Integrating preferences within multiobjective flexible job shop scheduling

Madani Bezoui, Alexandru-Liviu Olteanu, Marc Sevaux

Integrating preferences within multiobjective flexible job shop scheduling

Madani Bezoui[a,b], Alexandru-Liviu Olteanu[a,*], Marc Sevaux[a]

[a] *Lab-STICC, UMR 6285, CNRS, Université Bretagne Sud*
[b] *LINEACT Laboratory - CESI, 8 Rue Emmanuel Grout, 06200 Nice*

Abstract

When faced with a multiobjective optimization problem, it is necessary to consider the decision-maker preferences in order to propose the best compromise solution. We consider the multiobjective flexible job shop scheduling problem and a decision-maker that is best represented using a non-compensatory reference level-based preference model. We show how integrating this model into a multiobjective genetic algorithm allows to obtain solutions that surpass more aspiration levels when compared to classical multiobjective optimization approaches. Furthermore, these solutions are found faster and in greater numbers which facilitates their integration within the workshop.

Keywords: Multiobjective Optimization, Multi-criteria Decision Aiding, Flexible Job Shop Scheduling, Genetic Algorithm, Preference Models

1. Introduction

Taking into account multiple objectives in an optimization problem may prove difficult from the perspective of a decision-maker (DM), and often a compromise solution needs to be found. Classically, researchers have focused on constructing a set of efficient, i.e. non-dominated, solutions which are then submitted to the DM [30]. This approach, however, may spend a lot of computational resources on constructing solutions that would be deemed unsatisfactory by the DM. Integrating the preferences of the DM during the

*Corresponding author (alexandru.olteanu@univ-ubs.fr)

optimization process may help in focusing only on relevant solutions and thus provide desirable solutions in a more timely manner.

In this work, we focus on integrating the preferences of DMs within the optimization process and illustrate this using the multiobjective flexible job shop problem (MOFJSP). Multi-criteria decision aiding (MCDA) methods are generally classified into: methods based on multi-attribute value theory (MAVT) [26] and outranking methods [28]. We consider that providing one or multiple aspiration levels on the considered criteria may be a preferential information more easily provided by a DM, hence we focus on a preference model that is able to exploit this information in a meaningful way within an optimization process. Furthermore, we assume the criteria scales to be heterogeneous, therefore making it difficult to define tradeoffs between the evaluations of a given solution.

It is known that methods relying on pairwise comparisons to derive a ranking need to deal with Condorcet cycles in the outranking relation since a preference relation over alternatives resulting from a weighted majority of criteria are not necessarily transitive [20]. Most ranking methods based on outranking relations transform this relation into a transitive ranking by exploiting it [11], however, the Ranking with Multiple Points (RMP) method [27] does not compare alternatives one to each other, but to external profiles (as with Electre Tri [11]). As RMP imposes a dominance structure on the profiles, the outranking relation is guaranteed cycle-free, therefore providing a preorder on the set of alternatives. Hence, we propose to integrate the RMP model within multiobjective optimization, in particular for MOFJSP, in the form of an evolutionary population-based optimization approach using both the *a priori* and *a posteriori* strategies [33].

The paper is structured as follows. Section 2 presents the state of the art on the MOFJSP and on preference integration in multiobjective optimization in general. It is followed by Section 3 where the considered scheduling problem as well as the approach that will be used to solve it are presented. Section 4 then introduces the RMP model as well as the proposed approach of integrating it into the resolution method. Finally, we present the experimental protocol and several results in Section 5 before finishing with conclusions and perspectives for future work in Section 6.

2. State of the art

2.1. Multiobjective flexible job shop scheduling

The job shop scheduling problem (JSP) consists of assigning a number of jobs involving operations on a set of machines that are required to process them such that the operations of each job are sequenced in a given order and often trying to optimize a given objective such as, for instance, the makespan, i.e., the completion of all operations.

The flexible job shop scheduling problem (FJSP) extends the JSP by potentially allowing for multiple machines to handle a given operation of a job. The FJSP has a wide range of applications in several fields, such as environment, health, industry, etc. These multiple fields of application have attracted the interest of researchers to propose different approaches for its resolution. The FJSP is a strongly $\mathcal{NP}$-Hard problem even when reduced to two machines and a maximum of three operations per job [9]. The multiobjective FJSP (MOFJSP) has interested many researchers such as Kacem et al. [17] who combined evolutionary algorithms with fuzzy logic to solve the MOFJSP with the makespan, the total machine time, and the maximum machine workload as objectives to be minimized. Several other methods can also be found in [32]. Zhou et al. [38] introduce a multi-agent hyperheuristic based on several algorithms. This hyperheuristic integrates the prior knowledge of the workshop to evolve scheduling policies and aims to include domain knowledge in the algorithm to speed up the search process and improve the discovery of a solution. Their work is tested on real data from an aircraft engine blade manufacturing plant. Wu et al. [37] considered the problem of MOJSP and proposed a hybrid evolutionary algorithm, where the weighted sum of the following objective functions is taken into account: makespan, adjustment processing time, and total deviation. In [14], a general local search approach is suggested by García-León et al. for the multiobjective flexible scheduling problem (MOFJSP) to identify a Pareto front over any pattern of regular criteria.

2.2. Integrating DM preferences into multiobjective optimization

Multiple studies have focused on preference-based multiobjective evolutionary algorithms (MOEAs) [5, 24, 29, 34]. Based on these studies, the preference models may be classified into: goal specification, weights, utility functions, outranking methods, and fuzzy logic.

Goal specification consists of expressing the desired levels of the objectives, or aspiration levels, of a DM. [13] is among the earliest to incorporate DMs preferences into the search process of a MOEA by using a modified ranking scheme of the objectives in order to include goal information. This involved accommodating goal information as an additional criterion to the non-dominance principle used to rank the population. The main advantage of this method is its relative ease of use, as it does not require significant effort from the DM [12]. An interactive algorithm that incorporates the DMs preferences into an evolutionary approach based on achievement scalarization functions was introduced in [31].

Assigning weights to objectives can be used to reflect the relative importance of the objectives of a DM. A lexicographic order can be considered an extreme case of using weights, i.e., a more important objective is infinitely more important than a less important objective [35]. The best alternative is obtained by minimizing the objective functions in the given order. Furthermore, weights can also be viewed as a guidance direction, which is applied to find solutions along a particular search direction. For example, [7] linked this with the NSGA-II approach [8]. Weights can also be viewed as trade-offs, representing the extent to which the improvement on one (or several) objective can be compensated by a degradation on another (or several) objective. Inferring these weights when few objectives are considered may be easy and effective. However, this approach becomes more difficult to implement when the number of objectives increases. In [2] we find the G-MOEA method, where user preferences are taken into account by using the version of dominance proposed by Branke et al. [3]. A second version, called biased Crowding Distance, used a biased fitness sharing approach by adapting this distance to a weighted sum of lengths in the different dimensions of the objective space.

Utility functions are often difficult to provide due to the lack of knowledge of the problem. Wierzbicki presented in [36] a conceptual and mathematical model of the satisfaction of the decision-making process under multiple goals, in which information about the DMs preferences is expressed in the form of aspiration levels. The mathematical concept of a value (utility) function is modified to describe satisfaction behavior. In [25], Rădulescu et al. developed a taxonomy that classifies multi-agent multiobjective decision making contexts based on reward structures and on the way utility functions are applied. This mechanism is realized to provide a structured view of the field, clearly delineating the current state of the art of multi-agent multiobjec-

tive decision-making approaches, and identify promising directions for future research.

Outranking models have been integrated in a *a posteriori* way in [21]. The Pareto front is obtained in a first step, without external influences using a genetic algorithm, then the DMs preferences are taken into account to rank the solutions obtained using the PROMETHEE II method [6]. The same approach was applied in [23], but using a Gaussian preference and a global aggregation function in an NSGA-II algorithm, followed by the PROMETHEE II method, which is applied on the set of efficient solutions. Nevertheless, significant computational resources may be used in order to construct the set of efficient solutions, only to select one or a few using the PROMETHEE II method.

Fuzzy logic allows to integrate uncertainty in the parameter specification of a preference model. In [16], Jamwal et al. proposed an approach called 'Equitable Fuzzy Sorting Genetic Algorithm'. All criteria are defined as fuzzy objectives and the population is given a global activation score based on their respective fuzzy objective values. These scores are used to assign an explicit fuzzy dominance ranking to the population to enhance the sorting process.

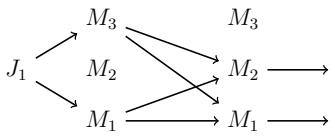
3. Solving the multiobjective flexible job shop problem

This section presents the problem that we aim to solve, using a small example to illustrate it, along with the general framework of the resolution method.

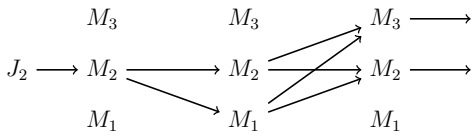
3.1. Problem description

The FJSP consists of assigning a number of jobs involving operations on a set of machines that are required to process them, where multiple machines are available to handle each operation, however, with different performance levels. We illustrate this problem using the small example shown in Figure 1.

Here we have two jobs consisting of three and respectively two operations to be performed on different subsets of three available machines (Figures 1a and 1b). Furthermore, the processing times for performing the same operation may differ based on the selected machine, as seen in Figure 1c. A possible schedule is provided in Figure 1d, where a single machine was selected for each operation, among those that could process them, and sequencing was performed on each machine. We observe that the duration of the project, i.e., its makespan, is equal to 6.



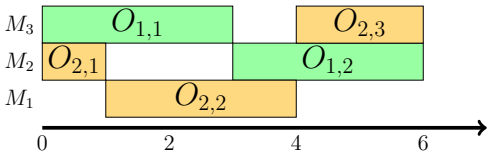
(a) First job possible operations sequences



(b) Second job possible operations sequences

Operation	M_1	M_2	M_3
$O_{1,1}$	2	-	3
$O_{1,2}$	4	3	-
$O_{2,1}$	-	1	-
$O_{2,2}$	3	2	-
$O_{2,3}$	-	3	2

(c) Processing times



(d) Feasible schedule

Figure 1: Illustration of a FJSP

In this work, we consider the MOFJSP with the following three objectives:

- f_1 : the makespan, or project duration;
- f_2 : the total machine processing time;
- f_3 : the balanced machine utilization.

The first objective is ensured by minimizing the finishing time of the last job and is mainly used in shipment planning. The second objective is given by the sum of the processing times of each operation on the machines that execute them, while the third objective is ensured by the minimization of the variance between the working time of each machine with respect to the average processing time of all machines. They are introduced to improve machine utilization and reduce the load on some machines at the expense of others.

3.2. Resolution approach

We recall that the FJSP involves a set of N jobs involving operations that need to be processed on a set of M machines. We denote with $O_{i,j}$ the j^{th} operation of job i, with $i = 1..N$ and $j = 1..N_i$. The operations of each

job have to follow fixed precedence constraints while each operation may potentially be processed on different machines. The processing time needed for each operation is also dependant on the selected machine and is denoted as $p_{i,j,k}, \forall i = 1..N, j = 1..N_i, k = 1..M$. For simplicity, we consider O to be the set of all operations of all jobs.

A complete solution s therefore consists of:

- $A = \{A_k, \forall k = 1..M\}$: the set of k disjoint subsets of operations ($\bigcup_{k \in 1..M} A_k = O$ and $\bigcap_{k=1..M} A_k = \emptyset$), where each subset $A_k \subseteq O$ contains the operations assigned to machine k;
- $C = \{c_{i,j}, \forall i \in 1..N, \forall j \in 1..N_i\}$: the list of completion dates of all operations.

We propose to solve this problem using a Multiobjective Genetic Algorithm (MOGA), whose main steps are presented in Algorithm 1. We consider an encoding of the previously defined solution s, or its chromosome, as a vector of machine assignments for all operations O. Decoding a solution involves solving an auxiliary problem, using mathematical programming in order to obtain the completion dates of all operations.

Algorithm 1: Used MOGA algorithm scheme

```
1 S ← INITIAL_POPULATION(pop_size)
2 F ← EVALUATE(S)
3 while stop condition not met do
4 |   S' ← REPRODUCE(S)
5 |   F' ← EVALUATE(S')
6 |   S ← SELECT(S ∪ S', F ∪ F')
7 return S
```

The MOGA algorithm begins by constructing a set of solutions as the initial population by randomly generating machine assignments (A) according to the machine compatibility of each operation.

Once the initial population is constructed, the solutions are decoded and evaluated on the three objectives. Objectives f_2 and f_3 are computed directly from the machine assignments, while f_1 is computed using the completion dates.

The main loop is repeated until a stopping condition is met. This condition may be a set number of iterations, a set amount of time or a convergence criterion linked to non-improvement of the best-found solutions across several iterations, to name a few.

Within the main loop of the algorithm, individuals are selected and used to construct new solutions using crossover and mutation operators. In our case, pairs of individuals are selected using a roulette mechanism based on the NSGA-II criterion [8], where solutions are sorted based on the dominance principle. From each pair of parents, two children are generated using the uniform crossover operator. This involves generating a uniformly distributed binary vector of size equal to the number of operations. The first (resp. second) child receives the machine assignments of the first (resp. second) parent for operations corresponding to a 1 in the binary vector and the machine assignments of the second (resp. first) parent for the remaining operations. Both children then go through a non-uniform mutation operator, where each machine assignment is changed based on a given probability parameter.

The new individuals are then evaluated on the three criteria, followed by a selection mechanism where both the old and new individuals are selected to form the new population. We use the same approach as the NSGA-II method [8], using an elitist scheme based on the non-dominance sorting criterion and the crowding distance, which is used in order to distance the solutions from each front apart.

In the end, the entire set of solutions is given, which may then be filtered further, using, for instance, a model of the DMs preferences. We will cover this part, as well as how to integrate this model within the MOGA approach in the following sub-section.

We can notice that the last two criteria (f_2 and f_3) depend only on the assignments A_k. To compute the value of the first criterion, we solve the following auxiliary problem:

$$
\begin{aligned}
\min \quad & C_{max} && && (1)\\
\text{s.t.} \quad & C_{ij} - P_{ijk} \geq C_{i'j'} - X_{ghk}L && \forall k = 1..M, \forall g > h \in 1..|A_k| \\
& && A_{kg} = O_{ij}, A_{kh} = O_{i'j'}, i \neq i' && (2)\\
& C_{i'j'} - P_{i'j'k} \geq C_{ij} - (1 - X_{ghk})L && \forall k = 1..M, \forall g > h \in 1..|A_k| \\
& && A_{kg} = O_{ij}, A_{kh} = O_{i'j'}, i \neq i' && (3)\\
& C_{i1} - P_{i1k} \geq 0 && \forall i = 1..N
\end{aligned}
$$

$$\text{with } k \in 1..M \text{ s.t. } O_{i1} \in A_k \quad (4)$$

$$C_{ij} - P_{ijk} \geq C_{i(j-1)} \qquad \forall i = 1..N, \forall j = 2..N_i$$

$$\text{with } k \in 1..M \text{ s.t. } O_{ij} \in A_k \quad (5)$$

$$C_{max} \geq C_{iN_i} \qquad \forall i = 1..N \quad (6)$$

The parameters of this model are:

- P_{ijk}, the processing time of the j^{th} operation of job i on machine k, $\forall i = 1..N, j = 1..N_i, k = 1..M$;
- A_k, the sets of subsets of operations assigned to each machine k, $\forall k = 1..M$; We use an additional index in order to denote a particular element from this set (e.g. the g^{th} element from A_k will be denoted as A_{kg});
- L, a large constant.

The variables of the model are:

- $C_{ij} \geq 0$, the complete time of the j^{th} operation of job i, $\forall i = 1..N, j = 1..N_i$;
- $C_{max} \geq 0$, the makespan, i.e. the complete time of the last operation of any job;
- $X_{ghk} = 1$ if the g^{th} operation from A_k is executed after the h^{th} operation, and 0 otherwise, $\forall g = 1..|A_k| - 1, h = g + 1..|A_k|, k = 1..M$.

The objective function in (1) seeks to minimize the makespan (the completion time of the last operation). Constraints (2) and (3) avoid overlapping of operations assigned to the same machine. Constraint (2) imposes that the the j^{th} operation of task i starts after the j'^{th} operation of task i' finishes when $X_{ghk} = 0$ while constraint (3) imposes the reverse when $X_{ghk} = 1$. Since the machine assignments are known, these constraints are expressed only when the two previously mentioned operations are assigned to the same machine k, i.e. they are both in A_k. Note that the indexes g and h are not indicative of the order in which the operations from A_k are executed on the k^{th} machine and are used only to define the pair of constraints once for each pair of operations within this set. Furthermore, these constraints are not

considered when operations of the same task are assigned to the same machine, since their execution order is dictated by constraints (5). Constraints (4) ensure that the first operation of each job finishes after its execution time while constraints (5) impose the precedence constraints linked to the operations of each task. Finally, constraint (6) is used to compute the makespan which corresponds to the largest completion time of the last operations of any task.

4. Integrating preferences within the optimization process

We consider here that a DM may be involved in the optimization process and that a model of his/her preferences may be constructed. We assume that this DM may be able to express multiple aspiration levels for the expected scheduling solution and that a non-compensatory preference model would be best suited. We propose to use the relatively recent RMP method [27], which is described below, and integrate it within the resolution algorithm using both an *a priori* and an a *posteriori* approach.

4.1. Ranking using reference profiles

We consider a finite set of alternatives $\mathcal{A}$ evaluated on a set of criteria, where we denote with $J = \{1, 2, \dots, j, \dots\}$ the set of criteria indices, and with $f_j(a)$ the performance of alternative $a \in \mathcal{A}$ on criterion j. We consider, without loss of generality, that preferences decrease with the evaluation on each criterion, i.e. the lower the better, so that an alternative may correspond to a solution to an optimization problem where the objectives, or criteria, are to be minimized. The RMP method uses three different types of parameters:

- $\mathcal{P} = \{p^h, h = 1, \dots, k\}$ a set of k reference profiles, with $p^h = \{p^h_j, \forall j \in J\}$, where p^h_j denotes the evaluation of profile p^h on criterion j; furthermore, each subsequent profile dominates the previous ones on all criteria, i.e, $p^{h+1}_j \leqslant p^h_j$, $\forall h = 1..k-1$, $\forall j \in J$;
- σ, a lexicographic order on the reference profiles, i.e., a permutation on $\{1, \dots, k\}$;
- criteria weights $w_j, \forall j \in J$, where $w_j \geq 0$ and $\sum_{j \in J} w_j = 1$

Each reference profile $p^h \in \mathcal{P}$ corresponds to a set of aspiration levels of the DM on all considered criteria. The entire set of reference profiles $\mathcal{P}$

furthermore corresponds to multiple sets of such aspiration levels, ordered with respect to the preferences of the DM, i.e. the first profile corresponds to the lowest aspiration levels on all criteria, the second profile corresponds to the second lowest aspiration levels on all criteria, and so on, up to the last profile which corresponds to the highest aspiration levels on all criteria.

RMP first identifies the criteria on which any alternative $a \in \mathcal{A}$ is at least as good as a profile $p^h \in \mathcal{P}$ as $C(a, p^h) = \{j \in J : f_j(a) \leqslant p_j^h\}$. Any two alternatives $a, b \in \mathcal{A}$ can then be compared according to each reference profile through a preference relation $\succsim_{p^h}$, such that $a \succsim_{p^h} b$ iff $\sum_{j \in C(a,p^h)} w_j \geq \sum_{j \in C(b,p^h)} w_j$. This means that a is at least as good as b according to profile p^h if a compares to p^h as well or better than b does. We will denote with $\succ_{p^h}$ the asymmetric part of this relation and with $\sim_{p^h}$, the symmetric part. The two alternatives are then ranked by sequentially considering the relations $\succsim_{p^{\sigma(1)}}, \succsim_{p^{\sigma(2)}}, \ldots, \succsim_{p^{\sigma(k)}}$. a is preferred to b if $a \succ_{p^{\sigma(1)}} b$, or if $a \sim_{p^{\sigma(1)}} b$ and $a \succ_{p^{\sigma(2)}} b$, or $\ldots$ Hence, a and b are indifferent iff $a \sim_{p^{\sigma(h)}} b$, for all $h = 1, \ldots, k$.

Let us consider a small illustrative example depicted in Figure 2.

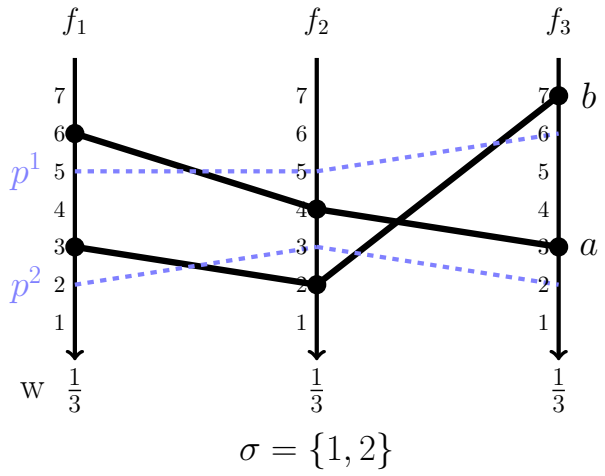


Figure 2: Illustration of an RMP model and two alternatives

Here we have a problem with three criteria to be minimized and an RMP model containing two reference profiles, p^1 and p^2, equal criteria weights, $w_1 = w_2 = w_3 = \frac{1}{3}$, and a lexicographic order consisting of p^1 followed by p^2. We also consider two alternatives, a and b, that we compare using the RMP procedure. We start by comparing these alternatives to p^1 and we

observe that $a \sim_{p^1} b$ since they are both at least as good as p^1 on two out of the three criteria. We therefore pass to the second profile according to σ. Here we observe that $b \succ_{p^2} a$ since b is at least as good as p^2 on the second criterion, while a is not at least as good as p^2 on any criterion. Hence $b \succ a$.

Using this approach, an entire set of alternatives $\mathcal{A}$ may be compared sequentially or through dichotomy in order to construct a preorder. In our particular case, these alternatives correspond to solutions to the MOFJSP, and their evaluations are the three considered criteria, although other criteria and even other optimization problems may easily be considered.

4.2. Preference model integration

As previously mentioned, we focus on the *a posteriori* and *a priori* integration of an RMP preference model within the optimization of a MOFJSP. Both of these approaches assume that one or multiple sessions are carried out with a DM in order to construct, either directly or indirectly, an RMP model that closely matches their perspective on the optimization problem. In the *a posteriori* approach, this step occurs after the optimization approach has constructed a set of efficient solutions, while in the *a priori* approach, this process occurs before. We do not focus on the preference model construction step in this work; the interested reader may refer to [22] for an exact approach for inferring an RMP model indirectly, while the work in [18] illustrates an incremental inference strategy.

The *a posteriori* approach is straightforward, as the NSGA-II algorithm is first used to construct the set of efficient solutions, then, following the inference of the RMP model, these solutions are ordered according to it, and the top solutions are proposed to the DM.

The *a priori* approach adapts the MOGA algorithm by integrating the RMP model in the evaluation of the solutions of each generation. We denote this approach as MOGA-RMP.

The RMP model outputs an order, which is perfectly compatible with the proposed approaches, however, one may also consider the RMP model as an ordered classification approach, similar to the MR-Sort approach [19]. The main difference consists in the fact that RMP constructs $(k+1)^{|J|}$ categories instead of the $k+1$ categories of MR-Sort. We illustrate this using Figure 3.

We observe a two-criterion problem and illustrate multiple potential RMP models with two profiles and the ordering of regions they induce. Considering that the four models have the same two profiles, we show how the order in which these profiles are used (σ) and how the order on the criteria weights (w)

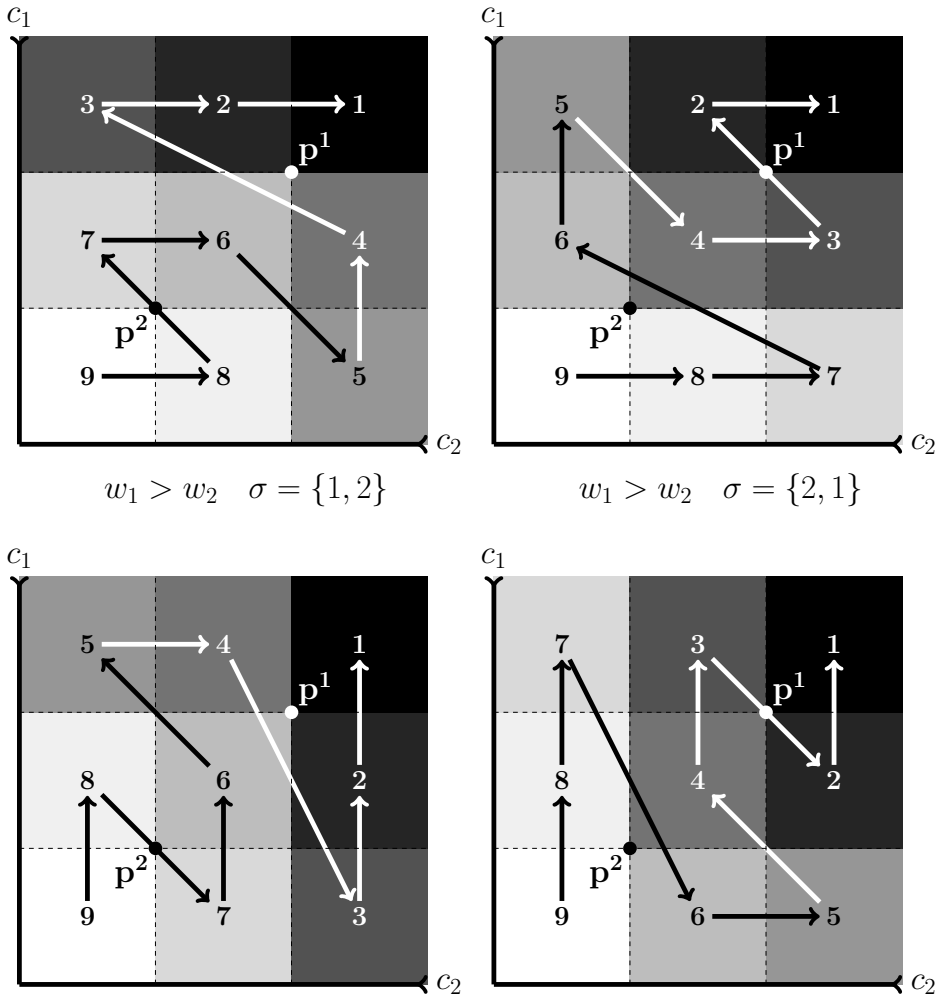


Figure 3: Several potential RMP ordered classification regions for 2 criteria and 2 profiles

influence the order on the regions. We find a total of 9 categories, with the best category denoted with 9 and the worst category denoted with 1. We have also depicted the preference order on these categories with straight arrows. Any solution that falls within a rectangular area defined by the two profiles will be indifferent to other solutions from the same area. This is due to the fact that all solutions from an area surpass the same aspiration levels, which are represented by the profiles evaluations. The order of the categories can be easily computed by generating an alternative for each area and then ranking them using the RMP procedure. The ranks of the alternatives become the ranks associated to their corresponding regions.

Let us take the second model as an example and detail in a more intuitive way how the regions induced by the profiles are ordered. We will denote a region by a tuple corresponding to the index of the interval of values induced by the two profiles on each criterion, i.e. the white region corresponds to $(1, 1)$, the region to its right corresponds to $(2, 1)$ and so on. Since $\sigma = \{2, 1\}$, we first use profile p^2 to partition the 9 regions leading to the following ordered partition $\{(1, 1)\} \succ \{(2, 1), (3, 1)\} \succ \{(1, 2), (1, 3)\} \succ \{(2, 2), (2, 3), (3, 2), (3, 3)\}$. The first partition contains regions that outrank p^2 on both criteria, followed by those outranking the profile on only the first criterion, then those outranking the profile on only the second criterion (since $w_1 > w_2$), and concluding with those not outranking the profile on any criterion. Profile p^1 is then used to partition each of these sets even further. The first partition only contains one region so it cannot be split further. The second partition can be split based on the second criterion, hence $\{(1, 2), (1, 3)\}$ becomes $\{(1, 2)\} \succ \{(1, 3)\}$. Similarly, partition $\{(2, 1), (3, 1)\}$ becomes $\{(2, 1)\} \succ \{(3, 1)\}$ based on how the two regions outrank or not p^1 on the first criterion. Finally, the last partition, $\{(2, 2), (2, 3), (3, 2), (3, 3)\}$ is split into four since region $(2, 2)$ outranks profile p^1 on both criteria, $(3, 2)$ outranks the profile on the first criterion only, $(2, 3)$ outranks the profile on the second criterion only, while $(3, 3)$ does not outrank the profile on any criterion. We are left with the following ordering on the regions $(1, 1) \succ (2, 1) \succ (3, 1) \succ (1, 2) \succ (1, 3) \succ (2, 2) \succ (3, 2) \succ (2, 3) \succ (3, 3)$. This corresponds to the order we see in the second image from Figure 3.

Within our RMP implementation, instead of constructing an order by comparing solutions pair-wisely, we construct a classification model based on how each solution compares to the profiles of the RMP model, hence rendering the ordering process linear with the number of solutions. The values linked to each area are computed at the start of the MOGA-RMP

algorithm and are normalized within a $[0, 1]$ interval simply by dividing the category index by the total number of categories. We will call this the RMP score and denote it using $RMP(s)$.

We also define $\Delta(s) = \min_{j \in J} \min_{\substack{h \in \{1,\ldots,k\} \\ p_j^{h+1} < f_j(s)}} \frac{f_j(s) - p_j^{h+1}}{p_j^h - p_j^{h+1}}$, which corresponds to the smallest improvement on any criterion between a solution and any reference level, normalized with respect to the two nearest bounding reference levels. This measure is useful in determining how close a solution is to surpassing any reference profile and thus move to an area corresponding to a better RMP score.

Finally, $\Delta RMP(s)$ is a normalized measure corresponding to the improvement in RMP score if solution s were to improve and surpass the closest better reference level.

The MOGA-RMP approach integrates these measures within two steps: the reproduction and the population selection. Let us denote with $\mathcal{P}$ the population of solutions of the genetic algorithm at a given time and with $NSGA(s)$ the NSGA-II criterion.

In the reproduction phase, we use a roulette selection where the probability of selecting a *parent* (s) depends not only on the NSGA-II criterion but also on the three previously defined measures. We combine these three measures into a secondary criterion as follows:

$$RMP^*(s) = \frac{RMP(s) + \Delta RMP(s) + (1 - \Delta(s))}{3}$$

The overall fitness of the parent then becomes:

$$\left(\frac{\max_{s' \in \mathcal{P}} NSGA(s') - NSGA(s) + RMP^*(s)}{\max_{s' \in \mathcal{P}} NSGA(s') + 1} \right)^{\alpha}$$

This normalized fitness value is then used in conjunction with a random variable drawn from a uniform distribution in order to select the parent if the variable is less or equal to the fitness of the individual. Therefore, higher fitness values correspond to higher probabilities of selecting an individual. In other words, the NSGA-II criterion is still used a the primary selection criterion, while the three previously mentioned measures are used as secondary criteria of equal importance. Higher values of the α parameter also shift the

probability of selection in favor of better individuals since small fitness values (in the $[0, 1]$ interval) are further reduced for higher values of α.

In the population selection phase, the NSGA-II criterion is first used to group individuals based on the dominance principle. The groups are then considered one by one, starting with the top one. If a group is smaller than the remaining spots needed by the new population then all of its individuals are selected. In the opposite case, individuals are further grouped based on their RMP scores and a number of individuals from each group, proportional to their RMP score, are selected. When a group contains more individuals than the required proportion, the crowding distance is further used to discriminate among them. In the opposite case, all individuals from the group are selected and the entire process is performed an additional time using the groups still containing unselected individuals.

5. Experimental validation

In order to test the proposed approaches, we follow the experimental protocol presented in Figure 4.

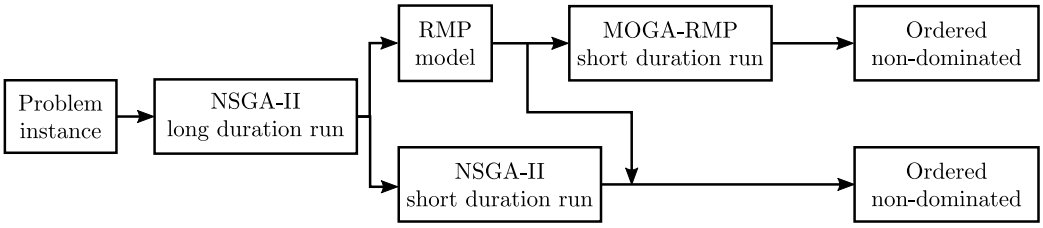


Figure 4: Experimental design

For each problem instance that we test, we begin by executing an NSGA-II algorithm. This approach is allowed to run for a long period of time so that a set of efficient solutions as close as possible to the Pareto front can be found.

We then use the generated solutions in order to construct an RMP model, since we need to simulate a DM. We do this so that the reference profiles of the RMP model correspond to realistic aspiration levels, which no solution may completely satisfy. We base this on the fact that a real DM would express very demanding aspiration levels, otherwise, if solutions that satisfy all aspiration levels are feasible, the problem would be easy to solve. We

will describe the process of generating the RMP model in the following subsection.

Once the RMP model is generated, both approaches (NSGA-II and MOGA-RMP) are executed for a shorter duration, up to one hour in particular. The NSGA-II approach uses the RMP model only at the end, whereas the MOGA-RMP approach requires it before starting. The results consist of a set of non-dominated solutions that are further ordered based on the RMP model. Comparing these solutions allows us then to compare the efficiency of the two approaches.

5.1. Benchmarks

To compare these approaches we have adapted 2 instances from [17] (denoted as K3 and K4), two instances from [10] (denoted as F19 and F20) amd two instances from [1] (denoted as B1 and B2). Some of these instances correspond to problems considered as medium (K3, F19 and F20) and large (K4, B1, B2). Moreover, the instances from [17] are known for their total flexibility (all operations can be processed on any machine) and the instances from [3] for their partial flexibility (operations can be processed on multiple machines but not all of them). Table 1 summarizes the characteristics of the instances used here.

Instance	K3	F19	F20	K4	B1	B2
# jobs	10	11	12	15	10	10
# machines	10	8	8	10	6	6
# operations per job	3	4	4	[2∼4]	[5∼7]	[5∼7]
Flexibility	total	partial	partial	total	partial	partial

Table 1: Characteristics of tested instances

As presented previously in the experimental design, each of these instances are first solved using the NSGA-II for a long period of time and with plenty of computational resources in order to construct a set of efficient solutions that are as close as possible to the Pareto front. We denote this set as $\mathcal{S}$. Using the solutions in $\mathcal{S}$, we then randomly generate the reference profiles of an RMP model. We use the ideal point (minimum objective values from $\mathcal{S}$), the closest solution from $\mathcal{S}$ w.r.t. the ideal point and finally the Nadir point (maximum objective values for $\mathcal{S}$) in order to partition each evaluation

scale (the three considered objective functions) into two ordered intervals intervals (values between the ideal point and the mid-point and values between the mid-point and the Nadir point). Two profiles are generated by randomly sampling the first interval for each evaluation scale and also making sure that one profile dominates the other, while three profiles are generated in a similar way using the second interval for each evaluation scale. The way in which the first two profiles are generated makes it so that it is very unlikely to find a solution satisfying all aspiration levels. This scenario corresponds to the RMP model representing a demanding albeit fictitious DM. The lexicographic order of the model is generated as a random permutation of the 5 profiles, while the criteria weights are generated using the method from [4].

5.2. Results

All the approaches have been implemented in Julia 1.6 and executed on an Intel Xeon Gold with 80 cores at 2.00 GHz and 64 GB of RAM. Each execution used a single thread, except for the mixed-integer linear program which was solved using the CPLEX v0.7.8 solver [15] and up to 4 threads. This decision was made in order to have the results correspond to those one would get using a standard personal computer available today.

Figure 5 illustrates the results following 50 executions of the classical NSGA-II approach together with an *a posteriori* integration of the RMP model and the approach we propose on the medium sized instances. On the left, we indicate the average normalized gap between the best category in a given solution population and the overall best found category for all algorithm executions. The category is computed according to the corresponding DM preference model for each execution. On the left side of the figure, we indicate the average number of solutions corresponding to the best category w.r.t. to the DM preference model as a function of time.

We notice that the MOGA-RMP approach is able to provide solutions with better RMP scores than those proposed by the NSGA approach, therefore corresponding to solutions perceived as better by the DM. Any difference in the RMP score corresponds to one or multiple levels of aspiration being surpassed by the corresponding solutions. MOGA-RMP also generally provides a larger number of such solutions. In the case of the F19 problem instance, the algorithm proposes fewer solutions than the NSGA approach, however, it should be noted that these solutions provide better performance based on the DM model as can be seen on the corresponding figure on the left.

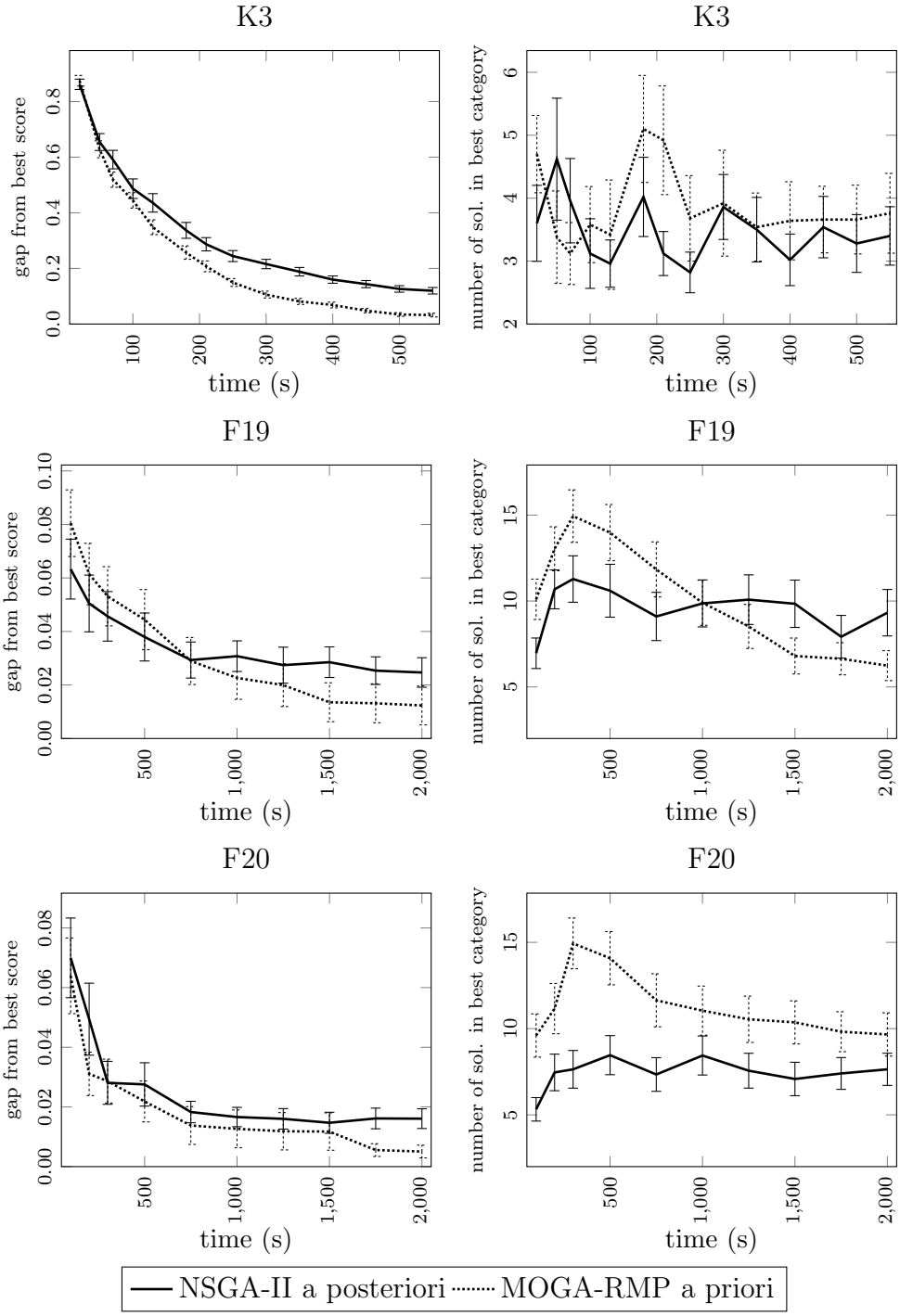


Figure 5: Gap from best RMP score (left) and number of solutions in best category (right) for medium instances

It should also be noted that, using the MOGA-RMP approach, we do not need to spend a lot of time or computational resources as NSGA-II does in order to propose solution of equivalent quality w.r.t. the DM preferences. For instance, for K3, MOGA-RMP can propose a solution within a category 20% worse than the best found solution after 200 seconds while NSGA-II can propose an equivalent solution after more than 300 seconds.

The previous remarks are also valid when considering the larger instances in Figure 6. Here the differences in solution quality are more pronounced on the K4 instance and slightly less on the B1 and B2 instances, however, in these cases, more solutions of such quality are proposed by the MOGA-RMP method, up to twice as many in some cases.

We can also notice that on the K4 problem instance, NSGA-II is not able to find a solution with a gap of 20% from the best found solution within 5000 seconds, while MOGA-RMP finds such a solution after less than 2000 seconds. Such gains can prove very important when a solution needs to be implemented in practice within a limited amount of time.

6. Conclusions and perspectives

In this work, we consider and compare the *a priori* and *a posteriori* integration of a non-compensatory preference model based on reference profiles within multiobjective optimization and illustrate it using the flexible job-shop scheduling problem. A multiobjective genetic algorithm is proposed and tested on state of the art problems from the literature. The results show that integrating a preference model within the optimization approach helps in providing better solutions faster as well as in greater numbers. Having more solutions of equivalent quality from the perspective of the DM may be helpful in integrating additional and more complex constraints that were not initially considered, thus increasing the chance of finding an operational solution.

As future perspectives, we will integrate and test additional operators within the genetic algorithm and improve its performance by either limiting the use of the mathematical program via heuristics or integrating it completely within a metaheuristic approach. Our current objective was not to propose solutions that perform better in terms of optimization w.r.t. to existing approaches, but mainly illustrate the gain that integrating a preference model could yield. We also intend to explore single solution metaheuristics, such as simulated annealing or tabu search, to give a couple of examples.

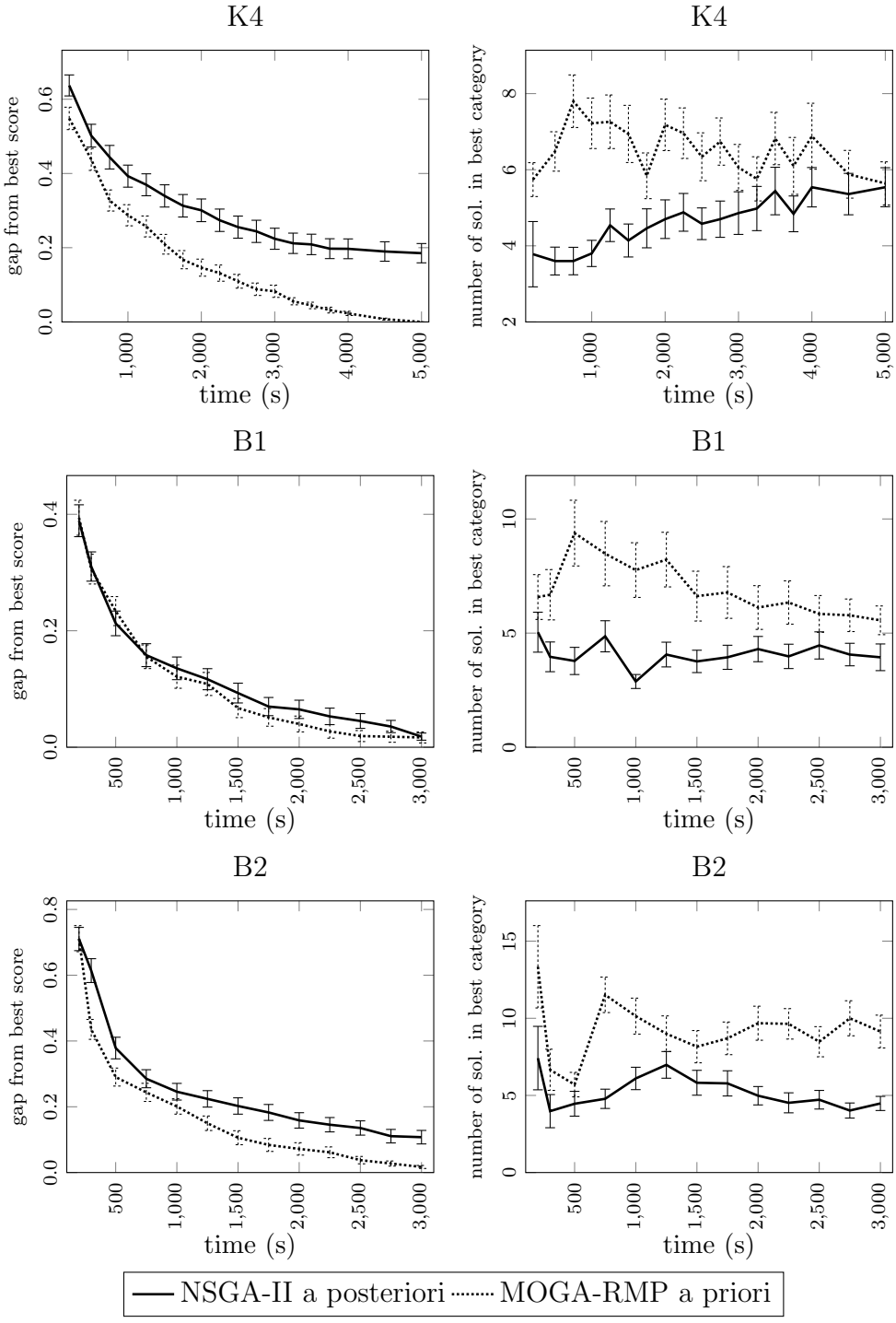


Figure 6: Gap from best RMP score (left) and number of solutions in best category (right) for large instances

Finally, we will further explore integrating the preference elicitation phase within the optimization phase in an interactive manner when dealing with difficult problems where such an interaction is possible.

Acknowledgment

This work has been partly supported by the Brittany Region in France, under the SAD program, reference SAD19046.

References

[1] P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.

[2] J. Branke and K. Deb. Integrating user preferences into evolutionary multi-objective optimization. In *Knowledge incorporation in evolutionary computation*, pages 461–477. Springer, 2005.

[3] J. Branke, T. Kaußler, and H. Schmeck. Guidance in evolutionary multi-objective optimization. *Advances in engineering software*, 32(6):499–507, 2001.

[4] J. Butler, J. Jia, and J. Dyer. Simulation techniques for the sensitivity analysis of multi-criteria decision models. *European Journal of Operational Research*, 103:531–546, 1997.

[5] C. A. C. Coello. Handling preferences in evolutionary multiobjective optimization: A survey. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, volume 1, pages 30–37. IEEE, 2000.

[6] W. De Keyser and P. Peeters. A note on the use of promethee multicriteria methods. *European Journal of Operational Research*, 89(3):457–461, 1996.

[7] K. Deb and A. Kumar. Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 781–788, 2007.

[8] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[9] Y. Demir and S. K. İşleyen. Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling*, 37(3):977–988, 2013.

[10] Parviz Fattahi, Mohammad Saidi Mehrabad, and Fariborz Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of intelligent manufacturing*, 18(3):331–342, 2007.

[11] J. Figueira, V. Mousseau, and B. Roy. ELECTRE methods. In J. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 133–162. Springer Verlag, Boston, Dordrecht, London, 2005.

[12] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 28(1):26–37, 1998.

[13] C. M. Fonseca, P. J. Fleming, et al. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In *Icga*, volume 93, pages 416–423. Citeseer, 1993.

[14] A. A. García-León, S. Dauzère-Pérès, and Y. Mati. An efficient pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria. *Computers & Operations Research*, 108:187–200, 2019.

[15] IBM. IBM CPLEX Optimizer, 2021.

[16] P. K. Jamwal, B. Abdikenov, and S. Hussain. Evolutionary optimization using equitable fuzzy sorting genetic algorithm (efsga). *IEEE Access*, 7:8111–8126, 2019.

[17] I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(1):1–13, 2002.

[18] A. Khannoussi, A-L. Olteanu, C. Labreuche, and P. Meyer. Simple ranking method using reference profiles: incremental elicitation of the preference parameters. *4OR: A Quarterly Journal of Operations Research*, July 2021.

[19] A. Leroy, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method. In R. Brafman, F. Roberts, and A. Tsoukiàs, editors, *Algorithmic Decision Theory*, volume 6992, pages 219–233. Springer, 2011.

[20] M. M. Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix.* Paris, 1785.

[21] S. Massebeuf, C. Fonteix, L. N. Kiss, I. Marc, F. Pla, and K. Zaras. Multicriteria optimization and decision engineering of an extrusion process aided by a diploid genetic algorithm. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 1, pages 14–21. IEEE, 1999.

[22] A-L. Olteanu, K. Belahcène, V. Mousseau, W. Ouerdane, A. Rolland, and J. Zheng. Preference elicitation for a ranking method based on multiple reference profiles. *4OR*, pages 1–22, 2021.

[23] R. O. Parreiras and J. A. Vasconcelos. Decision making in multiobjective optimization aided by the multicriteria tournament decision method. *Nonlinear Analysis: Theory, Methods & Applications*, 71(12):e191–e198, 2009.

[24] L. Rachmawati and D. Srinivasan. Preference incorporation in multiobjective evolutionary algorithms: A survey. In *2006 IEEE International Conference on Evolutionary Computation*, pages 962–968. IEEE, 2006.

[25] R. Rădulescu, P. Mannion, D. M. Roijers, and A. Nowé. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–52, 2020.

[26] H. Raiffa and R. L. Keeney. Decision analysis with multiple conflicting objectives, preferences and value tradeoffs. 1975.

[27] A. Rolland. Reference-based preferences aggregation procedures in multi-criteria decision making. *European Journal of Operational Research*, 225(3):479–486, 2013.

[28] B. Roy. The outranking approach and the foundations of ELECTRE methods. *Theory and Decision*, 31:49–73, 1991.

[29] L. B. Said, S. Bechikh, and K. Ghédira. The r-dominance: a new dominance relation for interactive evolutionary multicriteria decision making. *IEEE transactions on Evolutionary Computation*, 14(5):801–818, 2010.

[30] R. E. Steuer, Y. Qi, and M. Hirschberger. Multiple criteria decision making. *John Whiley*, 1986.

[31] L. Thiele, K. Miettinen, P. J. Korhonen, and J. Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary computation*, 17(3):411–436, 2009.

[32] A. Türkyılmaz, Ö. Şenvar, İ. Ünal, and S. Bulkan. A research survey: heuristic approaches for solving multi objective flexible job shop problems. *Journal of Intelligent Manufacturing*, pages 1–35, 2020.

[33] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

[34] H. Wang, M. Olhofer, and Y. Jin. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. *Complex & Intelligent Systems*, 3(4):233–245, 2017.

[35] Rui Wang, Robin C Purshouse, and Peter J Fleming. Preference-inspired coevolutionary algorithms for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 17(4):474–494, 2012.

[36] A. P. Wierzbicki. A mathematical basis for satisficing decision making. *Mathematical modelling*, 3(5):391–405, 1982.

[37] R. Wu, Y. Li, S. Guo, and X. Li. An efficient meta-heuristic for multi-objective flexible job shop inverse scheduling problem. *IEEE Access*, 6:59515–59527, 2018.

[38] Y. Zhou, J-J. Yang, and L-Y. Zheng. Multi-agent based hyper-heuristics for multi-objective flexible job shop scheduling: A case study in an aero-engine blade manufacturing plant. *Ieee Access*, 7:21147–21176, 2019.