A Sort & Search method for multicriteria optimization problems with applications to scheduling theory

Lei Shang, Vincent t'Kindt

A *Sort & Search* method for multicriteria optimization problems with applications to scheduling theory

Lei Shang
Université de Tours, LIFAT (EA 6300),
ERL ROOT CNRS 7002, 64 avenue Jean Portalis, 37200 Tours, France.
lei.shang@univ-tours.fr

Vincent T'kindt
Université de Tours, LIFAT (EA 6300),
ERL ROOT CNRS 7002, 64 avenue Jean Portalis, 37200 Tours, France.
tkindt@univ-tours.fr (corresponding author)

December 11, 2018

Abstract

This paper focuses on the *Sort & Search* method to solve a particular class of single criterion optimization problems called *Multiple Constraints Problems*. This general method enables to derive exponential-time algorithms for $\mathcal{NP}$-hard optimization problems. In this paper this method is further extended to the class of *Multicriteria Multiple Constraints Problems* for which the set of Pareto optima is enumerated. Then, the application of this new method on some multicriteria scheduling problems is proposed leading to new exponential-time algorithms for those problems.

Keywords. Multicriteria optimization, Exponential-time algorithms, Scheduling, *Sort & Search*

1 Introduction

This paper deals with multicriteria optimization and exponential-time algorithms with a special application focus to scheduling theory. Typically, scheduling problems consist in determining the optimal allocation of a set of jobs (or tasks) to machines (or resources) over time. Since the mid 1950's, scheduling problems have been the matter of numerous researches leading today to a well-defined theory at the crossroad of several research fields like operations research and combinatorial optimization, computer science and industrial engineering. Many of the scheduling problems dealt with in the literature are intractable problems, *i.e.* $\mathcal{NP}$-hard as

defined in complexity theory. Consequently, an optimal solution of such problems can only be achieved by super-polynomial time algorithms (unless $\mathcal{P} = \mathcal{NP}$), the design of which has been the matter of an important part of the literature over the last decades. These algorithms are often referred to as *exact exponential algorithms* whose worst-case time complexity can be expressed as $\mathcal{O}^*(c^n)$ with c a constant, n is the measure of input size, and the $\mathcal{O}^*$ notation suppresses multiplicative factors in the complexity, *i.e.*, $\mathcal{O}^*(c^n) = \mathcal{O}(p(n)c^n)$ where $p(n)$ is a polynomial on n. The interest in exponential-time algorithms relies on the desire of evaluating the complexity of computing optimal solutions by answering to the question: "What is the time I have to pay in the worst-case scenario?".

As met in the literature, the solution of scheduling problems often involves the minimization of a single criterion to optimize. However, in industrial scenarios, it is very natural to have multiple criteria to optimize. The optimization of several conflicting criteria relates to the general field of multicriteria optimization (Ehrgott, 2006). A comprehensive survey of multicriteria scheduling problems has been proposed by T'kindt and Billaut (2006).

To the best of the authors' knowledge, few studies on the existence of exact exponential-time algorithms for scheduling problems can be found in the literature. For single criterion problems, the exceptions are due to the pioneering work of Woeginger (2003) (also presented in the book of Fomin and Kratsch (2010)) who presented some preliminary results on single machine scheduling problems. More recently, Cygan et al. (2011), Lenté et al. (2011) and Lenté et al. (2014) have proposed several exponential-time algorithms for a set of classic scheduling problems. Some results on flowshop scheduling and single machine scheduling have been proposed by Shang (2017), Shang et al. (2017) and by Garraffa et al. (2018). Jansen et al. (2013) provided some results, under the *Exponential Time Hypothesis*, on the existence of lower bounds on the worst-case time complexities of some scheduling problems. Also notice that fixed-parameter algorithms, whose worst-case complexity are dependent on additional parameters, are closely related to exponential-time algorithms. Some recent results for scheduling problems have been proposed by Mnich and Wiese (2015), Knop et al. (2017), Knop and Koutecky (2018) and Mnich and van Bevern (2017). For multicriteria scheduling problems, very few results can be found. Carraway et al. (1990) generalized *dynamic programming* to solve multicriteria problems. The Pareto solutions are constructed from Pareto solutions of subpoblems. For an extended review of techniques related to exponential-time algorithms, we refer the reader to the book of Fomin and Kratsch (2010). Apart from the field of scheduling, exponential-time algorithms have been largely considered for graph or decision problems, but at most when a single criterion is optimized. To the best of our knowledge, no results are known for multicriteria optimization problems.

Lenté et al. (2013) extended the classic *Sort & Search* method to solve a class of problems referred to as *Multiple Constraints Problems* (MCP). They illustrated the application of the extended method to a set of scheduling problems with a single machine, parallel machines or in a flowshop environment. We further extend the *Sort & Search* method to multicriteria optimization and we illustrate its application to some multicriteria scheduling problems. Since more than one objective function is given, the algorithm enumerates all Pareto optimal solutions and return them all.

The complexity of the algorithm is therefore naturally dependent on the number of such solutions also referred to as the number of solutions in the Pareto front, denoted by $\mathcal{P}$. For $\mathcal{NP}$-hard scheduling problems, the instance size is usually measured by the number of jobs, denoted by n. The objective of this theoretical work is therefore to have an algorithm whose complexity is in the form of $\mathcal{O}^*(|\mathcal{P}| \cdot c^n)$ with c as small as possible. By the way, we provide an exact exponential-time algorithm applicable to some multicriteria optimization problems, and which worst-case time complexity is lower than that of a brute force approach. Notice that T'kindt et al. (2007) proposed complexity classes for qualifying the complexity of enumerating the set of Pareto optimal solutions. This works opens interesting research directions on the way of "quantifying" the complexity of enumerating Pareto optimal solutions.

The remainder is organized as follows. Section 2 describes the extension to *Multicriteria Multiple Constraint Problems* ($MMCP$), whilst section 3 is devoted to its application to some multicriteria scheduling problems. Conclusions and future research lines are provided at the end.

2 The *Sort & Search* method to solve ($MMCP$)

We propose a version of the *Sort & Search* method to *Multicriteria Multiple Constraint Problems* ($MMCP$) that can be formally introduced as follows.

Let $A = (\boldsymbol{a}_1,\ \boldsymbol{a}_2, \ldots \boldsymbol{a}_{n_A})$ be a table of n_A vectors of dimension d_A, $B = (\boldsymbol{b}_1,\ \boldsymbol{b}_2, \ldots \boldsymbol{b}_{n_B})$ be a table of n_B vectors $\boldsymbol{b}_k = (b_k^1, \ldots, b_k^K, b_k^{K+1}, \ldots, b_k^{K+d_B})$ of dimension $(d_B + K)$, f_h $(1 \leq h \leq K)$ and g_ℓ $(1 \leq \ell \leq d_B)$ be $(d_B + K)$ functions from $\mathbb{R}^{d_A+1}$ to $\mathbb{R}$ which are non-decreasing with respect to their last variable. The ($MMCP$) can be defined as follows.

$$
\begin{array}{ll}
\text{Min } f_1(\boldsymbol{a_j}, b_k^1) & \\
\ldots & \\
\text{Min } f_K(\boldsymbol{a_j}, b_k^K) & \\
s.t. & g_\ell(\boldsymbol{a_j}, b_k^{K+\ell}) \geq 0, \quad (1 \leq \ell \leq d_B) \\
 & \boldsymbol{a}_j \in A,\ \boldsymbol{b}_k \in B.
\end{array}
$$

Notice that functions f_h and g_ℓ can also be assumed to be non-increasing, which implies small changes in the presentation below but does not change fundamentally the proposed method.

Solving ($MMCP$) consists in computing all strict Pareto optimal solutions for criteria f_h.

Definition 1 *A solution $(\boldsymbol{a}_j, \boldsymbol{b}_k)$ is a strict Pareto optimum if there does not exist another solution $(\boldsymbol{a}_{j'}, \boldsymbol{b}_{k'})$ such that $f_h(\boldsymbol{a}_{j'}, \boldsymbol{b}_{k'}^h) \leq f_h(\boldsymbol{a}_j, \boldsymbol{b}_k^h)$, $1 \leq h \leq K$, with at least one strict inequality. We denote by $\mathcal{P}$ the minimal set of strict Pareto optima, i.e., the set of strict Pareto optima such that no two solutions in $\mathcal{P}$ have exactly the same criteria values.*

The computation of set $\mathcal{P}$ can be done by several approaches (T'kindt and Billaut, 2006), a well-known one being the ε-constraint approach. Assume a given

bound vector $\boldsymbol{\varepsilon} = [\varepsilon_2; \ldots; \varepsilon_K] \in \mathbb{R}^{K-1}$ is given. We denote by $(MMCP_\varepsilon)$ the associated ε-constraint problem, which is defined below.

$$\begin{array}{ll} \text{Min } & f_1(\boldsymbol{a_j}, b_k^1) \\ s.t. & g_\ell(\boldsymbol{a_j}, b_k^{K+\ell}) \geq 0, \quad (1 \leq \ell \leq d_B) \\ & f_h(\boldsymbol{a_j}, b_k^h) \leq \varepsilon_h, \quad (2 \leq h \leq K) \\ & \boldsymbol{a}_j \in A, \boldsymbol{b}_k \in B \end{array} \tag{1}$$

Solving $(MMCP_\varepsilon)$ for all vectors $\boldsymbol{\varepsilon}$ enables to compute set $\mathcal{P}$ since for any strict Pareto optimum there exists a vector ε such that the optimum is an optimal solution of the associated $(MMCP_\varepsilon)$. For a given $\boldsymbol{\varepsilon}$, the above problem can be reformulated as an (MCP) as follows:

$$\begin{array}{ll} \text{Min } & f_1(\boldsymbol{a_j}, b_k^1) \\ s.t. & g_\ell(\boldsymbol{a_j}, b_k^{K+\ell}) \geq 0, \quad (1 \leq \ell \leq d_B) \\ & g_{d_B+\ell}(\boldsymbol{a_j}, b_k^{\ell+1}) \geq 0, \quad (1 \leq \ell \leq K-1) \\ & \boldsymbol{a}_j \in A, \, \boldsymbol{b}_k \in B. \end{array}$$

The new functions $g_{d_B+\ell}$ are defined by $g_{d_B+\ell} = \varepsilon_{\ell+1} - f_{\ell+1}(\boldsymbol{a}_j, b_k^{\ell+1})$ and are non-increasing with respect to their last variable. As a consequence, $(MMCP_\varepsilon)$ can be solved by the algorithm `SolveMCP` proposed by Lenté et al. (2013).

Now let us deal with the whole enumeration of set $\mathcal{P}$. This can be achieved by solving as many $(MMCP_\varepsilon)$ as there are Pareto optima in $\mathcal{P}$. We first extend the definition of $(MMCP_\varepsilon)$.

Definition 2 *Let $\boldsymbol{\varepsilon^L}$ and $\boldsymbol{\varepsilon^U}$ be two bound vectors of $(K-1)$ values: $\boldsymbol{\varepsilon^L} = [\varepsilon_2^L; ..; \varepsilon_K^L]$ and $\boldsymbol{\varepsilon^U} = [\varepsilon_2^U; ..; \varepsilon_K^U]$. Then, $(MMCP_{\varepsilon^L, \varepsilon^U})$ defines the following ε-constraint problem:*

$$\begin{array}{ll} \text{Min } & f_1(\boldsymbol{a_j}, b_k^1) \\ s.t. & g_\ell(\boldsymbol{a_j}, b_k^{K+\ell}) \geq 0, \quad (1 \leq \ell \leq d_B) \\ & \varepsilon_h^L \leq f_h(\boldsymbol{a_j}, b_k^h) < \varepsilon_h^U, \quad (2 \leq h \leq K) \\ & \boldsymbol{a}_j \in A, \boldsymbol{b}_k \in B \end{array} \tag{2}$$

In other words, with respect to the $(MMCP_\varepsilon)$ problem defined in (1), we now also define lower bounds in $\varepsilon-$constraints. Notice that the constraints in the initial definition of (MCP) only take the form of upper bounds, but it can be easily adapted to handle constraints of lower bounds since range trees naturally support range queries with both lower bounds and upper bounds. $(MMCP_{\varepsilon^L, \varepsilon^U})$ is an instrumental problem which is solved by the algorithm `EnumMMCP`, presented in Figure 1, to enumerate set $\mathcal{P}$. The vector $\boldsymbol{\varepsilon^L}$ should be initialized with lower bounds of the criteria. For instance, by Definition 2 and assuming all objective function values are positive, $(MMCP_\varepsilon)$ can be represented as a $(MMCP_{\varepsilon^L, \varepsilon^U})$ with $\boldsymbol{\varepsilon^L} = [0; ..; 0]$ and $\boldsymbol{\varepsilon^U} = [\varepsilon_2; ..; \varepsilon_K]$.

Given an optimization problem with K objective functions to minimize, as defined at the beginning of this section, it is sufficient to reformulate the problem as an $(MMCP_{\boldsymbol{\varepsilon}^L,\boldsymbol{\varepsilon}^U})$, then call the algorithm `EnumMMCP`$(MMCP_{\boldsymbol{\varepsilon}^L,\boldsymbol{\varepsilon}^U})$ to enumerate all solutions in the Pareto front. The algorithm starts by calling the `SolveMCP` method (Step 1), to find the first Pareto optimum $\boldsymbol{x}$ with the first objective function minimized. Depending on the actual implementation of the `SolveMCP` method, it is possible that $\boldsymbol{x}$ corresponds to a *weak* Pareto optimum (T'kindt and Billaut, 2006) in multicriteria space, *i.e.*, there may exist another solution whose first criterion is as good as $\boldsymbol{x}$ but with smaller values on the other criteria. In order to reach a *strict* Pareto optimum starting from $\boldsymbol{x}$, a *tightening* procedure is applied. This corresponds to the `Tighten`() function called at Step 3. The idea is to formulate a series of ε-constraint problems where the criterion to minimize is the second one, the third one,..., until the K-th. For each reformulated problem, we tighten the constraints to search for strict Pareto optima. The definition of this function is provided in Figure 2.

Next we can add $\boldsymbol{x}$ as the first strict Pareto optimum (Step 4). In order to find the next one, constraints are updated such that all candidate solutions in the search space must be better than x on at least one criterion. This is done by the loop starting at Step 5. For each possible subset of criteria, we generate a new problem whose ε-constraints on these criteria are tightened to force the search towards solutions not dominated by x. It is also important to notice that the search space of subproblems generated at Step 12 does not overlap, *i.e.*, there is no dominance relation, in the sense of Pareto, between the solutions of these subproblems. This guarantees that the solutions of these problems can be directly joined to obtain the Pareto optima of the initial problem. This also ensures that the global number of calls to `SolveMCP` is bounded by the number of strict Pareto optima, denoted by $|\mathcal{P}|$, of the initial problem. This is essential for the complexity analysis.

In the worst-case, the call to `SolveMCP` has a time complexity in $O(n_B \log_2^{d_B+2}(n_B))$ whilst the space requirement is in $O(n_B \log_2^{d_B-1}(n_B))$, as established by Lenté et al. (2013). The complexity of the `Tighten`() function is mainly determined by $(K-1)$ calls to the `SolveMCP` procedure (Step 7 in Figure 2). The two loops at Step 5 and Step 8 yield an overall time complexity in $O(K \cdot 2^K)$, without considering the recursive cost at Step 12. Therefore, when the number of criteria K is fixed, the time and space complexity of `EnumMMCP` for a single run are mainly determined by the complexity of `SolveMCP` (K calls in total). Now, considering the fact that the number of calls to `EnumMMCP` is bounded by $|\mathcal{P}|$, the overall time complexity of the algorithm `EnumMMCP` is then in $O(|\mathcal{P}| \cdot (Kn_B \log_2^{d_B+2}(n_B) + K2^K))$ and the space requirement is in $O(n_B \log_2^{d_B-1}(n_B) + |\mathcal{P}|)$.

To conclude this section, we would like to emphasize the fact that all the results obtained here can be straightforwardly adapted for the cases in which functions f, and g_ℓ are non-increasing with respect to their last variable and for a maximization problem. In addition, for some multicriteria problems that cannot be directly formulated as an $(MMCP)$ the following corollary may apply. It follows from the construction of algorithm `EnumMMCP`.

Corollary 1 *Multicriteria optimization problems that can be modeled as an $(MMCP_\varepsilon)$ can still be solved by algorithm* ***EnumMMCP****.*

3 Application to scheduling problems

In this section we present the application of the extended *Sort & Search* approach to two multicriteria scheduling problems referred to as $P|d_i|C_{\max}, L_{\max}$ and $P|d_i|C_{\max}, \sum w_i U_i$ according to the standard three-field notation of Graham et al. (1979). The first problem cannot be formulated directly as an $(MMCP)$, but can be easily expressed as an $(MMCP_\varepsilon)$. In contrast, we formulate the second problem directly as an $(MMCP)$.

3.1 The $P|d_i|C_{\max}, L_{\max}$ problem

In this section we focus on the solution of a parallel machine scheduling problem with minimization of the makespan and the maximum lateness. This problem, referred to as $P|d_i|C_{\max}, L_{\max}$, can be defined as follows. A set J of n jobs has to be scheduled on m identical parallel machines and each job i has to be processed by one of the machines. Each job i is defined by a processing time p_i and a due date d_i by which job i has to be completed. A schedule is an assignment of jobs to machines over time. Let C_i be the completion time of job i in a given schedule. Two objective functions are to be minimized: $C_{\max}$ and $L_{\max}$, where $C_{\max} = \max_i(C_i)$ denotes the makespan and $L_{\max} = \max_i(C_i - d_i)$ denotes the maximum lateness. This problem is $\mathcal{NP}$-hard since the problem $P||C_{\max}$ is so (Garey and Johnson, 1979).

3.1.1 Preliminary results

In order to apply `EnumMMCP` on this problem, we show that it admits an $(MMCP_\varepsilon)$ formulation. We consider the case where function f_1 is the $C_{\max}$ criterion and f_2 is the $L_{\max}$ criterion. We first focus on the $C_{\max}$ criterion which is modeled in a similar way as for solving the $P||C_{\max}$ problem (see Lenté et al. (2013)).

A schedule s can be seen as a set of m sequences which correspond to the set of jobs processed consecutively on the machines. Let $P_\ell(s)$ be the sum of processing times of the jobs on machine ℓ in schedule s, $1 \leq \ell \leq m$, and let $P(s)$ be the sum of processing times of all jobs scheduled in s. We define by $\delta_\ell(s) = P_m(s) - P_\ell(s)$ the differences between the workload of the last machine and that of the ℓ^{th} machine.

As machines are identical we can assume, without loss of optimality, that the makespan is given by the set of jobs assigned on machine m: if this is not the case, then machines can be re-indexed accordingly. We denote by $\mathcal{O}_m$ the set of schedules such that the makespan is given by machine m.

Now, assume that two distinct partial schedules s and σ are given. Then, the makespan of the concatenated schedule $s\sigma$ is simply defined by $C_{\max}(s\sigma) = \max_{1\leq\ell\leq m}(P_\ell(s) + P_\ell(\sigma))$.

Property 1 *(Lenté et al., 2013) The makespan of a concatenated sequence $s\sigma$ is given by the jobs scheduled on machine m, if and only if the following conditions hold*

$$\delta_\ell(s) + \delta_\ell(\sigma) \geq 0,\ 1 \leq \ell \leq m-1.$$

Then, it follows that $C_{\max}(s\sigma) = P_m(s) + P_m(\sigma)$ *which can be rewritten as*

$$C_{\max}(s\sigma) = \frac{1}{m}\left(P(s) + P(\sigma) + \sum_{\ell=1}^{m-1}(\delta_\ell(s) + \delta_\ell(\sigma))\right). \quad (3)$$

This decomposition scheme enables to reduce the $P||C_{\max}$ problem to an (MCP). Property 2 provides some known facts on the $L_{\max}$ criterion.

Property 2 *Assume that a set of jobs has been assigned to a given machine and let* s *be any schedule of these jobs. We have the following properties:*

1. *Let* $L_{\max}(s|t)$ *be the maximum lateness of sequence* s *if it starts exactly at date* t*. Then,* $L_{\max}(s) = L_{\max}(s|0)$ *and* $L_{\max}(s|t) = t + L_{\max}(s)$*.*
2. *Assume that schedule* s *can be decomposed into two sequences* s_1 *and* s_2 *such that* $s = s_1 s_2$*. Then, we have*
   $L_{\max}(s_1 s_2) = \max\{L_{\max}(s_1)\ ,\ C_{\max}(s_1) + L_{\max}(s_2)\}$*.*
3. *If* $s = s_1 s_2$ *and all jobs in* s_1 *are early, then we have* $(L_{\max}(s_1 s_2) \leq 0 \iff L_{\max}(s_2) \leq -C_{\max}(s_1))$*.*

In the next section, we show that under the decomposition scheme of Property 1. It is possible to express the constraint $L_{\max} \leq \varepsilon$ so that the $P|d_i|C_{\max}, L_{\max}$ problem admits an $(MMCP_\varepsilon)$ formulation.

3.1.2 The *Sort & Search* method for the $P|d_i|C_{\max}, L_{\max}$ problem

Before entering into the details of the method, we introduce additional instrumental notations. For any set I of jobs, we can partition it into m disjoint subsets E_ℓ $(1 \leq \ell \leq m)$ such that $\bigcup_{\ell=1}^{m} E_\ell = I$. We refer to $\mathcal{E} = (E_1, E_2, \ldots, E_m)$ as an m-partition of I.

We assume that jobs in $\mathcal{J}$ are numbered according to the *Earliest Due Date* (EDD) order, *i.e* by non-decreasing value of d_i. Let us define by $I_1 = \{1, \ldots, \lfloor \frac{n}{2} \rfloor\}$ the subset made up of the $\lfloor \frac{n}{2} \rfloor$ first jobs of $\mathcal{J}$ and let $I_2 = \{\lfloor \frac{n}{2} \rfloor + 1, \ldots, n\}$ be the subset of the $\lceil \frac{n}{2} \rceil$ last jobs of $\mathcal{J}$.

To each m-partition $\mathcal{E}_1^j = (E_{1,1}^j, E_{1,2}^j, \ldots, E_{1,m}^j)$ of I_1 $(1 \leq j \leq m^{|I_1|})$ we associate a schedule s_1^j which contains the sequence of jobs in $E_{1,1}^j$ on machine 1, the sequence of jobs in $E_{1,2}^j$ on machine 2 and more generally the sequence of jobs in $E_{1,\ell}^j$ on machine ℓ $(1 \leq \ell \leq m)$. Similarly, we associate to each m-partition $\mathcal{E}_2^k$ of I_2 a schedule s_2^k (see Figure 3 for a simple illustration in the three-machine case). Besides, it is assumed that in each partition $E_{1,\ell}^j$ (resp. $E_{2,\ell}^k$) jobs are sorted by increasing value of their index. At last, to each couple $(\mathcal{E}_1^j, \mathcal{E}_2^k)$ we associate $f_1(\boldsymbol{a_j}, b_k^1) = \frac{1}{m}\left(P + \sum_{\ell=1}^{m-1}\delta_\ell(s_1^j) + \sum_{\ell=1}^{m-1}\delta_\ell(s_2^k)\right)$, with $P = \sum_{i=1}^{n} p_i$ the sum of processing times of all jobs, $a_j^\ell = \delta_\ell(s_1^j)$ and $b_k^1 = \sum_{\ell=1}^{m-1}\delta_\ell(s_2^k)$. Function

$f_1(\boldsymbol{a_j}, b_k^1)$ is the makespan of the concatenated schedule $s_1^j s_2^k$, which is also the function to minimize.

Note that when a set of jobs is assigned to a machine, the $L_{\max}$ criterion is minimized by scheduling these jobs in EDD order (Jackson, 1955). Due to the initial sorting of jobs according to the EDD rule, jobs in any concatenated schedule $s_1^j s_2^k$ on any machine also follow the EDD order. Therefore, $L_{\max}(s_1^j s_2^k)$ can be expressed as follows:

$$L_{\max}(s_1^j s_2^k) = \max(L_{\max}(s_1^j); \max_{1\leq \ell \leq m}(P_\ell(s_1^j) + L_{\max}(s_2^{k,\ell}))).$$

Given an upper bound ε on $L_{\max}(s_1^j s_2^k)$, the relation between ε and $L_{\max}(s_1^j s_2^k)$ can be formulated into several constraints, according to the above equation. These constraints correspond to $g'_\ell(\boldsymbol{a_j}, b_k^{m+\ell-1})$ in the following formulation.

$$\begin{array}{ll}
\text{Min } & f_1(\boldsymbol{a_j}, b_k^1) = \frac{1}{m}\left(P + \sum_{\ell=1}^{m-1}\delta_\ell(s_1^j) + \sum_{\ell=1}^{m-1}\delta_\ell(s_2^k)\right) \\
s.t. & g_\ell(\boldsymbol{a}_j, b_k^\ell) = \delta_\ell(s_1^j) + \delta_\ell(s_2^k) \\
& g'_\ell(\boldsymbol{a}_j, b_k^{m+\ell-1}) = \varepsilon - (P_m(s_1^j) - \delta_\ell(s_1^j) + L_{\max}(s_2^{k,\ell})) \\
\text{with} & \\
& 1 \leq j \leq m^{\lfloor \frac{m}{2} \rfloor},\ 1 \leq k \leq m^{\lfloor \frac{m}{2} \rfloor} \text{ and } 1 \leq \ell \leq m-1 \\
& \boldsymbol{a}_j = (\delta_1(s_1^j), \delta_2(s_1^j), ..., \delta_{m-1}(s_1^j), P_m(s_1^j), L_{\max}(s_1^j)) \\
& \boldsymbol{b}_k = (\sum_{\ell=1}^{m-1}\delta_\ell(s_2^k), \delta_1(s_2^k), ..., \delta_{m-1}(s_2^k), L_{\max}(s_2^{k,1}), ..., L_{\max}(s_2^{k,m-1}))
\end{array}$$

The above formulation corresponds to an $(MMCP_\varepsilon)$ formulation and, hence, the `EnumMMCP` can be called to solve the bicriteria problem.

It can be easily checked that functions f and g_ℓ $(1 \leq \ell \leq m-1)$ are non-decreasing functions with respect to their last variable.

According to the complexity analysis at the end of section 2, by taking $K = 2$, $d_B = (2m-1)$, $n_A = n_B = m^{\frac{n}{2}}$, the complexity of the *Sort & Search* method runs in $O(|\mathcal{P}|\cdot m^{\frac{n}{2}} \log_2^{2m+1}(m^{\frac{n}{2}}))$ or simply $O^*(|\mathcal{P}|\cdot m^{\frac{n}{2}}(\frac{n}{2})^{2m+2})$ in time and $O^*(m^{\frac{n}{2}}(\frac{n}{2})^{2m-2} + |\mathcal{P}|)$ in space. Whenever the number of machines $m \geq 2$ is fixed, we can rewrite the time and space complexities as $O^*(|\mathcal{P}| \cdot m^{\frac{n}{2}})$ and $O^*(m^{\frac{n}{2}} + |\mathcal{P}|)$. For $m = 2$, the time complexity is $O^*(|\mathcal{P}| \cdot 1.42^n)$ and the space complexity is $O^*(1.42^n + |\mathcal{P}|)$. An upper bound on these complexities can be obtained considering the fact that $|\mathcal{P}|$ is bounded by 2^n which is the number of two partitions of all jobs. In this case, the time complexity is $O^*(2.83^n)$ and the space complexity is at most $O^*(2^n)$.

3.2 The $P|d_i|C_{\max}, \sum w_i U_i$ problem

The scheduling problem considered in this section, referred to as $P|d_i|C_{\max}, \sum w_i U_i$, can be defined as follows. A set $\mathcal{J}$ of n jobs have to be scheduled on m identical parallel machines, each job being processed by one machine. Each job i is defined by a processing time p_i, a due date d_i and a tardiness weight w_i. A schedule is an assignment of jobs to machines over time. Two criteria have to be minimized to compute an optimal schedule. The first one denoted by $C_{\max}$, is the makespan of all jobs, and the second one, denoted

by $\sum w_i U_i$, is the weighted number of tardy jobs. We set $U_i = 1$ if $C_i > d_i$ in a given schedule, otherwise $U_i = 0$. The problem with only the second criterion, referred to as $P|d_i|\sum w_i U_i$, is $\mathcal{NP}$-hard (Brucker, 2007), which implies that the bicriteria problem is also $\mathcal{NP}$-hard.

3.2.1 Preliminary results

We first focus on the modeling of the $P|d_i|\sum w_i U_i$ to elaborate an $(MMCP)$ formulation for the bicriteria problem. First notice that, as soon as jobs are assigned to the machines, then m single machine problems have to be solved independently to minimize criterion $\sum w_i U_i$. Besides, this criterion does not take account of the amount of tardiness of jobs: either a job i is tardy in a schedule (*i.e.*, $C_i > d_i$), and it implies a cost of w_i, either it is early (*i.e.*, $C_i \leq d_i$). This feature yields useful properties.

First, let us remind that criterion $L_{\max} = \max_{1\leq i\leq n}(C_i - d_i)$ and criterion $\sum w_i U_i$ criteria have strong links. As recalled in section 3.1.2, the problem of minimizing the $L_{\max}$ criterion for a set of jobs on a single machine can be solved to optimality in polynomial time by the well-known EDD rule (*Earliest Due Date* first): jobs are sequenced by non-decreasing order of their due date d_i (Jackson, 1955). Property 2 remains valid here, in addition to which we add Property 3.

Property 3 *Assume that a set of jobs has been assigned to a given machine and let s be any schedule of these jobs. We have the following properties:*

1. $L_{\max}(s) \leq 0 \Longleftrightarrow \sum w_i U_i = 0$.
2. *There exists an optimal schedule s^* in which all early jobs are scheduled before the tardy jobs. Besides, all early jobs are sequenced according to the EDD rule.*

Now, let us turn to the m-machine problem for which the following property can be established.

Property 4 *(Lenté et al., 2013) There exists an optimal schedule in which all tardy jobs are assigned to the last machine and all early jobs assigned on the last machine are sequenced before the tardy jobs. The early jobs on each machine are sequenced according to the EDD rule.*

Note that properties 3 and 4 have been already used for designing an exact pseudopolynomial algorithm for the $1|d_i|\sum_i w_i U_i$ problem by Lawler and Moore (1969). Property 4 is illustrated in Figure 4 in the case of a two-machine problem.

3.2.2 The *Sort & Search* method for the $P|d_i|C_{\max}, \sum w_i U_i$ problem

Without loss of generality, assume that jobs are indexed in non-decreasing order of their due date. Let be $I_1 = \{1, \ldots, \lfloor \frac{n}{2} \rfloor\}$ the subset of the $\lfloor \frac{n}{2} \rfloor$ first jobs of $\mathcal{J}$ and let be $I_2 = \{\lfloor \frac{n}{2} \rfloor + 1, \ldots, n\}$ the subset of the $\lceil \frac{n}{2} \rceil$ last jobs of $\mathcal{J}$.

Considering $\sum w_iU_i$ as the only objective function, the problem can be decomposed as follows. To any $(m+1)$-partition $\mathcal{E}_1^j = (E_{1,1}^j, E_{1,2}^j, \ldots, E_{1,m+1}^j)$ of I_1 ($1 \leq j \leq$(m+1)$^{|I_1|}$) we associate a schedule s_1^j made up of $(m+1)$ sequences $(s_1^{j,1}, s_1^{j,2}, \ldots, s_1^{j,m+1})$. Each sequence $s_1^{j,\ell}$ contains jobs from set $E_1^{j,\ell}$ ordered according to the EDD rule ($1 \leq \ell \leq m+1$). Sequences $s_1^{j,1}$ to $s_1^{j,m}$ contain jobs that are completed early on machines 1 to m whilst sequence $s_1^{j,m+1}$ contains jobs that are completed tardy and, thus, assigned to machine m. We proceed similarly with any $(m+1)$-partition $\mathcal{E}_2^k$ of I_2. Besides, to each couple $(\mathcal{E}_1^j, \mathcal{E}_2^k)$ is associated a schedule o^{jk} made up of the concatenation, on machine ℓ ($1 \leq \ell \leq m-1$), of sequences $s_1^{j,\ell}$ and $s_2^{k,\ell}$, and of the concatenation, on machine m, of sequences $s_1^{j,m}$, $s_2^{k,m}$, $s_1^{j,m+1}$ and $s_2^{k,m+1}$ (see Figure 4).
At last, for any couple $(\mathcal{E}_1^j, \mathcal{E}_2^k)$, we define $f_1(\boldsymbol{a_j}, b_k^1) = W(s_1^{j,m+1}) + W(s_2^{k,m+1})$ with $W(\sigma)$ the sum of weights w_i of jobs in σ ($W(\sigma) = \sum_{i\in\sigma} w_i$), $\boldsymbol{a_j}$ contains $W(s_1^{j,m+1})$ and $b_k^1 = W(s_2^{k,m+1})$. Function $f_1(\boldsymbol{a_j}, b_k^1)$ is the $\sum w_iU_i$ value for schedule $s_1^j s_2^k$.

We denote by G the set of couples defined by $G = \{(j,k) \in [\![1, (m+1)^{|I_1|}]\!] \times [\![1, (m+1)^{|I_2|}]\!] \mid \forall \ell \in [\![1,m]\!],\ L_{\max}(s_1^{j,\ell} s_2^{k,\ell}) \leq 0\}$. Consequently, G is the set of couples for which the concatenation of sequences $s_1^{j,\ell}$ and $s_2^{k,\ell}$, $1 \leq \ell \leq m$, contains no tardy job. This can be ensured, according to Property 2, by adding constraints $L_{\max}(s_1^j) \leq 0$ and $C_{\max}(s_1^{j,\ell}) + L_{\max}(s_2^{k,\ell}) \leq 0$, $1 \leq \ell \leq m$. Notice that each schedule $s_1^{j,\ell} s_2^{k,\ell}$ follows the EDD order.

We now focus on the modeling of $C_{\max}$ criterion according to Property 1. We first add constraints to ensure that the $C_{\max}$ is given on machine m following Property 1. Then, we get

$$f_2(\boldsymbol{a_j}, b_k^{2m}) = \frac{1}{m}(P + \sum_{i=1}^{m-1} \delta_i(s_1^j) + \sum_{i=1}^{m-1} \delta_i(s_2^k))$$

with $\boldsymbol{a_j}$ contains $\delta_\ell(s_1^j)$, $1 \leq \ell \leq m-1$, and $b_k^{2m} = \sum_{i=1}^{m-1} \delta_i(s_2^k)$.

Consequently, the $P|d_i|C_{\max}, \sum w_iU_i$ problem can be formulated as the following $(MMCP)$.

$$\begin{array}{ll}
\text{Min } f_1(\boldsymbol{a_j}, b_k^1) = W(s_1^{j,m+1}) + W(s_2^{k,m+1}) & \\
\text{Min } f_2(\boldsymbol{a_j}, b_k^2) = \frac{1}{m}(P + \sum_{i=1}^{m-1} \delta_i(s_1^j) + \sum_{i=1}^{m-1} \delta_i(s_2^k)) & \\
s.t. \quad g_\ell(\boldsymbol{a}_j, b_k^{\ell+m+1}) = -L_{\max}(s_2^{\ell,k}) - (P_m(s_1^j) - \delta_\ell(s_1^j)),\ 1 \leq \ell \leq m & \\
\qquad g'_\ell(\boldsymbol{a}_j, b_k^{\ell+2}) = \delta_\ell(s_1^j) + \delta_\ell(s_2^k),\ 1 \leq \ell \leq m-1 &
\end{array}$$

with

$$\begin{array}{l}
1 \leq j \leq m^{\lfloor \frac{n}{2} \rfloor}, 1 \leq k \leq m^{\lceil \frac{n}{2} \rceil} \\
\boldsymbol{a}_j = (W(s_1^{j,m+1}),\ \delta_1(s_1^j), .., \delta_{m-1}(s_1^j), P_m(s_1^j), L_{\max}(s_1^j)) \\
\boldsymbol{b}_k = (W(s_2^{k,m+1}), \sum_{i=1}^{m-1} \delta_i(s_2^k), \delta_1(s_2^k), .., \delta_{m-1}(s_2^k),\ L_{\max}(s_2^{k,1}), .., \ L_{\max}(s_2^{k,m}))
\end{array}$$

The constraint $L_{\max}(s_1^j) \leq 0$ is not formulated above because this constraint

only depends on $\boldsymbol{a_j}$ and, hence, can be easily checked when generating $\mathcal{E}_1^j$. Starting from this formulation, we can then proceed as described in section 2 and finally solve the problem by calling `EnumMMCP`. Regarding worst-case complexities, the overall time complexity of the algorithm `EnumMMCP` applied to the above $(MMCP)$ is then in $O(|\mathcal{P}| \cdot (Kn_B \log_2^{d_B+2}(n_B) + K2^K))$ and the space requirement is in $O(n_B \log_2^{d_B-1}(n_B))$. Therefore, as $K = 2$, $n_B = n_A = (m+1)^{\frac{n}{2}}$, $d_B = 3m-2$, the time complexity is in $O^*(|\mathcal{P}| \cdot (m+1)^{\frac{n}{2}} \left(\frac{n}{2}\right)^{3m})$. The space requirement is in $O^*((m+1)^{\frac{n}{2}} \left(\frac{n}{2}\right)^{3m-3} + |\mathcal{P}|)$. For fixed m, the time complexity can be written as $O^*(|\mathcal{P}| \cdot (m+1)^{\frac{n}{2}})$ and the space complexity as $O^*((m+1)^{\frac{n}{2}} + |\mathcal{P}|)$. As an example, the two-machine case, *i.e.*, $m = 2$, the time complexity is in $O^*(|\mathcal{P}| \cdot 1.74^n)$ and the space complexity in $O^*(1.74^n + |\mathcal{P}|)$. Considering a simple upper bound of $|\mathcal{P}|$, which is 2^n, the time complexity becomes $O^*(3.42^n)$ and the space complexity becomes $O^*(2^n)$.

4 Conclusions

In this paper we have presented an extension of the *Sort & Search* method, initially introduced by Horowitz and Sahni (1974), to solve the class of *Multicriteria Multiple Constraint Problem*. The proposed method enumerates the set of strict Pareto optima by solving a series of single criterion problems. The strength of this approach is that it enables easily to derive exponential time algorithms for a large class of multicriteria problems. The worst-case complexity of these algorithms is shown to be defined by a polynomial of the worst-case time complexity of solving a single criterion problem and by the number of strict Pareto optima. The *Sort & Search* method is then applied to two multicriteria parallel machine scheduling problems, referred to as $P|d_i|C_{\max}, L_{\max}$ and $P|d_i|C_{\max}, \sum_i w_i U_i$. The time and space complexities are analyzed.

The contribution of this paper is two-fold. First, we provide the first known results for $\mathcal{NP}$-hard multicriteria scheduling problems when worst-case behaviors of exact methods is of interest. Second, we propose a generic exact method with worst-case guarantee for enumerating the set of strict Pareto optima for a large class of multicriteria optimization problems. Notice that even though this paper provides applications to some scheduling problems, the method can clearly be applied to more general multicriteria optimization problems as, for instance, *Multicriteria Knapsack* problems. Finally, this paper raises an interesting research direction related to the quantification of the difficulty of enumerating the set of Pareto optimal solutions of $\mathcal{NP}$-hard multicriteria optimization problems. We believe that this research direction is worthy to be investigated in the future.

References

Brucker, P. (2007). *Scheduling algorithms*. Springer.

Carraway, R. L., Morin, T. L., Moskowitz, H. (1990). Generalized dynamic program-

ming for multicriteria optimization. *European Journal of Operational Research*, 44(1), 95–104.

Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J. (2011). Scheduling partially ordered jobs faster than 2^n. In *C. Demetrescu and M.M. Halldorsson (Eds): Proceedings of 19th Annual European Symposium (ESA 2011), Lecture Notes in Computer Science*, volume 6942, 299–310.

De Berg, M., Cheong, O., van Kreveld, M., Overmars, M. (2008). *Computational geometry: algorithms and applications*. Springer, 3rd edition.

Ehrgott, M. (2006). *Multicriteria optimization*. Springer Science & Business Media.

Fomin, F., Kratsch, D. (2010). *Exact Exponential Algorithms*. Springer.

Garey, M. R., Johnson, D. S. (1979). Computers and intractability: a guide to the theory of NP-completeness *W.H. Freeman and Company*.

Garraffa, M., Shang, L., Della Croce, F., T'Kindt, V. (2018). An exact exponential branch-and-merge algorithm for the single machine total tardiness problem. *Theoretical Computer Science*, in press, `doi.org/10.1016/j.tcs.2018.05.040`.

Graham, R. L., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5,287–326.

Horowitz, E., Sahni, S. (1974). Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21, 277–292.

Jackson, J. (1955). Scheduling a production line to minimize maximum tardiness. *Management Science Research Project, University of California (USA), research report*, 43.

Jansen, K., Land, F., Land, K. (2013). Bounding the running rime of algorithms for scheduling and packing problems. In *F. Dehne, R. Solis-Oba, and J.-R. Sack, editors, Algorithms and Data Structures, Lecture Notes in Computer Science*, volume 8037, 439–450.

Knop, D., Koutecky, M. (2018). Scheduling meets n-fold integer programming. *Journal of Scheduling*, in press, `doi.org/10.1007/s10951-017-0550-0`.

Knop, D., Koutecky, M., Mnich, M. (2017). Combinatorial n-fold integer programming and applications. *25th Annual European Symposium on Algorithms (ESA 17)*, 4-8 september 2017, Vienna (Austria), `arxiv.org/abs/1705.08657`.

Lawler, E. L., Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16(1), 77–84.

Lenté, C., Liedloff, M., Soukhal, A., T'kindt, V. (2011). Exponential-time algorithms for scheduling problems. In *MAPSP'11*, Nymburk (Czech Republic), 3 pages.

Lenté, C., Liedloff, M., Soukhal, A., T'kindt, V. (2013). On an extension of the sort & search method with application to scheduling theory. *Theoretical Computer Science*, 511, 13–22.

Lenté, C., Liedloff, M., Soukhal, A., T'Kindt, V. (2014). Exponential algorithms for scheduling problems. Technical report, University of Tours. URL `https://hal.archives-ouvertes.fr/hal-00944382`.

Mnich, M., van Bevern, R. (2017). Parametrized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100, 254–261.

Mnich, M., Wiese, A. (2015). Scheduling and fixed-parameter tractability. *Mathematical Programming: Series A*, 154:1-2, 533–562.

Shang, L. (2017). *Exact algorithms with worst-case guarantee for scheduling: from theory to practice*. PhD thesis, Université de Tours, France, URL `https://hal.archives-ouvertes.fr/tel-01651097`.

Shang, L., Lenté, C., Liedloff, M., T'Kindt, V. (2017). Exact exponential algorithms for 3-machine flowshop scheduling problems. *Journal of Scheduling*, 21:2, 1–7.

T'kindt, V., Billaut, J.-C. (2006) *Multicriteria scheduling: theory, models and algorithms*. Springer, 2nd edition.

T'kindt, V., Bouibede-Hocine, K., Esswein, C. (2007) *Counting and enumeration complexity with application to multicriteria scheduling. Annals of Operations Research*, 153:1, 215–234.

Woeginger, G. J. (2003). Exact algorithms for NP-hard problems: a survey. *Lecture Notes in Computer Science*, 2570, 185–207.

Algorithm `EnumMMCP`$(MMCP_{\boldsymbol{\varepsilon}^{\boldsymbol{L}},\boldsymbol{\varepsilon}^{\boldsymbol{U}}})$

1. Apply `SolveMCP` to solve $(MMCP_{\boldsymbol{\varepsilon}^{\boldsymbol{L}},\boldsymbol{\varepsilon}^{\boldsymbol{U}}})$
2. Let $\boldsymbol{x} = (x_1, ..., x_K)$ be the objective values obtained
3. $\boldsymbol{x} \longleftarrow$ `Tighten`$(MMCP_{\boldsymbol{\varepsilon}^{\boldsymbol{L}},\boldsymbol{\varepsilon}^{\boldsymbol{U}}}, \boldsymbol{x})$
4. $\mathcal{P} \longleftarrow \{\boldsymbol{x}\}$
5. **For all** $S \subset \{2, ..., K\}$ with $0 < |S| \leq K - 1$
6. Let $\boldsymbol{\varepsilon}^{\boldsymbol{l}} \longleftarrow \boldsymbol{\varepsilon}^{\boldsymbol{L}}$
7. Let $\boldsymbol{\varepsilon}^{\boldsymbol{u}} \longleftarrow \boldsymbol{\varepsilon}^{\boldsymbol{U}}$
8. **For** $i = 2, ..., K$
9. **If** $i \in S$ **Then** $\varepsilon_i^u \longleftarrow x_i$
10. **Else** $\varepsilon_i^l \longleftarrow x_i$
11. **End If**
12. $\mathcal{P} \longleftarrow \mathcal{P} \cup$ `EnumMMCP`$(MMCP_{\boldsymbol{\varepsilon}^{\boldsymbol{l}},\boldsymbol{\varepsilon}^{\boldsymbol{u}}})$
13. **End For**
14. **End For**
15. **Return** $\mathcal{P}$

Figure 1: Algorithm for enumerating Pareto optima

Algorithm `Tighten`($MMCP_{\boldsymbol{\varepsilon}^L,\boldsymbol{\varepsilon}^U}, \boldsymbol{x}$)

1. **For** $j = 2, ..., K$
2. **For** $i = j, ..., K$
3. **If** $x_i < \varepsilon_i^U$ **Then** $\varepsilon_i^U \longleftarrow x_i$
4. **End If**
5. **End For**
6. Reformulate ($MMCP_{\boldsymbol{\varepsilon}^L,\boldsymbol{\varepsilon}^U}$) to ($MMCP^j_{\boldsymbol{\varepsilon}^{L'},\boldsymbol{\varepsilon}^{U'}}$) such that the function to minimize is now the j-th criterion. The first criterion is put as an ε-constraint with a upper bound of x_1
7. Apply `SolveMCP` to solve ($MMCP^j_{\boldsymbol{\varepsilon}^{L'},\boldsymbol{\varepsilon}^{U'}}$)
8. $\boldsymbol{x} \longleftarrow$ the objective values obtained
9. **End For**
10. **Return** $\boldsymbol{x}$

Figure 2: Removing the non-strict Pareto optima

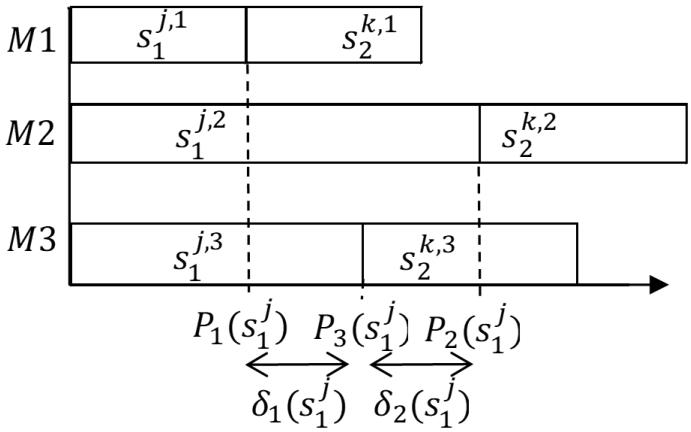

Figure 3: Decomposition of the 3-machines scheduling problem

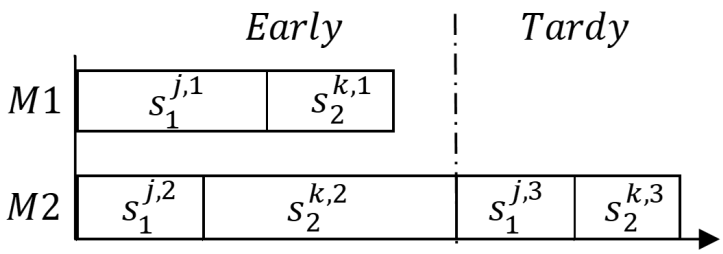

Figure 4: Decomposition of the 2-machines scheduling problem