Despite the growing interest on many-objective evolutionary algorithms, studies on combinatorial problems are still rare. In this study, we choose many-objective knapsack problem (MaOKP) as the benchmark and target the challenges imposed by many-objectives in discrete search spaces by investigating several reference set handling approaches and combining several prominent evolutionary strategies in an innovative fashion. Our proposed algorithm uses elitist nondominated sorting and reference set based sorting, however reference points are mapped onto a fixed hyperplane obtained at the beginning of the algorithm. All evolutionary mechanisms are designed in a way to complement the reference set based sorting. Reference point guided path relinking is proposed as the recombination scheme for this purpose. Repair and local improvement procedures are also guided by reference points. Moreover, the reference set co-evolves simultaneously with the solution set, using both cooperative and competitive interactions to balance diversity and convergence, and adapts to the topology of the Pareto front in a self-adaptive parametric way. Numerical experiments display the success of the proposed algorithm compared to state-of-art approaches and yield the best results for MaOKP. The findings are inspiring and encouraging for the use of co-evolutionary reference set based techniques for combinatorial optimization.</p>
Suggested Reviewers:	
Opposed Reviewers:	

Dear Prof. Słowiński,

We are submitting our manuscript titled "*A Reference Set Based Many-Objective Co-evolutionary Algorithm with an Application to the Knapsack Problem*" to be considered for possible publication in the European Journal of Operational Research.

The manuscript comprises the first part of the Ph.D. dissertation of my student H. Mert Şahinkoç. He developed a co-evolutionary algorithm for many-objective combinatorial optimization problems by using the Knapsack problem as the benchmark, where he obtained remarkable results. Later, he extended the algorithm to problems with permutation encoding like many-objective versions of TSP and QAP, which also gave very good results. The current paper introduces the approach through the knapsack problem; we plan to handle the latter problems in another paper. We hope that the paper can bring this line of research, which mostly takes place within the computer science community, into the attention of a wider OR community.

Mert will be the corresponding author for our paper, but I will still be keeping track of the process very closely.

Thanks in advance for your consideration.

With best regards,

Prof. Ümit Bilge

Department of Industrial Engineering

Boğaziçi University, Turkey

Highlights

A Reference Set Based Many-Objective Co-evolutionary Algorithm with an Application to the Knapsack Problem

H. Mert Şahinkoç, Ümit Bilge

- A reference set based evolutionary approach for many-objective combinatorial problems
- Use of a fixed hyperplane provides a significant improvement in performance
- Innovative complementary strategies such as reference point guided path relinking
- Co-evolves reference and solution sets for self-adaptive divergence and convergence
- Yields the best results so far in the literature for many-objective knapsack problem

A Reference Set Based Many-Objective Co-evolutionary Algorithm with an Application to the Knapsack Problem

H. Mert Şahinkoç*, Ümit Bilge

Boğaziçi University, Department of Industrial Engineering, 34342, Istanbul, Turkey

Abstract

Despite the growing interest on many-objective evolutionary algorithms, studies on combinatorial problems are still rare. In this study, we choose many-objective knapsack problem (MaOKP) as the benchmark and target the challenges imposed by many-objectives in discrete search spaces by investigating several reference set handling approaches and combining several prominent evolutionary strategies in an innovative fashion. Our proposed algorithm uses elitist nondominated sorting and reference set based sorting, however reference points are mapped onto a fixed hyperplane obtained at the beginning of the algorithm. All evolutionary mechanisms are designed in a way to complement the reference set based sorting. Reference point guided path relinking is proposed as the recombination scheme for this purpose. Repair and local improvement procedures are also guided by reference points. Moreover, the reference set co-evolves simultaneously with the solution set, using both cooperative and competitive interactions to balance diversity and convergence, and adapts to the topology of the Pareto front in a self-adaptive parametric way. Numerical experiments display the success of the proposed algorithm compared to state-of-art approaches and yield the best results for MaOKP. The findings are inspiring and encouraging for the use of co-evolutionary reference set based techniques for combinatorial optimization.

Keywords: Evolutionary computations, Multiple objective programming, Combinatorial optimization, Many-objective, Co-evolution

*Corresponding author. Email: hayrullah.sahinkoc@boun.edu.tr

1. Introduction

Multi-objective optimization problems (MOPs) involve two or more conflicting objectives. Due to this conflicting nature, the optimal solutions to a MOP consist of a set of compromised solutions known as the Pareto optimal set. A *Pareto optimal* (or, *efficient*) solution is characterized such that no single objective can be improved without degrading at least one of the others, and as such, it corresponds to a *nondominated* point in the objective space. In MOPs, the goal is usually to determine the entire Pareto optimal set, or at least as many Pareto optimum solutions as possible. Multi-objective evolutionary algorithms (MOEAs), which can produce a set of efficient solutions in a single run due to their population-based structures, are often referred as “the method” to obtain good *Pareto front approximations* (Deb, 2001).

On the other hand, when MOPs involve more than three objectives, they are considered to deserve a new name as many-objective optimization problems (MaOPs). This is because, as the number of objectives increases, the number of nondominated solutions becomes huge and maintaining diversity as well as convergence for a good and well-spread approximation of the true Pareto front turns into a true challenge (Chand & Wagner, 2015) rendering classical MOEAs mostly ineffective. Several recent studies try to characterize the challenges posed by MaOPs and to propose approaches that adapt or extend existing MOEAs in an attempt to overcome these challenges. Consequently, the so-called many-objective evolutionary algorithms (MaOEAs) have become an emerging topic (Li et al., 2015).

Reference set based approaches are one of the most promising branches of MaOEAs. They use a set of references (reference points, reference vectors, or weight vectors) to assist diversification and increase the span of the resulting Pareto set approximation. A good reference set should be less sensitive to the problem type and Pareto front geometry and should be scalable to keep the algorithm successful despite the increase in the number of objectives. Thus, effective strategies for their positioning and utilization are critical. One research question of this study is the investigation of several reference set handling approaches in an attempt to develop an enhanced version.

Secondly, it has been observed that most of the methodological studies in this field use test suites composed of well-defined continuous mathematical functions such as DTLZ (Deb et al., 2005) and WFG (Huband et al., 2006). These problems are repeatedly preferred because they are scalable to any number of objectives and decision variables, and the exact positions and

shapes of the resulting Pareto optimal fronts are known in advance. Combinatorial optimization problems, on the other hand, still remain uncharted for the studies of many-objective approaches, despite having a vast amount of applications in real-life. However, these problems, which display specific characteristics due to their discrete nature, exhibit particular needs. Thus, the problem domain of this study is combinatorial optimization problems. Many-objective knapsack problem (MaOKP) is selected as the benchmark, as in some recent pioneering work in the field of combinatorial many-objective optimization research (Ishibuchi et al., 2014a). This choice makes a good starting point within the domain of combinatorial optimization because it is easy to generate knapsack problems; and infeasibility, hence the constrained nature, is seldom a problem since it is very easy to repair infeasible solutions.

After exploring the prominent aspects of existing approaches, some of these features are combined with several effective evolutionary strategies in an innovative fashion to design a successful MaOEA. The proposed MaOEA uses elitist nondominated sorting and reference set based sorting, however the reference points are mapped onto a “fixed hyperplane” obtained at the beginning of the algorithm. The use of a fixed hyperplane is the first novel idea that provides a significant improvement in performance. Next, the reference points are permitted to move on the fixed hyperplane in an attempt to adapt to the progression of evolution in terms of diversity. Finally, the reference point set co-evolves simultaneously with the population of candidate solutions. This co-evolutionary approach, which constitutes the key innovative part of our algorithm, provides the necessary adaptability to achieve both convergence and divergence, bringing also a conclusive success. The path relinking recombination scheme integrated with complementing selection mechanisms and repair/local improvement procedures, as well as the use of an external population structure to store nondominated solutions are other successful features of our proposed MaOEA.

The rest of the paper is organized as follows: The next section starts with a review of MaOEAs, which particularly emphasizes those that inspired our work. Co-evolutionary algorithms are also briefly outlined. Then, our benchmark problem, the MaOKP, is introduced. In Section 3, after outlining the main framework of the proposed algorithm, we explain its important features in detail. In Section 4, we present and discuss the numerical results where we compare the proposed approaches with the existing ones. Finally, conclusions and future work are provided in Section 5.

2. Related Work

Many-objective evolutionary approaches can be broadly categorized into seven classes based on the key idea used in their algorithms (Li et al., 2015). In the first class, the dominance rule is relaxed in order to differentiate among the excess amount of nondominated solutions and conduct a selection pressure towards the Pareto front. Diversity-based approach introduces a density-based criterion to support the dominance-based primary criterion in the evolutionary selection. In the indicator-based approach, performance indicators used to evaluate the Pareto approximations are also used to guide the evolutionary process. In preference-based approaches, the decision makers' preferences are incorporated in the search process to focus on the relevant subset of the Pareto front. Dimensionality reduction approach aims to deal with many-objective problems with redundant objectives. In aggregation-based or decomposition based approaches, the objective functions of a many-objective problem are aggregated into a single objective function and the Pareto front is decomposed into several sub-regions.

Reference set based approaches constitute the last class of many-objective evolutionary approaches. Since the approach we proposed in this study can be considered within this class, it will be reviewed in detail in the next subsection. In Section 2.2, we shall specifically overview a subset of this type of approaches which involves the adaptation the reference set during the evolutionary process. Furthermore, in Section 2.3, we will provide a brief overview for the co-evolutionary algorithms, an interesting extension to classical evolutionary algorithms and an important feature of our proposed approach. Finally, our benchmark problem, namely the many-objective knapsack problem (MaOKP), will be introduced in Section 2.4 together with its formulation and an overview of related literature.

2.1. Reference Set Based MOEAs

Some pioneering approaches in the literature adopt only a single reference point which represents the preferred ideal solution of the decision maker. For instance, in Wierzbicki (1980) the aim is to obtain the closest Pareto optimal solution to a given reference point, whereas the aim in Deb & Sundar (2006) is to obtain a set of solutions near a reference point. On the other hand, when the main interest is obtaining an approximation of the entire Pareto-front for a posteriori decision making, as is the case with our study, a set of well-distributed reference points can be employed. Reference set based

approaches use a set of reference points or solutions to guide the search process and measure the quality of solutions. Performance depends on how successfully the balance between convergence and diversity is sustained by the reference set.

Reference set based methods differ from one another in how they construct the reference set and measure the quality of the solutions with respect to the reference set during the evolution. Regarding the first issue, while some algorithms use real solutions in the reference set by treating some members of the population as reference points, other algorithms use a virtual reference set where a set of virtual points in the objective space is used to form an ideal front to lead the search process. Although more information about the population might be contained in a real reference set, virtual reference set approaches are more common as their locations and distributions can be better organized. The second issue is about how solutions are associated to the reference points. Different distance metrics and scalarizing functions such as Euclidean and Tchebycheff are adopted by different algorithms.

In Figueira et al. (2010), a set of reference points is generated after approximating the boundaries of the Pareto front, namely ideal and nadir points, and solutions close to each reference point are found in parallel by decomposing the objective space using the reference points. Fitness values of individuals are assessed using achievement scalarizing functions.

In the NSGA-III algorithm proposed by Deb & Jain (2013), the entire objective space is expected to be covered by a hyperplane which is constructed based on the current population. Reference points are uniformly distributed over this hyperplane for diversity maintenance. After normalizing the objective functions values of population members, the reference points are placed on a normalized hyperplane. The number of solutions associated with a reference point is denoted as its niche count. Niche counts of all reference points are aimed to be balanced. The conventional Pareto dominance rule is followed by a niche-preservation strategy when measuring the fitness of individuals. In this algorithm, although the hyperplane containing all reference points evolves as its base population evolves, the locations of the reference points on the hyperplane are fixed.

In MOEA/D proposed by Zhang & Li (2007), a multi-objective problem is decomposed into a number of single-objective problems identified using the same scalarizing function with different weight vectors. While this algorithm is categorized as a decomposition based algorithm, there is a strong analogy between its weight vectors and the reference points. The solution

corresponding to each weight vector is generated by recombining a pair of parent solutions randomly selected from the neighborhood. The new solution is compared with the existing solution of the corresponding weight vector and the existing solutions of its neighbors. It replaces those with a worse scalarizing function value in terms of the corresponding weight vector.

In Two Archive Algorithm (Two_Arch2) proposed by Wang et al. (2014), the solution set of each generation is separated into two groups, namely a convergence archive and a divergence archive. The former holds only non-dominated solutions which once dominated some existing archive members, and the latter contains solutions which have largest distances to the convergence archive members. In this approach, the solutions of the convergence archive can be seen as adaptive, online-updated real reference points.

2.2. Reference Set Adaptation Methods

The use of adaptive reference points is advised to deal with irregular Pareto fronts and different scales of objectives. Irregular Pareto fronts include degenerate fronts, discontinuous fronts, inverted fronts and highly nonlinear fronts (e.g. strongly convex/concave fronts). The difficulties faced by the decomposition based and reference set based methods on the irregular Pareto fronts have been pointed out by several researchers (Ishibuchi et al., 2019; Hua et al., 2018). It is claimed that sampling the reference vectors from a uniform set of points inherently assumes that the Pareto front is bounded by the reference vectors, and it is nondegenerate, continuous, smooth, and without significant nonlinearity. On the other hand, methods with dynamic reference points can be challenged as their convergence speed may be degraded (Giagkiozis et al., 2013) and the new parameters they introduce to the algorithm may cause additional complexity. As a result, it is crucial to develop effective positioning and adaption strategies for the reference set.

In RPEA by Liu et al. (2017), reference points are updated only in some iterations and the reference set regeneration frequency is determined by a predefined parameter value. Reference points are formed as superior hypothetical solutions based on the nondominated solutions with largest crowding distances. As a result, reference points provide up-to-date information of the Pareto approximation. In RVEA proposed by Cheng et al. (2016), two sets of reference vectors are used. One set preserves a uniform distribution over the objective space and the other set is adaptively adjusted based on the normalized values of the solutions. In generalized decomposition based evolutionary algorithm (g -DBEA) proposed by Asafuddoula et al. (2017), reference vectors

are periodically adapted based on the information collected over a learning period. There are two sets of reference vectors: an active set that consists of adapting reference vectors and an inactive set that contains discarded vectors which have a chance to return back to the active set over the course of evolution. Adaptive reference point mechanisms are employed in Tian et al. (2017) to an indicator-based evolutionary algorithm and in Zhou et al. (2018) to an entropy based evolutionary algorithm.

Two other noteworthy algorithms with adaptive reference set strategies are proposed as adaptations of NSGA-III. In *A*-NSGA-III (Jain & Deb, 2014), reference points are adjusted according to the association of solutions to reference points in each generation. The adaption method has two stages: addition of new reference points near to crowded reference points with high niche counts and occasional deletion of reference points with zero niche counts. In the addition stage, a simplex is built around a crowded reference point and at most m (objective count) new reference points that are believed to share the niche count with the existing reference point are added. The addition of new reference points continues until no reference point with niche count greater than one is left. In a similar adaptive algorithm, named as A^2 -NSGA-III, proposed by (Jain & Deb, 2013), a more efficient approach to create new reference points is claimed to be followed. In this approach, the simplex structure results in the addition of fewer new reference points and creation of new reference points is allowed only when the niche counts of crowded reference points remain constant for a predefined number of generations. In this algorithm, there is also a cap over the maximum number of reference points, and as a result, deletion process is triggered more often.

2.3. Co-evolutionary Algorithms

Co-evolutionary algorithms, first developed in the 90s, provide a way to decompose and solve complex problems consisting of multiple interrelated subproblems, while respecting the interdependency between these subproblems. As an extension of classical evolutionary algorithms, co-evolutionary algorithms imitate the process of symbiotic evolution in nature, where different species evolve in a way to adapt to each other. The problem is decomposed into subproblems and multiple populations are used to optimize different subproblems simultaneously. During evolution, populations of different species can interact with each other in different ways. The fitness value of a solution is calculated based on its interaction with other species.

Antonio & Coello (2017) provide a survey for the application of co-evolutionary techniques in multi-objective evolutionary algorithms based on the different ways of interaction among the species. In this regard, coevolutionary approaches can be classified as cooperative or competitive. In the cooperative structure, good collaborators are encouraged and rewarded. Under the competitive scheme, the interaction is negative, as in the case where one population tries to develop increasingly difficult inputs for the other. It is also possible to have a hybrid structure that combines cooperative and competitive motives to promote diversity and avoid early convergence as the species work together to obtain good solutions to the problem. Coello & Sierra (2003), and Goh et al. (2010) are among the rare studies that use such an approach in the multi-objective optimization area.

In applications involving multi-objective problems, the cooperative structure is more common, and the decomposition is based on either the decision space or the objective space. For instance, Antonio & Coello (2016) apply co-evolutionary techniques to an indicator-based algorithm where each subpopulation represents a particular component of the problem in the decision space, so members from all subpopulations are required to assemble a complete solution. On the other hand, Zhan et al. (2013) adopt co-evolutionary techniques to a particle swarm optimization algorithm in which each subpopulation is responsible for one objective.

There are also a few studies that involve many-objective cases. Zhang et al. (2017) implement a cooperative co-evolutionary structure in a decomposition based many-objective algorithm. Each of the multiple cooperative subpopulations corresponds to a decomposition weight vector, while evolving in parallel, mating pools collect solutions from these subpopulations for recombination. Liu et al. (2018) introduce a co-evolutionary particle swarm optimization methodology to many-objective problems where multiple swarms are distributed to different areas of the Pareto front and concentrate on these limited areas. Preference-inspired co-evolutionary algorithm using weight vectors (PICEA-w) by Wang et al. (2015) proposes adaptive modification of decomposition weights (equivalent to reference points in our case) by co-evolving them with candidate solutions. At each iteration, the solution population duplicates its size by means of genetic recombination, while decomposition weights population is duplicated by random generation of weights. Then, the neighborhood relations between solutions and decomposition weights create a ranking matrix which is used to sort and truncate both populations. In order for a candidate decomposition weight to be se-

lected in the truncation procedure, it must be one of the weights that ranks a survived solution as the best, or in the case of ties, it should be the furthest from the corresponding solution.

2.4. Many-Objective Knapsack Problem

The 0-1 knapsack problem is one of the most well-known combinatorial optimization problems as it has real-life applications in many different fields. Briefly, in the knapsack problem, given a set of items each having a weight and a profit value, the objective is to decide whether each item will be included in a collection so that total profit is as large as possible while total weight is less than the given capacity.

Among the relatively few studies devoted on many-objective combinatorial optimization problems, most choose the many-objective version of knapsack problem as their benchmark. Two algorithms, δ -MOSS and k -EMOSS, with the goal of objective reduction are suggested in Brockhoff & Zitzler (2006). Further examples are a correlation-based weighted sum approach by Murata & Taki (2009), a hyperplane-based evolutionary algorithm HypE by Bader & Zitzler (2011), and an artificial fish swarm optimization algorithm by Azad et al. (2014). Sato et al. (2007), Tanigaki et al. (2014), and Ishibuchi et al. (2014b) apply modified versions of NSGA-II to MaOKP; while a search for correct parametrization for MOEA/D is carried out in Sato (2015). In another line of research, the performance of existing multi-objective evolutionary algorithms is evaluated as the number of objectives in MaOKP increases (Ishibuchi et al., 2013; Sato et al., 2013; Ishibuchi et al., 2014a).

We use MaOKP in the development and performance evaluation of our proposed algorithm, and compare it with some existing MaOEAs. In the formulation provided below, profit coefficients p_{ij} are defined for each objective $i = \{1, \dots, m\}$ and for each item $j = \{1, \dots, n\}$. The binary decision variable x_j denotes whether item j is included in the knapsack collection or not, and each of the m objective functions are expressed as in Equation set 1. In multi-constrained MaOKP, the knapsack constraints are given as in Equation set 2, where w_{jk} denotes the weight parameter for item j and knapsack k , and C_k denotes the capacity parameter for knapsack k .

$$\max_x f_i(x) = \sum_j p_{ij} x_j \quad \forall i \quad (1)$$

s.t.

$$\sum_j w_{jk} x_j \leq C_k \quad \forall k \quad (2)$$

$$x_j \in \{0, 1\} \quad \forall j \quad (3)$$

Binary encoding is used as the chromosome structure for an evolutionary algorithm for the MaOKP, hence the genotype of a solution is represented by a binary string in the n -dimensional decision space and the phenotype of the solution corresponds to its location in the m -dimensional objective space.

3. Proposed Algorithm

The proposed MaOEA uses a virtual reference point set and adopts an elitist strategy, where at each generation, existing and offspring population are combined and sorted first by the nondominated sorting and then by the reference set based sorting, similar to that of NSGA-III (Deb & Jain, 2013). The main novel features of the proposed algorithm are highlighted below:

- The algorithm starts by solving each objective separately and uses m individual optima (or near optimal) solutions to construct a hyperplane which remains fixed over the course of the algorithm.
- Reference points are mapped uniformly onto the fixed hyperplane. Their locations on the fixed hyperplane can be changed when the algorithm has a reference point update feature (FHMR), or when the reference set is allowed to co-evolve together with the solution population (FHCo).
- The recombination operator and selection mechanisms are designed in a way to complement and support the reference set based sorting. Reference point guided path relinking is proposed as the recombination scheme for this purpose. Additionally, repair and local improvement procedures are also guided by the reference points.
- An archive structure called external population stores all nondominated solutions and reports them at the end of the algorithm.

These features are presented within the main framework in Algorithm 1 and will be explained in detail in the following subsections.

Algorithm 1: Proposed MaOEA

Input: *parameterSetting, TerminationCriteria*

1. Find **optimal (near optimal) solutions** of single objective problems
 2. Set **fixed_hyperplane** and *referencePoints* : \mathcal{R}
 3. Generate initial *population*: \mathcal{P}
 4. Set **external_population**: \mathcal{P}_{ext}
 5. **While not** *TerminationCriteria*:
 6. *parentPairs* : $\mathcal{P}_{pairs} := \text{Select_Pairs}(\mathcal{P})$
 7. *offspring* : $\mathcal{P}_{off} := \text{Reference_Point_Guided_Path_Relinking}(\mathcal{P}_{pairs})$
 8. *Mutation*(\mathcal{P}_{off}), *Repair*(\mathcal{P}_{off}) and *Local_Improvement*(\mathcal{P}_{off})
 9. **Reference_Set_Based_Sorting**($\mathcal{P} \cup \mathcal{P}_{off}, \mathcal{R}$)
 10. Trim $\mathcal{P} \cup \mathcal{P}_{off}$ and update \mathcal{P}
 11. Update \mathcal{P}_{ext} using eliminated, nondominated solutions in \mathcal{P}_{off}
 12. **Reference_Set_Update_Procedure** (for *FHMR*)
 13. **Reference_Set_Co-evolution** (for *FHCo*)
 14. **End while**
- Output:**
- \mathcal{P}_{ext}
-

3.1. Fixed Hyperplane

In the NSGA-III algorithm proposed by Deb & Jain (2013), the Pareto front is attempted to be embraced by a hyperplane which is constructed repeatedly at each iteration of the algorithm based on the evolving population. The reference points are distributed uniformly on this hyperplane using the systematic simplex lattice design by Das & Dennis (1998) for diversity maintenance. When d divisions are declared for each objective, the total number of reference points, H in an m -objective problem is given by Equation 4. An example reference point structure for four divisions in a three-objective problem is presented in Figure 1.

$$H = \binom{m + d - 1}{d} \quad (4)$$

In the original implementation by Deb & Jain (2013), the hyperplane containing all reference points evolves as its base population evolves. The positions of the reference points remain unchanged relative to the hyperplane, but a shift occurring in the hyperplane location results also in the transformation of the entire set of reference points. A number of drawbacks of this implementation are observed. First, since the members of the population in the early stages of the evolutionary algorithm might all be dominated, the

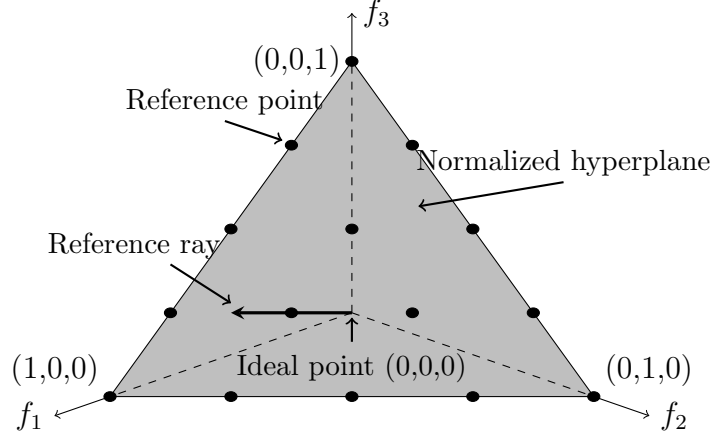


Figure 1: Reference points on the normalized hyperplane.

hyperplane that is built from these solutions might be misleading. As a result, the reference points might be poorly constructed and the algorithm may proceed in the wrong directions. Additionally, when the hyperplane changes together with the evolving population, the locations of the reference points might shift severely and the sorting procedure used until then becomes useless. In other words, it is highly probable that instabilities of the reference points and the association of population members disrupt the optimization efforts. Furthermore, degeneracy is observed occasionally during hyperplane construction. This degeneracy may occur due to failing to obtain m distinct extreme solutions to construct the hyperplane or obtaining a hyperplane with negative intercept values for some objective axes. Finally, the construction of the hyperplane in each iteration brings a heavy computational burden and the algorithm becomes impractical for many-objective problems.

A fundamental novelty of our algorithm includes placing the reference points on a fixed hyperplane contrary to the original approach. This fixed hyperplane is constructed at the beginning of the algorithm using the lexicographic optimal solutions of the multi-objective problem and never moves throughout evolution. These solutions are obtained by solving m single objective problems obtained by considering each objective separately. This turns out to be a very effective strategy for problems where it is possible to find a good single-objective solution within a reasonable time (as in the case of 0-1 knapsack problem) for several reasons:

- time performance improves,
- degeneracy issues encountered in the construction of hyperplanes are eliminated,
- and most importantly, the performance is improved since the algorithm does not have to search for the hyperplane over the course of generations, while a good one is readily available from the beginning.

As a side remark, the above benefits of using a fixed hyperplane are observed even when the hyperplane is constructed based on near-optimal single-objective solutions. Since it takes negligible time to obtain optimum solutions even for large Knapsack problems, this issue is experimented and verified on problems where it can be difficult to obtain optimal solutions such as TSP and QAP. But this issue is not the focus of this article.

3.2. External Population

Another modification is keeping an external population which is basically an archive structure storing all the nondominated points created throughout evolution. Archive mechanism is reported to be very useful for the evolutionary algorithms whose sorting principles are not based on dominance relationship, e.g., MOEA/D (Zhang & Li, 2007). Although nondominated sorting based algorithms do not have to use external populations in their original framework, we believe that archiving is necessary for many-objective problems, since the huge number of nondominated points that get eliminated from the internal populations throughout iterations is a waste that may cause loss of important information. The computational effort required to update the external population can be reduced by limiting its size or applying some clustering techniques.

3.3. Genetic Operators

All evolutionary mechanisms in our proposed MaOEA are designed in a way to complement the reference set based sorting, to enhance its power and success. For this purpose, selection and recombination processes are integrated with each other and with the reference set based sorting.

Selection. Instead of an independent selection procedure that first creates a parent pool and then randomly selects parents from the pool, we devise a selection mechanism that facilitates the crossover scheme by choosing parents deliberately as couples that will enter recombination together. The target

is to associate an equal number of solutions to each reference point, e.g., when the population size is equal to the number of reference points, all reference points need to have niche counts equal to one. Thus, the selection mechanism tries to generate parent couples whose offspring might populate empty reference points.

Algorithm 2: *Select_Pairs*

Input: *population* : \mathcal{P} , *referencePoints* : $\mathcal{R} = \mathcal{R}_A \cup \mathcal{R}_E$

1. Set *parentPairs* : $\mathcal{P}_{pairs} = \emptyset$
 2. Set *guidingReferencePoints* : $\mathcal{R}_G = \emptyset$
 3. **For each** reference point r **in** \mathcal{R}_E :
 4. Set *neighbors* : \mathcal{R}_N from \mathcal{R}_A in the neighborhood of r
 5. **If** $\mathcal{R}_N.Count \geq 2$:
 6. Choose two references points: r_1 and r_2 from \mathcal{R}_N
 7. Set the closest associated solution of r_1 as p_1
 8. Set the closest associated solution of r_2 as p_2
 9. Add (p_1, p_2) to \mathcal{P}_{pairs}
 10. Add r to \mathcal{R}_G
 11. **Else if** $\mathcal{R}_N.Count = 1$:
 12. Choose the reference point: r_1 from \mathcal{R}_N
 13. Set the closest associated solution of r_1 as p_1
 14. Choose two candidate solutions: s_1, s_2 randomly from \mathcal{P}
 15. Set p_2 by *BinaryTournament*(s_1, s_2) using *Dissimilarity_with_p1*
 16. Add (p_1, p_2) to \mathcal{P}_{pairs}
 17. Add r to \mathcal{R}_G
 18. **Else if** $\mathcal{R}_N.Count = 0$:
 19. **Continue**
 20. **End if**
 21. **End for each**
 22. **While** $\mathcal{P}_{pairs}.Count < \mathcal{P}.Count$:
 23. Choose two candidate solutions: s_1, s_2 randomly from \mathcal{P}
 24. Set p_1 by *BinaryTournament*(s_1, s_2) using *Reference_Set-Based_Sorting*
 25. Choose two candidate solutions: s_1, s_2 randomly from \mathcal{P}
 26. Set p_2 by *BinaryTournament*(s_1, s_2) using *Dissimilarity_with_p1*
 27. Add (p_1, p_2) to \mathcal{P}_{pairs}
 28. **End while**
- Output:** $\mathcal{P}_{pairs}, \mathcal{R}_G$
-

On that account, two solutions associated with two reference points which are neighbors to an empty reference point (target reference point) are se-

lected as a couple. It is anticipated that a path built between such a pair may generate an offspring that will fit into the purpose. To avoid short and inadequate paths, the parent pair is not selected from the same reference point or from reference points that are themselves neighbors. In case the second parent cannot be found using this rule, it is selected by means of a binary tournament scheme. The decision in the binary tournament is based on the dissimilarity to the already selected parent. The candidate having a more dissimilar genotype, i.e. larger Hamming distance, is preferred in order to ensure a diverse gene pool in the recombination scheme. If none of the parents can be selected as above, then the binary tournament replaces the selection. The pseudo-code of the parent pair selection is shown in Algorithm 2 where the reference point set $\mathcal{R} : \mathcal{R} = \mathcal{R}_A \cup \mathcal{R}_E$ consists of two sets, associated reference points and empty reference points.

Recombination. It is reported that conventional recombination schemes may perform unsatisfactorily in many-objective problems since the offspring whose parents are close to the Pareto optimal front do not need to be close to the front themselves (Jaimes & Coello, 2015).

The proposed algorithm employs reference point guided path relinking as its recombination method. Using a randomized mixed path relinking strategy (Glover et al., 2004), two paths are initiated simultaneously from both parents. The moves selected at each step are guided by the reference points. The nodes of the path generated at each step correspond to the objective space locations of candidate solutions for offspring.

Specifically, in a knapsack problem, the move corresponds to switching a binary gene in one parent from 0 to 1 and vice versa in the other parent. Starting from a random position of the chromosomes, the genes of the two parents at that position are examined. If the corresponding genes of the two parents are different, a flip operation is performed for both parents. The two resulting solutions are inspected and the move that yields the “better” solution is performed by the corresponding parent, thus forming a new node on the path. In the next move iteration, this new node replaces the parent from which it is obtained. Then, the move operation is applied for this new solution along with the solution at the end of the other path. Moves continue until all the genes of the two endpoints are the same, meaning that the full path has been formed. In the end, the last node generated on the path is declared as offspring solution. It should be noted that this procedure yields a single offspring.

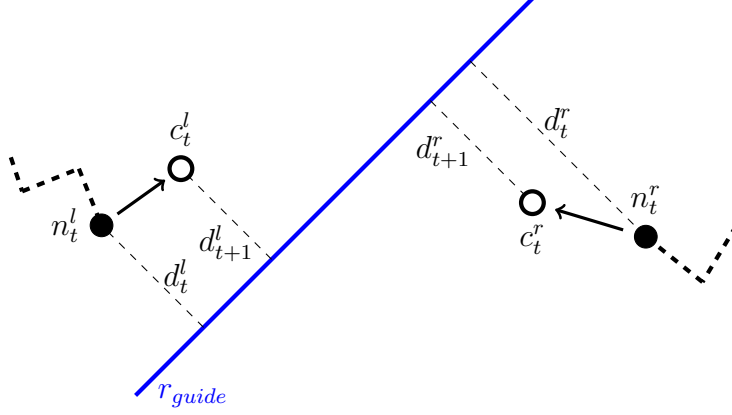


Figure 2: Reference point guided path relinking.

In each move iteration t , the end nodes of the two paths, n_t^l and n_t^r , create two candidate solutions, c_t^l and c_t^r , respectively. If only one of these candidates is feasible, it is chosen as the next node of the path. It must be noted that at most one of the candidates (the one obtained by switching the binary gene of 0 to 1) can be infeasible given the fact that the move is made from two feasible knapsack solutions. When both candidates are feasible, the dominance is checked. This is very likely to be inconclusive, so the candidate solutions are compared based on their move-values. Denoting the Euclidean distance between the parent node and the guiding reference ray (the ray starting from origin and passing through the target reference point) as d_t , and similarly, the distance between the candidate node and the guiding ray as $d_{(i+1)}$, the move-value δ is calculated as their difference.

The candidate solution with a minimum move-value, i.e. the move operation that shortens the distance to the guiding reference ray more than the other is chosen as the next node of the path. An illustrative example of how the move-values are calculated is shown in Figure 2.

After each move, if the most recent node happens to get associated with an empty reference point (does not have to be the guiding reference point), the corresponding solution is declared as the offspring and the crossover is terminated before forming the full path. The pseudo-code of the reference point guided path relinking recombination scheme is shown in Algorithm 3.

Mutation, Repair and Local improvement. Following recombination, each offspring enters a bit-flip mutation operation and randomly selected binary

genes are reversed with a mutation rate parameter. After the mutation, each offspring is associated with a reference point and if the offspring is infeasible, it enters a repair procedure. Among the feasible offspring solutions, randomly selected ones enter local improvement procedure based on a local improvement rate parameter.

Algorithm 3: Reference_Point_Guided_Path_Relinking

Input: $parentPairs : \mathcal{P}_{pairs}$, $referencePoints : \mathcal{R} = \mathcal{R}_A \cup \mathcal{R}_E$,
 $guidingReferencePoints : \mathcal{R}_G$

1. Set $offspring : \mathcal{P}_{off} = \emptyset$
 2. **For each** $parentPair : (p_1, p_2)$ **in** \mathcal{P}_{pairs} :
 3. Set $r_g = \mathcal{R}_G[0]$
 4. **If** $r_g \neq \emptyset$
 5. Set n_{total} as total difference in genotypes of p_1 and p_2
 6. Set $n_1^l = p_1$ and $n_1^r = p_2$
 7. Set $path = \emptyset$
 8. **For** t **in** $(1, n_{total})$:
 9. Apply $Move_Operation(n_t^l, n_t^r)$ to obtain (c_t^l, c_t^r)
 10. $\delta^l = Euclidean_Distance(r_g, c_t^l) - Euclidean_Distance(r_g, n_t^l)$
 11. $\delta^r = Euclidean_Distance(r_g, c_t^r) - Euclidean_Distance(r_g, n_t^r)$
 12. **If** $\delta^l < \delta^r$:
 13. Add c_t^l to $path$
 14. Set $n_{t+1}^l = c_t^l$ and keep n_t^r as n_{t+1}^r
 15. **Else:**
 16. Add c_t^r to $path$
 17. Set $n_{t+1}^r = c_t^r$ and keep n_t^l as n_{t+1}^l
 18. **End if**
 19. Find $r_{associate}$ as association of the most recent node in $path$
 20. **If** $r_{associate} \in \mathcal{R}_E$: **Break**
 21. **End for**
 22. Select s_{off} as the most recent node in $path$
 23. $\mathcal{R}_G = \mathcal{R}_G \setminus r_g$
 24. **Else:**
 25. Select s_{off} from a generic crossover between (p_1, p_2)
 26. **End if**
 27. Add s_{off} to \mathcal{P}_{off}
 28. **End for each**
- Output:** \mathcal{P}_{off}
-

Both repair and local improvement procedures are guided by the reference point to which the corresponding offspring is associated. When a reference

point $\vec{r} = [r_1, \dots, r_m] : \sum_{i=1}^m r_i = 1, \forall r_i \geq 0$ is used to guide procedures, knapsack items are sorted by calculating the $q_j(\vec{r})$ values for each item as in Equation 5. In this way, the coordinate of the associated reference point is used to assess the importance of the knapsack items.

$$q_j(\vec{r}) = \frac{\sum_i r_i p_{ij}}{\max_k \{w_{jk}\}}, \quad \forall j \quad (5)$$

High $q_j(\vec{r})$ value in the local improvement procedure indicates that the inclusion of item j into the knapsack provides great benefit in the direction of \vec{r} whereas low \vec{r} value in the repair procedure means that item j can be removed from the knapsack with no major loss that would take away from \vec{r} . The repair and local improvement procedures obtained in this manner are called “*weighted greedy*”.

This basic form of our proposed algorithm uses a fixed positioned reference point set on the fixed hyperplane with nondominated sorting, hence will be named FHFR. Two other versions (FHMR and FHCo) which allow the reference set to move on the fixed hyperplane (as indicated in lines 12 and 13 of the pseudo-code of Algorithm 1) will be explained next.

3.4. Reference Set Update Procedure

No matter how well-spread the reference points are, they may still fail to represent the Pareto front effectively. Due to the irregularities in Pareto front shapes, some reference points may have no solution that can be associated with them. As it is usually not possible to know this landscape a priori, it might be useful to update the positions of the reference points during the search. In other words, although the systematic mesh approach proposed by Das & Dennis (1998) is used at the beginning, this can be changed during the course of the evolutionary algorithm. The update mechanism must be generic so that it can be applied to various different types of problems. In this study, we tried two approaches to cope up with the landscape irregularity issue. The first of these, which involves mobile reference points is explained in this section.

While using the algorithm in its basic form (FHFR), it is observed that a large number of reference points may remain unassociated despite complementary genetic operators’ attempts to fill. The algorithm should be equipped to detect such reference points and to avoid further unnecessary efforts. In our implementation, when a reference point stays unassociated for

a certain number of iterations, it is concluded that its surrounding region in the objective space is void and its position needs to be updated.

Reference set update procedure works on the reference set as the second stage of the algorithm after the reference set based sorting stage works on the solution set to establish the population of the next iteration. This procedure, which relocates some reference points based on the information gathered during the first stage, is designed as an independent module so that it can be used in a plug-in manner. It must be noted that the reference set update procedure does not necessarily need to be applied at every iteration. When this stage is deactivated, the algorithm with a fixed reference set is obtained.

In fact, the first time the reference set update procedure is activated is delayed for some “learning period” (Asafuddoula et al., 2017) thus allowing the population that is mostly composed of dominated solutions in the initial iterations some time to get mature. The population is said to be sufficiently mature when the number of solutions in its first front has grown large enough such that nondominated points start to get truncated. In other words, reference set update procedure is applied for the first time when the front count in the current population drops to one. In this manner, the solution associations to the initial uniform map of reference points get settled to convey meaningful information. In many-objective problems, the front count rapidly decreases to one since the number of nondominated points is extremely high.

By the same line of reasoning, after each reference set update epoch, the reference set is allowed to settle for some time until the evolutionary process becomes more stable in terms of how the solutions are associated with the reference points. Our preliminary experiments supported this concept showing that when the update procedure is applied at each iteration, a deterioration occurs in the performance due to its overuse. As a result, the update procedure is applied only at the end of the “cycles”. Based on the preliminary numerical studies, when there is no change in how the solution set is associated with the reference point set for 10 iterations, the cycle is said to be complete and the reference point update procedure is activated.

In our implementation, when no solution is found to be associated with a particular reference point throughout a cycle, this reference point is relocated near another densely populated reference point. In this manner, the moved reference point is hoped to relieve the crowded one and to create balanced niche counts for reference set based sorting algorithm. When a solution is randomly picked and its associated reference point is chosen, reference points with higher niche counts are given a higher chance, similar to a roulette

wheel selection mechanism. The new reference point is positioned between the selected reference point and one of its randomly chosen neighbors. The exact location is given by a randomly generated ratio that determines the convex combination of chosen neighbors. In this way, it is guaranteed that the new reference point is located on the fixed hyperplane and the number of reference points remains the same. This reference point update procedure yields the second version of the proposed algorithm, called Fixed Hyperplane and Mobile Reference points: FHMR. The pseudo-code of the reference set update procedure in FHMR is shown in Algorithm 4.

Algorithm 4: *Reference_Set_Update_Procedure* (for *FHMR*)

Input: *population* : \mathcal{P} , *referencePoints* : $\mathcal{R} = \mathcal{R}_A \cup \mathcal{R}_E$, *Association*(\mathcal{P}, \mathcal{R})

1. Detect empty reference points throughout the cycle \mathcal{R}_{DEL} from \mathcal{R}_E
2. **For each** reference points r in \mathcal{R}_{DEL} :
3. Select a solution s randomly from \mathcal{P}
4. Set $r_1 = \text{Association}(s, \mathcal{R})$
5. Select a reference point r_2 randomly from the neighborhood of r_1
6. Relocate $r = \alpha r_1 + (1 - \alpha)r_2$ using a random $\alpha : 0 \leq \alpha \leq 1$
7. Update the neighborhoods of r , r_1 and r_2
8. Update *Association*(\mathcal{P}, \mathcal{R})
9. **End for each**

Output: \mathcal{R} , *Association*(\mathcal{P}, \mathcal{R})

3.5. Co-evolution of the Reference Set

While the reference point update procedure described above occasionally relocates some of the reference points, the co-evolutionary approach defines a coherent and natural mechanism that allows more flexible adaptation of the entire reference set. In the co-evolutionary structure of the proposed MaOEA, the solution set and the reference point set are defined as two concurrently evolving populations. Solutions are evaluated based on how strongly they are associated with the reference point set, and reference points evolve based on the number of solutions associated with them. The evolution of a reference point corresponds to an update of its location on the fixed hyperplane. The co-evolutionary structure possesses both cooperative and competitive aspects, since the balance between cooperation and competition is necessary to avoid stagnation in the algorithm and to ensure that the co-evolutionary framework is self-adaptive.

In general, the two species evolve in a cooperative manner, and the members of both sets are rewarded if they manage to associate decisively with

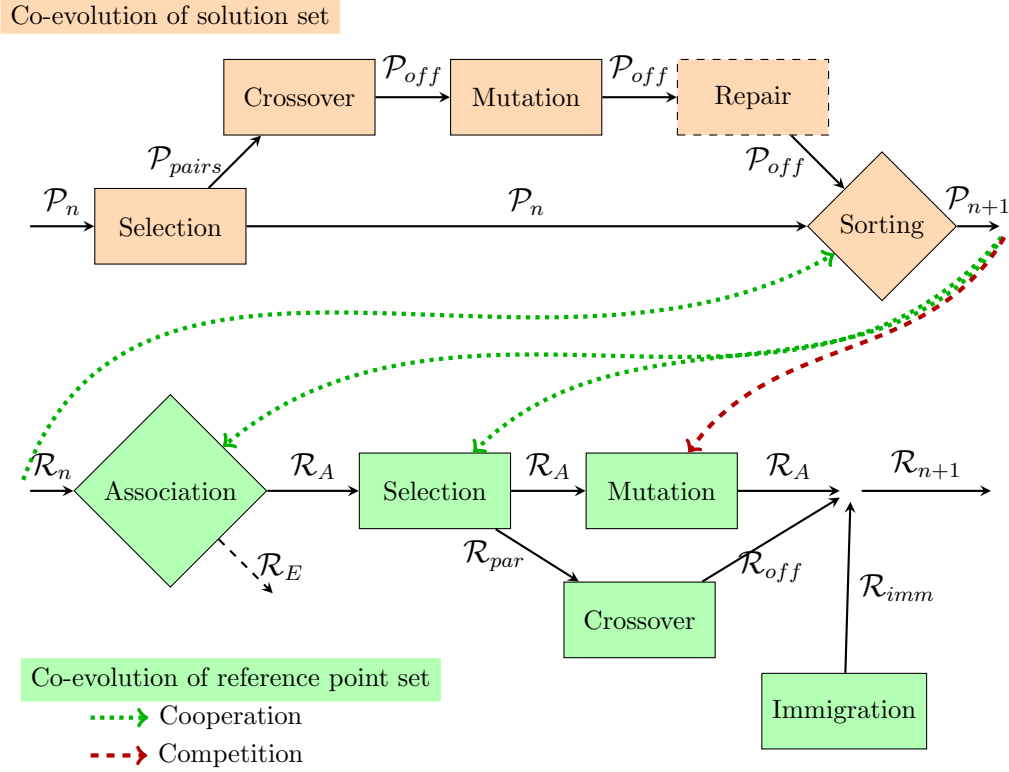


Figure 3: Flow chart of the co-evolution of solution and reference point sets.

the counter set. On the other hand, they are also involved in a competition to diversify the search area. The diagram in Figure 3 presents the sequential progress of the evolutionary processes and their co-evolutionary relationship. All interactions, except for the mutation of the reference set, include cooperative features. All these interactions are essential to improve the performance of both species. It is worth mentioning that the evolution of the two sets have different paces, i.e. while the evolution of solution set occurs at every iteration, the evolution of reference set is activated only at the end of the cycles. The reference set evolution resembles to a classic evolutionary approach with typical genetic operators such as selection, crossover, immigration and mutation. Its pseudo code is presented in Algorithm 5.

A predefined proportion of the associated reference points mutates and changes their locations on the hyperplane in order to force the evolutionary

process to search different regions of the objective space. Empty reference points on the other hand, are replaced by newly generated reference points. This can be achieved by introducing new offspring through crossover or by immigrating new reference points. Empty reference points will be completely replaced by these new reference points.

Algorithm 5: Reference_Set_Co-evolution (for *FHCo*)

Input: *referencePoints* : $\mathcal{R} = \mathcal{R}_A \cup \mathcal{R}_E$, *Association*(\mathcal{P}, \mathcal{R})

1. $\mathcal{R}_{par} := \text{Selection}(\mathcal{R})$

2. $\mathcal{R}_{off} := \text{Crossover}(\mathcal{R}_{par})$

3. $\mathcal{R}_{img} := \text{Immigration}$

4. $\text{Mutation}(\mathcal{R}_A)$

5. $\mathcal{R} \leftarrow \mathcal{R}_A \cup \mathcal{R}_{off} \cup \mathcal{R}_{img}$

6. Update *Association*(\mathcal{P}, \mathcal{R})

Output: \mathcal{R} , *Association*(\mathcal{P}, \mathcal{R})

Selection. Binary tournament, where the candidate with higher niche count is selected as the first parent, is used. In the case of a tie, the distances to the closest solutions of the candidates are measured and the more distant one is selected. The choice of the second parent is made randomly, to give every search direction a chance.

Crossover. The crossover scheme involves a convex combination of the locations of the two parent reference points in the objective space. Denoting the locations of the parent points as $L(\vec{\mathbf{r}}_x)$ and $L(\vec{\mathbf{r}}_y)$, the location of the offspring reference point $L(\vec{\mathbf{r}}_o)$ is given by Equation 6.

$$L(\vec{\mathbf{r}}_o) = \alpha L(\vec{\mathbf{r}}_x) + (1 - \alpha) L(\vec{\mathbf{r}}_y) \quad (6)$$

where, $\alpha : 0 \leq \alpha \leq 1$ is a randomly generated non-negative coefficient. It is observed that the offspring usually does not bear any common features with either of its parents if the ratio α is generated uniformly. To this end, we employ a quadratic piecewise probability density function which puts some bias towards either one of the parents, with more bias towards the parent selected with the binary tournament. The offspring reference points produced by this scheme are guaranteed to be located on the normalized hyperplane.

Immigration. In order to explore different regions of the objective space, new reference points with random coordinates are also generated. These immigrants are used to complement the crossover scheme in replacing the empty

reference points. The relative ratio for the usage of these two methods depends on current status of the evolution. The ratio of immigrant count to crossover count is equal to the ratio of number of empty reference points to the number of associated reference points. In this way, immigration process shows a self-adaptive behavior such that its usage diminishes as the share of empty reference points decreases throughout the evolutionary process. In order to obtain better immigrants, twice as many are created and a binary tournament is applied among them. Immigrants with a prospective associated solution are chosen.

Mutation. Since the number of nondominated solutions increases exponentially in many-objective problems, avoiding premature convergence becomes even more challenging. In the application of a cooperative co-evolutionary structure, it is natural to prefer solutions/reference points that are closely associated with the counter set. However, there is a clear threat that the reference points and their best associated solutions will “*stick*” to each other never to be eliminated in the sorting procedure. This will result in the stagnation of the co-evolutionary framework and thus a premature convergence.

To avoid this behavior, we use mutation in a way that brings a competitive nature to the co-evolutionary structure. In order to direct the search to unexplored regions of the objective space, mutation is applied to associated reference points such that they move away from their closest associated solutions. Consequently, the other species is forced to cope up with this disruption. The opposite direction on the line segment between the reference point and its closest associated solution is used as the move direction. The length of this move is critical since large values cause the reference point to drift away and small values are ineffective. After investigating different alternatives in preliminary experiments, it is decided to use a random coefficient between 0 and 1 to determine the move length. In this manner, the move length can be at most equal to the distance between the corresponding reference point and its closest associated solution. The mutated reference points are also ensured to stay on the normalized hyperplane. The mutation is applied only to a percentage, not to all associated reference points. This percentage is equal to the ratio of associated reference points across the entire set. In this manner, as the percentage of associated reference points increases throughout the evolutionary process, the use of mutation also increases.

4. Numerical Studies

In our numerical experimentation we assess the contribution of different crucial aspects of the proposed algorithm, and compare its performance with that of the following seven state-of-the-art MaOEAs: NSGA-III (Deb & Jain, 2013), A-NSGA-III (Jain & Deb, 2014), A^2 -NSGA-III (Jain & Deb, 2013), Two_Arch2 (Wang et al., 2014), RPEA (Liu et al., 2017), MOEA/D (Zhang & Li, 2007), and PICEA-w (Wang et al., 2015).

All algorithms are coded in C# programming language in Microsoft Visual Studio 2017 and all experiments are carried out on a PC with 3.60 GHz Intel® Core™ i7-3820 CPU and 16 GB RAM running on a 64-bit Windows 10 operating system. Before presenting the results, we first describe the experimental settings in the next subsection.

4.1. Experimental Settings

500-item knapsack problems with six to 30 objectives are used for numerical experiments. The test problems are generated using the procedure in Zitzler & Thiele (1999), where the profit p_{ij} and weight w_{jk} coefficients are specified randomly using discrete uniform distribution $[10, 100]$. The problems have a single constraint and the capacity of the knapsack C is 50% of the sum of all the weights.

4.1.1. Parameters

In the numerical studies for the proposed algorithm, the sizes of the solution set and the reference point set are equal to each other. In this way, the algorithm is expected to associate exactly one solution with each reference point. It should be reminded that when the systematic simplex lattice design by Das & Dennis (1998) is used, the number of reference points depends on the objective and division counts as in Equation 4. The number of reference points required for different division counts corresponding to each objective count is given in Table 1 where the selected set sizes are shown in bold.

To make all algorithms comparable, the population size (or the total size of the two archives in the case of Two Arc.2) and the number of reference points (or the number of decomposition vectors in the case of MOEA/D and PICEA-w) are kept the same for all algorithms. Note that the number of reference points is allowed to increase in A-NSGA-III and A^2 -NSGA-III. In MOEA/D, the same reference set neighborhood structure in the proposed MaOEA is used for the neighborhood structure of the decomposition vectors.

Table 1: Reference point set and population size setting.

Division count (d)	Objective Count (m)					
	6	8	10	15	20	30
2	21	36	55	120	210	465
3	56	120	220	680	1540	4960
4	126	330	715	3060	8855	40920

Based on the parameter fine-tuning experiments across the algorithms, the mutation rate is set to 0.004 and the local improvement rate is increased at 0.001 per iteration where applicable. As suggested in Zhang & Li (2007), all algorithms are terminated when $500 \times \mathcal{P}$ (population size) repair calls are made. The rest of the parameters specific to each algorithm are determined based on the recommendations in the respective references.

4.1.2. Performance Measure

Various performance indices have been suggested in the literature for quality assessment and comparison of different Pareto front approximations (see Knowles & Corne (2002); Okabe et al. (2003); Tan et al. (2006); Zitzler et al. (2003) for comprehensive reviews). For some of these, the true Pareto front must be known in advance. Furthermore, some performance metrics can only be used in two or three-dimensional objective spaces. Among the various metrics, the hypervolume indicator (Zitzler & Thiele, 1999) is one of the most common, as it uniquely characterizes approximation sets in a strictly monotonic manner with regard to Pareto dominance (Bader & Zitzler, 2011). This means that when comparing two approximation sets, the hypervolume value of the dominant set is always better than other. Higher values indicate that the corresponding Pareto approximation achieves to dominate more volume in the objective space. Hence, the hypervolume ratio (HR) is employed as the performance metric in our study. The HR value is calculated for each approximation obtained by different algorithms using Monte Carlo simulation by uniformly generating 10^5 random coordinate points within the hypercube, where the origin and the ideal point are the corner points.

Table 2: Percentage of associated reference points in MaOKP.

Algorithm	Objective Count (m)					
	6	8	10	15	20	30
FHFR	27.3%	38.2%	39.4%	45.7%	57.6%	42.1%
FHMR	54.5%	65.8%	54.2%	73.7%	78.4%	75.2%
FHCo	63.3%	79.8%	56.3%	79.3%	92.1%	90.3%

Table 3: Means and standard deviations of HR values on MaOKP.

Algorithm	Objective Count (m)					
	6	8	10	15	20	30
NSGA-III	5.91e-1(4.7e-2)	4.54e-1(5.9e-2)	3.95e-1(5.9e-2)	1.73e-1(5.0e-3)	3.20e-2(6.9e-6)	2.24e-3(7.3e-4)
NSGA-III (fixed)	7.22e-1(2.5e-2)	5.75e-1(3.6e-2)	4.51e-1(5.5e-2)	2.07e-1(1.5e-2)	4.29e-2(9.3e-4)	3.15e-3(4.7e-4)
A-NSGA-III	5.32e-1(3.7e-2)	4.18e-1(4.8e-2)	3.48e-1(5.2e-2)	1.67e-1(5.0e-3)	4.28e-2(5.4e-4)	3.43e-3(1.4e-4)
A-NSGA-III (fixed)	7.62e-1(2.3e-2)	5.90e-1(3.4e-2)	5.24e-1(4.9e-2)	2.29e-1(6.8e-3)	5.67e-2(6.4e-4)	4.44e-3(1.6e-4)
A ² -NSGA-III	6.11e-1(2.9e-2)	5.52e-1(3.8e-2)	4.45e-1(5.0e-2)	2.28e-1(5.2e-3)	6.26e-2(4.9e-4)	4.89e-3(1.0e-3)
A ² -NSGA-III (fixed)	7.71e-1(2.8e-2)	6.18e-1(2.2e-2)	5.43e-1(5.0e-2)	2.31e-1(7.3e-3)	6.55e-2(2.1e-4)	5.44e-3(2.1e-6)
Two_Arch2	7.75e-1(3.0e-2)	6.35e-1(3.0e-2)	5.21e-1(5.0e-2)	2.25e-1(1.4e-2)	6.64e-2(5.5e-4)	5.25e-3(7.2e-4)
RPEA	7.35e-1(2.5e-2)	4.95e-1(4.4e-2)	4.12e-1(4.5e-2)	2.14e-1(1.2e-2)	5.50e-2(3.5e-4)	4.31e-3(1.0e-4)
MOEA/D (ws)	6.17e-1(2.6e-2)	4.55e-1(5.0e-2)	3.89e-1(4.7e-2)	2.02e-1(1.2e-2)	4.08e-2(8.8e-5)	3.05e-3(2.5e-4)
MOEA/D (tc)	5.95e-1(4.3e-2)	4.15e-1(5.6e-2)	3.59e-1(5.2e-2)	1.67e-1(3.6e-3)	3.33e-2(6.2e-5)	2.18e-3(5.2e-4)
PICEA-w	7.67e-1(2.1e-2)	6.22e-1(2.2e-2)	5.68e-1(4.6e-2)	2.63e-1(6.4e-3)	7.22e-2(9.0e-4)	5.83e-3(5.1e-4)
FHFR	7.83e-1(2.6e-2)	6.62e-1(2.4e-2)	5.60e-1(4.9e-2)	3.01e-1(5.8e-2)	7.15e-2(1.0e-4)	5.51e-3(9.1e-6)
FHMR	8.11e-1(2.3e-2)	7.08e-1(2.9e-2)	5.90e-1(3.3e-2)	3.26e-1(4.7e-2)	8.44e-2(9.0e-4)	6.74e-3(6.7e-4)
FHCo	8.35e-1(2.5e-2)	7.35e-1(2.4e-2)	6.31e-1(2.8e-2)	3.52e-1(5.9e-2)	9.70e-2(2.7e-4)	7.81e-3(6.4e-4)

4.2. Results and Discussion

Three versions of the proposed algorithm are included in the numerical analysis to explore the effects of the reference point set adaptation strategies. The first version FHFR using a fixed reference point set is kept in the experimentation to demonstrate the benefits gained solely from using a fixed hyperplane. The second and third versions, namely FHMR and FHCo, both work on the fixed hyperplane and are designed to bring further improvements to performance. The percentages of the associated reference points in the reference set at the termination of the algorithms are depicted in Table 2 and the success of the reference point update procedures is observed.

We compare the performance of different versions of our proposed al-

Table 4: Wilcoxon rank-sum test and Friedman test results on MaOKP.

Algorithm	Wilcoxon Rank-sum Test, Objective Count (m)						Friedman Test
	6	8	10	15	20	30	
NSGA-III	0-4-9	0-4-9	0-6-7	2-0-11	0-0-13	0-1-12	12.5 (12)
NSGA-III (fixed)	5-1-7	5-3-5	3-4-6	3-3-7	3-1-9	2-2-9	9.333 (10)
A-NSGA-III	0-1-12	0-3-10	0-3-10	0-1-12	3-1-9	3-1-9	12.5 (12)
A-NSGA-III (fixed)	7-4-2	6-2-5	5-6-2	6-3-4	6-0-7	6-1-6	7 (7)
A^2 -NSGA-III	1-3-9	5-1-7	2-5-6	6-3-4	7-0-6	5-3-5	8.333 (8)
A^2 -NSGA-III (fixed)	7-4-2	6-4-3	7-5-1	6-3-4	8-0-5	8-1-4	5.333 (5)
Two_Arch2	7-4-2	8-3-2	7-2-4	4-5-4	9-0-4	7-4-2	5.667 (6)
RPEA	5-1-7	2-2-9	2-3-8	4-2-7	5-0-8	5-1-7	9.167 (9)
MOEA/D (ws)	1-3-9	0-4-9	0-6-7	3-1-9	2-0-11	2-1-10	11.333 (11)
MOEA/D (tc)	1-3-9	0-3-10	0-3-10	0-1-12	1-0-12	0-1-12	13.333 (14)
PICEA-w	7-4-2	8-2-3	8-4-1	10-1-2	11-0-2	9-2-2	4 (4)
FHFR	7-4-2	10-1-2	8-4-1	10-3-0	10-0-3	9-2-2	3.5 (3)
FHMR	12-1-0	12-0-1	9-4-0	11-2-0	12-0-1	12-0-1	2 (2)
FHCo	12-1-0	13-0-0	12-1-0	11-2-0	13-0-0	13-0-0	1(1)

gorithm with that of the seven state-of-the-art MaOEAs mentioned before. For MOEA/D, two versions with different decomposition methods, weighted sum (ws) and Tchebysheff (tc), are included in the experiments. It should be noted that lexicographic optimal solutions are fed into the initial populations of all algorithms for a fair comparison. However, when we provide single objective optimal solutions to an algorithm which works on hyperplanes, namely NSGA-III, A-NSGA-III or A^2 -NSGA-III, its hyperplane becomes fixed throughout the evolution, resulting into a new algorithm whose behavior changes drastically. Therefore, our numerical experimentation includes both the original and the fixed hyperplane versions of these algorithms. As a result, the numerical analysis covers a total of 14 algorithm versions.

An analysis is carried out to investigate the performance of these algorithms as the number of objectives increases. All algorithms are allowed to carry external populations, and the evaluation is made on the final external populations. Means and standard deviations (listed in parentheses) of HR results over ten replications are displayed in Table 3. In Table 4, the non-parametric Wilcoxon rank-sum test at 95% confidence level is carried out to verify statistical differences between the HR results for each objective count instance, and the Friedman’s test is used to determine the overall ranks of the algorithms. Wilcoxon rank-sum test results are presented in three digits: *w-t-l* representing win, tie and lose, respectively. The average of Friedman’s

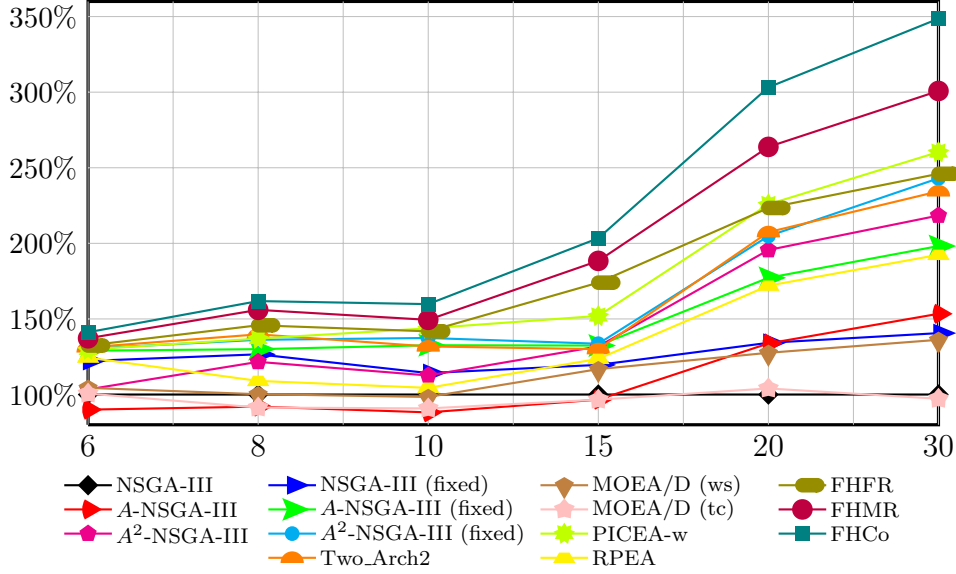


Figure 4: Relative HR values with regard to the objective count.

test results over all problem instances and the overall final results (given in parentheses) are provided. In both tables, the last three rows correspond to three versions of our proposed MaOEAs. Additionally, the behavior of algorithms as the number of objectives increases can be observed in terms of relative HR values in Figure 4 where results are normalized according to NSGA-III results. In this way, the relative performances of the algorithms can be observed and another view is obtained to interpret the results.

These results reveal that all three algorithms, NSGA-III, A-NSGA-III and A²-NSGA-III, benefit significantly when their hyperplanes are fixed. Among them, the fixed hyperplane version of A²-NSGA-III performs the best. However, although this algorithm employs a reference set update mechanism, it is still surpassed by our FHFR. In fact, FHFR is more successful than all of the competitors, only to be surpassed by the adaptive versions of itself. Its performance can be significantly enhanced by using mobile reference points (FHMR). In spite of this improvement, the reference point relocation strategy of FHMR remains rather restrictive compared with its co-evolutionary counterpart FHC0 which allows all reference points to evolve in a manner more responsive to the information gathered from the population about the shape of the Pareto front. As a result, the algorithm is capable of searching

Table 5: Mean computation time (in seconds) for single replication on MaOKP.

Algorithm	Objective Count (m)					
	6	8	10	15	20	30
NSGA-III	0.7	0.9	2.1	1.2	2.6	10.1
NSGA-III (fixed)	1.0	1.2	2.6	2.0	5.5	24.2
A-NSGA-III	1.4	1.9	4.1	3.1	7.0	31.4
A-NSGA-III (fixed)	1.6	2.1	4.4	3.9	12.4	65.3
A ² -NSGA-III	1.0	1.6	3.8	2.9	6.9	31.5
A ² -NSGA-III (fixed)	1.3	1.8	4.2	3.5	11.5	57.9
Two_Arch2	1.2	1.9	4.1	3.4	7.0	29.9
RPEA	0.8	1.0	2.3	1.6	3.5	16.4
MOEA/D (ws)	4.6	5.2	12.5	10.0	18.2	68.9
MOEA/D (tc)	4.3	5.1	11.3	9.9	17.8	67.1
PICEA-w	3.9	4.9	9.7	8.7	19.2	77.8
FHFR	3.4	3.8	7.5	5.2	12.7	54.5
FHMR	2.9	3.1	7.1	4.9	13.6	67.6
FHCo	4.1	4.6	9.5	8.4	18.1	83.4
<i>CPLEX</i>	0.4	0.5	0.6	0.8	2.2	3.4

different and relevant regions in the objective space and producing a crowded external archive. For the proposed co-evolutionary algorithm, the relative HR results increase as the number of objectives increases, indicating that it responds better to the objective count, and is therefore more suitable for many objectives. Finally, the Friedman’s test results reveal a perfect score for the co-evolutionary algorithm FHCo. In conclusion, the proposed algorithm is the best approach to solve MaOKP among the tested algorithms.

Finally, Table 5 presents the mean time requirement for a single replication of each algorithm alternative. The last row shows the computation times of the CPLEX solver to find single-objective optimal solutions. The computation times given for all algorithms, except the original versions of NSGA-III, A-NSGA-III and A²-NSGA-III, include these reported times.

5. Conclusion

In this study, we target the challenges imposed by many-objectives in discrete search spaces and develop a successful MaOEA for combinatorial optimization using MaOKP as the benchmark problem. Our proposed MaOEA uses a reference set based approach and is based on the co-evolution of the reference set together with the solution set.

The proposed algorithm increases the efficiency of reference set based approaches through the innovative use of several strategies. While reference set based approaches are very powerful methodologies when it comes to many-objective algorithms, our study demonstrates that their power can be considerably enhanced by fixing the hyperplane that encompasses the reference points at the beginning. It has also been shown that the efficiency can be improved further by using evolutionary strategies and genetic operators specifically designed to complement the reference set concept. The reference point guided path relinking recombination scheme, complementing parent selection mechanisms, mutation and local improvement operators are the main features that contribute to this success.

Nonetheless, the main strength of proposed algorithm comes from allowing the reference set to adapt to the underlying Pareto front by moving on a fixed hyperplane that has a good grip of the front. The proposed algorithm provides a practical and robust co-evolutionary structure that involves both cooperative and competitive aspects to achieve adaptation of the reference set to the topology of the Pareto front. The self-adaptive parametric nature not only prevents overfitting to test problem instances, but also creates a flexible and convenient way to balance diversity and convergence. Since the reference points are defined as coordinate points in the normalized objective space, the methods used in their evolution do not need to be problem specific. For instance, a problem such as creating an infeasible reference point never arises. Thus, the development and use of the evolution of the reference set can be generic.

The proposed algorithm solves MaOKP very effectively. As such, to the best of our knowledge, the presented results are the best so far in the literature for MaOKP. Moreover, the proposed algorithm can be easily extended to other combinatorial optimization problems such as many-objective TSP and QAP, once genetic operators are converted to handle permutation encoding instead of binary. Hence, in our ongoing research, we aim to assess the performance of the algorithm in different problems, especially when lexicographic optimal solutions are difficult to obtain at the initial stage of the algorithm and near optimal solutions need to be used to construct the hyperplane.

Acknowledgments

This work was supported by Boğaziçi University Research Fund Grant Number: 13843.

References

- Antonio, L. M., & Coello, C. A. C. (2016). Indicator-based cooperative coevolution for multi-objective optimization. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 991–998). IEEE.
- Antonio, L. M., & Coello, C. A. C. (2017). Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 22, 851–865.
- Asafuddoula, M., Singh, H. K., & Ray, T. (2017). An enhanced decomposition-based evolutionary algorithm with adaptive reference vectors. *IEEE transactions on cybernetics*, 48, 2321–2334.
- Azad, M. A. K., Rocha, A. M. A., & Fernandes, E. M. (2014). Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems. *Swarm and Evolutionary Computation*, 14, 66–75.
- Bader, J., & Zitzler, E. (2011). Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19, 45–76.
- Brockhoff, D., & Zitzler, E. (2006). Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. In *Parallel Problem Solving from Nature-PPSN IX* (pp. 533–542). Springer.
- Chand, S., & Wagner, M. (2015). Evolutionary many-objective optimization: A quick-start guide. *Surveys in Operations Research and Management Science*, 20, 35–42.
- Cheng, R., Jin, Y., Olhofer, M., & Sendhoff, B. (2016). A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans. Evolutionary Computation*, 20, 773–791.
- Coello, C. C., & Sierra, M. R. (2003). A coevolutionary multi-objective evolutionary algorithm. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03*. (pp. 482–489). IEEE volume 1.
- Das, I., & Dennis, J. E. (1998). Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8, 631–657.

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10 Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*
11 volume 16. John Wiley & Sons.
- 12
- 13 Deb, K., & Jain, H. (2013). An evolutionary many-objective optimization al-
14 gorithm using reference-point-based nondominated sorting approach, part
15 i: solving problems with box constraints. *IEEE transactions on evolution-
16 ary computation*, 18, 577–601.
- 17
- 18 Deb, K., & Sundar, J. (2006). Reference point based multi-objective opti-
19 mization using evolutionary algorithms. In *Proceedings of the 8th annual
20 conference on Genetic and evolutionary computation* (pp. 635–642). ACM.
- 21
- 22
- 23 Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable test prob-
24 lems for evolutionary multiobjective optimization. In *Evolutionary multi-
25 objective optimization* (pp. 105–145). Springer.
- 26
- 27
- 28 Figueira, J. R., Liefvooghe, A., Talbi, E.-G., & Wierzbicki, A. P. (2010). A
29 parallel multiple reference point approach for multi-objective optimization.
30 *European Journal of Operational Research*, 205, 390–400.
- 31
- 32
- 33 Giagkiozis, I., Purshouse, R. C., & Fleming, P. J. (2013). Towards un-
34 derstanding the cost of adaptation in decomposition-based optimization
35 algorithms. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE Inter-
36 national Conference on* (pp. 615–620). IEEE.
- 37
- 38
- 39 Glover, F., Laguna, M., & Martí, R. (2004). Scatter search and path relink-
40 ing: Foundations and advanced designs. In *New optimization techniques
41 in engineering* (pp. 87–99). Springer.
- 42
- 43
- 44 Goh, C. K., Tan, K. C., Liu, D., & Chiam, S. C. (2010). A competitive and
45 cooperative co-evolutionary approach to multi-objective particle swarm
46 optimization algorithm design. *European Journal of Operational Research*,
47 202, 42–54.
- 48
- 49 Hua, Y., Jin, Y., & Hao, K. (2018). A clustering-based adaptive evolution-
50 ary algorithm for multiobjective optimization with irregular pareto fronts.
51 *IEEE transactions on cybernetics*, 49, 2758–2770.
- 52
- 53
- 54 Huband, S., Hingston, P., Barone, L., & While, L. (2006). A review of
55 multiobjective test problems and a scalable test problem toolkit. *IEEE
56 Transactions on Evolutionary Computation*, 10, 477–506.
- 57
- 58

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9 Ishibuchi, H., Akedo, N., & Nojima, Y. (2013). A study on the specification
10 of a scalarizing function in moea/d for many-objective knapsack problems.
11 In *International Conference on Learning and Intelligent Optimization* (pp.
12 231–246). Springer.
- 13
- 14
- 15 Ishibuchi, H., Akedo, N., & Nojima, Y. (2014a). Behavior of multiobjec-
16 tive evolutionary algorithms on many-objective knapsack problems. *IEEE*
17 *Transactions on Evolutionary Computation*, 19, 264–283.
- 18
- 19
- 20 Ishibuchi, H., He, L., & Shang, K. (2019). Regular pareto front shape is not
21 realistic. In *2019 IEEE Congress on Evolutionary Computation (CEC)*
22 (pp. 2034–2041). IEEE.
- 23
- 24
- 25 Ishibuchi, H., Tanigaki, Y., Masuda, H., & Nojima, Y. (2014b). Distance-
26 based analysis of crossover operators for many-objective knapsack prob-
27 lems. In *International Conference on Parallel Problem Solving from Nature*
28 (pp. 600–610). Springer.
- 29
- 30
- 31 Jaimes, A. L., & Coello, C. A. C. (2015). Many-objective problems: chal-
32 lenges and methods. In *Springer handbook of computational intelligence*
33 (pp. 1033–1046). Springer.
- 34
- 35
- 36 Jain, H., & Deb, K. (2013). An improved adaptive approach for elitist
37 nondominated sorting genetic algorithm for many-objective optimization.
38 In *International Conference on Evolutionary Multi-Criterion Optimization*
39 (pp. 307–321). Springer.
- 40
- 41
- 42 Jain, H., & Deb, K. (2014). An evolutionary many-objective optimization al-
43 gorithm using reference-point based nondominated sorting approach, part
44 ii: Handling constraints and extending to an adaptive approach. *IEEE*
45 *Trans. Evolutionary Computation*, 18, 602–622.
- 46
- 47
- 48 Knowles, J., & Corne, D. (2002). On metrics for comparing nondominated
49 sets. In *Proceedings of the 2002 Congress on Evolutionary Computation.*
50 *CEC'02 (Cat. No. 02TH8600)* (pp. 711–716). IEEE volume 1.
- 51
- 52
- 53 Li, B., Li, J., Tang, K., & Yao, X. (2015). Many-objective evolutionary
54 algorithms: A survey. *ACM Computing Surveys (CSUR)*, 48, 13.
- 55
- 56
- 57
- 58
- 59
- 60
- 61
- 62
- 63
- 64
- 65

- 1
 - 2
 - 3
 - 4
 - 5
 - 6
 - 7
 - 8
 - 9
 - 10
 - 11
 - 12
 - 13
 - 14
 - 15
 - 16
 - 17
 - 18
 - 19
 - 20
 - 21
 - 22
 - 23
 - 24
 - 25
 - 26
 - 27
 - 28
 - 29
 - 30
 - 31
 - 32
 - 33
 - 34
 - 35
 - 36
 - 37
 - 38
 - 39
 - 40
 - 41
 - 42
 - 43
 - 44
 - 45
 - 46
 - 47
 - 48
 - 49
 - 50
 - 51
 - 52
 - 53
 - 54
 - 55
 - 56
 - 57
 - 58
 - 59
 - 60
 - 61
 - 62
 - 63
 - 64
 - 65
- Liu, X.-F., Zhan, Z.-H., Gao, Y., Zhang, J., Kwong, S., & Zhang, J. (2018). Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 23, 587–602.
- Liu, Y., Gong, D., Sun, X., & Zhang, Y. (2017). Many-objective evolutionary optimization based on reference points. *Applied Soft Computing*, 50, 344–355.
- Murata, T., & Taki, A. (2009). Many-objective optimization for knapsack problems using correlation-based weighted sum approach. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 468–480). Springer.
- Okabe, T., Jin, Y., & Sendhoff, B. (2003). A critical survey of performance indices for multi-objective optimisation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC’03.* (pp. 878–885). IEEE volume 2.
- Sato, H. (2015). Analysis of inverted pbi and comparison with other scalarizing functions in decomposition based moeas. *Journal of Heuristics*, 21, 819–849.
- Sato, H., Aguirre, H., & Tanaka, K. (2013). Variable space diversity, crossover and mutation in moea solving many-objective knapsack problems. *Annals of Mathematics and Artificial Intelligence*, 68, 197–224.
- Sato, H., Aguirre, H. E., & Tanaka, K. (2007). Controlling dominance area of solutions and its impact on the performance of moeas. In *International conference on evolutionary multi-criterion optimization* (pp. 5–20). Springer.
- Tan, K. C., Yang, Y., & Goh, C. K. (2006). A distributed cooperative coevolutionary algorithm for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 10, 527–549.
- Tanigaki, Y., Narukawa, K., Nojima, Y., & Ishibuchi, H. (2014). Preference-based nsga-ii for many-objective knapsack problems. In *Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on* (pp. 637–642). IEEE.

- Tian, Y., Cheng, R., Zhang, X., Cheng, F., & Jin, Y. (2017). An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. *IEEE Transactions on Evolutionary Computation*, 22, 609–622.
- Wang, H., Jiao, L., & Yao, X. (2014). Two_arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19, 524–541.
- Wang, R., Purshouse, R. C., & Fleming, P. J. (2015). Preference-inspired co-evolutionary algorithms using weight vectors. *European Journal of Operational Research*, 243, 423–441.
- Wierzbicki, A. P. (1980). The use of reference objectives in multiobjective optimization. In *Multiple criteria decision making theory and application* (pp. 468–486). Springer.
- Zhan, Z.-H., Li, J., Cao, J., Zhang, J., Chung, H. S.-H., & Shi, Y.-H. (2013). Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems. *IEEE transactions on cybernetics*, 43, 445–463.
- Zhang, Q., & Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11, 712–731.
- Zhang, Y.-H., Gong, Y.-J., Gu, T.-L., Yuan, H.-Q., Zhang, W., Kwong, S., & Zhang, J. (2017). Decal: Decomposition-based coevolutionary algorithm for many-objective optimization. *IEEE transactions on cybernetics*, 49, 27–41.
- Zhou, C., Dai, G., Zhang, C., Li, X., & Ma, K. (2018). Entropy based evolutionary algorithm with adaptive reference points for many-objective optimization problems. *Information Sciences*, 465, 232–247.
- Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *evolutionary computation, IEEE transactions on*, 3, 257–271.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7, 117–132.