An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria

Andrés Alberto García-León, Stéphane Dauzere-Peres, Yazid Mati

Version of Record: https://www.sciencedirect.com/science/article/pii/S0305054819300954
Manuscript_9cddd2d654176beb7bb2cef84abdf023

An Efficient Pareto Approach for Solving the Multi-Objective Flexible Job-shop Scheduling Problem with Regular Criteria

Andrés Alberto García-León[a,b,*], Stéphane Dauzère-Pérès[b,c], Yazid Mati[d]

[a] *Programa de Ingeniería Industrial, Universidad de Ibagué, Ibagué, Colombia*
[b] *Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, CMP, Department of Manufacturing Sciences and Logistics, Gardanne, France*
[c] *Department of Accounting, Auditing and Business Analytics, BI Norwegian Business School, Oslo, Norway*
[d] *College of Business and Economics, Qassim University, Saudi Arabia*

Abstract

In this paper, a general local search approach for the Multi-Objective Flexible Job-shop Scheduling Problem (MOFJSP) is proposed to determine a Pareto front for any combination of regular criteria. The approach is based on a disjunctive graph, a fast estimation function to evaluate moves and a hierarchical test to efficiently control the set of non-dominated solutions. Four search strategies using two neighborhood structures are developed. Numerical experiments are conducted on test instances of the literature with three sets of criteria to minimize and using metrics to evaluate and compare Pareto fronts. The results show that our approach provides sets of non-dominated solutions of good quality.

Keywords: Flexible job-shop scheduling, Multi-objective, Regular criteria, Pareto optimization, Local search

1. Introduction

In a strong competitive environment and with more and more demanding customers, manufacturing and service systems must be flexible (Johnzen et al. (2011)) and able to deal with different objectives dynamically. Among

*Corresponding author

the important operational decisions are scheduling decisions where several types of flexibility have been considered, and in particular operation flexibility that refers to the ability of an operation to be performed in different ways. The scheduling literature dealing with operation flexibility in the classical Job-shop Scheduling Problem (JSP) has accumulated over the last twenty five years (see the recent survey of Chaudhry and Khan (2016)), leading to the so-called Flexible Job-shop Scheduling Problem (FJSP). The FJSP is more realistic for modeling a wide range of real-life applications, as it can capture key features of modern manufacturing and service systems.

The Flexible Job-shop Scheduling Problem (FJSP) is defined as follows. A set $\mathcal{M}$ of m machines are always available to process a set of n jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$. Each machine can only perform one operation at a time. Each job consists of a sequence of operations, called routing, which can differ from one job to another, i.e. there is not a single pre-specified order of machines for all jobs. The preemption of operations is not allowed, i.e. an operation cannot be interrupted once started. Each job J_i has a release date r_i, a weight w_i related to the priority of job J_i, and a due date d_i that specifies the date before which J_i should be completed. An important feature of the FJSP is that the machine needed to perform an operation j is not given but must be selected from a subset $\mathcal{R}_j \subseteq \mathcal{M}$ of eligible machines. The processing time p_j of an operation j depends on the selected machine in $\mathcal{R}_j$. Let us assume that these processing times are non-negative integer, known and include non sequence-dependent setup times between operations. The FJSP consists in both assigning a machine to each operation and sequencing operations on the selected machines, to optimize a single criterion or multiple criteria.

In single criterion optimization, the most studied criterion for the FJSP is the minimization of the makespan $C_{\max}$, which corresponds to the completion time of all jobs. However, minimizing other criteria that include the weight of jobs and their due dates are better suited to capture critical factors that affect the competitiveness of a firm (Zhang and Wu (2011)). Regular criteria are among the most common objectives considered in the scheduling literature. A criterion is said to be regular if it is an increasing function of the completion times of the jobs (see e.g. Mati et al. (2011) for the JSP). In addition to the makespan, the following regular criteria are among the most popular ones in the scheduling literature: (1) Maximum tardiness $T_{\max} = \max T_i$, where $T_i = \max(0, C_i - d_i)$, C_i is the completion time of job J_i and d_i is its due date, (2) Total tardiness $\sum_i T_i$, (3) Total completion time $\sum_i C_i$, and (4)

Number of tardy jobs $\sum_i U_i$, where $U_i = 1$ if $T_i > 0$ and 0 otherwise.

The Multi-Objective Flexible Job-shop Scheduling Problem (MOFJSP) is the optimization of the FJSP with multiple criteria that are in conflict to some extent. In this paper, we develop a general local search approach that optimizes any combination of regular criteria for the MOFJSP. The remainder of the paper is organized as follows. Section 2 reviews the related literature. Section 3 details the disjunctive graph model and the neighborhood structures that are used to solve the FJSP. Section 4 proposes a theoretical framework to evaluate the quality of the set of non-dominated solutions. Section 5 describes the proposed local search approach based on Pareto optimization with four new search strategies. Experiments that validate the efficiency of our approach are presented and discussed in Section 6. Finally, Section 7 concludes the paper and provides some directions for future research.

2. Literature review

The FJSP has been extensively studied in the literature to optimize a single criterion or multiple criteria. A recent survey covering the various techniques to solve the FJSP with a single objective and multiple objectives, can be found in Chaudhry and Khan (2016). This survey includes different comparative tables to classify the literature according to the performance measures and the types of techniques. It also gives, for each paper, the algorithm and shop details, the objective functions considered and the number of citations. Another survey can be found in Genova et al. (2015) that only covers techniques developed to solve the MOFJSP between 2005 and 2014. A recent literature review on genetic algorithms to solve the FJSP is presented in Amjad et al. (2018) where, for each paper, the considered objective functions, the parameters of the genetic algorithms and the benchmarks are presented.

By taking a closer look at the literature on the MOFJSP, one can observe that most papers focus on optimizing the makespan, the total workload of machines, and the workload of the critical machine. Chaudhry and Khan (2016) report (Table 3) that the makespan is used in combination with another objective function in 39.59% of papers, and 23.35% of them use the workload of machines. Even though the list of papers in Chaudhry and Khan (2016) is not exhaustive as they missed some papers, the observation and trend are the same. The makespan remains the most studied criterion

for the FJSP, and is generally combined with the workload of machines in the MOFJSP. A similar observation can be drawn from Amjad et al. (2018) (Table 10) where 88.88% (i.e. 32 out of 36) of papers relying on genetic algorithms to solve the MOFJSP consider the makespan and workload of machines. Since there are a very large number of papers on the MOFJSP, we only focus in this section on the recent, state-of-the-art and closely related works where multiple criteria are optimized, with special attention to papers that consider regular criteria.

The MOFJSP is usually tackled in the literature using two types of approaches. The first one consists in transforming the multi-objective problem into a mono-objective problem by assigning different weights for each objective. Various heuristics in this category were proposed in the literature. A Tabu Search algorithm is presented in Li et al. (2014) that uses several neighborhood search rules for machine assignment and operation scheduling. A heuristic method that starts from an initial solution, and improves it using two move search algorithms, is introduced in Xing et al. (2009). Several hybrid heuristics are proposed such as the hybridization of particle swarm optimization and Tabu Search in Zhang et al. (2009), genetic algorithms and Shifting Bottleneck in Gao et al. (2007), and Particle Swarm Optimization and Simulated Annealing in Xia and Wu (2005).

The second type of approaches that started about fifteen years ago is based on Pareto optimization where the goal is to determine the set of non-dominated solutions, i.e. the Pareto front. A hierarchical heuristic algorithm that is an adaptation of the Newton's method for continuous multi-objective unconstrained optimization problems is proposed in Pérez and Raupp (2016). Two adapted genetic algorithms are presented in Rahmati et al. (2013). A simple and effective evolutionary algorithm that needs only two parameters is developed in Chiang and Lin (2013), and a filtered-beam-search-based heuristic in Shi-Jin et al. (2008).

Many authors developed hybrid methods that combine two or more algorithms to improve the convergence while ensuring the diversity of solutions. Chun et al. (2013) combine an evolutionary algorithm with a local search heuristic. A Non-dominated Sorting Genetic Algorithm II is combined with a local search in Yuan and Xu (2015), a Scatter Search algorithm that uses Tabu Search and Path-Relinking is proposed in González et al. (2015), a Path-Relinking based on a Tabu Search algorithm with back-jumping tracking is developed in Jia and Hu (2014), a hybrid discrete Particle Swarm Optimization and Simulated Annealing algorithm are proposed in Shao et al.

(2013), a Pareto-based estimation of distribution algorithm is combined with a local search heuristic in Wang et al. (2013), a genetic algorithm and local search are combined in Xiong et al. (2012), a genetic algorithm is combined with a Simulated Annealing in Li (2011), and an approach hybridizing a discrete Artificial Bee Colony algorithm and local search approaches is proposed in Li et al. (2011).

All the above mentioned papers focus on optimizing the makespan, the total workload of machines, and the workload of the critical machine. There are only very few papers that consider other objectives such as regular criteria. In the FJSP with mono-objective, García-León et al. (2015) propose a general approach for optimizing any regular criteria, which presents new concepts to be used in local search methods. To minimize the total tardiness, Türkyılmaz and Bulkan (2015) combine a genetic algorithm and a Variable Neighborhood Search, and Mousakhani (2013) presents a Mixed Integer Linear Programming model and an iterated local search. For the MOFJSP, Singh et al. (2016) propose a Particle Swarm Optimization algorithm to simultaneously minimize the makespan, mean flow time, and mean tardiness. Gao et al. (2014) minimize a weighted combination of the makespan and the mean of earliness and tardiness, using a discrete Harmony Search algorithm that makes use of several heuristics. A Variable Neighborhood Search algorithm is proposed in Bagheri and Zandieh (2011) to minimize a weighted sum of the makespan and the mean tardiness. Vilcot and Billaut (2011) present a version of Tabu Search that minimizes a linear combination of $C_{\max}$, $\sum T_i$ and $L_{\max}$. Tay and Ho (2008) consider the minimization of the weighted sum of the makespan, the mean tardiness, and the mean flow time, by using priority rules and the concept of genetic programming. A heuristic inspired from Particle Swarm Optimization and Variable Neighborhood Search is proposed in Liu et al. (2006) for minimizing a weighted linear combination of the makespan and the sum of completion times.

The MOFJSP has also been addressed under a variety of constraints, assumptions and practical issues. In Lei et al. (2018), the makespan and total tardiness are minimized under the constraint that the total energy consumption does not exceed a given threshold. Mokhtari and Hasani (2017) develop an evolutionary algorithm to minimize the makespan, the total availability of the system, and the total energy cost of both production and maintenance operations. The uncertainty in processing times is addressed in Shen et al. (2017) to simultaneously minimize makespan, maximal machine workload, and robustness to uncertainties. Lu et al. (2017) investigate the problem un-

der controllable processing times, i.e. the processing times of operations can be controlled by allocating additional resources, to find an efficient trade-off between the makespan and the total additional resource consumption. Fuzzy processing times and fuzzy due dates are addressed in Chun et al. (2013) using a memetic algorithm that combines genetic global optimization with a local search method. Ahmadi et al. (2016) address random machine breakdowns by considering the makespan and stability measures. Li et al. (2014) consider maintenance activities on machines and propose a discrete Artificial Bee Colony algorithm to deal with the makespan, the total workload of machines, and the workload of the critical machine. Random machine breakdowns are also considered in Xiong et al. (2013) with the objective of minimizing the makespan and the robustness. The dynamic FJSP with job release dates is addressed in Nie et al. (2013) to minimize the makespan, the mean flow time, and the mean tardiness. Sadrzadeh (2013) considers sequence-dependent setups using an Artificial Immune System and Particle Swarm Optimization to minimize an aggregate function of the makespan and the mean tardiness. Setup times are also considered in Bagheri and Zandieh (2011) using a Variable Neighborhood Search approach to minimize the makespan and the mean tardiness.

To conclude, the MOFJSP has been solved in the literature using different methods that range from simple heuristics to sophisticated metaheuristics. Although the MOFJSP gained considerable attention from researchers during the last ten years, most studies consider the optimization of the makespan and two non-regular criteria (total workload and maximum workload of machines). A very limited number of papers address regular criteria even when optimizing a single criterion. Indeed, Chaudhry and Khan (2016) report that the optimization of the makespan combined with other regular criteria has little been studied, e.g. 2.5% of the papers consider maximum tardiness and 1.5% deal with total tardiness. When regular criteria are combined with the makespan, most papers do not aim at Pareto optimization and instead aggregate the criteria in one objective using a weight for each criterion. Moreover, the concepts of disjunctive graph and estimation functions are not exploited. One of the contributions of this paper is the design of an efficient Pareto optimization approach for the MOFJSP with regular criteria by developing different strategies to efficiently determine a set of non-dominated solutions.

3. Problem modeling and neighborhood structures for the FJSP

This section introduces the different concepts used in this paper to model and solve the MOFJSP, and illustrates these concepts using the example in Table 1 with three jobs and four machines. Each job J_i has four operations which are denoted O_{ij} ($i = 1, 2, 3$ and $j = 1, 2, 3, 4$). For example, the first operation of job J_1 has two eligible machines M_1 and M_3 with processing times of 3 and 5, respectively. The third operation of J_2 has no flexibility, since it can only be performed on machine M_4.

	Eligible machines and processing times for operations			
Job	1	2	3	4
J_1	$M_1(3)/M_3(5)$	$M_2(3)/M_4(4)$	$M_1(5)/M_3(1)$	$M_3(1)$
J_2	$M_1(5)/M_3(4)$	$M_1(4)/M_2(5)$	$M_4(1)$	$M_2(2)$
J_3	$M_1(2)$	$M_3(3)/M_4(4)$	$M_2(8)$	$M_3(2)/M_4(2)$

Table 1: An illustrative example of the FJSP

The FJSP with regular criteria can be modeled using a disjunctive graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{A} \cup \mathcal{E}$ is the set of arcs (see Dauzère-Pérès and Paulli (1997)). Let us recall some important definitions. The set $\mathcal{V}$ includes operations of jobs, a dummy node 0 that represents the start of each job, and n dummy nodes ϕ_i associated to the completions of jobs (see e.g. Mati et al. (2011)). Nodes ϕ_i are necessary since regular criteria depend on the completion times of jobs. The set $\mathcal{A}$ contains conjunctive arcs that connect two consecutive operations (i.e. in the routing) of a job, the node 0 and every first operation of each job, and the last operation of each job J_i to its corresponding node ϕ_i. The set $\mathcal{E} = \cup_{m \in \mathcal{M}} \mathcal{E}_m$ contains disjunctive arcs where $\mathcal{E}_m$ includes arcs between pairs of operations that may use machine m. The arc from 0 to the first operation of a job J_i has a length which is equal to the release date r_i of J_i, and any remaining conjunctive or disjunctive arc has a length which is equal to the processing time of the operation from which it starts. Figure 1(a) shows the disjunctive graph for the example in Table 1.

A feasible solution of the FJSP is obtained by assigning a machine to each operation (thus keeping only the relevant disjunctive arcs in E) and by fixing a direction to each disjunctive arc in $\mathcal{E}$ such that the induced graph does not contain any directed cycle. To effectively exploit the structure and properties of the graph in a local search approach, the graph must be

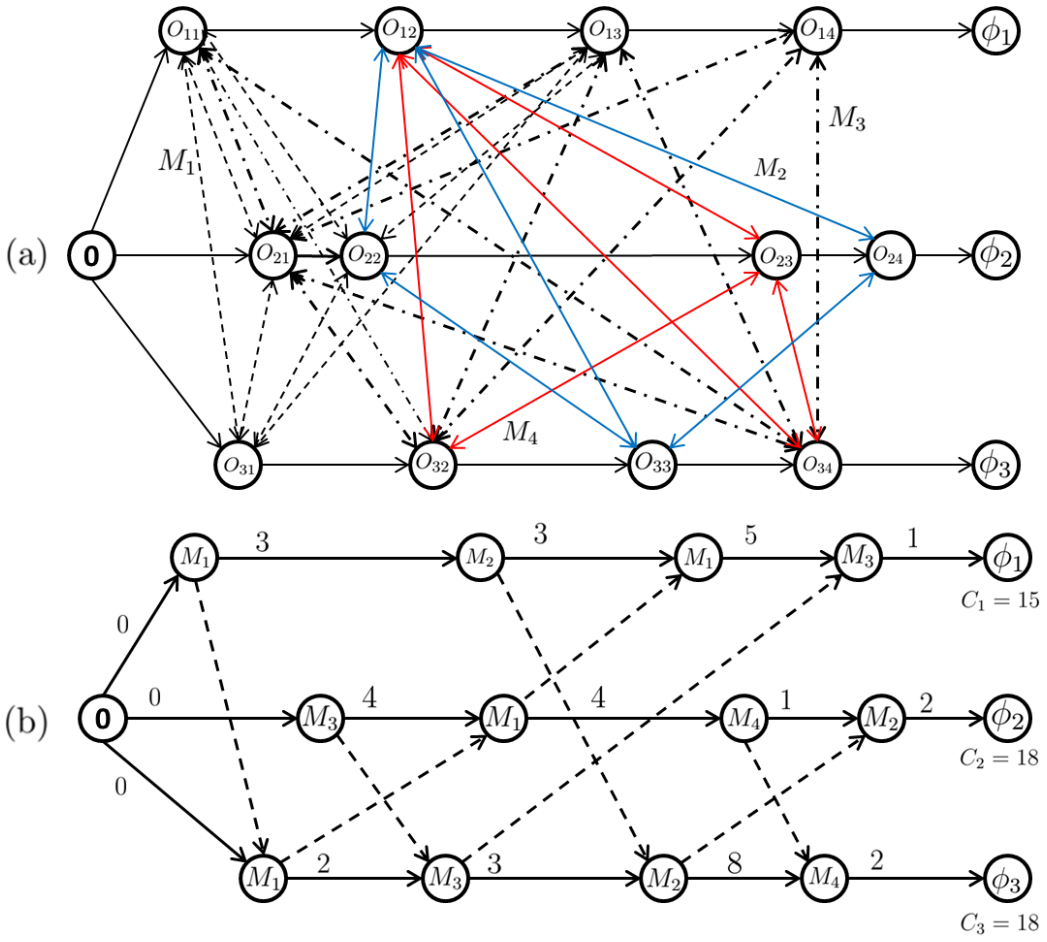

Figure 1: Disjunctive graph model and a feasible solution for the example in Table 1

simplified by removing redundant arcs so that every node x has at most one predecessor and one successor on the machine that performs x. Figure 1(b) shows a feasible solution for the example in Table 1. For example, the first operation O_{21} of job J_2 is assigned to machine M_3. The sequences of jobs with their operations on machines are the following: $J_1(O_{11}) \rightarrow J_3(O_{31}) \rightarrow J_2(O_{22}) \rightarrow J_1(O_{13})$ on M_1, $J_1(O_{12}) \rightarrow J_3(O_{33}) \rightarrow J_2(O_{24})$ on M_2, $J_2(O_{21}) \rightarrow J_3(O_{32}) \rightarrow J_1(O_{14})$ on M_3, and $J_2(O_{23}) \rightarrow J_3(O_{34})$ on M_4.

The starting time h_x (called *head*) of a node x is given by the length of a longest path from 0 to x. The *level* l_x of node x is the maximum number of

arcs from node 0 to x. The *tail* q_x^i of x to a dummy node ϕ_i is the maximum length from the completion of x to ϕ_i if a path exists from x to ϕ_i and $-\infty$ otherwise. Tails are needed since regular criteria are considered in this paper. For example, looking at Figure 1(b), the head of operation O_{23} is 9, its level is equal to 4 and its tail to ϕ_3 is equal to 2. However, the tail of operation O_{23} to ϕ_1 is equal to $-\infty$ since there is no path from O_{23} to ϕ_1. The longest path from node 0 to node ϕ_i is called the critical path from 0 to ϕ_i, and its length is equal to h_{ϕ_i}, which corresponds to the completion time of J_i. Every node x belonging to a critical path is critical according to J_i, and satisfies $h_x + p_x + q_x^i = h_{\phi_i}$. Each arc (x, y) belonging to the critical path from 0 to ϕ_i is critical if nodes x and y are assigned to the same machine and belong to the routing of different jobs. A *block* is a maximum sequence of critical nodes assigned to the same machine. Table 2 shows the critical paths of jobs and their corresponding blocks for the solution in Figure 1(b). Note that O_{31} is critical for all jobs, O_{24} is only critical for job J_2, whereas operations O_{12}, O_{21}, and O_{23} are not critical. The critical path of job J_2 contains two blocks.

Job	Critical path	Block
J_1	$0 \rightarrow O_{11} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{13} \rightarrow O_{14} \rightarrow \phi_1$	$(O_{11} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{13})$
J_2	$0 \rightarrow O_{11} \rightarrow O_{31} \rightarrow O_{32} \rightarrow O_{33} \rightarrow O_{24} \rightarrow \phi_2$	$(O_{11} \rightarrow O_{31})$,$(O_{33} \rightarrow O_{24})$
J_3	$0 \rightarrow O_{11} \rightarrow O_{31} \rightarrow O_{32} \rightarrow O_{33} \rightarrow O_{34} \rightarrow \phi_3$	$(O_{11} \rightarrow O_{31})$

Table 2: Critical paths of jobs and their corresponding blocks for solution in Figure 1(b)

Neighborhood structures are used in local search to generate new solutions by performing small perturbations of a current solution. In the FJSP, a well-known perturbation proposed in Dauzère-Pérès and Paulli (1997) consists in moving (i.e. resequencing or reassigning) a critical operation in the graph of the current solution. In this paper, we consider two neighborhood structures (N_1 and N_2), which differ from one another in the selection of operations that are moved. Neighborhood structure N_1 focuses on all critical operations of jobs, while neighborhood structure $N_2 \subset N_1$ focuses on operations that belong to blocks of critical paths of the jobs that affect the value of the criterion (e.g. jobs that are late for lateness criteria). Our motivation in defining these two neighborhood structures is to analyze whether the concept of blocks is helpful to generate sets of non-dominated solutions for the MOFJSP. To understand the difference between N_1 and N_2, let us consider the minimization of $\sum C_i$ in the solution of Figure 1(b). According to the

critical paths and blocks of Table 2, neighborhood structure N_1 considers the critical operations O_{11}, O_{13}, O_{14}, O_{22}, O_{24}, O_{31}, O_{32}, O_{33} and O_{34}, whereas neighborhood structure N_2 "only" focuses on O_{11}, O_{13}, O_{22} ,O_{24}, O_{31} and O_{33}. Table 3 gives the possible resequencing and reassignment moves for each critical operation in both neighborhood structures. For a given critical operation O_{ij}, the notation $[a - b]$ means that O_{ij} is moved between operations a and b. If $a = 0$ (resp. $b = *$), O_{ij} is moved in the first (resp. last) position of the sequence of the machine on which it is resequenced or reassigned. For example, O_{11} can be resequenced between O_{22} and O_{13}, and reassigned on machine M_3 between O_{32} and O_{14}.

Critical operation	Resequencing *Move*	Reassignment *Machine*	*Move*
O_{11}	$[O_{31} - O_{22}]^{\ddagger}$ $[O_{22} - O_{13}]^{\ddagger}$ $[O_{13} - *]^{\otimes}$	M_3	$[0 - O_{21}]^{\ddagger}$ $[O_{21} - O_{32}]^{\ddagger}$ $[O_{32} - O_{14}]^{\ddagger}$ $[O_{14} - *]^{\otimes}$
O_{31}	$[0 - O_{11}]^{\ddagger}$ $[O_{22} - O_{13}]^{\ddagger}$ $[O_{13} - *]^{\dagger}$		
O_{22}	$[0 - O_{11}]^{\ddagger}$ $[O_{11} - O_{31}]^{\ddagger}$ $[O_{13} - *]^{\ddagger}$	M_2	$[0 - O_{12}]^{\ddagger}$ $[O_{12} - O_{33}]^{\ddagger}$ $[O_{33} - O_{24}]^{\ddagger}$ $[O_{24} - *]^{\otimes}$
O_{32}	$[0 - O_{21}]^{\ddagger}$ $[O_{14} - *]^{\ddagger}$	M_4	$[0 - O_{23}]^{\ddagger}$ $[O_{23} - O_{34}]^{\ddagger}$ $[O_{34} - *]^{\otimes}$
O_{13}	$[0 - O_{11}]^{\otimes}$ $[O_{11} - O_{31}]^{\ddagger}$ $[O_{31} - O_{22}]^{\ddagger}$	M_3	$[0 - O_{21}]^{\ddagger}$ $[O_{21} - O_{32}]^{\ddagger}$ $[O_{32} - O_{14}]^{\ddagger}$ $[O_{14} - *]^{\otimes}$
O_{33}	$[0 - O_{12}]^{\ddagger}$ $[O_{24} - *]^{\ddagger}$		
O_{34}	$[0 - O_{23}]^{\ddagger}$	M_3	$[0 - O_{21}]^{\otimes}$ $[O_{21} - O_{32}]^{\otimes}$ $[O_{32} - O_{14}]^{\ddagger}$ $[O_{14} - *]^{\ddagger}$
O_{24}	$[0 - O_{12}]^{\dagger}$ $[O_{12} - O_{33}]^{\ddagger}$		
O_{14}	$[0 - O_{21}]^{\otimes}$ $[O_{21} - O_{32}]^{\ddagger}$		

Table 3: Possible resequencing and reassignment moves in Figure 1(b)

Moving an operation in both neighborhood structures N_1 and N_2 can generate directed cycles in the resulting graph, thus leading to unfeasible solutions of the FJSP. To check the feasibility of a move, the sufficient conditions proposed in García-León et al. (2015) are used. Without actually transforming the graph, they validate that a cycle is not created in the new graph. These conditions generalize previous conditions of the literature by using the concepts of heads, tails and levels of operations. In Table 3, feasible moves that are obtained by the sufficient conditions are denoted by the superscript ‡, unfeasible moves by the superscript $^{\otimes}$, and moves that are feasible but cannot be validated by the sufficient conditions are denoted by the superscript †.

The best move in the neighborhood of a solution is generally obtained using the value of the criterion of the generated solution. Previous studies on the FJSP have shown that using estimation functions is more appropriate to evaluate the quality of moves, because significant computational times can be saved and much more iterations can be performed to reach better solutions. Since regular criteria are considered in this paper, we need to estimate the new completion times of nodes ϕ_i after moving an operation (see Mati et al. (2011) for the classical job-shop scheduling problem). To do so, we use the estimation function proposed in García-León et al. (2015) by considering forward and backward moves. A forward (resp. backward) move of a node x, currently sequenced between nodes p and q, between two nodes u and v is defined when $l_x \leq l_u$ (resp. $l_x > l_u$). The idea of the estimation function consists in considering the newly created paths after the move together with a suitable subset of paths that are available in the current and new graphs. This is performed by focusing not only on the operation x, but also on the operations involved in the move p, q, u and v, as well as on operations w for which $l_w = l_x$ (see Mati et al. (2011)). In addition to its efficiency in estimating the value of the criterion, the estimation function is fast and guarantees whenever possible the lower bound property, i.e. the quality of a move is not overestimated (García-León et al. (2015)).

4. Evaluating sets of non-dominated solutions

An effective approach for solving the MOFJSP with regular criteria is the Pareto approach, which aims at finding a set of non-dominated solutions $\mathcal{S}$, called the Pareto front. In this section, let us briefly recall the main notions

related to Pareto optimization and introduce the measures to evaluate the quality of the set $\mathcal{S}$.

4.1. Dominance of Pareto

Let $\mathcal{C}$ be the set of criteria to minimize in Pareto manner and $f_c(s)$ be the value of the criterion c of a feasible solution s. Solution s_1 dominates solution s_2 if the following two conditions are true:

1. Solution s_1 is not worse than solution s_2 for all criteria, i.e. $\forall c \in \mathcal{C}$, $f_c(s_1) \leq f_c(s_2)$,
2. Solution s_1 is strictly better than s_2 for at least one criterion, i.e. $\exists c \in \mathcal{C}$ such that $f_c(s_1) < f_c(s_2)$.

Accordingly, any two solutions of $\mathcal{S}$ are non-dominated with respect to each other, and any solution not in $\mathcal{S}$ is dominated by at least one solution in $\mathcal{S}$.

4.2. Quality measures of the set of non-dominated solutions

A good set of non-dominated solutions should satisfy two goals: Convergence and diversity. Convergence ensures that the set of solutions is as close as possible to the optimal Pareto front, and diversity is related to the sparsity of solutions to ensure that the decision maker has multiple representative trade-off solutions among conflicting objectives. Zitzler et al. (2003) state that it is difficult to define appropriate measures to approximate the optimal Pareto front when analyzing both goals, and that the discrepancies increase when considering stochastic approaches.

For the MOFJSP, most previous studies aim at improving the convergence and increasing the number of non-dominated solutions without considering diversity (see e.g. Jia and Hu (2014)). In this paper, we consider both the convergence and diversity to better evaluate the quality of sets of non-dominated solutions. Three measures are selected to evaluate the convergence: (1) The elite solutions, (2) The mean ideal distance and (3) The hypervolume. Elite solutions correspond to the best values of the criteria. The Mean Ideal Distance (MID) is the average distance between non-dominated solutions and the origin point (Singh et al. (2016)), i.e. the point $(0, 0)$ if two criteria are analyzed. MID is calculated using (1), where $|\mathcal{S}|$ is the number of non-dominated solutions.

$$MID = \frac{\sum_{s \in \mathcal{S}} \sqrt{\sum_{c \in \mathcal{C}} f_c^2(s)}}{|\mathcal{S}|} \tag{1}$$

The HyperVolume (HV) is the volume covered by the solutions of the front. When all criteria are minimized, a reference point having as coordinates the worst values of the criteria is used to limit this coverage (Zitzler et al. (2007)). Thus, $HV = \sum_{s \in \mathcal{S}} V_s$, where V_s is the hypercube of s with respect to the reference point. Since the hypervolume can lead to large values,(2) is used to calculate HV, which corresponds to the ratio of the total volume $\mathcal{V}_{\mathcal{T}}$ covered by the reference point and the origin point.

$$HV = \frac{\sum_{s \in \mathcal{S}} V_s}{|\mathcal{S}| \times \mathcal{V}_{\mathcal{T}}} \tag{2}$$

The maximum spread (D) and spacing (SP) are selected to evaluate the diversity. The metric D is the longest diagonal of the hyperbox formed by the extreme values of the criteria in $\mathcal{S}$ (Zitzler (1999)), and is calculated using (3), where $f_c^{\max}$ and $f_c^{\min}$ are the maximum and minimum values of criterion c for all solutions in $\mathcal{S}$:

$$D = \sqrt{\sum_{c \in \mathcal{C}} (f_c^{\max} - f_c^{\min})^2} \tag{3}$$

The metric SP is the average distance between consecutive solutions in $\mathcal{S}$ (Schott (1995)). Let $\hat{d}_i$ be the distance between solution s_i and its nearest solution, i.e. $\hat{d}_i = \min_{s_p \in \mathcal{S}; p \neq i} \sum_{c \in \mathcal{C}} |f_c(s_i) - f_c(s_p)|$, and let $\bar{d}$ be the average of these distances for all solutions in $\mathcal{S}$. Spacing is calculated using (4).

$$SP = \sqrt{\frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} (\hat{d}_i - \bar{d})^2} \tag{4}$$

To ensure the quality of $\mathcal{S}$, the spacing and mean ideal distance must be minimized, the maximum spread and hypervolume must be maximized and elite solutions must be as close as possible to the optimal values of the criteria.

5. Solving the MOFJSP

The proposed Pareto approach for the MOFJSP with regular criteria aims at finding a set of non-dominated solutions $\mathcal{S}$ whose convergence and diversity are optimized. Let us first describe how $\mathcal{S}$ is managed, then present the framework of the approach and the initial solution, and finally propose four search strategies.

5.1. Controlling the set of non-dominated solutions

The control of the set of non-dominated solutions consists in managing solutions entering and leaving $\mathcal{S}$ each time a new solution s is obtained by the search process. A schedule $s \in \mathcal{S}$ is called a reference schedule for criterion c if $f_c(s)$ is the best possible value for criterion c. The reference schedule for c is denoted by s_c^{ref}, and the subset of $\mathcal{S}$ with the reference schedules is denoted by $\mathcal{S}^{ref}$.

To efficiently control $\mathcal{S}$, we propose a fast hierarchical test in three steps to avoid performing too many evaluations to check whether s should be added to $\mathcal{S}$. The test is illustrated in Figure 2. It first evaluates if the value of any criterion c of s is strictly lower than the best value for criterion c. If it is the case, then s becomes the reference schedule for criterion c, and s is added to $\mathcal{S}^{ref}$ and $\mathcal{S}$, maybe replacing other solutions. Otherwise, the test validates the dominance between $s_i \in \mathcal{S}$ and s starting with the reference schedules. If no dominance is found, then s can be added to $\mathcal{S}$, maybe replacing other solutions. Hence, the step *Update* $\mathcal{S}$ consists in adding s and removing the dominated solutions. Note that multiple solutions are not considered, i.e. if the values of all the criteria of s and of a solution $s_i \in \mathcal{S}$ are equal.

5.2. Framework of the approach

The approach consists of two alternating phases, namely an improving phase and a diversification phase. The improving phase is a steepest descent that performs iterative improvements until a local optimum is reached for a given criterion or all criteria. At each iteration, a set of neighbor solutions is generated using the neighborhood structures, the feasibility test and the move evaluation described in Section 3. The diversification phase starts from the local optimum of the improving phase and performs at most b iterations. During this phase, a critical operation is randomly selected and a move is randomly chosen. If the selected move is feasible, the heuristic advances to the next iteration, otherwise the above process is repeated until a feasible

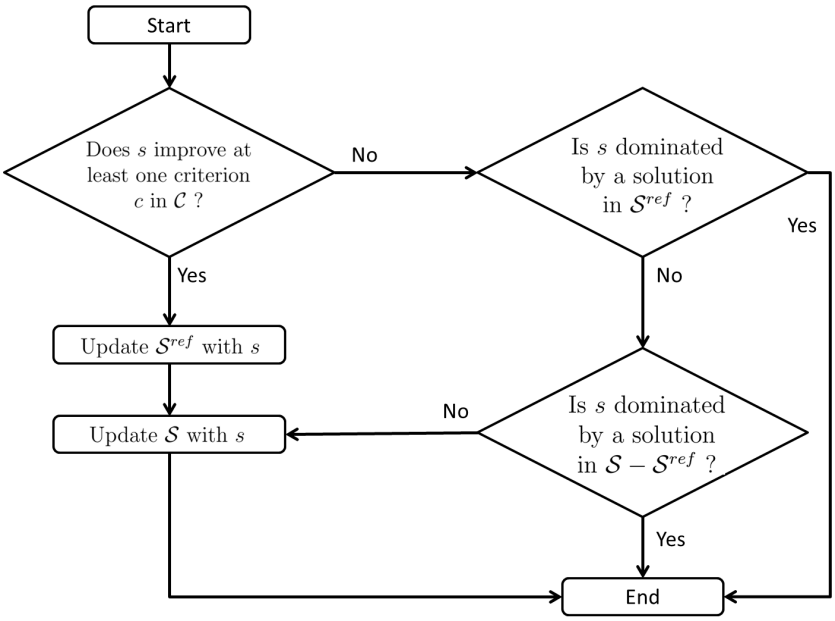

Figure 2: Test to check whether s is added to $\mathcal{S}$

move is obtained. If a new best solution is obtained in the diversification phase for a given criterion, the search returns to the improving phase, otherwise it continues until b iterations are performed. The value of b is randomly selected in $[4, 10]$ which is fixed experimentally.

After performing a move in both phases, all local values of the criteria are updated, the hierarchical test to determine whether s is added to $\mathcal{S}$ is performed, and $\mathcal{S}$ is updated if s is added as described in Figure 2. Additionally, the best values of the criteria are updated.

To deal with multiple criteria for the MOFJSP, we propose four versions of the above approach that differ mainly in the way the criterion to optimize is selected, the way the approach is alternating between the improving and diversification phases, and the way the solution is selected when each phase is resumed.

5.3. Initial solution

The initial solution is obtained using a constructive heuristic that selects an operation at each step according to an established order of the jobs. The main idea of the heuristic is to complete the selected operation as soon as possible to try to minimize the completion times of jobs. The jobs are ordered

by non-decreasing weights when at least one criterion considers weights. The ties are broken using the due dates, and then the average processing times $\sum_{j=1}^{n_i} \frac{1}{|R_j|} \sum_{a \in R_j} p_j$ where n_i is the number of operations of job J_i. For a given operation x and for each eligible machine $M_k \in R_x$, the time t_k at which the machine completes its previous operation v on the sequence (if it exists) is calculated. Operation x is then assigned to the machine that completes x as soon as possible, i.e. the machine $M_k \in R_x$ that minimizes $t_k + p_x$. The graph is updated by adding arc (v, x) and the heuristic continues until all operations have been selected.

5.4. Search strategy $\mathsf{T_1}$

The idea of this strategy is to concentrate on optimizing a given criterion by performing an improving phase followed by a diversification phase. More precisely, a random criterion c is selected in $\mathcal{C}$, and the improving phase performs iterative improvements of c until reaching a local optimum for this criterion. The diversification phase starts from this local optimum and, if the value of the criterion c is improved during this phase, the search returns to the improving phase with the same criterion c. However, if the maximum number of iterations b is reached, the search sets all local values of the criteria to ∞, randomly selects a new criterion to minimize from the set $\mathcal{C} - \{c\}$ and returns to the improving phase.

5.5. Search strategy $\mathsf{T_2}$

This strategy gives more attention to the improving phase since most of the promising solutions are obtained in this phase. The strategy intensifies the search in the improving phase until reaching the local optimum of all criteria. To apply this strategy, the concept of forbidden criterion or criteria is introduced. This concept is defined and applied, for a given criterion c, only during the improving step when the local optimum of c is reached. More precisely, a criterion becomes forbidden when, in the improving phase, it is selected to create a move and it cannot generate an improving move. A criterion is authorized to be selected as soon as its local value is improved or in the finalization of the diversification phase.

More precisely, Strategy $\mathsf{T_2}$ starts by setting the set of forbidden criteria $\mathcal{C}^{for}$ to $\emptyset$ to specify that initially all criteria are authorized. Then, at each iteration of the improving phase, a criterion c is randomly selected from the set $\mathcal{C} - \mathcal{C}^{for}$ of authorized criteria. The search optimizes c whenever it is possible to generate an improving move, and any forbidden criterion becomes

authorized if its local value is improved. However, if an improving move is not possible with c, this criterion becomes forbidden and the search continues with a criterion randomly selected in the set of authorized criteria. If all criteria are forbidden, i.e. $\mathcal{C} - \mathcal{C}^{for} = \emptyset$, the search goes to the diversification phase.

An important problem with the continuity of the search can be caused by criterion $T_{\max}$ since, if it is equal to zero, criteria $\sum T_i$ and $\sum U_i$ will also be equal to zero, and it is not possible to create a move. In this case, the search removes all forbidden criteria from $\mathcal{C}^{for}$ and the selected criterion c is either $C_{\max}$ or $\sum C_i$ if the latter criterion belongs to $\mathcal{C}$.

The diversification phase starts from the solution generated by the neighborhood structure of the last forbidden criterion. If a local value of any criterion is improved in this phase, the criterion becomes authorized, and the search returns to the improving phase using the neighborhood of this criterion. In case of several improved criteria, a random choice is performed. If it is not possible to improve any criterion during b iterations, all criteria are authorized, all local values of the criteria are set to ∞ and the search goes to the improving phase with a random criterion.

5.6. Search strategy $\mathsf{T_3}$

This strategy is a variant of $\mathsf{T_2}$, the only difference is in the improving phase in which it is possible that criterion c is changed in each iteration even if the last iteration was an improving move for c. This means that, rather than continuing with a single criterion until reaching its local optimum, $\mathsf{T_3}$ can modify the optimized criterion by using a random selection from the set of non-forbidden criteria. More precisely, in each iteration, a random criterion c is selected to create a move from the set $\mathcal{C} - \mathcal{C}^{for}$. If this move improves the criterion, the set of forbidden criteria $\mathcal{C}^{for}$ is emptied. If it is not possible to create an improving move using c, this criterion becomes forbidden and it is added to $\mathcal{C}^{for}$. If it is not possible to create an improving move with all criteria, the search goes to the diversification phase considering the neighborhood of the last forbidden criterion and the same guidelines than Strategy $\mathsf{T_2}$.

5.7. Search strategy $\mathsf{T_4}$

Strategy $\mathsf{T_4}$ operates as Strategy $\mathsf{T_3}$ but uses the concept of global value of the criterion c. The only difference is in the improving phase in which, if the global value of a criterion $c' \neq c$ is improved, this criterion becomes the

optimized criterion in the next iteration even if the search with the current criterion was improving. The motivation is that it is more suitable to shift the search to optimize c' with the aim of finding new reference schedules for c', since these schedules can be lost if the search process does not focus on c' at this iteration. If several global values are improved, a random choice is performed. Further, in the diversification phase, the search can return to the improving phase with a criterion that improves its global value.

6. Computational results

This section analyzes the efficiency of the general approach proposed in the previous section, which was developed in Java. In the remainder of the paper, this approach is denoted *GMD*. The experiments were conducted on a PC with 3.40 GHz and 8GB RAM. The computational time for each search strategy and each neighborhood structure was set to 300 seconds. Hence, the computational time for an instance is 2,400 seconds for a set of criteria to optimize in Pareto manner, i.e. 300 seconds multiplied by four search strategies and two neighborhood structures. Three sets of criteria to optimize are considered: $\mathcal{C}_A$, $\mathcal{C}_B$ and $\mathcal{C}_C$. $\mathcal{C}_A$ includes three criteria: $C_{\max}$, $T_{\max}$ and $\sum T_i$. $\mathcal{C}_B$ adds criterion $\sum U_i$ to the criteria in $\mathcal{C}_A$, and $\mathcal{C}_C$ adds criterion $\sum C_i$ to the criteria in $\mathcal{C}_B$.

The analysis was conducted in six phases that are described in the following sections. Sections 6.1 to 6.5 use the problem instances from Brandimarte (1993) by setting the due date of each job J_i to $1.3 \times \sum_{j=1}^{n_i} \frac{1}{|R_j|} \sum_{a \in R_j} p_j$, where n_i is the number of operations of job J_i. Section 6.1 analyzes the Net Front Contribution (NFC) and the Weak OutPerformance (WOP) of the two neighborhood structures to check if one is dominating the other. Then, the same analysis is performed for the search strategies. The impact of adding criteria in the set of criteria to optimize on the number of non-dominated solutions is studied in Section 6.2. Elite solutions for five regular criteria are analyzed in Section 6.3. The diversity is studied in Section 6.4. The analysis of the hypervolume and the mean ideal distance is presented in Section 6.5. Finally, Section 6.6 compares our approach to previous approaches to optimize the makespan and the total tardiness.

6.1. Analysis of the Net Front Contribution (NFC) and Weak OutPerformance (WOP)

The Net Front Contribution (NFC) is the percentage of solutions of the reference front that are included in a specified set of non-dominated solutions (Deb et al. (2001)). For example, if the NFC of Strategy T_1 is 25%, then 25% of the solutions of the reference front belong to T_1. The Weak OutPerformance metric ($\text{WOP}_{x,o}$) evaluates the dominance between two sets of non-dominated solutions s_x and s_o (see Vilcot and Billaut (2011)). The set s_x weakly outperforms s_o if no solution in s_x is dominated by a solution in s_o and at least one solution in s_x dominates a solution in s_o. Hence, $\text{WOP}_{x,o}$ takes value 1 if s_x weakly outperforms s_o and 0 otherwise. To further improve the analysis of dominance between s_x and s_o, we extend the numerical scale to three values -1, 0 and 1: $\text{WOP}_{x,o}$ is equal to 1 (resp. -1) if s_x (resp. s_o) weakly outperforms s_o (resp. s_x) and 0 otherwise.

Table 4 presents, over ten runs of the algorithm, the average NFC and the average percentage for WOP_{N_1,N_2} (WOP) when it is equal to 1 or -1 for each set of criteria and for each neighborhood structure. As an example, *mk01* has six machines (m), 10 jobs (n) and a flexibility level ($flex$) of 2.09, i.e. one operation has on average 2.09 eligible machines. For $\mathcal{C}_A$, the average NFC for neighborhood N_1 is 50% and 53.3% for N_2. N_1 weakly outperforms N_2 (1 in column WOP) in 33.3% of cases and N_2 weakly outperforms N_1 (-1 in column WOP) in 60% of cases. Additionally, the neighborhood structure with the average largest NFC and WOP are written in bold.

Inst	*Size*	*flex*	$\mathcal{C}_A$				$\mathcal{C}_B$				$\mathcal{C}_C$			
			NFC(%)		WOP(%)		NFC(%)		WOP(%)		NFC(%)		WOP(%)	
	$(m \times n)$		N_1	N_2	1	-1	N_1	N_2	1	-1	N_1	N_2	1	-1
mk01	6×10	2.09	50.0	**53.3**	33.3	**60.0**	**85.2**	20.8	**26.7**	0.0	**71.0**	29.6	**6.7**	0.0
mk02	6×10	4.1	**66.7**	56.7	**33.3**	26.7	53.3	**80.0**	13.3	**40.0**	**67.8**	33.3	**53.3**	26.7
mk03	8×15	3.01	43.3	**63.3**	40.0	**53.3**	39.2	**60.8**	6.7	**53.3**	32.4	**67.6**	20.0	**53.3**
mk04	8×15	1.91	**55.7**	44.3	33.3	**53.3**	47.6	**52.9**	0.0	0.0	31.8	**68.2**	0.0	0.0
mk05	4×15	1.71	**61.8**	38.2	**66.7**	26.7	48.4	**51.8**	0.0	0.0	49.5	**50.5**	0.0	0.0
mk06	15×10	3.27	**93.3**	46.7	**53.3**	6.7	**100.0**	93.3	**6.7**	0.0	40.0	**60.0**	33.3	**53.3**
mk07	6×10	2.83	41.6	**58.4**	26.7	**53.3**	**60.9**	39.3	**6.7**	0.0	47.7	**52.3**	0.0	0.0
mk08	5×20	1.43	46.7	**53.3**	46.7	**53.3**	6.5	**93.5**	0.0	**80.0**	8.3	**91.7**	0.0	**80.0**
mk09	10×20	2.53	**63.9**	36.1	**46.7**	26.7	49.6	**52.2**	**20.0**	0.0	**64.2**	35.8	**20.0**	0.0
mk10	15×20	2.98	46.6	**53.4**	40.0	40.0	**53.0**	47.4	**6.7**	0.0	**55.2**	44.8	**13.3**	0.0

Table 4: Average NFC and WOP for N_1 and N_2

Table 4 shows that there is not a dominant neighborhood structure, even though N_2 is slightly better, which confirms the benefit of using the concept of blocks to solve the flexible job-shop scheduling problem. Using the NFC

metric, neighborhood structure N_2 generates larger contributions for 17 instances over 30 instances: *mk01*, *mk03*, *mk07*, *mk08* and *mk10* for $\mathcal{C}_A$; *mk02*, *mk03*, *mk04*, *mk05*, *mk08* and *mk09* for $\mathcal{C}_B$, and *mk03*, *mk04*, *mk05*, *mk06*, *mk07* and *mk08* for $\mathcal{C}_C$. N_1 generates larger contributions for the remaining 13 instances. Concerning the metric WOP, N_2 weakly outperforms N_1 in 11 instances: *mk01*, *mk03*, *mk04*, *mk07* and *mk08* for $\mathcal{C}_A$; *mk02*, *mk03* and *mk08* for $\mathcal{C}_B$ and *mk03*, *mk06* and *mk08* for $\mathcal{C}_C$, and N_1 weakly outperforms N_2 in 13 instances (4 for $\mathcal{C}_A$, 5 for $\mathcal{C}_B$ and 4 for $\mathcal{C}_C$). Additionally, no neighborhood structure weakly outperforms the other in five instances: *mk04* and *mk05* for $\mathcal{C}_B$ and $\mathcal{C}_C$, and *mk07* for $\mathcal{C}_C$.

Table 5 helps us to analyze the contribution of each search strategy to the reference front by considering only neighborhood structure N_2 for each set of criteria. Note that the values in this table are average values over ten runs of the algorithm.

Inst	$\mathcal{C}_A$				$\mathcal{C}_B$				$\mathcal{C}_C$			
	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
mk01	0.0	**100.0**	**100.0**	**100.0**	0.0	0.0	25.0	**75.0**	0.0	7.7	**92.3**	0.0
mk02	0.0	**100.0**	**100.0**	**100.0**	33.3	0.0	**66.7**	33.3	0.0	**100.0**	**100.0**	**100.0**
mk03	0.0	**100.0**	0.0	0.0	0.0	**50.0**	25.0	25.0	0.0	0.0	**80.0**	20.0
mk04	0.0	**100.0**	50.0	50.0	0.0	18.4	**46.9**	36.7	0.0	7.9	**47.4**	44.7
mk05	0.0	**83.3**	16.7	0.0	0.0	30.8	**48.7**	20.5	0.0	7.7	30.8	**61.5**
mk06	**100.0**	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	**100.0**	**100.0**	**100.0**
mk07	0.0	0.0	**100.0**	**100.0**	0.0	4.6	**50.0**	45.5	0.0	0.0	**88.0**	12.0
mk08	0.0	**100.0**	**100.0**	**100.0**	0.0	**81.5**	18.5	0.0	0.0	0.0	0.0	**100.0**
mk09	20.0	**80.0**	0.0	0.0	0.0	19.1	38.1	**42.9**	0.0	0.0	24.0	**76.0**
mk10	0.0	0.0	**100.0**	20.0	0.0	33.3	22.2	**44.4**	0.0	0.0	10.0	**90.0**

Table 5: Analysis of NFC for the different strategies and N_2

The results of Table 5 reveals that the contribution of T_1 is equal to zero except for *mk09* with $\mathcal{C}_A$, *mk02* with $\mathcal{C}_B$, and *mk06* for $\mathcal{C}_A$ and $\mathcal{C}_B$. When using T_2, the entire solutions of the reference front is obtained in 5 instances for $\mathcal{C}_A$ (*mk01*, *mk02*, *mk03*, *mk04* and *mk08*) and 2 instances for $\mathcal{C}_C$ (*mk02* and *mk06*). However, the NFC of T_2 is equal to zero in 11 instances (3 for $\mathcal{C}_A$, 3 for $\mathcal{C}_B$ and 5 for $\mathcal{C}_C$). Besides, T_2 has the largest number of non-dominated solutions in 2 instances (*mk05* and *mk09* for $\mathcal{C}_A$), and 2 instances (*mk03* and *mk08*) for $\mathcal{C}_B$. The contribution of T_2 for the $\mathcal{C}_C$ is really low, which implies that this search strategy is not adequate for minimizing $\sum C_i$. Further, T_3 and T_4 seem to be the best search strategies and their results are comparable with a slight advantage to the former for the three sets of criteria. Indeed,

$\mathsf{T_3}$ generates the highest contribution in 9 instances (1 for $\mathcal{C}_A$, 4 for $\mathcal{C}_B$, and 4 for $\mathcal{C}_C$) whereas $\mathsf{T_4}$ obtains the highest contribution in 7 instances (3 for $\mathcal{C}_B$ and 4 for $\mathcal{C}_C$). Both $\mathsf{T_3}$ and $\mathsf{T_4}$ reach all the solutions of the reference front in 6 instances (4 for $\mathcal{C}_A$ and 2 for $\mathcal{C}_C$). The largest number of non-dominated solutions is obtained by $\mathsf{T_3}$ or $\mathsf{T_4}$ in 73.33% of instances.

To extend the analysis of the four search strategies, Table 6 shows the WOP by considering again neighborhood structure N_2. It can be seen from WOP_{1-2} (column $1-2$), WOP_{1-3} (column $1-3$) and WOP_{1-4} (column $1-4$) that $\mathsf{T_1}$ is dominated by the other search strategies except for instance *mk06* for $\mathcal{C}_A$. Strategy $\mathsf{T_2}$ dominates Strategies $\mathsf{T_3}$ and $\mathsf{T_4}$ in only two instances (*mk03* and *mk09*) for $\mathcal{C}_A$, although $\mathsf{T_2}$ is better for $\mathcal{C}_A$ regarding the metric NFC. The comparison between $\mathsf{T_3}$ and $\mathsf{T_4}$ does not show any obvious dominance since WOP_{3-4} is equal to zero in the largest number of instances (90% of instances for $\mathcal{C}_A$, 50% of instances for $\mathcal{C}_B$, and 70% of instances for $\mathcal{C}_C$). $\mathsf{T_3}$ is better than $\mathsf{T_4}$ in 4 instances and worst in 5 instances.

Inst	$\mathcal{C}_A$						$\mathcal{C}_B$						$\mathcal{C}_C$					
	$1-2$	$1-3$	$1-4$	$2-3$	$2-4$	$3-4$	$1-2$	$1-3$	$1-4$	$2-3$	$2-4$	$3-4$	$1-2$	$1-3$	$1-4$	$2-3$	$2-4$	$3-4$
mk01	-1	-1	-1	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
mk02	-1	-1	-1	0	0	0	1	0	0	-1	-1	0	-1	-1	-1	0	0	0
mk03	-1	-1	-1	1	1	0	-1	-1	-1	-1	0	1	-1	-1	-1	-1	-1	0
mk04	-1	-1	-1	0	0	0	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0
mk05	-1	0	0	-1	1	1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0
mk06	1	1	1	0	0	0	-1	-1	-1	-1	-1	0	-1	-1	-1	0	0	0
mk07	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0
mk08	-1	-1	-1	0	0	0	-1	-1	-1	0	-1	1	-1	-1	-1	-1	-1	-1
mk09	-1	-1	-1	1	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0
mk10	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Table 6: Analysis of WOP for N_2

It can be concluded from the analysis in this section that Strategy $\mathsf{T_1}$ is not suitable to optimize the three sets of criteria, $\mathsf{T_2}$ is effective for solving $\mathcal{C}_A$ regarding the NFC but does not appear interesting when considering the WOP, and Strategies $\mathsf{T_3}$ and $\mathsf{T_4}$ are the most efficient when considering both NFC and WOP, in particular for sets $\mathcal{C}_B$ and $\mathcal{C}_C$.

6.2. Analysis of the number of non-dominated solutions

Table 7 is used to analyze the number of non-dominated solutions considering three outputs: The minimum (Min), average (Av), and maximum (Max) numbers of non-dominated solutions when running ten times each set of criteria. For example, after executing the approach ten times with set $\mathcal{C}_A$ for instance *mk01*, the minimum number of non-dominated solutions was 1,

the maximum was 2 and on average 1.3 solutions were obtained. In an ideal situation, it is desired that the three outputs are equal.

Inst	*Size*	*flex*	$\mathcal{C}_A$			$\mathcal{C}_B$			$\mathcal{C}_C$		
	$m \times n$		Min	Av	Max	Min	Av	Max	Min	Av	Max
mk01	6×10	2.09	1	1.3	2	4	4.8	5	10	15.8	28
mk02	6×10	4.10	1	1.5	2	1	1.5	2	1	1.5	2
mk03	8×15	3.01	1	1.3	2	1	1.8	2	2	7.3	17
mk04	8×15	1.91	1	2.3	4	22	35.0	51	30	36.0	41
mk05	4×15	1.71	4	6.8	11	17	26.0	30	27	30.3	32
mk06	15×10	3.27	1	1	1	1	1	1	1	1	1
mk07	5×20	2.83	4	6.0	10	20	27.8	32	12	20.8	26
mk08	10×20	1.43	1	1.3	2	2	16.8	27	3	6.0	9
mk09	10×20	2.53	1	2.8	4	19	33.0	51	22	27.8	38
mk10	15×20	2.98	4	6.8	10	16	23.8	30	14	17.0	24

Table 7: Analysis of the number of non-dominated solutions

The addition of criteria and the flexibility level affects the number of non-dominated solutions, but there is not a clear behavior in function of the flexibility level since different trends can be identified. More precisely, in instances with a large flexibility level ($flex > 3$), one or two solutions solve the problem, i.e. in instance *mk06* ($flex = 3.27$) only one solution is generated for the three sets of criteria. In instance *mk02* ($flex = 4.10$), no more than two non-dominated solutions for the three sets of criteria were obtained, and this is also the case for instance *mk03* when optimizing sets $\mathcal{C}_A$ and $\mathcal{C}_B$. In instances with a small flexibility level ($flex < 2$), the average number of non-dominated solutions increases when adding criteria, such as the case of instances *mk04* and *mk05*. However, in instance *mk08* ($flex = 1.43$), this average number increases from set $\mathcal{C}_A$ to set $\mathcal{C}_B$ (1.3 to 16.8) and decreases when $\sum C_i$ is added (6.0). The same observation can be drawn for the maximum number of non-dominated solutions. The addition of criterion $\sum U_i$ to $\mathcal{C}_A$ increases the number of non-dominated solutions, and adding $\sum C_i$ to $\mathcal{C}_B$ increases the number of non-dominated solutions in 50% of instances, i.e. instances *mk01*, *mk03*, *mk04*, *mk05* and *mk09*.

6.3. Analysis of elite solutions

In the previous analysis of the number of non-dominated solutions, the values of the criteria were not considered. Now, let us evaluate elite solutions to determine which set of criteria effectively leads to improving the

convergence of the front. The first criterion considered is the makespan since it is the most studied in the literature and its quality can strongly explain the efficiency of our approach. Then the other criteria are analyzed. Only neighborhood structure N_2 is considered in the analysis.

6.3.1. Elite solutions for the makespan

The results for the makespan for each set of criteria are illustrated in Table 8, which details the minimum value Min, the average value Av and the percentage of error $Per(\%)$ generated by the average value with respect to the best known value BKV (González et al. (2015)) after running ten times our approach. The value of the makespan is underlined for instances where the best known value is obtained, and the minimum value is written in bold for the remaining instances.

Inst	*BKV*	$\mathcal{C}_A$			$\mathcal{C}_B$			$\mathcal{C}_C$		
		Min	Av	$Per(\%)$	Min	Av	$Per(\%)$	Min	Av	$Per(\%)$
mk01	40	42	42.0	5.0	<u>40</u>	40.0	0.0	<u>40</u>	40.0	0.0
mk02	26	**28**	29.0	11.5	29	30.0	15.4	**28**	29.0	11.5
mk03	204	<u>204</u>	204.8	0.4	<u>204</u>	204.0	0.0	<u>204</u>	204.0	0.0
mk04	60	67	68.8	14.6	<u>60</u>	61.5	2.5	62	63.8	6.3
mk05	172	176	176.3	2.5	**174**	174.8	1.6	175	175.5	2.0
mk06	57	**67**	69.5	21.9	69	71.3	25	70	70.5	23.7
mk07	139	144	146.3	5.3	**142**	143.0	2.9	148	148.3	6.7
mk08	523	<u>523</u>	523.0	0.0	<u>523</u>	523.0	0.0	<u>523</u>	523.0	0.0
mk09	307	320	327.5	6.7	<u>307</u>	308.0	0.3	<u>307</u>	309.3	0.7
mk10	196	221	223.0	13.8	**216**	216.5	10.5	228	231.3	18.0

Table 8: Elite solutions for makespan

Minimizing the makespan performs better when the approach optimizes Set $\mathcal{C}_B$ since, in 6 instances, the best known value is obtained (*mk01*, *mk03*, *mk04*, *mk07*, *mk08* and *mk09*). Figure 3 shows the sequences of operations on machines for instance *mk01* when the makespan is minimized. Besides, Set $\mathcal{C}_B$ has the best performance compared to Sets $\mathcal{C}_A$ and $\mathcal{C}_C$ in instances *mk05* with an error of 1.6% and *mk10* in spite of the large error (9.9%). In the instances *mk02* and *mk06*, the best known value is not obtained for any set and the error is quite large, e.g. 11.5% and 21.9% with Set $\mathcal{C}_A$. As illustrated in Section 6.2, few solutions are determined for these two instances that have large flexibility levels. This means that, to improve the convergence of the makespan, new strategies might be needed, in particular for instances with large flexibility levels.

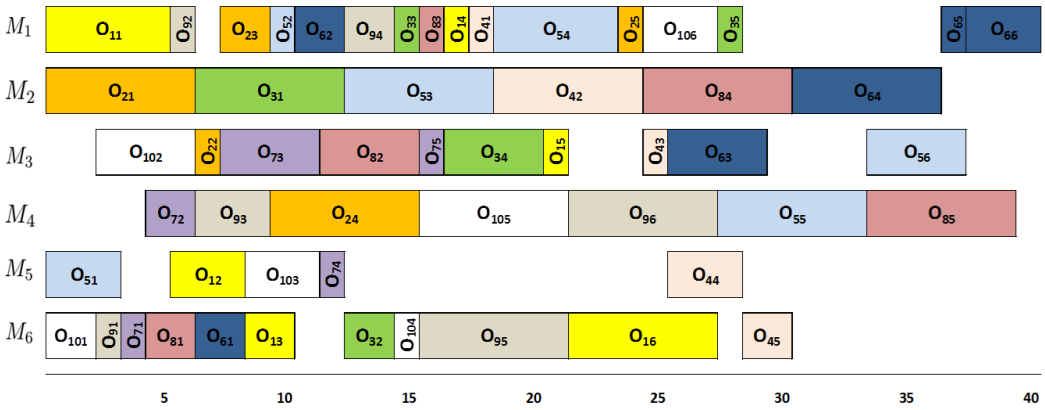

Figure 3: Gantt diagram when minimizing makespan in instance *mk01*

6.3.2. Elite solutions for maximum tardiness and total tardiness

The performances of the approach with criteria $T_{\max}$ and $\sum T_i$ are shown in Table 9 and Table 10. The meaning of columns is similar to Table 8, except for Column $Per(\%)$ that corresponds to the percentage of error generated by the average and the minimum value. This is due to the fact that there is a lack of results in the literature on instances from Brandimarte (1993). However, $Per(\%)$ gives insights on the homogeneity of our approach in generating good solutions in all ten runs.

Inst	$\mathcal{C}_A$			$\mathcal{C}_B$			$\mathcal{C}_C$		
	Min	*Av*	*Per*(%)	*Min*	*Av*	*Per*(%)	*Min*	*Av*	*Per*(%)
mk01	10	11.5	15.0	**9**	9.0	0.0	**9**	9.0	0.0
mk02	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0
mk03	77	77.0	0.0	**77**	77.0	0.0	**77**	77.0	0.0
mk04	27	27.5	1.9	**20**	20.0	0.0	**20**	20.3	1.3
mk05	100	101.0	0.5	**98**	98.0	0.0	99	99.8	0.8
mk06	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0
mk07	73	78.0	6.8	**70**	70.3	0.4	76	77.3	1.6
mk08	**310**	310.0	0.0	**310**	310.0	0.0	**310**	310.0	0.0
mk09	137	143.0	4.4	**123**	127.3	3.5	**123**	125.5	2.0
mk10	38	42.8	12.5	**30**	31.3	4.2	50	50.5	1.0

Table 9: Elite solutions for $T_{\max}$

It can be observed from Table 9 that optimizing Set $\mathcal{C}_B$ provides the best

results for $T_{\max}$ in all instances. This means that the addition of $\sum U_i$ improves the convergence not only for the makespan but also for $T_{\max}$. Besides, including $\sum C_i$ does not affect the quality of $T_{\max}$, since the minimum value is obtained in 7 instances. For the remaining 3 instances, the obtained $T_{\max}$ is very close to the best value in instance *mk05* (99 for set $\mathcal{C}_C$ against 98 for set $\mathcal{C}_B$), and the only bad result is obtained for instance *mk10* where the gap is really large (30 for $\mathcal{C}_B$ and 50 for $\mathcal{C}_C$). This table also shows that the approach is quite stable for $T_{\max}$ as the average value is always close to the minimum value for the three sets $\mathcal{C}_A$, $\mathcal{C}_B$ and $\mathcal{C}_C$.

Inst	$\mathcal{C}_A$			$\mathcal{C}_B$			$\mathcal{C}_C$		
	Min	*Av*	*Per*(%)	*Min*	*Av*	*Per*(%)	*Min*	*Av*	*Per*(%)
mk01	39	44.8	14.7	**18**	18.2	1.1	19	20.5	7.9
mk02	<u>0</u>	0.0	0.0	<u>0</u>	0.0	0.0	<u>0</u>	0.0	0.0
mk03	523	576.3	10.2	**275**	278.0	1.1	321	325.8	1.5
mk04	170	198.0	16.5	**108**	112.4	4.1	125	130.5	4.4
mk05	1102	1179.3	7.0	**788**	811.2	2.9	854	916.0	7.3
mk06	<u>0</u>	0.0	0.0	<u>0</u>	0.0	0.0	<u>0</u>	0.0	0.0
mk07	895	995.3	11.2	**808**	840.0	4.0	845	887.5	5.0
mk08	4786	4832.0	1.0	**2932**	3102.4	5.8	3387	3496.5	3.2
mk09	2060	2292.5	11.3	**1048**	1144.0	9.2	1270	1321.8	4.1
mk10	496	566.8	14.3	**340**	395.2	16.2	482	548.0	13.7

Table 10: Elite solutions for $\sum T_i$

For $\sum T_i$, optimizing Set $\mathcal{C}_B$ provides much better results than when optimizing the two other sets, since not only the best solutions are always obtained for $\mathcal{C}_B$, but there is also a large gap between the value obtained with $\mathcal{C}_B$ and the ones obtained with the two other sets (e.g. in instance *mk08* where 2932 is obtained with $\mathcal{C}_B$ whereas 4786 and 3387 are obtained with $\mathcal{C}_A$ and $\mathcal{C}_C$, respectively). Contrary to the case with $T_{\max}$, the value of $Per(\%)$ is sometimes large such as in instance *mk10* for the three sets. This can be explained by the fact that the criterion $\sum T_i$ is more difficult to solve than $T_{\max}$, which implies that additional strategies might be needed to make the approach more stable for $\sum T_i$.

6.3.3. Elite solutions for total number of tardy jobs and total completion times

$\sum U_i$ is one of the most difficult criteria to minimize, since it is nonlinear. This is because U_i is equal to 1 as soon as the completion time of job J_i is

strictly larger than its due date. The results obtained wih sets $\mathcal{C}_B$ and $\mathcal{C}_C$ are illustrated in Table 11. Note that $\mathcal{C}_B$ still leads to the best results for all instances. Further, adding criterion $\sum C_i$ to $\mathcal{C}_C$ has a small effect on the value of $\sum U_i$, in particular for instances with 15 and 20 jobs, i.e. *mk03*, *mk04*, *mk08*, *mk09* and *mk10*. $Per(\%)$ is sometimes large for $\mathcal{C}_B$, which can be explained by the small values of $\sum U_i$ such as in instance *mk05* in which $Min = 5$ and $Av = 5.8$ leading to $Per = 15.0$. For criterion $\sum C_i$, the only observation that can be drawn is that $Per(\%)$ is quite small, which confirms the homogeneity of our approach for this criterion.

Inst	$\sum U_i$						$\sum C_i$		
	$\mathcal{C}_B$			$\mathcal{C}_C$			$\mathcal{C}_C$		
	Min	Av	$Per(\%)$	Min	Av	$Per(\%)$	Min	Av	$Per(\%)$
mk01	**3**	3.0	0.0	**3**	3.0	0.0	272	274.0	0.7
mk02	0	0.0	0.0	0	0.0	0.0	237	241.0	1.7
mk03	**5**	5.8	15.0	6	6.5	8.3	2100	2159.0	2.8
mk04	**6**	6.3	4.2	7	7.3	3.6	653	656.0	0.5
mk05	**12**	12.5	4.2	**12**	12.8	6.3	1789	1851.0	3.5
mk06	0	0.0	0.0	0	0.0	0.0	657	662.0	0.7
mk07	**15**	15.5	3.3	**15**	15.0	0.0	2142	2177.0	1.6
mk08	**16**	17.0	6.3	17	17.8	4.4	7021	7130.0	1.5
mk09	**13**	14.8	13.5	15	15.8	5.0	4873	4919.0	0.9
mk10	**11**	12.3	11.4	14	14.3	1.8	3773	3835.0	1.6

Table 11: Elite solutions for $\sum U_i$ and $\sum C_i$

6.4. Analysis of the diversity

This section aims at validating that the front of Pareto is diverse in solutions, so that the decision maker can choose among representative solutions in different regions of the front. The evaluation of the diversity is only presented for set $\mathcal{C}_B$, because this is the set of criteria for which our approach found the best results. Table 12 includes the minimum value (Min), the average value (Av), the maximum value (Max) and the standard deviation (σ) for each instance. Column *AvNDS* recalls the average number of non-dominated solutions of Table 7. For example, for instance *mk01*, on average 4.8 non-dominated solutions are generated (Column *AvNDS*) and, when considering the maximum spread D, the minimum (resp. maximum) distance between extreme solutions is 7.9 (resp. 10.2), the average distance is 9.4 and

$\sigma = 1.0$. Besides, for the average distance between nearest solutions (spacing SP), the minimum (resp. maximum) distance is 1.2 (resp. 2.6) with an average of 2.0 and $\sigma = 0.6$. If an ideal situation is taken as reference, the tendency of σ is to be equal to zero.

Inst	*AvNDS*	D				SP			
		Min	*Av*	*Max*	σ	*Min*	*Av*	*Max*	σ
mk01	4.8	7.9	9.4	10.2	1.0	1.2	2.0	2.6	0.6
mk02	1.5	0.0	1.3	3.2	1.6	0.0	0.1	0.5	0.2
mk03	1.8	0.0	17.6	30.7	13.5	0.0	0.3	0.8	0.3
mk04	35.0	74.7	79.9	84.2	4.3	3.8	6.3	11.4	3.5
mk05	26.0	293.9	341.6	396.7	43.7	5.3	8.7	10.0	2.3
mk06	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
mk07	27.8	154.1	243.9	311.1	65.9	8.3	13.1	16.0	3.6
mk08	16.8	98.1	482.5	728.9	289.0	0.5	18.0	29.8	12.4
mk09	33.0	410.5	567.4	710.8	136.1	13.7	18.5	21.7	3.4
mk10	23.8	251.1	321.6	384.1	60.6	19.3	23.7	32.2	5.9

Table 12: Results for maximum spread D and spacing SP for $\mathcal{C}_B$

When analyzing the spacing SP, the ideal performance corresponds to small values of SP and σ. Looking at Table 12, it is possible to infer that, in the instances with high flexibility *mk02*, *mk03* and *mk06* (i.e. $flex > 3$), the spacing values are very close to zero, which is mainly explained by the results of Table 7 in which only there are very few solutions in the set of non-dominated solutions. Further, it is possible to infer two other trends. The first trend is that spacing seems to increase with the size of the problem, in particular for instances *mk07*, *mk09* and *mk10* with 20 jobs. The second trend is that the number of non-dominated solutions does not affect the spacing since, even though there are large values for *AvNDS* in instances *mk04*, *mk05* and *mk08*, the spacing is small (3.8, 5.3 and 0.5). Moreover, for instance *mk04*, *AvNDS* = 35.0 and the average spacing is 6.3 while, in instance *mk10* with a lower number of non-dominated solutions (*AvNDS* = 23.8), the average spacing is 23.7.

Concerning the maximum spread D, an efficient value corresponds to a large average and a low σ. When looking at Table 12, it is possible to infer that the maximum spread is large for instances with medium flexibility ($2 < flex < 3$) such as instances *mk07*, *mk09* and *mk10*. However, the size of the

problem, especially the number of jobs, seems to not influence the maximum spread as the values (i.e. Min, Av, Max) obtained in instance *mk05* with 15 jobs are larger than the corresponding values in instances *mk07*, *mk08* and *mk10*. It is also possible to infer, by observing for example instances *mk07* and *mk10*, that there is no trend between the number of non-dominated solutions and the maximum spread. These observed trends of diversity can be considered as additional arguments to confirm the performance of our approach.

6.5. Analysis of HyperVolume (HV) and Mean Ideal Distance (MID)

In this section, we expand the analysis for $\mathcal{C}_B$ using the HV and MID measures, which must be maximized and minimized, respectively. Table 13 gives the minimum value (Min), the average value (Av) and the maximum value (Max) of these measures for each instance after running ten times our approach. The value $Per(\%) = (Av - Min)/Min$ for MID and $Per(\%) = (Max - Av)/Max$ for HV. To calculate HV, the coordinates of the reference point is fixed to 10,000 for $C_{\max}$ and $T_{\max}$, to 100,000 for $\sum T_i$ and to two times the number of jobs for $\sum U_i$. For example, the coordinates of the reference point are $(10,000; 10,000; 100,000; 20)$ for instances with 10 jobs. To improve the scale of distance when calculating MID, the criterion $\sum T_i$ of each solution has been divided by the number of jobs.

It is very difficult to assess the quality of our approach due to the lack of previous values for HV and MID for the regular criteria studied in this paper. However, we can observe that the values of HV tend to 100% for instances *mk02* and *mk06* with few solutions and large flexibility. Our approach is stable since σ is closer to zero for all instances and $Per(\%)$ is lower than 3.4%. The analysis of MID also reveals uniformity in the results, which is explained by small values of σ and $Per(\%)$ except for two instances: *mk08* ($\sigma = 6.27$) and *mk09* ($\sigma = 5.35$), which could be explained by the number of jobs.

6.6. Comparison with previous approaches

The performance of our approach is compared against the Multi-Objective Differential Evolution algorithm ($MODE$) proposed in Wisittipanich and Kachitvichyanukul (2014) to minimize the makespan and the total tardiness. In $MODE$, the Pareto front was obtained by evaluating five search strategies (MODE-$ms1$, MODE-$ms2$, MODE-$ms3$, MODE-$ms4$ and MODE-$ms5$) and the MOPSO algorithm proposed in Nguyen and Kachitvichyanukul (2010).

Inst	*Size*	*flex*	*HyperVolume (HV)*					*Mean Ideal Distance (MID)*				
	$m \times n$		*Min*	*Av*	*Max*	*Per*(%)	σ	*Min*	*Av*	*Max*	*Per*(%)	σ
mk01	6×10	2.09	0.80	0.82	0.85	3.4	0.01	43.22	43.95	45.29	1.69	0.49
mk02	6×10	4.10	0.95	0.98	1.00	1.3	0.02	29.00	30.21	31.00	4.19	0.65
mk03	8×15	3.01	0.75	0.77	0.80	3.7	0.01	219.02	224.34	231.47	2.43	4.60
mk04	8×15	1.91	0.65	0.67	0.69	3.2	0.01	73.86	75.41	77.68	2.10	1.15
mk05	4×15	1.71	0.50	0.51	0.52	2.3	0.00	218.26	221.08	227.37	1.29	2.38
mk06	15×10	3.27	0.99	0.99	0.99	0.0	0.00	69.00	71.33	73.00	3.38	1.50
mk07	5×20	2.83	0.53	0.54	0.55	2.0	0.01	173.05	176.19	182.29	1.82	3.01
mk08	10×20	1.43	0.47	0.48	0.49	2.2	0.01	640.25	652.11	662.71	1.85	6.27
mk09	10×20	2.53	0.54	0.55	0.57	2.8	0.01	364.68	376.42	384.45	3.22	5.35
mk10	15×20	2.98	0.55	0.58	0.60	3.4	0.01	230.97	238.25	246.64	3.15	4.72

Table 13: Analysis of HV and MID for $\mathcal{C}_B$

Note that the solutions of the Pareto front are mainly obtained from those determined by MODE-$ms1$, MODE-$ms2$, MODE-$ms3$ and MODE-$ms5$. The comparison is based on a set of eight problem instances used in Wisittipanich and Kachitvichyanukul (2014), which includes five problem instances from Dauzère-Pérès et al. (1998) (*dpp02a*, *dpp09a*, *dpp11a*, *dpp16a* and *dpp18a*) and three problem instances from Brandimarte (1993) (*mk04*, *mk07* and *mk09*). The due dates of jobs were determined using the expression provided in He et al. (1993). Figure 4 shows the non-dominated solutions obtained by N_1 and N_2 for instance *dpp02a* with set $\mathcal{C}_A$.

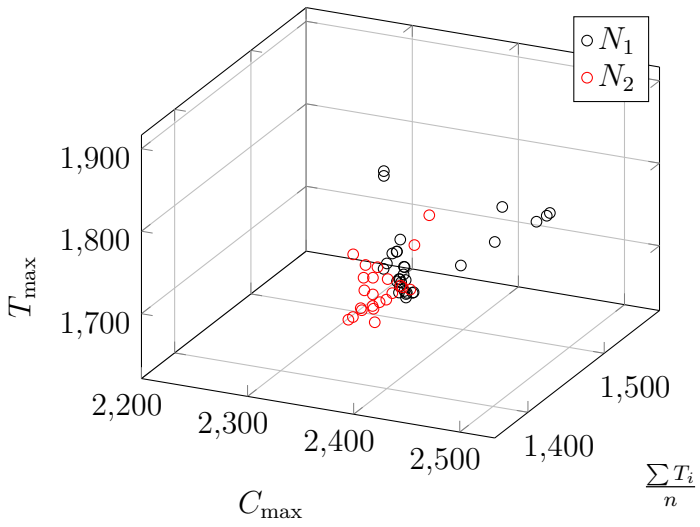

Figure 4: Set of non-dominated solutions for instance *dpp02a* and set $\mathcal{C}_A$

To compare the results of GMD with $MODE$, the Weak OutPerformance metric ($\text{WOP}_{x,o}$) and the set coverage metric ($SCM_{x,o}$) are analyzed. $\text{WOP}_{x,o}$ evaluates the dominance between two sets of non-dominated solutions s_x and s_o. $SCM_{x,o}$ is the ratio of solutions of s_o weakly dominated by solutions of s_x (Zitzler (1999)).

Table 14 provides for both GMD and $MODE$ the number of non-dominated solutions in column NDS, the Pareto set in column *Solutions*, the set coverage metric in column SCM_1 and SCM_2, and the Weak OutPerformance metric in column WOP. For example in instance *dpp09a*, our approach finds 12 non-dominated solutions when $MODE$ finds 14 non-dominated solutions, and 71% (10 of 14) of the solutions of $MODE$ are weakly dominated by at least one solution of GMD, and $WOP = 0$ means that there are weakly dominated solutions in both sets of non-dominated solutions. Table 14 also shows that the performance of GMD and $MODE$ are comparable. Our approach is better for three instances (*dpp*02*a*, *dpp*11*a* and *dpp*16*a*, since WOP and SCM_1 are equal to 1 and SCM_2 is equal to 0), and $MODE$ is better for three instances (*dpp*18*a*, *mk*07 and *mk*09). Figure 5 depicts the sets of non-dominated solutions obtained by GMD and $MODE$ for instance *dpp02a*. Note that GMD was ran with criteria $C_{\max}$ and $\sum T_i$.

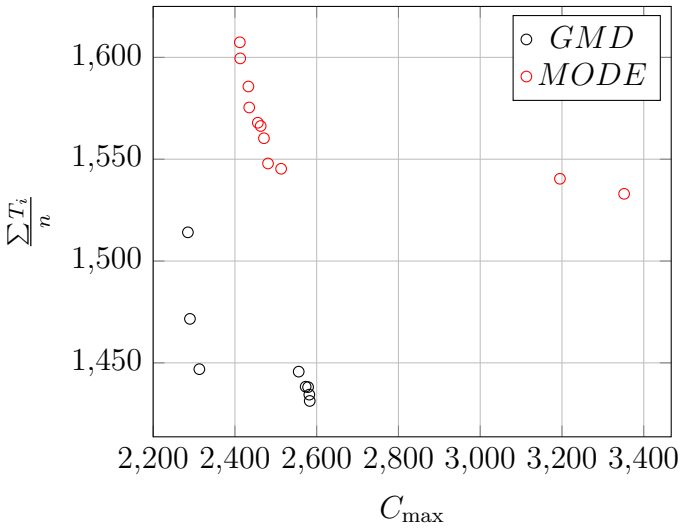

Figure 5: Comparison of GMD and $MODE$ using the instance *dpp02a*

Inst	*GMD*		*MODE*				
	NDS	*Solutions*	*NDS*	*Solutions*	SCM_1	SCM_2	WOP
dpp02a	8	(2285, 15141) (2573, 14383) (2290, 14716) (2579, 14381) (2313, 14469) (2582, 14344) (2556, 14457) (2583, 14313)	11	(2412, 16073.65) (2471, 15602.65) (2413, 15994.65) (2481, 15478.65) (2433, 15857.65) (2513, 15452.65) (2435, 15754.65) (3195, 15403.65) (2456, 15679.65) (3352, 15339.65) (2463, 15663.65)	1.00	0.00	1
dpp09a	12	(2146, 21689) (2239, 19839) (2148, 21337) (2280, 19692) (2150, 20253) (2284, 19659) (2155, 20244) (2288, 19637) (2176, 19953) (2290, 19571) (2180, 19951) (2306, 19474)	14	(2168, 21022.3) (2196, 20098.3) (2170, 20909.3) (2198, 20088.3) (2172, 20734.3) (2200, 20068.3) (2173, 20691.3) (2201, 20015.3) (2174, 20228.3) (2204, 19650.3) (2191, 20101.3) (2207, 19600.3) (2193, 20099.3) (2220, 19430.3)	0.71	0.50	0
dpp11a	17	(2155, 20829) (2194, 20070) (2172, 20821) (2195, 20033) (2176, 20503) (2196, 19948) (2177, 20298) (2203, 19935) (2178, 20265) (2332, 19894) (2184, 20246) (2345, 19886) (2185, 20181) (2459, 19489) (2186, 20137) (2461, 19486) (2193, 20086)	9	(2234, 21190.61) (2330, 20385.61) (2238, 21109.61) (2866, 20301.61) (2273, 21073.61) (2951, 20296.61) (2282, 20537.61) (3152, 20226.61) (2307, 20496.61)	1.00	0.00	1
dpp16a	10	(2389, 31441) (2545, 30303) (2392, 30779) (2597, 30180) (2399, 30760) (2607, 30177) (2409, 30360) (2607, 30127) (2435, 30313) (2865, 30020)	11	(2582, 33004.83) (2643, 31674.83) (2584, 32930.83) (2644, 31577.83) (2607, 32909.83) (2652, 31294.83) (2610, 32426.83) (2657, 31187.83) (2636, 32159.83) (3552, 31130.83) (2642, 32009.83)	1.00	0.00	1
dpp18a	7	(2237, 29053) (2298, 28626) (2238, 29033) (2321, 28563) (2246, 28924) (2327, 28530) (2280, 28722)	16	(2227, 26698.92) (2254, 25651.92) (2229, 26431.92) (2256, 25447.92) (2233, 26428.92) (2276, 25307.92) (2234, 26363.92) (2277, 25190.92) (2237, 26354.92) (2278, 25123.92) (2238, 26135.92) (2280, 24991.92) (2251, 26097.92) (2282, 24789.92) (2253, 25754.92) (2789, 24078.92)	0.00	1.00	-1
mk04	6	(61, 479) (64, 455) (62, 464) (73, 438) (63, 457) (75, 429)	6	(64, 445.46) (67, 390.46) (65, 439.46) (69, 388.46) (66, 401.46) (74, 386.46)	0	0.50	-1
mk07	5	(144, 1661) (184, 1618) (147, 1660) (198, 1599) (150, 1629)	7	(143, 1789.49) (150, 1424.49) (144, 1492.49) (152, 1413.49) (146, 1476.49) (154, 1341.18) (147, 1465.49)	0	1.00	-1
mk09	7	(307, 3498) (328, 3342) (311, 3454) (332, 3338) (325, 3386) (334, 3295) (326, 3347)	7	(307, 3216.1) (403, 3166.1) (309, 3206.1) (409, 3084.1) (311, 3191.1) (418, 2993.1) (315, 3190.1)	0	1.00	-1

Table 14: Comparison of *GMD* and *MODE*

7. Conclusions

In this paper, we proposed a general local search approach to determine Pareto fronts for the Multi-Objective Flexible Job-shop Scheduling Problem (MOFJSP) for any combination of regular scheduling criteria. Regular criteria correspond to various customer service objectives, which are important in a competitive environment. The local search approach is based on two neighborhood structures (N_1 and N_2), that consist in moving a critical operation in the conjunctive graph, sufficient conditions to determine the feasibility of a move without transforming the graph, and an estimation function to select the best move. A hierarchical test is proposed to quickly update the set of non-dominated solutions during the search, and four search strategies ($\mathsf{T_1}$, $\mathsf{T_2}$, $\mathsf{T_3}$ and $\mathsf{T_4}$) have been proposed.

Three sets of criteria to optimize are considered in our experiments. The experiments showed that N_2 is the dominant neighborhood structure and generates the largest number of non-dominated solutions. Besides, a combination of Strategies $\mathsf{T_3}$ and $\mathsf{T_4}$ is sufficient to solve the MOFJSP with all sets of criteria.

In future research, we would like to study how our approach can be improved for specific regular criteria. New dedicated properties could be used to accelerate the search or avoid being stuck in local optima for criteria such as $\sum T_i$ or $\sum U_i$, that are more complex to handle. Another research avenue is the use of sophisticated metaheuristics that could help to better diversify the search process to reach promising regions. We also intend to work on extending our approach to search for more diverse solutions by better considering different types of criteria.

Acknowledgements

This research is partially supported by the government of Colombia in the program of fellowships Colfuturo, the embassy of France in Colombia, and by the project *"Formulation and validation of heuristics for optimizing the customer service in flexible configurations in the Tolima region"*, identified with the code 17-465-INT, which is financed by the Universidad de Ibagué(Colombia) and with the advisory of École des Mines de Saint Etienne (France) and Qassim University (Saudi Arabia).

References

References

Ahmadi, E., Zandieh, M., Farrokh, M., Emami, S. M., 2016. A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. Computers & Operations Research 73, 56 – 66.

Amjad, M., Butt, S., Kousar, R., Ahmad, R., Hassam, M., Faping, Z., Anjum, N., Asgher, U., 2018. Recent research trends in genetic algorithm based flexible job shop scheduling problems. Mathematical Problems in Engineering 2018, 1–32.

Bagheri, A., Zandieh, M., 2011. Bi-criteria flexible job-shop scheduling with sequence-dependent setup times–variable neighborhood search approach. Journal of Manufacturing Systems 30 (1), 8–15.

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 41 (3), 157–183.

Chaudhry, I. A., Khan, A. A., 2016. A research survey: review of flexible job shop scheduling techniques. International Transactions in Operational Research 23 (3), 551–591.

Chiang, T.-C., Lin, H.-J., 2013. A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. International Journal of Production Economics 141 (1), 87–98.

Chun, W., Zhicheng, J., Yan, W., 2013. A novel memetic algorithm based on decomposition for multiobjective flexible job shop scheduling problem. Mathematical Problems in Engineering 2017, 1 –20, meta-heuristics for manufacturing scheduling and logistics problems.

Dauzère-Pérès, S., Paulli, J., 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Annals of Operations Research 70, 281–306.

Dauzère-Pérès, S., Roux, W., Lasserre, J., 1998. Multi-resource shop scheduling with resource flexibility. European Journal of Operational Research 107 (2), 289–305.

Deb, K., et al., 2001. Multi-objective optimization using evolutionary algorithms. Vol. 2012. John Wiley and Sons Chichester.

Gao, J., Gen, M., Sun, L., Zhao, X., 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Computers & Industrial Engineering 53 (1), 149–162.

Gao, K., Suganthan, P., Pan, Q., Chua, T., Cai, T., Chong, C., 2014. Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. Journal of Intelligent Manufacturing, 1–12.

García-León, A., Dauzère-Pérès, S., Mati, Y., 2015. Minimizing regular criteria in the flexible job job-shop scheduling problem. 7th Multidisciplinary International Scheduling Conference: Theory and Applications, 443–456.

Genova, K., Kirilov, L., Guliashki, V., 2015. A survey of solving approaches for multiple objective flexible job shop scheduling problems. Cybernetics and Information Technologies 15 (2), 3–22.

González, M. A., Vela, C. R., Varela, R., 2015. Scatter search with path relinking for the flexible job shop scheduling problem. European Journal of Operational Research 245 (1), 35–45.

He, Z., Yang, T., Deal, D., 1993. A multiple-pass heuristic rule for job shop scheduling with due dates. International Journal of Production Research 31 (11), 2677–2692.

Jia, S., Hu, Z.-H., 2014. Path-relinking tabu search for the multi-objective flexible job shop scheduling problem. Computers & Operations Research 47, 11–26.

Johnzen, C., Dauzère-Pérès, S., Vialletelle, P., 2011. Flexibility measures for qualification management in wafer fabs. Production Planning and Control 22 (1), 81–90.

Lei, D., Li, M., Wang, L., 2018. A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. IEEE Transactions on Cybernetics, 1–13.

Li, J. J. W. M. L., 2011. Hybrid genetic algorithm for flexible job-shop scheduling with multi-objective. Journal of Information & Computational Science 8 (11), 2197.

Li, J.-Q., Pan, Q.-K., Gao, K.-Z., Aug 2011. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. The International Journal of Advanced Manufacturing Technology 55 (9), 1159–1169.

Li, J.-Q., Pan, Q.-K., Tasgetiren, M. F., 2014. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. Applied Mathematical Modelling 38 (3), 1111–1132.

Liu, H., Abraham, A., Choi, O., Moon, S. H., 2006. Variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems. In: Simulated Evolution and Learning. Springer, pp. 197–204.

Lu, C., Li, X., Gao, L., Liao, W., Yi, J., 2017. An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times. Computers & Industrial Engineering 104, 156–174.

Mati, Y., Dauzère-Pérès, S., Lahlou, C., 2011. A general approach for optimizing regular criteria in the job-shop scheduling problem. European Journal of Operational Research 212 (1), 33–42.

Mokhtari, H., Hasani, A., 2017. An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. Computers & Chemical Engineering 104, 339 – 352.

Mousakhani, M., 2013. Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. International Journal of Production Research 51 (12), 3476–3487.

Nguyen, S., Kachitvichyanukul, V., Jul. 2010. Movement strategies for multi-objective particle swarm optimization. Int. J. Appl. Metaheuristic Comput. 1 (3), 59–79.

Nie, L., Gao, L., Li, P., Li, X., Aug 2013. A gep-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. Journal of Intelligent Manufacturing 24 (4), 763–774.

Pérez, M. A. F., Raupp, F. M. P., Apr 2016. A newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem. Journal of Intelligent Manufacturing 27 (2), 409–416.

Rahmati, S. H. A., Zandieh, M., Yazdani, M., Feb 2013. Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. The International Journal of Advanced Manufacturing Technology 64 (5), 915–932.

Sadrzadeh, A., Dec 2013. Development of both the ais and pso for solving the flexible job shop scheduling problem. Arabian Journal for Science and Engineering 38 (12), 3593–3604.

Schott, J. R., 1995. Fault tolerant design using single and multicriteria genetic algorithm optimization. Tech. rep., AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH.

Shao, X., Liu, W., Liu, Q., Zhang, C., Aug 2013. Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem. The International Journal of Advanced Manufacturing Technology 67 (9), 2885–2901.

Shen, X.-N., Han, Y., Fu, J.-Z., Nov 2017. Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. Soft Computing 21 (21), 6531–6554.

Shi-Jin, W., Bing-Hai, Z., Li-Feng, X., 2008. A filtered-beam-search-based heuristic algorithm for flexible job-shop scheduling problem. International Journal of Production Research 46 (11), 3027–3058.

Singh, M. R., Singh, M., Mahapatra, S. S., Jagadev, N., Aug 2016. Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem. The International Journal of Advanced Manufacturing Technology 85 (9), 2353–2366.

Tay, J. C., Ho, N. B., 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers & Industrial Engineering 54 (3), 453–473.

Türkyılmaz, A., Bulkan, S., 2015. A hybrid algorithm for total tardiness minimisation in flexible job shop: genetic algorithm with parallel vns execution. International Journal of Production Research 53 (6), 1832–1848.

Vilcot, G., Billaut, J.-C., 2011. A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem. International Journal of Production Research 49 (23), 6963–6980.

Wang, L., Wang, S., Liu, M., 2013. A pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem. International Journal of Production Research 51 (12), 3574–3592.

Wisittipanich, W., Kachitvichyanukul, V., 2014. A pareto-archived differential evolution algorithm for multi-objective flexible job shop scheduling problems. Logistics Operations, Supply Chain Management and Sustainability, 325–339.

Xia, W., Wu, Z., 2005. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computers & Industrial Engineering 48 (2), 409–425.

Xing, L.-N., Chen, Y.-W., Yang, K.-W., Jun 2009. An efficient search method for multi-objective flexible job shop scheduling problems. Journal of Intelligent Manufacturing 20 (3), 283–293.

Xiong, J., ning Xing, L., wu Chen, Y., 2013. Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. International Journal of Production Economics 141 (1), 112 – 126, metaheuristics for manufacturing scheduling and logistics problems.

Xiong, J., Tan, X., Yang, K.-w., Xing, L.-n., Chen, Y.-w., 2012. A hybrid multiobjective evolutionary approach for flexible job-shop scheduling problems. Mathematical Problems in Engineering 2012, 1–27.

Yuan, Y., Xu, H., Jan 2015. Multiobjective flexible job shop scheduling using memetic algorithms. IEEE Transactions on Automation Science and Engineering 12 (1), 336–353.

Zhang, G., Shao, X., Li, P., Gao, L., 2009. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Computers & Industrial Engineering 56 (4), 1309–1318.

Zhang, R., Wu, C., 2011. A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardinessobjective. Computers & Operations Research 38 (5), 854–867.

Zitzler, E., 1999. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. thesis, ETH Zurich, Switzerland.

Zitzler, E., Brockhoff, D., Thiele, L., 2007. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: Evolutionary multi-criterion optimization. Springer, pp. 862–876.

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