

A Parallel Multiple Reference Point Approach for Multi-objective Optimization

José Figueira, Arnaud Liefoghe, El-Ghazali Talbi, Andrzej Wierzbicki

► **To cite this version:**

José Figueira, Arnaud Liefoghe, El-Ghazali Talbi, Andrzej Wierzbicki. A Parallel Multiple Reference Point Approach for Multi-objective Optimization. [Research Report] RR-6938, INRIA. 2009. <inria-00385079>

HAL Id: inria-00385079

<https://hal.inria.fr/inria-00385079>

Submitted on 18 May 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

*A Parallel Multiple Reference Point Approach for
Multi-objective Optimization*

José R. Figueira — Arnaud Liefooghe — El-Ghazali Talbi — Andrzej P. Wierzbicki

N° 6938

April 2009

Thème NUM

 *rapport
de recherche*

A Parallel Multiple Reference Point Approach for Multi-objective Optimization

José R. Figueira^{*}, Arnaud Liefooghe[†], El-Ghazali Talbi[‡],
Andrzej P. Wierzbicki[§]

Thème NUM — Systèmes numériques
Équipe-Projet Dolphin

Rapport de recherche n 6938 — April 2009 — 24 pages

Abstract: This document presents a multiple reference point approach for multi-objective optimization problems of discrete and combinatorial nature. When approximating the Pareto Frontier, multiple reference points can be used instead of traditional techniques. These multiple reference points can easily be implemented in a parallel algorithmic framework. The reference points can be uniformly distributed within a region that covers the Pareto Frontier. An evolutionary algorithm is based on an achievement scalarizing function that does not impose any restrictions with respect to the location of the reference points in the objective space. Computational experiments are performed on a bi-objective flow-shop scheduling problem. Results, quality measures as well as a statistical analysis are reported in the paper.

Key-words: multi-objective optimization, reference point, achievement scalarizing function, parallelization, bi-objective scheduling problems

^{*} CEG-IST, Center for Management Studies, Instituto Superior Técnico, Technical University of Lisbon, Tagus Park, Av. Cavaco Silva, 2780 - 990 Porto Salvo, Portugal (phone: +351 21 423 32 99, fax: +351 21 423 35 68, e-mail: figueira@ist.utl.pt). Also Associate Researcher at LAMSADE, Université Paris-Dauphine, France.

[†] Laboratoire d'Informatique Fondamentale de Lille (LIFL), UMR CNRS 8022, INRIA Lille-Nord Europe, Université Lille 1, Parc Scientifique de la Haute Borne, 40 avenue Halley, 59650 Villeneuve d'Ascq, France (phone: +33 3 59 57 79 30, fax: +33 3 59 57 78 50, e-mail: Arnaud.Liefooghe@lifl.fr).

[‡] Laboratoire d'Informatique Fondamentale de Lille (LIFL), UMR CNRS 8022, INRIA Lille-Nord Europe, Université Lille 1, Parc Scientifique de la Haute Borne, 40 avenue Halley, 59650 Villeneuve d'Ascq, France (phone: +33 3 59 57 78 88, fax: +33 3 59 57 78 50, e-mail: talbi@lifl.fr).

[§] National Institute of Telecommunications, Szachowa Str. 1, 04-894 Warsaw, Poland and Japan Advanced Institute of Science and Technology (JAIST), 1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan (phone: +48 22 51 28 100, fax: +48 22 51 28 625, e-mail: a.wierzbicki@itl.waw.pl).

A Parallel Multiple Reference Point Approach for Multi-objective Optimization

Résumé : This document presents a multiple reference point approach for multi-objective optimization problems of discrete and combinatorial nature. When approximating the Pareto Frontier, multiple reference points can be used instead of traditional techniques. These multiple reference points can easily be implemented in a parallel algorithmic framework. The reference points can be uniformly distributed within a region that covers the Pareto Frontier. An evolutionary algorithm is based on an achievement scalarizing function that does not impose any restrictions with respect to the location of the reference points in the objective space. Computational experiments are performed on a bi-objective flow-shop scheduling problem. Results, quality measures as well as a statistical analysis are reported in the paper.

Mots-clés : multi-objective optimization, reference point, achievement scalarizing function, parallelization, bi-objective scheduling problems

1 Introduction

Multi-objective Optimization (MO) is one of most fundamental areas in the field of Multiple Criteria Decision Analysis (MCDA). Over the last decades, a large number of papers were published in this field comprising both theoretical and applied works. The most challenging problems in MO are related to the identification of the Pareto Frontier (PF), or an approximation of it (PF^A) for large-size and rather difficult MO problems. For such a purpose, evolutionary algorithms seem more adequate than exact methods. However, the identification of the whole set or of an approximation of the PF is frequently not necessary, an approximation of some specific regions suffices. Indeed, when some preference information is provided by the Decision-Maker (DM), diverse methods can guide, in an interactive manner, the search of the potentially best compromise solution(s) to a particular region of interest. Reference point methods are particularly adequate to deal with this kind of situations; the preference information needed by them has mainly the form of reference point(s) (or also any other information that can be translated into reference point(s)). Reference point-based methods use then an achievement scalarizing function to make projection of the reference point or points onto the PF.

Contrary to the single-objective case, typically there is no unique optimal solution for a MO problem. Instead, a set of solutions, called compromise solutions, efficient solutions, etc, represent to the PF when transformed into the objective space. A fundamental issue while trying to solve MO problems relates to the cooperation between the DM and a computerized Decision Support System (DSS). In general, the DSS includes a mathematical model of the problem being solved along with a data base, an optimization solver, and an interactive solution method. There are several approaches to the roles that the DM could play in a decision-making process []. Firstly, the *a priori* approaches, where the DM is supposed to provide some knowledge or preferences about the problem to be solved in order to help the DSS in its search; practical experience shows that such methods are seldom effective. Separately, the *a posteriori* approaches, where the DSS aims at finding, or approximating the whole set of efficient solutions; the DM then has to choose his/her most preferred one. Finally, the interactive approaches where there is a progressive direct interaction/cooperation between the DM and the DSS.

Over the last two decades, most of MO resolution methods proposed in the literature were rather the *a posteriori* ones. A large part of them consists of approximating the set of efficient solutions and the corresponding PF using an evolutionary algorithm. On this one hand, this is based on the belief that the computing power of modern computers is unlimited, we can use them for any complex problems and solution methods. This belief, however, is contradicted by computational experience of solving complex problems: even the most powerful computer of any generation can be easily saturated, due to the nonlinear dependencies of computational complexity on the amount of data processed. Thus, a reasonable use of the existing computing power, even if this power is tremendous due to the possibilities of parallel use of computers in the network, still remains and will probably always remain a fundamental problem. On the other hand, many interactive approaches are based on the reference point method using achievement scalarizing functions as proposed by Wierzbicki [25] and developed by many other researchers (see, for example, [26]). The reference

point method results in projecting a given reference point (or a pair of them, usually called reservation and aspiration points), that represents the objective or criteria values desired by the DM, onto the set of efficient solutions. There are diverse methods of interaction with the DM in the reference point method, starting with just fully sovereign change of reference points by the DM, through using additional trade-off information, up to visual interfaces based on fuzzy specification of reference values. However, the result is focusing on a specific region of the objective space, thus avoiding the loss of computational resources for searching solutions that may not interest the DM at the end.

In this paper, we propose a new method combining the use of reference points while trying to approximate the whole set of efficient solutions, or a selected part of it. Instead of using a single reference point, the idea of this *a posteriori* approach is to automatically define a set of points in such a way that the objective space is uniformly divided, but entirely covered. Each point gives rise to a corresponding achievement scalarizing function that concentrates on a specific sub-region of the objective space. Thus, the set of efficient solutions can be rebuilt by combining the output of all solvers. Note that the solvers can be launched in parallel since the problems of optimizing a given achievement function can all be solved independently. The proposed parallel multiple reference point approach can be used to solve difficult real-world optimization problems, and is applied on a bi-objective combinatorial scheduling problem.

The outline of the paper is as follows. Section 2 introduces some fundamental concepts related to MO. Section 3 is devoted to the multiple reference point approach proposed in the paper; the key issues being widely detailed. Section 4 presents the parallel model and the implementation of the method. Section 5 formulates a bi-objective Flowshop Scheduling Problem (FSP). Section 6 shows the effectiveness of our approach by conducting experiments on the FSP. Finally, the last section concludes the paper and draws some perspectives.

2 Multi-objective Optimization (MO)

Many areas of the industry as, for example, telecommunications, transportation, aeronautics, chemistry, mechanical, and environment, deal with MO, where various conflicting objectives have to be considered simultaneously. This section briefly presents some basic concepts, definitions, and notation for MO. The interested reader is referred to [3, 4, 18] for more details about this field.

2.1 Basic Concepts

A general MO problem consists of optimizing a set of $n \geq 2$ objective functions $f_1(x), f_2(x), \dots, f_n(x)$. Each objective function can be either minimized or maximized (or even stabilized, kept close to a given target level [26]). Here we assume, without loss of generality, that they are all to be minimized. A decision vector $x = (x_1, x_2, \dots, x_k)$ is represented by a vector of k decision variables. Let X denote the set of *feasible solutions* in the *decision space*. To each decision vector $x \in X$ is assigned exactly one¹ objective vector, $z \in Z$, on the basis of a vector function $f : X \rightarrow Z$ with $z = (z_1, z_2, \dots, z_n) = f(x) =$

¹As distinguished from a distribution of objective vectors, which occurs in the case of stochastic multi-objective optimization. Even if such a stochastic case could be approached

$(f_1(x), f_2(x), \dots, f_n(x))$, where Z denotes the set of feasible points in the *objective* (or *criterion*) *space*. Therefore, a MO problem can be formulated as follows:

$$\begin{aligned} \text{“min”} \quad & f(x) = (f_1(x), f_2(x), \dots, f_n(x)) \\ \text{subject to:} \quad & x \in X \end{aligned} \quad (1)$$

Whereas solving a single-objective optimization problem generally results in a unique optimal solution, a MO problem obtains rather a set of solutions known as the *Pareto optimal*. Crucial concepts are the ones of *dominance* and of *efficiency*, that can be defined as follows:

Definition 1 (Dominance) *A solution $x_1 \in X$ dominates another solution $x_2 \in X$ if and only if $\forall i \in \{1, \dots, n\}$, $f_i(x_1) \leq f_i(x_2)$ and $\exists j \in \{1, \dots, n\}$ such that $f_j(x_1) < f_j(x_2)$.*

Definition 2 (Efficiency) *A solution $x^* \in X$ is efficient if and only if there is not another solution $x \in X$ such that x dominates x^* .*

The whole set of efficient solutions is the Pareto optimal set, and is denoted by X_P . The image of a Pareto optimal solution in the objective space results in a *non-dominated outcome vector*.

Definition 3 (Non-dominated outcome vector) *A point $z \in Z$ is a non-dominated outcome vector if there exists at least one efficient solution $x \in X_P$ such that $y = f(x)$.*

The set of all non-dominated outcome vectors is the *Pareto Frontier* (PF). One of the possible approaches for solving MO problems consists of finding PF or an approximation PF^A . This depends on the practical computational complexity of the problem, because finding a representation of PF is practically possible only if the resulting computational complexity is rather low.

Now, suppose that the optimum is known for each objective function, then we can define the concept of *ideal vector*:

Definition 4 (Ideal vector) *The ideal vector $z^* = (z_1^*, z_2^*, \dots, z_n^*)$ is the vector that optimizes each objective function individually, i.e. $z_i^* = \min_{x \in X} f_i(x)$.*

Of course, this ideal vector optimizing each objective function is rarely feasible as the objectives are often in conflict. Besides, the upper bounds for all objectives of the PF can be represented by the *nadir point* z^n . This nadir point is much more difficult to compute [18], especially when the number of objectives is more than two. A rough approximation of the nadir point can be provided by recording the maximal values of all objective functions obtained from their separate minimization while determining the ideal point.

2.2 Achievement Scalarizing Functions

The *achievement scalarizing functions* approaches, proposed by Wierzbicki [25], are often used to solve MO problems. These techniques are particularly well suited to work with reference points. A reference point gives desirable or acceptable values for each one of the objective functions. These objective values are

by similar techniques of parallelization as proposed in this paper. We address here, for the sake of simplicity, the deterministic case only.

called *aspiration levels* and the resulting objective vector is called a *reference point* and can be defined either in the feasible or in the unfeasible region of the objective space. One of the families of achievement functions can be stated as follows:

$$\sigma(z, z^0, \lambda, \rho) = \max_{i=1,2,\dots,n} \left\{ \lambda_j (z_i - z_i^0) \right\} + \rho \sum_{j=1}^n \lambda_j (z_j - z_j^0) \quad (2)$$

where σ is an application of Z into \mathbb{R} , $z = (z_1, z_2, \dots, z_n)$ is an objective vector, $z^0 = (z_1^0, z_2^0, \dots, z_n^0)$ is a reference point vector, $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ is a scaling coefficients vector (weighting coefficients), and ρ is an arbitrary small positive number ($0 < \rho \ll 1$). The word family is used here to state that several functions can be built according to the variability of the weighting coefficients and the reference points.

Now, the following achievement problem can be built:

$$\begin{aligned} \min \quad & \sigma(z, z^0, \lambda, \rho) \\ \text{subject to:} \quad & x \in X \end{aligned} \quad (3)$$

For a given reference point z^0 , two properties can be proved [20]:

- i) if $x^* = \arg \min_{x \in X} \sigma(z, z^0, \lambda, \rho)$, then x^* is an efficient solution;
- ii) if x^* is an efficient solution, then there exists a function $\sigma(z, z^0, \lambda, \rho)$ such that x^* is a (global) optimal solution of the achievement problem given in (3).

Note that the weighting vectors can be normalized, and the set of all feasible normalized weighting vectors can be represented as follows:

$$\Lambda = \left\{ \lambda = (\lambda_1, \lambda_2, \dots, \lambda_n) \mid \sum_{j=1}^n \lambda_j = 1, \lambda_j > 0, j = 1, 2, \dots, n \right\}$$

Using different λ vectors for the same reference point z^0 can lead to different optimal solutions for the achievement problem defined in (3). This aspect can be used to design a parallel algorithm dealing with the same reference point and different weighting vectors.

3 A Multiple Reference Point Approach

The goal of the approach proposed in the paper is to approximate the whole PF by using multiple reference points. Using several reference points is particularly well-suited to parallel computing since the resulting problems can all be solved independently. The general principle of the algorithm consists of generating a given number of reference points. These points are generated according to, either a rough approximation of the bounds of the PF, or by reasonable upper and lower bounds provided by the DM. For each reference point, a solver is assigned with. This solver aims at finding or approximating the efficient set corresponding to the pre-defined reference point (and some weighting coefficients). The reference point solvers are all launched in a parallel way and a master process preserves the approximated efficient set found by these sub-processes.

In this section, the multiple reference point approach is described after having introduced some essential issues relating to its design.

3.1 Fundamental Issues

This section introduces some fundamental aspects for the design of multiple reference point method. Firstly, a way of estimating the bounds of the PF is given. Secondly, a description of how to generate multiple reference points is detailed. Finally, a single reference point solver is proposed to be used as a sub-routine of the main algorithm.

3.1.1 Estimating the Bounds of the Pareto Frontier

As pointed out in Section 2.1, computing the nadir point z^n is rather a hard task, and even more difficult for discrete MO problems of large-size. Furthermore, computing the exact ideal point z^* is usually unfeasible as soon as we deal with large-size instances. However, in the work established in this paper, they are required to generate the initial set of reference points to be used by the algorithm. We should then find only a sufficiently good approximation of these two points. These approximations will be denoted z^{*^A} and z^{n^A} respectively.

Let us consider two legitimate options for performing such a task. On the one hand, if the DM is able to provide reasonable upper and lower bounds for each objective function, ideal and nadir computations or approximations rely then on such an initial scaling information. From our experience, it is always conceivable to ask the DM to give initial knowledge about the problem to be solved. Furthermore, note that metaheuristic algorithms can, in general, not run fully automatically without some initial information. At least, for instance, the starting solution or population, even if provided by some random number generator, usually needs some initial scaling.

On the other hand, if the DM is not able to provide any piece of information, another option is to approximate these points by means of a related search method to be performed over a relatively small number of iterations. Generally speaking, evolutionary multi-objective algorithms give set-valued but discrete approximations of the PF; the ideal and the nadir points of such approximations can then be computed. The difficulty of estimating nadir points is related to the original problem and to the fact that such evolutionary approximations might badly approximate a PF close to the nadir point. Therefore, it is important to apply at least two aspects of an evolutionary multi-objective algorithm. One is to use a strategy stressing diversity, or generally enhancing the importance of the extreme values of the discrete set-valued approximation of the PF; there might be several such strategies (see [21]). Another is to use a stopping criterion that involves the diameter of the approximation of the PF, measured as the distance between the nadir and the ideal points. If this diameter stabilizes in subsequent iterations, it indicates that either the approximation is already good (because a good approximation of the nadir point is one of the most difficult aspects of the entire approximation of the PF) or that the algorithm specifically applied cannot produce further improvement (which can always happen with heuristic algorithms). See [21] for a good overview of diverse examples. Notice that the approximation must be corrected by broadening it by some factor. This “broadening” is necessary for many reasons, starting with the fact that an evolutionary multi-objective algorithm approximates the distance ideal-nadir from below, and such “broadening” is even used when running classical nonlinear multicriteria optimization.

To approximate the ideal and the nadir points, Deb et al. [6] proposed a modified NSGA-II procedure [5] that focuses its search towards the extremities of the PF. Then, the best and worst objective values are computed among the final PF^A , and can be used to constitute the approximated ideal and nadir objective vectors, z^{*A} and z^{nA} . To do so, this algorithm uses a diversity maintaining strategy, based on an *extremized crowding distance* that emphasizes the importance of the worst objective solutions. The reader is referred to [6] for more details about this approach.

3.1.2 Generating Multiple Reference Points

There might exist several ways of generating multiple reference points. The most intuitive is perhaps the random generation over a predefined area of the objective space. This simple way, however, does not seem to be adequate to our case. The random generation can be so random that it does not cover a predefined area. This is the reason why we propose to define a quite uniform distribution of the multiple points over a predefined area. After having defined the bounds for each objective function, an area is built in such a way that it covers all the feasible region of the objective space. Figure 1 shows how to proceed when two objectives are involved. Let us assume, for the sake of simplicity, that the bounds can be defined in a more or less exact way.

How to proceed? A possible way is to perform the following steps:

1. Compute the ideal point z^* or a good approximation of it z^{*A} ;
2. Compute the nadir point z^n or a good approximation of it z^{nA} ;
3. Consider the box formed by these points, and define a cutting plane as the “diagonal” of this box;
4. If necessary, relax the extreme points of such a cutting plane and obtain the cutting plane used to generate the reference points;
5. Uniformly generate the reference points, according to the number of processors available.

Figure 1 illustrates how to define this cutting plane. There are two embedded rectangles. The small one was built from z^* and z^n . The diagonal, represented by the dashed line connecting the two opposing vertices can be used as a first cutting plane. Then, we relax this line in order to get the solid style one. This is done to avoid the use of the extreme points as reference ones, since they are non-dominated. Obviously, when dealing with approximations of the ideal and nadir points, we do not need to perform such a relaxation. In continuous multi-objective linear models, the plan covers the whole non-dominated region which is above the plan (when all the objective functions are to be minimized). In discrete MO, it is, however, not always the case as we can observe in the figure. There are non-dominated outcome points below and above the cutting line. This does not raise any problem since the achievement function that will be used does not impose any constraints on the location of the reference points. Such a conclusion is not true when using a (weighting) Chebychev metric, as it is the case in many research works done in interactive decision-making tools.

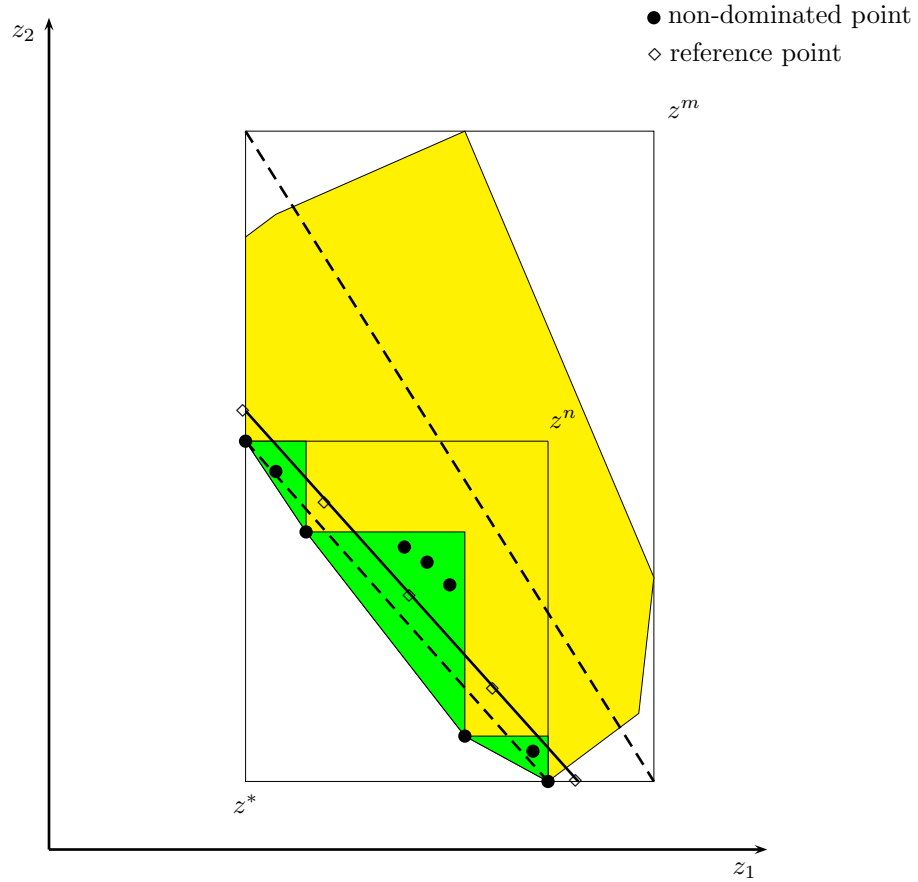


Figure 1: Cutting plane in the objective space for a two-objective problem.

The following figure shows how to define initial reference points for a three objectives problem. In such a case, the area is represented by a triangle. It shows a grid of fifteen reference points. Note that this principle can be easily generalized for n -dimensional problems, but the required number of reference points grows exponentially with n .

3.1.3 A Single Reference Point Evolutionary Algorithm

In this subsection, we present an evolutionary algorithm based on the achievement functions introduced in Section 2.2 and designed to be used with a single reference point z^0 and a single weighting coefficients vector λ . This single reference point search method will appear as a subroutine of the main algorithm in order to solve the single-objective optimization problem resulting from z^0 and λ . Different kinds of methods can be used to tackle such a problem (exact methods, heuristics, or metaheuristics), and here we choose to design an evolutionary algorithm. It consists of the Preference-Based Evolutionary Algorithm (PBEA) proposed in [23]. As the problem function is used to get efficient solutions, now we can talk in terms of approximate efficient solutions. Then, PBEA aims at producing a good, probably small, set of approximate efficient solutions related to z^0 and λ .

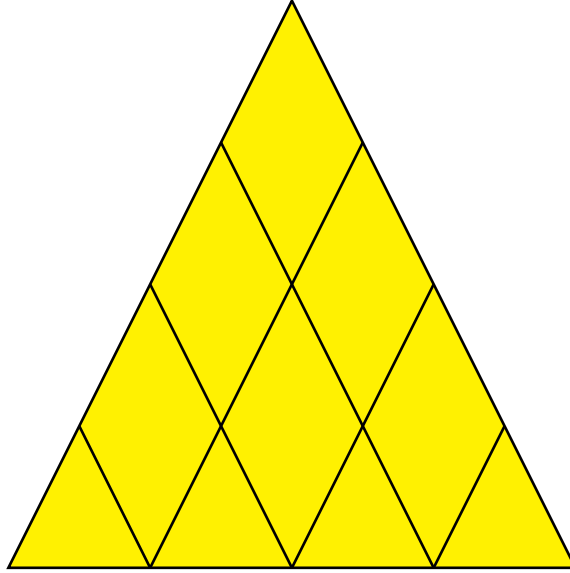


Figure 2: Generating initial reference points for a three objectives problem.

PBEA is a variant of the Indicator-Based Evolutionary Algorithm (IBEA) proposed by Zitzler and Künzli [27], where preference information is taken into account through a reference point. The main idea behind IBEA is to introduce a total order between solutions by generalizing the dominance relation given in Definition 1 by means of an arbitrary binary quality indicator I . Indeed, its fitness assignment scheme is based on a pairwise comparison of solutions from the current population, based on a user-given indicator I . To each individual x is then assigned a fitness value $F(x)$ measuring the “loss in quality” if x was removed from the current population [27]. Selection for reproduction consists of a binary tournament between randomly chosen individuals. And selection for replacement consists of iteratively removing the worst solution from the current population until the required population size is reached. Fitness information of the remaining individuals is updated each time there is a deletion. A detailed description of IBEA [27] is reproduced below.

Indicator-Based Evolutionary Algorithm (IBEA)

1. **Initialization.** Start with a user-given initial population P of size N , or generate it randomly.
2. **Fitness assignment.** Compute the fitness values of individuals in P :

$$F(x) \leftarrow \sum_{x' \in P \setminus \{x\}} (-e^{-I(x',x)/\kappa}) \quad (4)$$

where $\kappa > 0$ is a user-defined scaling factor.

3. **Environmental selection.** Iterate the following steps until the size of the population P does not exceed N :

- (a) Choose an individual $x^* \in P$ with the smallest fitness value: $F(x^*) \leq F(x)$ for all $x \in P$.
- (b) Remove x^* from P .
- (c) Update the fitness values of the remaining individuals. For all $x \in P$:

$$F(x) \leftarrow F(x) + e^{-I(x^*, x)/\kappa} \quad (5)$$

4. **Termination.** If a stopping condition is satisfied, return the non-dominated solutions of P . Stop.
5. **Mating pool selection.** Perform binary tournament selection with replacement on P in order to fill the temporary mating pool P' .
6. **Variation.** Apply recombination and mutation operators to the mating pool P' and add the resulting offspring to P . Go to Step 2.

Several binary indicators can be used in (4). As an example, the authors [27] define the binary additive ϵ -indicator ($I_{\epsilon+}$) as follows:

$$I_{\epsilon+}(x, x') = \max_{i \in \{1, \dots, n\}} \{f_i(x) - f_i(x')\} \quad (6)$$

It computes the minimum value by which a solution $x \in X$ has to be, or can be, translated in the objective space to weakly dominate another solution $x' \in X$.

In order to take preference information into account by means of a reference point, Thiele et al. [23] proposed the so-called *preference-based quality indicator* based on the achievement function given in Definition 2. First, a normalized achievement function is defined.

$$\sigma(f(x), z^0, \lambda, \rho, \delta) = \sigma(f(x), z^0, \lambda, \rho) + \delta - \min_{x' \in P} \{\sigma(f(x'), z^0, \lambda, \rho)\} \quad (7)$$

Parameter $\delta > 0$ a *specificity* representing the minimal value of the normalized function. Then, the preference based quality indicator I_p can be defined as follows:

$$I_p(x, x') = I_{\epsilon+}(x', x) / \sigma(f(x), z^0, \lambda, \rho, \delta) \quad (8)$$

This quality indicator can now be used in (4) to set the fitness values of a population P . The resulting algorithm is referred to as PBEA.

In order to (approximately) give the same amplitude to each objective function, note that we replace each element z_i of an objective vector z with a normalized value by using the (approximated) ideal and nadir points:

$$\frac{z_i - z_i^*}{z_i^n - z_i^*} \quad (9)$$

Moreover, an external archive A has been added to PBEA in order to store the current approximated efficient set. It is updated with new non-dominated solutions at each iteration of the algorithm.

3.2 Outline of the Method

The parallel multiple reference point approach proposed in this paper can be divided into two consecutive phases. The first one, called the preparation phase, is devoted to the design of several versions of a PBEA by *i*) estimating the bounds of the PF, *ii*) generating multiple reference points, and *iii*) designing a version of the solver for each reference point. The second phase is the running phase and consists of launching a PBEA version for every reference point in every processor, until a stopping condition is fulfilled. Details about the search method are summarized above.

3.2.1 Preparation Phase

The preparation phase is devoted to assign an equal number of *CPUs* and reference point vectors by:

1. Getting a very rough common sense estimation of lower bounds and upper bounds for all objective functions. This estimation can either be given by the DM, or approximated automatically (see Subection 3.1.1).
2. A possible stopping test for the latter algorithm can be stated as follows. First, determine the outer diameter (Chebychev norm of the distance between the current lower and upper bound) for the current efficient set approximation. Then, compare it with the former estimation of the outer diameter. If the difference between both values remains constant over a user-given number of iterations, stop.
3. Imagining a cutting plane through the area (of normalized ranges of objectives), and defining a uniform distribution of reference points in this area.
4. Assigning to every computer a PBEA version with a quality indicator function related to an achievement scalarization function defined by a reference point z^0 and a weighting vector λ .

3.2.2 Running Phase

The running phase consists of performing a PBEA version in parallel until a stopping condition is verified. The output of each reference point solver is then unified to obtain the efficient set approximation. The global algorithm is outlined next.

Parallel Multiple Reference Point Evolutionary Algorithm (PMRPEA)

1. Start with a user-given approximation of objective ranges; use this approximation as an initial estimation of the bounds of the PF to determine the initial ideal and nadir point approximations, z^{*A} and z^{nA} .
2. Set $d^* \leftarrow$ outer diameter of these bounds, measured by $\|z^{nA} - z^{*A}\|$.
3. Set $\lambda = (1/n, 1/n, \dots, 1/n)$, where n is the number of objective functions.

4. Generate m initial reference points $z^{0(1)}, z^{0(2)}, \dots, z^{0(m)}$ in a uniform way by using the (approximated) bounds of the PF.
5. Generate m initial populations P_1, P_2, \dots, P_m of size N .
6. For $i \leftarrow 1$ to m , perform in parallel (on m processors):

$$A_i \leftarrow \text{PBEA}(P_i, G, z^{0(i)}, \lambda, \rho)$$
 until a stopping condition is verified.
7. Return the non-dominated solutions of $A_1 \cup A_2 \cup \dots \cup A_m$.

4 Parallel Model and Implementation

To design the algorithm proposed in the paper, two levels of parallelism may be used: the reference point level and the evolutionary algorithmic level. First, at the reference point level, using multiple reference points can naturally be implemented in a parallel algorithmic framework, since the resulting single-reference point solvers can all be launched independently. Moreover, we saw that multiple weighting coefficients vectors can also be used for the same reference point, what gives rise to a second level of parallelism. Next, at the evolutionary algorithmic level, three models can be distinguished [17]. The first one is the Island model and consists of deploying different evolutionary algorithms simultaneously. The second one, the parallel evaluation of the population, consists of distributing the evaluation of the objective value(s) of solutions contained in the population between different “workers”. The last one is the distributed evaluation of a single solution, where the evaluation of one solution is itself parallelized (when it is possible and interesting to do it). For the sake of simplicity, we here only focus our attention on the parallelism at the reference point level. To this end, we will implement the single reference point-based evolutionary algorithm as described in Subection 3.1.3. But, of course, any other reference point evolutionary algorithm can be used. For instance, a more advanced version of this evolutionary algorithm that would intrinsically involve parallelism can be considered instead of this simple model. The interested reader is referred to [17] for more details about parallel and distributed metaheuristics.

In terms of implementation, ParadisEO² has been used. ParadisEO [2] is a white-box object-oriented software framework devoted to the flexible design of metaheuristics. It is composed of four connected modules: ParadisEO-EO for population-based metaheuristics, ParadisEO-MO for single solution-based metaheuristics, ParadisEO-MOEO for multi-objective metaheuristics, and finally ParadisEO-PEO for parallel and distributed metaheuristics. Moreover, ParadisEO also includes tools for the design of hybrid and cooperative models. The single reference point evolutionary algorithm introduced in Section 3.1.3, PBEA, has been implemented using the ParadisEO-MOEO [16] module of ParadisEO. The ParadisEO-PEO module is based on the other modules of ParadisEO, so that PBEA instances can directly be used in the implementation of the parallel multiple reference point algorithm. Using ParadisEO-PEO, different types of parallel and distributed architectures may be used. In our case, the

²ParadisEO is available at the URL: <http://paradiseo.gforge.inria.fr>

implementation of the parallel algorithm is based on the Message Passing Implementation (MPI) library. MPI is a parallel programming environment allowing a portable deployment on networks of heterogeneous workstations.

5 A Case Study

The FSP is one of the most well-known scheduling problems and has been widely studied in the literature. The majority of works devoted to this problem considers it on a single-objective form and mainly aims at minimizing the makespan (*i.e.* the total completion time). However, many objective functions, varying according to the particularities of the tackled problem, may be considered and some multi-objective approaches have also been proposed. For a survey on this topic, see for instance [13, 19, 24].

Following the formulation of a multi-objective permutation FSP, this section presents some complexity issues and various works related to the problem under consideration.

5.1 Problem Definition

Solving the FSP consists of scheduling N jobs $\{J_1, J_2, \dots, J_N\}$ on M machines $\{M_1, M_2, \dots, M_M\}$. Machines are critical resources, *i.e.* one machine cannot process more than one job at a time. Each job J_i is composed of M consecutive tasks $\{t_{i1}, t_{i2}, \dots, t_{iM}\}$, where t_{ij} represents the j^{th} task of the job J_i , requiring the machine M_j . A processing time p_{ij} is associated to each task t_{ij} ; and a due date d_i is given to each job J_i (the deadline of the job). In this study, we focus on the permutation FSP, where the operating sequences of the jobs are identical and unidirectional for every machine, as illustrated in Figure 3. Then, for a problem instance of N jobs, there exists $N!$ feasible solutions.

M ₁	J ₁	J ₂	J ₃				
M ₂		J ₁	J ₂	J ₃			
M ₃			J ₁	J ₂	J ₃		
M ₄				J ₁	J ₂	J ₃	

Figure 3: Example of a solution for a permutation FSP where 3 jobs (J_1, J_2, J_3) have to be scheduled on 4 machines (M_1, M_2, M_3, M_4).

Many objective functions may be tackled while scheduling tasks on several machines [19]. The FSP that we consider here aims at minimizing the following ones: the makespan (C_{max}) and the total tardiness (\bar{T}). These objectives are among the most widely investigated in the literature [13, 19, 24]. For each task t_{ij} being scheduled at the time s_{ij} , they are computed as follows:

$$C_{max} = \max_{i \in \{1, \dots, N\}} \{s_{iM} + p_{iM}\} \quad (10)$$

$$\bar{T} = \sum_{i=1}^N \left\{ \max\{0, s_{iM} + p_{iM} - d_i\} \right\} \quad (11)$$

According to [8], this problem can be denoted by $F/perm, d_i/(C_{max}, \bar{T})$.

5.2 Complexity Issues

Even if it can be solved in polynomial time by using the Johnson's algorithm [10] for two machines, the FSP of minimizing the makespan has been proven to be *NP-hard* for three or more machines [14]. The objective of minimizing the total tardiness is already *NP-hard* for one machine [7], what possibly explains the small number of studies dealing with minimizing the total tardiness in the case of M machines [11]. Hence, as minimizing the makespan and the total tardiness independently is already *NP-hard*, medium- and large-size instances can generally not be solved exactly.

5.3 Related Works

The methods proposed in the literature for the resolution of multiple objective scheduling problems vary from exact methods, specific heuristics and metaheuristics. In their survey, Nagar et al. [19] classify the scheduling problems according to different characteristics, including shop configuration (single or multiple machines) and objectives (two- or more-than-two objectives). The majority of FSP works on multiple machine has been restricted to the treatment of one objective at a time (generally the makespan or the sum of completion times). Moreover, most of multi-objective scheduling applications deal with two-machine and/or bi-objective problems, and concentrate on FSPs. In [24], T'Kindt and Billaut provide an overview on multi-objective scheduling for both researcher and industrial points of view. They provide models and a topology for single- and multi-objective scheduling problems, and describe some exact and heuristic algorithms to tackle them. More recently, Landa Silva et al. [13] provided a survey about metaheuristics for solving multi-objective scheduling problems. It seems that the most commonly used approaches are genetic algorithms and local searches, but also algorithms hybridizing both or hybridizing metaheuristics with exact methods.

5.4 Problem-related Implementation Issues

The problem-related components involved in the evolutionary algorithm, and used for the specific case of the FSP under consideration are the following ones:

Individual representation: sequence of jobs scheduled in one machine. A solution of an instance with N jobs and M machines is represented by a permutation of size N .

Initialization: Randomly generated solutions.

Mutation: Insert mutation [9] (see Figure 4).

Crossover: Two-point crossover [9] (see Figure 5).

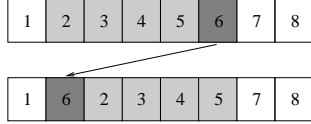


Figure 4: Insert mutation.

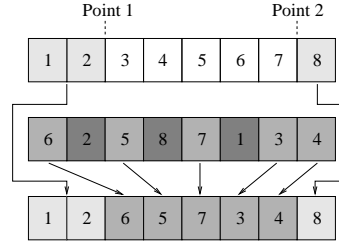


Figure 5: Two-point crossover.

6 Computational Experiments

In this section, computational experiments are performed on the parallel multiple reference point algorithm applied to the multi-objective FSP introduced in the previous section. The experimental protocol used to evaluate the quality of the algorithm is first described. Then, some results are given and discussed in order to extract some useful information about the proposed approach.

6.1 Experimental Protocol

In order to quantify the effectiveness of the proposed approach, we compare its behavior to both NSGA-II [5] and IBEA [27]. The first one is often used as a reference to be compared with for MO. The latter is also investigated because the PMRPEA sub-procedures are based on an IBEA-variant, so that the impact of the proposed parallelization scheme will be fairly measured at the implementation level.

6.1.1 Characteristics of the Computer Network

The experiments have been conducted on a cluster of 2 nodes, each one being composed of 2 Xeon 5060 CPUs (3.2 GHz, 2×2 MB, 4 Gb RAM). Hence, a total number of 16 processors interconnected in a distributed computing environment were available.

6.1.2 Benchmark Test Instances

To evaluate the performance of the mechanisms introduced in this paper, we consider various benchmark test instances proposed in [15]³. They have been built from Taillard instances for the single-objective FSP [22] and extended to the bi-objective case by adding a due date for every job. These due dates are fixed using a value randomly generated between $\bar{p} \times M$ and $\bar{p} \times (N + M - 1)$, where N stands for the number of jobs, M for the number of machines and \bar{p} for the average processing time for the instance under consideration. Thus, a due date roughly lies between the average completion date of the first scheduled job and the average completion date of the last scheduled job. During our experiments, we will consider a set of 8 instances involving from 20 to 100 jobs and from 5 to 20 machines. An instance denoted by $N \times M \times i$ represents the i^{th} instance composed of N jobs and M machines.

³These instances are available at the URL: <http://www.lifl.fr/~liefvooga/benchmarks/>

6.1.3 Parameter Values Settings

The parameter values that we used during our experiments are summarized in Table 1. The stopping condition has been motivated by the fact that, after 5000 generations with no improvement, we can reasonably think that the evolutionary algorithm has reached its convergence. However, a non-improving iteration is difficult to define while dealing with MO. Here, we state that an iteration is non-improving if there are no potential non-dominated solution that can be included into the archive. In addition, a maximum runtime of 30 minutes was allowed.

Table 1: Parameter Setting.

	Parameter	Value
	Number of reference points	10
	Population size	100
	Max. number of iterations without improvement	5000
	Maximum runtime	30 minutes
	Crossover rate	0.8
	Mutation rate	1.0
	κ (PBEA and IBEA)	0.05
	δ (PBEA)	10^{-2}
	ρ (PMRPEA)	10^{-4}

6.1.4 Performance Assessment

In the frame of MO, the performance assessment of a number of algorithms in solving the same problem is a key issue. In this study, a set of 30 runs per instance is performed for each evolutionary algorithm. In order to evaluate the quality of the approximations for every instance we experimented, the protocol proposed by Knowles et al. in [12] was adopted. For a given instance, let Z^{all} denotes the union of the outputs we obtained during all our experiments. Note that this set probably contains both dominated and non-dominated points, as a given approximation may contain vectors dominating the ones of another approximation, and *vice-versa*. We first compute a reference set PF^A containing all the non-dominated points of Z^{all} plus any other existing best know non-dominated set for the problem under consideration. Second, we define $z^{max} = (z_1^{max}, \dots, z_n^{max})$, where z_k^{max} denotes the upper bound of the k^{th} objective for all the points contained in Z^{all} . In order to give a roughly equal range to the objective functions, values are normalized with respect to z^{max} . Now, to measure the quality of an output set A in comparison to Z_N^* , we compute the difference between these two sets by using the unary hypervolume metric [28], z^{max} being the reference point. The hypervolume difference indicator (I_H^-) computes the portion of the objective space that is weakly dominated by PF^A and not by A . The closer this measure is to 0, the better is the approximation A . Furthermore, we also consider the additive ϵ -indicator proposed in [28]. The unary additive ϵ -indicator ($I_{\epsilon+}^1$) gives the minimum factor by which an approximation A can or has to be translated in the objective space to weakly dominate the reference set Z_N^* . As a consequence, for each test instance, we obtain 30 I_H^- measures and 30 $I_{\epsilon+}^1$ measures, corresponding to the 30 simulation

runs, *per* algorithm. Once all these values are computed, we first compute an average value *per* metric. Additionally, we perform a statistical analysis for a pairwise comparison of methods. To this end, we use the Wilcoxon signed rank test. Such a non-parametric statistical test is motivated by the fact that samples collected here correspond to matched samples. Indeed, for a given simulation run, the random seed numbers are identical for all the algorithms, so that the final indicator values can be taken as pairs. Details for this statistical testing procedure are given in [12]. Hence, for a given test instance, and with respect to the p -value and to the metric under consideration, this statistical test reveals that if the sample of approximation sets obtained by a given search method is significantly better than the one of another search method, or if there is no significant difference between both. Note that all the performance assessment procedures have been achieved using the performance assessment tool provided in PISA⁴ [1].

6.2 Results and Discussion

Results are summarized in Tables 2, 3 and 4. PMRPEA denotes the parallel multiple reference point evolutionary algorithm introduced in this paper. According to the experimental protocol defined above, the first set of experiments revealed that PMRPEA results was significantly better than the ones of IBEA and NSGA-II on all instances with respect to both metrics. However, as it can be seen in Table 2, the average runtime of PMRPEA is always higher than the one of the other algorithms under investigation. That is the reason why we performed another set of experiments where the PMRPEA search process was stopped after a smaller runtime. The corresponding algorithm is denoted by PMRPEA-2. For a given benchmark test instance, the PMRPEA-2 maximum runtime was set as the minimum value between the average runtime of IBEA and the average runtime of NSGA-II, observed from previous experiments. However, PMRPEA-2 also obtained significantly better I_H^- - and I_{ϵ^+} -values than IBEA and NSGA-II on every instance we tested. Besides, PMRPEA-2 is predominantly outperformed by PMRPEA, except for large-size instances of 100 jobs, where the difference between both is not significant, see Tables 3 and 4.

7 Conclusion and Directions for Future Research

In this paper, a parallel multiple reference point approach has been proposed for solving multi-objective optimization problems. The parallelization of evolutionary multi-objective optimization algorithms is combined with the use of a reference point approach. On the one hand, the problem can be formulated as follows: how to parallelize the resolution method by separating a number of regions in the objective space, then applying an evolutionary algorithm in each region, then combining the results? The clear motivation is to exploit the power of computer networks. On the other hand, the concept of reference points can be naturally used to separate regions of interest in the objective space, and then to parallelize the computations.

The outcomes of our tests show that the parallelization of evolutionary multi-objective optimization algorithms based on reference point methods gives very

⁴The package is available at <http://www.tik.ee.ethz.ch/pisa/assessment.html>.

Table 2: Comparison of the different algorithms with respect the average runtime, the average I_H^- -value and the average I_{ϵ^+} -value.

benchmark instance		average runtime (seconds)	average I_H^- -value ($\times 10^{-1}$)	average I_{ϵ^+} -value ($\times 10^{-1}$)
$020 \times 05 \times 01$	PMRPEA	311.1	0.942	1.582
	PMRPEA-2	111.0	0.967	1.618
	IBEA	111.5	1.905	2.046
	NSGA-II	124.9	1.722	2.000
$020 \times 10 \times 01$	PMRPEA	345.0	0.212	0.492
	PMRPEA-2	147.0	0.229	0.508
	IBEA	147.4	0.912	1.090
	NSGA-II	182.6	0.760	1.035
$020 \times 20 \times 01$	PMRPEA	510.0	0.493	0.852
	PMRPEA-2	210.0	0.514	0.881
	IBEA	210.7	1.900	2.446
	NSGA-II	218.1	2.023	2.532
$050 \times 05 \times 01$	PMRPEA	517.1	0.740	0.738
	PMRPEA-2	254.0	0.750	0.745
	IBEA	258.0	1.630	1.265
	NSGA-II	254.2	1.522	1.231
$050 \times 10 \times 01$	PMRPEA	794.4	2.759	2.267
	PMRPEA-2	640.0	2.776	2.275
	IBEA	640.1	3.398	2.855
	NSGA-II	767.6	3.421	2.870
$050 \times 20 \times 01$	PMRPEA	1000.7	2.119	1.886
	PMRPEA-2	561.0	2.184	1.923
	IBEA	561.0	3.558	2.995
	NSGA-II	934.9	3.109	2.767
$100 \times 10 \times 01$	PMRPEA	1550.6	2.748	2.252
	PMRPEA-2	1060.0	2.817	2.303
	IBEA	1060.5	4.643	4.028
	NSGA-II	1260.3	3.889	3.367
$100 \times 20 \times 01$	PMRPEA	1740.5	2.666	2.280
	PMRPEA-2	1620.0	2.665	2.285
	IBEA	1620.3	3.424	2.872
	NSGA-II	1702.8	3.277	2.965

Table 3: Wilcoxon rank test p -values with respect to the I_H^- metric. Either the algorithm located at a specific row significantly outperforms the algorithm located at a specific column (represented by a gray or light-gray area for a p -value less or equal to 0.01 or to 0.05, respectively), or there is no significant difference between both (represented by a white area).

		IBEA	NSGA-II	PMRPEA-2
020 × 05 × 01	PMRPEA			
	PMRPEA-2			-
020 × 10 × 01	PMRPEA			
	PMRPEA-2			-
020 × 20 × 01	PMRPEA			
	PMRPEA-2			-
050 × 05 × 01	PMRPEA			
	PMRPEA-2			-
050 × 10 × 01	PMRPEA			
	PMRPEA-2			-
050 × 20 × 01	PMRPEA			
	PMRPEA-2			-
100 × 10 × 01	PMRPEA			
	PMRPEA-2			-
100 × 20 × 01	PMRPEA			
	PMRPEA-2			-

Table 4: Wilcoxon rank test p -values with respect to the $I_{\epsilon+}$ metric. Either the algorithm located at a specific row significantly outperforms the algorithm located at a specific column (represented by a gray or light-gray area for a p -value less or equal to 0.01 or to 0.05, respectively), or there is no significant difference between both (represented by a white area).

		IBEA	NSGA-II	PMRPEA-2
020 × 05 × 01	PMRPEA			
	PMRPEA-2			-
020 × 10 × 01	PMRPEA			
	PMRPEA-2			-
020 × 20 × 01	PMRPEA			
	PMRPEA-2			-
050 × 05 × 01	PMRPEA			
	PMRPEA-2			-
050 × 10 × 01	PMRPEA			
	PMRPEA-2			-
050 × 20 × 01	PMRPEA			
	PMRPEA-2			-
100 × 10 × 01	PMRPEA			
	PMRPEA-2			-
100 × 20 × 01	PMRPEA			
	PMRPEA-2			-

promising and statistically significantly better results than the comparable non-parallel evolutionary multi-objective algorithm. However, many further issues should be studied in future work. One of them is defining and analyzing statistically the speed-up factor due to parallelization. Another is analyzing diverse possible versions of parallelization, the impact of a larger number of objectives, etc. Finally, interactive versions of the algorithms proposed in this paper are also possible, together with a question of possible dynamic modifications of reference points.

Acknowledgments. José Rui Figueira acknowledges the COST Action Research Grant IC0602 on “Algorithmic Decision Theory”.

References

- [1] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA — a platform and programming language independent interface for search algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *Lecture Notes in Computer Science*, pages 494–508, Faro, Portugal, 2003. Springer-Verlag.
- [2] S. Cahon, N. Melab, and E.-G. Talbi. ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.
- [3] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, Boston, MA, USA, 2002.
- [4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.
- [5] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 849–858, Paris, France, 2000.
- [6] K. Deb, S. Chaudhuri, and K. Miettinen. Towards estimating nadir objective vector using evolutionary approaches. In *Proceedings of the 8th Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 643–650, Seattle, Washington, USA, 2006.
- [7] J. Du and J. Y.-T. Leung. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15:483–495, 1990.
- [8] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.
- [9] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 28:392–403, 1998.

-
- [10] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1:61–68, 1954.
- [11] Y.-D. Kim. Minimizing total tardiness in permutation flowshops. *European Journal of Operational Research*, 33:541–551, 1995.
- [12] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006. (revised version).
- [13] J. D. Landa Silva, E. Burke, and S. Petrovic. An introduction to multiobjective metaheuristics for scheduling and timetabling. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 91–129. Springer-Verlag, Berlin, Germany, 2004.
- [14] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.
- [15] A. Liefvooghe, M. Basseur, L. Jourdan, and E.-G. Talbi. Combinatorial optimization of stochastic multi-objective problems: an application to the flow-shop scheduling problem. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 of *Lecture Notes in Computer Science*, pages 457–471, Matsushima, Japan, 2007. Springer-Verlag.
- [16] A. Liefvooghe, M. Basseur, L. Jourdan, and E.-G. Talbi. ParadisEO-MOEO: A framework for evolutionary multi-objective optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 of *Lecture Notes in Computer Science*, pages 386–400, Matsushima, Japan, 2007. Springer-Verlag.
- [17] N. Melab, E.-G. Talbi, S. Cahon, E. Alba, and G. Luque. Parallel metaheuristics: Models and frameworks. In E.-G. Talbi, editor, *Parallel Combinatorial Optimization*, chapter 6, pages 149–162. John Wiley & Sons, Chichester, UK, 2006.
- [18] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12. Kluwer Academic Publishers, Boston, MA, USA, 1999.
- [19] A. Nagar, J. Haddock, and S. Heragu. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research*, 81:88–104, 1995.
- [20] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley & Sons, Chichester, UK, 1986.
- [21] M. Szczepański and A.P. Wierzbicki. Application of multiple criteria evolutionary algorithms to vector optimisation, decision support and reference point approaches. *Journal of Telecommunications and Information Technology*, 3:16–33, 2003.

-
- [22] E. D. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285, 1993.
- [23] L. Thiele, K. Miettinen, P. Korhonen, and J. Molina. A preference-based interactive evolutionary algorithm for multiobjective optimization. Working Papers W-412, Helsinki School of Economics, Helsinki, 2007.
- [24] V. T'Kindt and J.-C. Billaut. *Multicriteria Scheduling: Theory, Models and Algorithms*. Springer-Verlag, Berlin, Germany, 2002.
- [25] A. Wierzbicki. The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, editors, *Multiple Objective Decision Making, Theory and Application*, number 177 in Lecture Notes in Economics and Mathematical Systems, pages 468–486. Springer-Verlag, 1980.
- [26] A. P. Wierzbicki, M. Makowski, and J. Wessels, editors. *Model-Based Decision Support Methodology with Environmental Applications*. Kluwer Academic Publishers, Dordrecht, 2000.
- [27] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842, Birmingham, UK, 2004. Springer-Verlag.
- [28] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Contents

1	Introduction	3
2	Multi-objective Optimization (MO)	4
2.1	Basic Concepts	4
2.2	Achievement Scalarizing Functions	5
3	A Multiple Reference Point Approach	6
3.1	Fundamental Issues	7
3.1.1	Estimating the Bounds of the Pareto Frontier	7
3.1.2	Generating Multiple Reference Points	8
3.1.3	A Single Reference Point Evolutionary Algorithm	9
3.2	Outline of the Method	12
3.2.1	Preparation Phase	12
3.2.2	Running Phase	12
4	Parallel Model and Implementation	13
5	A Case Study	14
5.1	Problem Definition	14
5.2	Complexity Issues	15
5.3	Related Works	15
5.4	Problem-related Implementation Issues	15
6	Computational Experiments	16
6.1	Experimental Protocol	16
6.1.1	Characteristics of the Computer Network	16
6.1.2	Benchmark Test Instances	16
6.1.3	Parameter Values Settings	17
6.1.4	Performance Assessment	17
6.2	Results and Discussion	18
7	Conclusion and Directions for Future Research	18



Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)

ISSN 0249-6399