Cooperation between Branch and Bound and Evolutionary Approaches to solve a BiObjective Flow Shop Problem

Matthieu Basseur, Julien Lemesre, Clarisse Dhaenens and El-Ghazali Talbi

Laboratoire d'Informatique Fondamentale de Lille (LIFL), UMR CNRS 8022,
University of Lille, 59655 Villeneuve d'Asq Cedex, France.
{basseur,lemesre,dhaenens,talbi}@lifl.fr

Abstract. Over the years, many techniques have been established to solve NP-Hard Optimization Problems and in particular multiobjective problems. Each of them are efficient on several types of problems or instances. We can distinguish exact methods dedicated to solve small instances, from heuristics - and particularly metaheuristics - that approximate best solutions on large instances. In this article, we firstly present an efficient exact method, called the two-phases method. We apply it to a biobjective Flow Shop Problem to find the optimal set of solutions. Exact methods are limited by the size of the instances, so we propose an original cooperation between this exact method and a Genetic Algorithm to obtain good results on large instances. Results obtained are promising and show that cooperation between antagonist optimization methods could be very efficient.

1 Introduction

A large part of real-world optimization problems are of multiobjective nature. In trying to solve Multiobjective Optimization Problems (MOPs), many methods scalarize the objective vector into a single objective. Since several years, interest concerning MOPs area with Pareto approaches always grows. Many of these studies use Evolutionary Algorithms to solve MOPs [1–3] and only few approaches propose exact methods such as a classical branch and bound with Pareto approach, an ϵ-constraint method and the two-phases method.

In this paper, we propose to combine the two types of approaches: a metaheuristic and an exact method. Therefore, we firstly present a two-phases method developed to exactly solve a BiObjective Flow Shop Problem (BOFSP) [4]. In order to optimize instances which are too large to be solved exactly, we propose and present cooperation methods between Genetic Algorithms (GAs) and the two-phases method.

In section II, we define MOPs and we present a BOFSP. In section III, we present the two-phases method applied to the BOFSP, and computational results. In section IV, we present cooperation schemes between GA and the two-phases method. Section V presents results on non-solved instances. In the last

section, we discuss on effectiveness of this approach and perspectives of this work.

2 A BiObjective Flow Shop Problem (BOFSP)

2.1 Multiobjective Optimization Problems (MOPs

Although single-objective optimization may have a unique optimal solution, MOPs present a set of optimal solutions which are proposed to a decision maker. So, before presenting BOFSP, we have to describe and define MOPs in a general case. We assume that a solution x to such a problem can be described by a decision vector $(x_1, x_2, ..., x_n)$ in the decision space X. A cost function $f : X \rightarrow Y$ evaluates the quality of each solution by assigning it an objective vector $(y_1, y_2, ..., y_p)$ in the objective space Y (see Fig. 1). So, multiobjective optimization consists in finding the solutions in the decision space optimizing (minimizing or maximizing) p objectives.

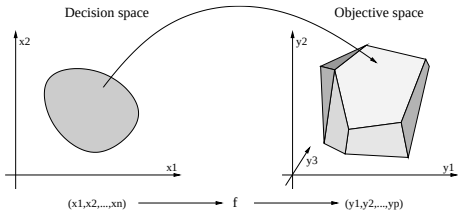

Fig. 1. Example of MOP

For the following definitions, we consider the minimization of p objectives. In the case of a single objective optimization, comparison between two solutions x and x' is immediate. For multiobjective optimization, comparing two solutions x and x' is more complex. Here, there exists only a partial order relation, known as the Pareto dominance concept:

Definition 1. *A solution x dominates a solution x' if and only if:*

$$\begin{cases} \forall k \in [1..p], f_k(x) \leq f_k(x') \\ \exists k \in [1..p] / f_k(x) < f_k(x') \end{cases}$$

In MOPs, we are looking for Pareto Optimal solutions:

Definition 2. *A solution is Pareto optimal if it is not dominated by any other solution of the feasible set.*

The set of optimal solutions in the decision space X is denoted as the Pareto set, and its image in the objective space is the Pareto front. Here we are interested in a apriori approach where we want to find every Pareto solutions.

2.2 Flow Shop Problem (FSP)

The FSP is one of the numerous scheduling problems. Flow-shop problem has been widely studied in the literature. Proposed methods for its resolution vary between exact methods, as the branch & bound algorithm [5], specific heuristics [6] and meta-heuristics [7]. However, the majority of works on flow-shop problem studies the problem in its single criterion form and aims mainly to minimize makespan, which is the total completion time. Several bi-objective approaches exist in the literature. Sayin et al. proposed a branch and bound strategy to solve the two-machine flow-shop scheduling problem, minimizing the makespan and the sum of completion times [5]. Sivrikaya-Serifoglu et al. proposed a comparison of branch & bound approaches for minimizing the makespan and a weighted combination of the average flowtime, applied to the two-machine flow-shop problem [8]. Rajendran proposed a specific heuristic to minimize the makespan and the total flowtime [6]. Nagar et al. proposed a survey of the existing multicriteria approaches of scheduling problems [7].

FSP can be presented as a set of N jobs $J_1, J_2, \ldots, J_N$ to be scheduled on M machines. Machines are critical resources: one machine cannot be assigned to two jobs simultaneously. Each job J_i is composed of M consecutive tasks $t_{i1}, \ldots, t_{iM}$, where t_{ij} represents the j^{th} task of the job J_i requiring the machine m_j. To each task t_{ij} is associated a processing time p_{ij}. Each job J_i must be achieved before its due date d_i. In our study, we are interested in permutation FSP where jobs must be scheduled in the same order on all the machines (Fig. 2).

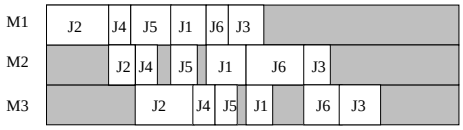

Fig. 2. Example of permutation Flow-Shop

In this work, we minimize two objectives: C_{max}, the makespan (Total completion time), and T, the total tardiness. Each task t_{ij} being scheduled at the time s_{ij}, the two objectives can be computed as follows:

$$C_{max} = Max\{s_{iM} + p_{iM} | i \in [1 \ldots N]\}$$

$$T = \sum_{i=1}^{N} [max(0, s_{iM} + p_{iM} - d_i)]$$

In the Graham et. al. notation, this problem is denoted [9]: F/perm, $d_i/(C_{max}, T)$. C_{Max} minimization has been proven to be NP-hard for more than two machines, in [10]. The total tardiness objective T has been studied only a few times for M machines [11], but total tardiness minimization for one machine has been proven to be NP-hard [12]. The evaluation of the performances of our algorithm

has been realized on some Taillard benchmarks for the FSP [13], extended to the bi-objective case [14] (bi-objective benchmarks and results obtained are available on the web at `http://www.lifl.fr/~basseur`).

3 An exact approach to solve BOFSP: the Two-Phases Method (TPM)

On the Pareto front two types of solutions may be distinguished : the supported solutions, that may be found thanks to linear combinations of criteria, and non supported solutions [15]. As supported solutions are the vertices of the convex hull, they are nearer to the ideal optimal solution, and we can ask why it is important to look for non supported solutions. Figure 3, shows the importance of non-supported solutions. It represents the Pareto front for one instance of the bicriteria permutation flowshop with 20 jobs and 5 machines. This figure shows that for this example, only two Pareto solutions are supported (the extremes) and to get a good compromise between the two criteria, it is necessary to choose one of the non-supported solutions.

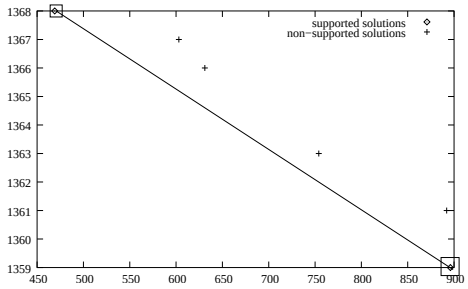

Fig. 3. Importance of non supported solutions (Pb: ta_20_5_02)

A lot of heuristic methods exist to solve multicriteria (and bicriteria) problems. In this section we are interested in developing an exact method able to enumerate all the Pareto solutions for a bicriteria flowshop problem.

A method, called the Two-Phases Method, has been proposed by Ulungu and Teghem to initially solve a bicriteria assignment problem [15]. This method is in fact a very general scheme that could be applied to other problems at certain conditions. It has not yet been very often used for scheduling applications where the most famous exact method for bicriteria scheduling problems is the ϵ-constraint approach, proposed by Haimes et al. [16]. This section presents the application of the scheme of the two-phases method to the bicriteria flow shop problem under study.

3.1 The two-phases method

Here we present the general scheme of the method. It proceeds in two phases. The first phase finds all the supported solutions and the second all the non-supported ones.

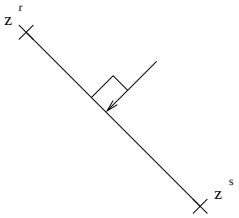

Fig. 4. Search direction.

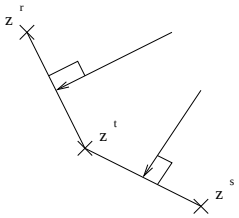

Fig. 5. New searches.

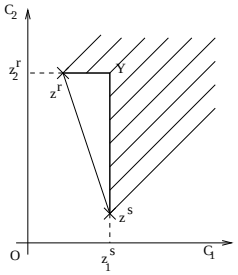

Fig. 6. Non supported solutions.

- The first phase consists in finding supported solutions with aggregations of the two objectives C_1 and C_2 in the form $\lambda_1 C_1 + \lambda_2 C_2$. It starts to find the two extreme efficient solutions that are two supported solutions. Then it looks recursively for the existence of supported solutions between two already found supported solutions z^r and z^s (we suppose $z_1^{(r)} < z_1^{(s)}$ and $z_2^{(r)} > z_2^{(s)}$) according to a direction perpendicular to the line $(z^r\ z^s)$ (see figure 4), while defining λ_1 and λ_2 as follows: $\lambda_1 = z_2^{(r)} - z_2^{(s)}$, $\lambda_2 = z_1^{(s)} - z_1^{(r)}$. Each new supported solution generates two new searches (see figure 5).
- The second phase consists in finding non-supported solutions. Graphically, any non-supported solution between z^r and z^s belongs to the triangle represented in figure 6. This triangle is defined by z^r, z^s and Y, which is the point $[z_1^{(s)}, z_2^{(r)}]$. Hence, the second phase consists in exploiting all the triangles, underlying each pair of adjacent supported solutions, in order to find the non-supported solutions.

3.2 Applying the two-phases method to a bicriteria flow shop problem

The interesting point of the two-phases method, is that it solves exactly a bicriteria problem without studying the whole search space. Hence we want to apply it to solve BOFSP for which the complete search space is too large to enable a complete enumeration. But this method is only a general scheme and applying it to a given problem requires a monocriterion exact method to solve aggregations.

As this scheduling problem (even in its monocriterion form) is NP-Hard, we decided to develop a branch-and-bound method. A large part of the success of a branch-and-bound is based on the quality of its lower bounds. As the makespan minimization has been widely studied, we have adapted an existing bound for

this criterion whereas for the total tardiness we propose a new bound. Details about these bounds may be found in [4].

The search strategy used is "a depth first search" where at each step, the node with the best bound is chosen. Moreover, a large part of the tardiness (T) value is generated by the last scheduled jobs. So the construction of solutions places jobs either at the beginning or at the end of the schedule, in order to have a precise estimation of the final T value fastly.

3.3 Improvements of the two-phases method

The two-phases method can be applied to any bicriteria problem. Applying it to scheduling problems allows improvements:

- Search of the extremes: The calculation of the extremes may be very long for scheduling problems as there exists a lot of solutions with the same value for one criterion. Hence, we propose to find extremes in a lexicographic order. A criterion is first optimized and then the second, without degrading the first one.
- Search intervals: The objective of the first phase is to find all the supported solutions in order to reduce the search space of the second phase. But when supported solutions are very near to each other, it is not interesting to look for all of them, as it will be very time consuming. Moreover, in the second phase, the search is, in fact, not reduced to the triangle shown on figure 6 but to the whole rectangle $(z_2^{(r)}, Y, z_1^{(s)}, 0)$. Hence, during the second phase, it is possible to find supported solutions that still exists. Then to avoid uninteresting branch-and-bounds we propose to execute a first phase only between solutions far from each other (a minimal distance is used).

3.4 Results

Table 1 presents results obtained with the two-phases method on the studied problems. The first column describes the instances of the problem: ta_number of jobs_number of machines_number of the instance. Then the three following columns indicate computational time with three different versions : the original two-phases method, the method with improvements proposed, and its parallel version[1]. It shows that both, improvements and parallelization allow to solve problems faster. Sequential runs have been executed on a 1.00Ghz machine. The parallel version has been executed on eight 1.1Ghz machines.

4 Using the two-phases approach to approximate the optimal Pareto front

4.1 Motivations

Exact methods are always limited by the size of the problem. Moreover, when the optimal Pareto front is not reached, these methods do not give good solutions. So,

[1] The parallel version is described in [4]

Table 1. Results of two-phases method.

Instances	Time		
	Original method	**With improvements**	**With parallelization**
ta_20_5_01	30"	17"	no need
ta_20_5_02	15'	14'	no need
ta_20_10_01	one week	2 days	1 day
ta_20_10_02	one week	2 days	1 day
ta_20_20_01	Unsolved	Unsolved	few weeks

for these problems, heuristics are usually proposed. In this section, we propose to use the adaptation of the TPM to improve Pareto fronts obtained with a heuristic. Firstly, we briefly present the hybrid GA which will cooperate with TPM. Then we propose several cooperation mechanisms between TPM and the hybrid GA.

4.2 An Adaptive Genetic/Memetic Algorithm (AGMA)

In order to optimize solutions of FSP, AGMA algorithm has been proposed in [17]. AGMA is firstly a genetic algorithm (GA) which proposes an adaptive selection between mutation operators. Crossover, selection and diversification operators are described in [18]. Moreover, AGMA proposes an original hybrid approach: the search alternates adaptively between a Genetic Algorithm and a Memetic Algorithm (MA). The hybridization works as follows: Let P_{PO*} be the value of the modification rate done on the Pareto front $PO*$ computed on the last generations of the GA. If this value goes below a threshold α, the MA is launched on the current GA population. When the MA is over, the Pareto front is updated, and the GA is re-run with the previous population (Algorithm 1).

Computational results presented in [17] show that we have a good approximation of the Pareto front. In order to improve these results, we propose some cooperative schemes between AGMA and TPM.

4.3 Cooperation between AGMA and TPM

Recently, interest for cooperation methods grows. A large part of them are hybrid methods, in which a first heuristic gives solution(s) to a second one which upgrades its (their) quality [19]. But different Optimization Methods (OMs) can cooperate in several ways as shown in figure 7. This cooperation can be sequential (a), often called hybridization. The search can also alternate between two OMs (b). The alternativity may be decided thanks to thresholds (c). Finally a cooperation can be established, with one method integrated in a mechanism of the second one with or without threshold (d).

Here we present three cooperation methods that combine the two-phases method (TPM) and the Adaptive Genetic/Memetic Algorithm (AGMA) pre-

Algorithm 1 AGMA algorithm

Create an initial population
while run time not reached **do**
  Make a GA generation with adaptive mutation
  Update $PO*$ an P_{PO*}
  if $P < \alpha$ **then**
    / Make a generation of MA on the population */*
    Apply crossover on randomly selected solutions of PO to create a set of new solutions.
    Compute the non-dominated set PO' on these solutions
    while New solutions found **do**
      Create the neighborhood N of each solution of PO'
      Let PO' be the non-dominated set of $N \bigcup PO'$
    end while
    Update $PO*$ an P_{PO*}
  end if
  Update selection probability of each mutation operator
end while

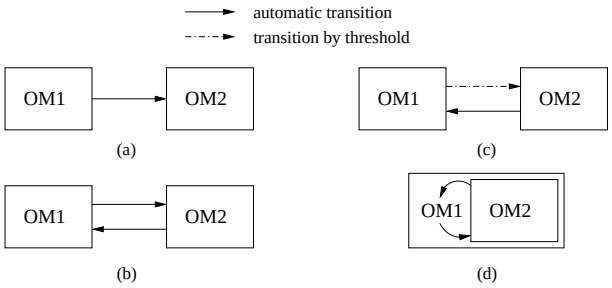

Fig. 7. Examples of cooperation scheme

sented before. The first one is an exact method which uses the Pareto set obtained with AGMA to speed up TPM. But the computational time of TPM still grows exponentially with the size of the instances. So, for the next approaches running on larger problems, we add constraints to TPM, to guide the algorithm despite of the loss of the guaranty to obtain the optimal Pareto set.

These three methods use the cooperation scheme (a). But we can apply these methods with the other cooperation schemes, which are more evolved.

Approach 1 - An improved exact approach: Using AGMA solutions as initial values: In this approach, we run the whole two-phases method. For every branch-and-bounds of the TPM, we consider the best solutions given by the meta-heuristic as initial values. Therefore we can cut a lot of nodes of the branch-and-bound and find all optimal solutions with this method.

The time required to solve a given problem is of course smaller if the distance between the initial front (given by the meta-heuristic) and the optimal front is

small. If the distance between them is null, the TPM will be used to prove that solutions produced by AGMA are optimal.

Even if this approach reduces the time needed to find the exact Pareto front, it does not allow to increase a lot the size of the problems solved.

Approach 2 - Using TPM as a Very Large Neighborhood Search (VLNS): Neighborhood search algorithms (also called local search algorithms) are a wide class of improvement heuristics where at each iteration an improving solution is found by exploring the "neighborhood" of the current solution. Ahuja et. al remark that a critical issue in the design of a neighborhood search is the choice of the neighborhood structure [20]. In a general case, larger is the neighborhood, more efficient is the neighborhood search. So, VLNS algorithms consist in exploring exponential neighborhood in a practical time to get better results. In [20], several exponential neighborhoods techniques are exposed. Here, we propose to use TPM as a VLNS algorithm.

The idea is to reduce the space explored by the TPM by cutting branches when the solution in construction is too far from the initial Pareto solution. An efficient neighborhood operator for FSP is the insertion operator [17]. So, we allow TPM to explore only the neighborhood of an initial Pareto solution which consists of solutions that are distant from less than δ_{max} insertion operator applications from it:

The following example represents an example of solution construction using VLNS approach from the initial solution $abcdefghij$. In this example two sets of jobs are used: The first one (initialized to {}) represents the current partially constructed solution and the second one (initialized to $\{abcdefghij\}$) represents jobs that have to be placed. During the solution construction, δ value (initially set to 0) is incremented for each breaking order with the initial solution. If $\delta = \delta_{max}$, then no more breaking order is allowed, so in this case, only one schedule is explored:
Example (constraint: $\delta_{max} = 2$):

- Initialization: {},$\{abcdefghij\}$ (represents: {jobs scheduled},{jobs to be placed})
- We firstly place the two first jobs: $\{ab\}, \{cdefghij\}$. $\delta = 0$.
- Then we place job g, so we apply insertion operator on the remaining jobs: $\{abg\}, \{cdefhij\}$. For the moment, the distance δ between the initial solution and the current solution is 1 (one breaking order).
- Then we place jobs c and d: $\{abgcd\}, \{efhij\}$. $\delta = 1$.
- Then we place job h: $\{abgcdh\}, \{efij\}$. $\delta = 2$.
- Here, $\delta = \delta_{max}$, so the last jobs have to be scheduled without breaking order. So, the single solution explored is the schedule $\{abgcdhefij\}, \{\}$, with $\delta = 2$. Others possible schedules are too far from the initial solution.

The size of the insertion neighborhood is in $\Theta(N^2)$, so the size of the space explored by TPM may be approximated (for $d_{max} << N$) by $\Theta(N^{2d_{max}})$. Hence, we have to limit d_{max} value, especially on instances with many jobs.

Approach 3 - A local optimization with TPM: This third cooperation limits the size of the explored space, while reducing it to a partition of Pareto solutions proposed by AGMA. So TPM is applied on regions of Pareto solutions.

The main goal of this approach is to limit the size of the trees obtained by the TPM, in order to apply this approach to large instances. In this section, we will present a non-exact two-phases method in which we only explore a region of the decisional space. So, we select partitions of each solution obtained by AGMA algorithm. Then we explore all the solutions obtained with modifications of these partitions using the two-phases method. After having explored a partition for all the Pareto set, we extract the new Pareto set from the obtained solutions.

This Simple Partitionning Post Optimization Branch & Bound (SPPOBB) works as follows:

The two-phases method explores the tree by placing jobs either at the beginning or at the end of the schedule. So, if the partition is defined from job X_i to job X_j, it places, jobs $0..X_i - 1$ at the beginning of the schedule and jobs $X_j + 1..N$ at the end. Then it explores the remaining solutions of the tree by using the two-phases method technique.

Figure 8 shows an example of hybridization by the two-phases method - it can be applied for other branch and bound methods. In this figure, we consider an initial solution $a, b, \ldots, i, j$, which is on the Pareto front obtained by AGMA algorithm. In this example, $N = 10$, the partition size is 4, and is applied from job number 4 to 7, i.e jobs d, e, f, g in the schedule. The first phase consists in placing the first three jobs at the beginning of the schedule. Then, it places the last three jobs at the end of the schedule (a job j placed in queue is symbolized by $-j$). Then, we apply the two-phases method on the remaining jobs. After cutting several nodes, 5 complete schedules have been explored:

- a,b,c,-j,-i,-h,d,-e,f,g which corresponds to the schedule abcdfgehij
- a,b,c,-j,-i,-h,d,-g,e,f which corresponds to the schedule abcdefghij (the initial solution)
- a,b,c,-j,-i,-h,d,-g,f,e which corresponds to the schedule abcdfeghij
- a,b,c,-j,-i,-h,g,-d,-e,f which corresponds to the schedule abcgfedhij
- a,b,c,-j,-i,-h,g,-d,-f,e which corresponds to the schedule abcgefdhij

Parameters:

Different parameters have to be set to have an efficient search without having a too large time expense.

- Size of the partitions: The cardinality of the Pareto set obtained with AGMA algorithm varies between several tens and two hundred solutions. In order to obtain solutions rapidly, we limit the size of partitions to 15 jobs for the 10-machines instances, and 12 jobs for the 20-machines instances. So each two-phases execution can be solved in several seconds or minutes.
- Number of partitions for each solution: Enough partitions of the complete schedule have to be considered to treat each job at least once by TPM approach. Moreover, it is interesting to superpose consecutive partitions to authorize several moves of a same job during optimization. Then, a job which

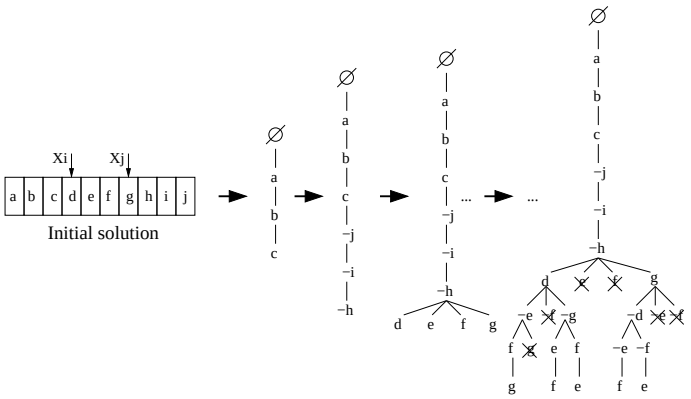

Fig. 8. Example: one partition exploration

is early scheduled could be translated at the end of the schedule by successive moves. On the other side, the more partitions we have, the more processing time is needed. So we take 8 partitions for the 50-jobs instances, 16 partitions for the 100-jobs instances and 32 partitions for the 200-jobs instances.

5 Results

We test the first approach to prove optimality of Pareto fronts on small instances. This approach reduces the time needed by the TPM to exactly solve these instances. Then we test the last two approaches. Results are comparable on 50 machines instances, but the computational time of the VLNS approach is exponential, so we present here only the results obtained with SPPOBB. However, the other approaches give some perspectives about cooperation mechanisms.

In this part, we firstly present performance indicators to evaluate effectiveness of this approach. Then we apply these indicators to compare the fronts obtained before and after cooperation with SPPOBB.

5.1 Quality assessment of Pareto set approximation

Solutions' quality can be assessed in different ways. We can observe graphically progress realized as in figures 9 and 10. Here, we use the contribution metric [21] to evaluate the proportion of Pareto solutions given by each front, and the S metric, as suggested in [22], to evaluate the dominated area.

Contribution metric: The contribution of a set of solutions PO_1 relatively to a set of solutions PO_2 is the ratio of non-dominated solutions produced by PO_1 in PO^*, where PO^* is the set of Pareto solutions of $PO_1 \cup PO_2$.

- Let PO be the set of solutions in $PO_1 \cap PO_2$.

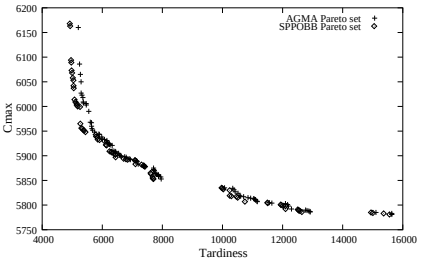

Fig. 9. SPPOBB results: instance with 100 jobs and 10 machines.

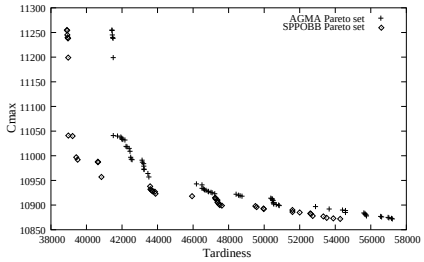

Fig. 10. SPPOBB results: instance with 200 jobs and 10 machines.

- Let W_1 (resp. W_2) be the set of solutions in PO_1 (resp. PO_2) that dominate some solutions of PO_2 (resp. PO_1).
- Let L_1 (resp. L_2) be the set of solutions in PO_1 (resp. PO_2) that are dominated by some solutions of PO_2 (resp. PO_1).
- Let N_1 (resp. N_2) be the other solutions of PO_1 (resp. PO_2): $N_i = PO_i \setminus (PO \cup W_i \cup L_i)$.

$$Cont(PO_1/PO_2) = \frac{\frac{\|PO\|}{2} + \|W_1\| + \|N_1\|}{\|PO^*\|}$$

***S* metric:** A definition of the S metric is given in [23]. Let PO be a non-dominated set of solutions. S metric calculates the hyper-volume of the multi-dimensional region enclosed by PO and a reference point Z_{ref}.

Let PO_1 and PO_2 be two sets of solutions. To evaluate quality of PO_1 against PO_2, we compute the ratio $(S(PO_1) - S(PO_2))/S(PO_2)$. For the evaluation, the reference point is the one with the worst value on each objective among all the Pareto solutions found over the runs.

5.2 Computational results

We use S and Contribution metrics to compute improvements realized on fronts. Tests were realized for 10 runs per instance, on a 1.6Ghz machine. Tables 2 and 3 show the results obtained for these metrics.

Table 2 shows that improvements realized on 50*10 and 50*20 instances were small in a general case. In fact we have an average improvement of 18.8 per cent of the initial Pareto set for the 50*10 instance, and 4.8 per cent for the 50*20 instance. For the other problems, a large part of the new Pareto set dominates the initial set of Pareto solutions. Table 3 shows a good progression of the Pareto front for large problems, especially for the 200 jobs* 10 machines instance.

Table 4 shows that the time required to realize the set of two phases is almost regular despite of the branch & bound approach.

Table 2. Quality assessment (contribution metric): C(SPPOBB/AGMA)

Problem	**ta_50_10_01**	**ta_50_20_01**	**ta_100_10_01**	**ta_100_20_01**	**ta_200_10_01**
C_{Min}	0.54	0.51	0.96	0.73	1.00
C_{Max}	0.63	0.55	1.00	0.96	1.00
Average	0.594	0.525	0.986	0.876	1.000
Std dev	0.026	0.015	0.015	0.062	0.000

Table 3. Quality assessment (S metric): S(SPPOBB)/S(AGMA)

Problem	**ta_50_10_01**	**ta_50_20_01**	**ta_100_10_01**	**ta_100_20_01**	**ta_200_10_01**
S_{Min}	0.02%	0.01%	0.75%	0.28%	8.35%
S_{Max}	0.46%	0.27%	2.10%	1.92%	15.57%
Average	0.185%	0.093%	1.199%	0.970%	13.094%
Std dev	0.122%	0.095%	0.387%	0.412%	1.974%

6 Conclusion and perspectives

In this paper, we have first presented an exact approach and a metaheuristic approach to solve MOPs. These approaches have been applied on a BOFSP. Then we have proposed original approaches to upgrade metaheuristic results by using an exact method i.e. the two-phases method. These approaches were tested, and their effectivenesses were shown by improvements realized on Pareto fronts obtained with AGMA algorithm. These results show the interest of this type of methods, which can be improved by adding other mechanisms to explore a large region of the search space without exploring a great part of the solutions. In the future, cooperation could be made in a hybrid way to combine the partitionning and the VLNS approaches. Another way for cooperation between evolutionary and exact approaches, without considering partitions of optimal solution, is to extract information from these solutions to reduce sufficiently the size of the search space.

References

1. Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary algorithms for solving Multi-Objective Problems. Kluwer, New York (2002)
2. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, UK (2001)
3. Zitzler, E., Deb, K., Thiele, L., Coello, C.A.C., Corne, D., eds.: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001). Volume 1993 of Lecture Notes in Computer Science., Berlin, Springer-Verlag (2001)
4. Lemesre, J.: Algorithme parallèle exact pour l'optimisation multi-objectif: application à l'ordonnancement. Master's thesis, University of Lille (2003)

Table 4. Run time

Problem	**ta_50_10_01**	**ta_50_20_01**	**ta_100_10_01**	**ta_100_20_01**	**ta_200_10_01**
T_{Min}	3h16'	5h33'	28h11'	37h26'	63h24'
T_{Max}	5h26'	6h19'	52h44'	65h02'	122h45'
Average	4h19'	5h49'	35h54'	50h25'	90h53'
Std dev	42'	25'	8h05'	7h58'	17h16'

5. Sayin, S., Karabati, S.: A bicriteria approach to the two-machine flow shop scheduling problem. European journal of operational research (1999) 435–449
6. Rajendran, C.: Heuristics for scheduling in flowshop with multiple objectives. European journal of operational research (1995) 540–555
7. Nagar, A., Haddock, J., Heragu, S.: Multiple and bicriteria scheduling: A litterature survey. European journal of operational research (1995) 88–104
8. Sivrikaya, F., Ulusoy, G.: A bicriteria two-machine permutation flowshop problem. European journal of operational research (1998) 414–430
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of Discrete Mathematics. Volume 5. (1979) 287–326
10. Lenstra, J.K., Kan, A.H.G.R., Brucker, P.: Complexity of machine scheduling problems. Annals of Discrete Mathematics **1** (1977) 343–362
11. Kim, Y.D.: Minimizing total tardiness in permutation flowshops. European Journal of Operational Research **33** (1995) 541–551
12. Du, J., Leung, J.Y.T.: Minimizing total tardiness on one machine is np-hard. Mathematics of operations research **15** (1990) 483–495
13. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operations Research **64** (1993) 278–285
14. Talbi, E.G., Rahoual, M., Mabed, M.H., Dhaenens, C.: A hybrid evolutionary approach for multicriteria optimization problems : Application to the flow shop. In Zitzler, E., et al., eds.: Evolutionary Multi-Criterion Optimization. Volume 1993 of LNCS., Springer-Verlag (2001) 416–428
15. Visée, M., Teghem, J., Pirlot, M., Ulungu, E.: The two phases method and branch and bound procedures to solve the bi-objective knapsack problem. Journal of Global Optimization **Vol. 12** (1998) p. 139–155
16. Haimes, Y., Ladson, L., Wismer, D.: On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Transaction on system, Man and Cybernetics (1971) 269–297
17. Basseur, M., Seynhaeve, F., Talbi, E.G.: Adaptive mechanisms for multi-objective evolutionary algorithms. In: Congress on Engineering in System Application CESA'03, Lille, France (2003)
18. Basseur, M., Seynhaeve, F., Talbi, E.G.: Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem. In: Congress on Evolutionary Computation CEC'02, Honolulu, USA (2002) 1151–1156
19. Talbi, E.G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics **8** (2002) 541–564
20. Ahuja, R.K., zlem Ergun, Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Appl. Math. **123** (2002) 75–102

21. Meunier, H., Talbi, E.G., Reininger, P.: A multiobjective genetic algorithm for radio network optimisation. In: CEC. Volume 1., Piscataway, New Jersey, IEEE Service Center (2000) 317–324
22. Knowles, J.D., Corne, D.W.: On metrics for comparing non-dominated sets. In Center, I.S., ed.: Congress on Evolutionary Computation (CEC'2002). Volume 1., Piscataway, New Jersey (2002) 711–716
23. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: Methods and applications. Master's thesis, Swiss federal Institute of technology (ETH), Zurich, Switzerland (1999)