SIAM J. DISCRETE MATH.
Vol. 27, No. 1, pp. 186–204


# A SUBMODULAR OPTIMIZATION APPROACH TO BICRITERIA SCHEDULING PROBLEMS WITH CONTROLLABLE PROCESSING TIMES ON PARALLEL MACHINES*

AKIYOSHI SHIOURA†, NATALIA V. SHAKHLEVICH‡, AND VITALY A. STRUSEVICH§

**Abstract.** In this paper, we present a general methodology for designing polynomial-time algorithms for bicriteria scheduling problems on parallel machines with controllable processing times. For each considered problem, the two criteria are the makespan and the total compression cost, and the solution is delivered in the form of the break points of the efficient frontier. We reformulate the scheduling problems in terms of optimization over submodular polyhedra and give efficient procedures for computing the corresponding rank functions. As a result, for two of the considered problems we obtain the first polynomial-time algorithms, while for the third problem we considerably improve the known running time.

**Key words.** submodular optimization, parallel machine scheduling, controllable processing times, bicriteria problems

**AMS subject classifications.** 90C27, 90B35, 90C05

**DOI.** 10.1137/110843836

## 1. Introduction.

In this paper, we study preemptive scheduling problems on parallel machines with controllable processing times. In the model under consideration, the jobs of set $N = \{1, 2, \ldots, n\}$ have to be processed on parallel machines $M_1, M_2, \ldots, M_m$, where $m \geq 2$. Throughout this paper, it is assumed that $n \geq m$. For a job $j \in N$, its processing time $p(j)$ is not given in advance but has to be chosen by the decision maker from a given interval $[l(j), u(j)]$. That selection process can be seen as either *compressing* (also known as *crashing*) the longest processing time $u(j)$ down to $p(j)$ or *decompressing* the shortest processing time $l(j)$ up to $p(j)$. In the former case, the value $x(j) = u(j) - p(j)$ is called the *compression amount* of job $j$. Compression may decrease the completion time of job $j$ but incurs additional cost $w(j)x(j)$, where $w(j)$ is a given nonnegative unit compression cost. The total cost associated with a choice of the actual processing times is represented by the linear function $W = \sum_{j \in N} w(j)x(j)$.

Each job $j \in N$ can be given a *release date* $r(j)$, before which it is not available. In the processing of any job, *preemption* and *migration* are allowed, so that the processing can be interrupted on any machine at any time and resumed later on, possibly on another machine. It is not allowed to process a job on more than one machine at a time, and a machine processes at most one job at a time.

*Received by the editors August 8, 2011; accepted for publication (in revised form) November 28, 2012; published electronically January 31, 2013. This research was supported by the EPSRC funded project EP/J019755/1, "Submodular Optimisation Techniques for Scheduling with Controllable Parameters."

http://www.siam.org/journals/sidma/27-1/84383.html

†Graduate School of Information Sciences, Tohoku University, Sendai, Japan (shioura@dais.is.tohoku.ac.jp). This author was partially supported by the Humboldt Research Fellowship of the Alexander von Humboldt Foundation and by Grant-in-Aid of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

‡School of Computing, University of Leeds, Leeds LS2 9JT, UK (N.Shakhlevich@leeds.ac.uk).

§School of Computing and Mathematical Sciences, University of Greenwich, Old Royal Naval College, London SE10 9LS, UK (V.Strusevich@greenwich.ac.uk).

Given a schedule, let $C(j)$ denote the completion time of job $j$, i.e., the time at which the last portion of job $j$ is finished on the corresponding machine. A schedule is called *feasible* if the processing of a job $j \in N$ takes place no earlier than its release date $r(j)$. The value $C_{\max} = \max\{C(j)|j \in N\}$ determines the maximum completion time of all jobs and is called the *makespan*.

The machines can be either *identical*, i.e., they have the same speed, or *uniform*, i.e., machine $M_v$ has speed $s_v$, $1 \le v \le m$. Without loss of generality, throughout this paper we assume that the uniform machines are numbered in nonincreasing order of their speeds, i.e.,

$$s_1 \ge s_2 \ge \cdots \ge s_m. \tag{1}$$

For some schedule, denote the total time during which a job $j \in N$ is processed on machine $M_v$, $1 \le v \le m$, by $q^v(j)$. Taking into account the speed of the machine, we call the quantity $s_v q^v(j)$ the *processing amount* of job $j$ on machine $M_v$. It follows that

$$p(j) = \sum_{v=1}^{m} s_v q^v(j).$$

Scheduling problems with controllable processing times have received considerable attention since the 1980s; see, e.g., surveys by Nowicki and Zdrzałka [9] and by Shabtay and Steiner [13]. The models with controllable processing times have found applications in supply chain management and scheduling, imprecise computation, make-or-buy decision making, etc.

Traditionally, in this area two functions determine the quality of a schedule: (a) total compression cost $W$ given by a linear function $\sum w(j)x(j)$, and (b) a function $F$ of the job completion times. The following four types of models are mainly considered in the literature:

- $\Pi 1$: to minimize $W$, subject to a bounded value of $F$;
- $\Pi 2$: to minimize $F$, subject to a bounded value of $W$;
- $\Pi 3$: to minimize some aggregated function, e.g., a linear combination of $W$ and $F$;
- $\Pi 4$: to minimize both functions $W$ and $F$, i.e., to determine the set of the Pareto-optimal solutions.

The problems considered in this paper fall in the category $\Pi 4$. We need to find the set of Pareto-optimal solutions defined by the break points of the so-called efficiency frontier; see [18] for definitions and a state-of-the-art survey of multicriteria scheduling. Recall that a schedule $S'$ is called *Pareto-optimal* if there exists no schedule $S''$ such that $C_{\max}(S'') \le C_{\max}(S')$ and $W(S'') \le W(S')$, where at least one of these inequalities is strict.

Adapting standard notation for scheduling problems by Lawler et al. [7], we denote a bicriteria problem in the most general setting by $Q|r(j), p(j) = u(j) - x(j), pmtn|(C_{\max}, W)$. Here, in the first field we write $Q$ to define a processing environment that consists of $m \ge 2$ uniform parallel machines; this parameter is replaced by $P$ if the machines are identical. In the middle field, the item $r(j)$ implies that the jobs have individual release dates; this parameter is omitted if the release dates are equal. We write $p(j) = u(j) - x(j)$ to indicate that the processing times are controllable and $x(j)$ is the compression amount of job $j$ to be found. The abbreviation *pmtn* is used to point out that preemption is allowed. Finally, in the third field we write

$(C_{\max}, W)$, which means that we are searching for a set of Pareto-optimal solutions with respect to the two given criteria.

Notice that the bicriteria models are most general and if we know how to solve the Π4 version of a scheduling model, then we can deduce a solution to any other single-criterion counterpart, Π1–Π3. The bicriteria models are also most important since a solution delivers to a decision maker the whole range of options to choose from. Although the first studies on the Π4 problems date to as early as 1982 [21], positive results for these problems are still quite rare, as admitted in the survey [13]. Besides, attempts to handle the bicriteria problems lack a general methodology.

Below we mainly review the previously known results on the bicriteria scheduling problems (model Π4), as most relevant to this study.

Van Wassenhove and Baker [21], Tuzikov [19], and Hoogeveen and Woeginger [5] consider various versions of a bicriteria single machine problem with a function $F$ representing the maximum completion penalty. The break points of the efficiency frontier are found by tracking the changes in the structure of a schedule as the processing times of the jobs change.

A similar method is applied to problem $P|u(j)-x(j), pmtn|(C_{\max}, W)$ with identical parallel machines by Nowicki and Zdrzałka [10], who obtain an $O(n^2)$-time algorithm.

For problem $Q|u(j)-x(j), pmtn|(C_{\max}, W)$ with uniform machines, Nowicki and Zdrzałka in [9] describe an approach that allows finding an $\varepsilon$-approximation of the efficiency frontier in pseudopolynomial $O(nm(\overline{d}-\underline{d})/\varepsilon)$ time, where $\overline{d}$ and $\underline{d}$ are the optimal makespan values if all jobs are fully decompressed and fully compressed, respectively.

A systematic development of a general framework for solving scheduling problems with controllable processing times via submodular methods has been initiated by Shakhlevich and Strusevich [14, 15] and further advanced by Shakhlevich, Shioura, and Strusevich [16] and Shioura, Shakhlevich, and Strusevich [17]. This paper makes an additional contribution to the development of this approach.

In [14, 15] a number of scheduling problems with controllable processing times have been formulated in terms of maximization linear programming problems defined over special polyhedra with submodular constraints. For several models, including those studied in this paper, the corresponding rank functions of submodular polyhedra have been developed. Still, at that stage, the solution methods developed in those papers have mainly remained schedule-based and implemented the greedy procedures of compression or decompression of the processing times of the jobs. The only advantage of the submodular reformulations has been that of an easy justification of the greedy approach for the whole range of related problems. Prior to [14, 15], researchers justified the greedy reasoning from the first principles and in a problem-dependent way. As far as the bicriteria problems are concerned, Shakhlevich and Strusevich [14] combine submodular and scheduling reasoning to develop an $O(n \log n)$ time algorithm for problem $P|u(j)-x(j), pmtn|(C_{\max}, W)$, while in [15] they design the first polynomial-time algorithm for problem $Q|u(j)-x(j), pmtn|(C_{\max}, W)$, that requires $O(n \log n + nm^4)$ time. While submodular reasoning has been crucial in deriving those results, its potential was not explored in [14, 15] in depth.

Further advantages of applications of submodular optimization have been demonstrated in [16]. For the bicriteria problem of minimizing the total compression cost and the maximum completion penalty $F$ a fast algorithm based on a reformulation in terms of a maximization linear programming problem over a (parametric) submodular polyhedron intersected with a box has been designed. An important statement

TABLE 1
*Time complexity of the algorithms.*

| Machines | Release dates | Previously known | This paper |
|---|---|---|---|
| Identical parallel | Zero | $O(n \log n)$ [14] | N/A |
| Identical parallel | Different | N/A | $O(n^2 \log m)$ |
| Uniform parallel | Zero | $O(n \log n + nm^4)$ [15] | $O(nm \log m)$ |
| Uniform parallel | Different | N/A | $O(n^2 m)$ |

contained in that paper has become one of the essential tools that is used in all our subsequent papers, including this study. This statement, quoted below as Theorem 1, states that a linear programming problem over a submodular polyhedron intersected with a box can be reduced to a problem defined over a better structured polyhedron, a so-called base polyhedron with a modified rank function. It is well known that the resulting problem admits a greedy solution algorithm with the optimal decision variables written in closed form; see Theorem 2 in section 2.

For the single criterion counterparts (model Π1) of the models considered in this paper, Shioura, Shakhlevich, and Strusevich [17] develop the fastest known algorithms based on a submodular reformulation and decomposition. These results are yet to appear in the form of a journal publication.

In this paper, we continue the line of research that links scheduling with submodular optimization and present fast algorithms that solve bicriteria problems $P|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$, $Q|u(j) - x(j), pmtn|(C_{\max}, W)$, and $Q|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$. Our reasoning is schedule-free and is based on reformulation of the corresponding problems as optimization problems with submodular constraints. For each of these problems we develop an appropriate routine for computing the corresponding rank functions as piecewise-linear functions followed by computing their sum, with several stages of the solution process being problem-independent, either technically or at least ideologically.

The summary of the results relevant to this paper is given in Table 1. Notice that here we do not consider problem $P|u(j) - x(j), pmtn|(C_{\max}, W)$, because the algorithm from [14] has the running time of $O(n \log n)$, leaving no room for further improvements, since solving any bicriteria problem under consideration requires the sorting of the jobs with respect to their unit compression costs. Observe that for problem $Q|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$ no polynomial-time algorithm has been previously known, even for its special case with identical machines, i.e., problem $P|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$.

The remainder of this paper is organized as follows. Section 2 gives a brief review of the necessary facts on submodular optimization, demonstrates how the scheduling problems under consideration can be formulated in terms of maximization linear programming problems with submodular constraints, and provides the explicit expressions for the corresponding rank functions. The key new outcome of that section is a collection of general principles that are applicable to solving any bicriteria scheduling problem under consideration. Those principles are aimed at obtaining an expression for the cost function as a piecewise linear function of makespan. Sections 3 through 5 consider each of the three problems individually and contain algorithms for computing the corresponding rank functions and for finding the cost functions in the piecewise linear form. Some concluding remarks are given in section 6.

**2. General principles.** In this section, we describe algorithms for finding the set of Pareto-optimal solutions to the bicriteria problems $Q|u(j) - x(j), pmtn|(C_{\max}, W)$, $Q|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$, and $P|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$.

We start with the features that are common to all problems. Let $S(d)$ denote a feasible preemptive schedule in which all jobs are completed by time $d$. The task of checking whether such a schedule exists for a given $d$ can be formulated in terms of submodular optimization.

For completeness, we introduce the necessary definitions. For a set $N = \{1, 2, \ldots, n\}$, let $2^N$ denote the family of all subsets of $N$. For a subset $X \subseteq N$, let $\mathbb{R}^X$ denote the set of all vectors $\mathbf{p}$ with real components $p(j)$, where $j \in X$. For a vector $\mathbf{p} \in \mathbb{R}^N$, define $p(X) = \sum_{j \in X} p(j)$ for every nonempty set $X \in 2^N$ and define $p(\emptyset) = 0$.

A set function $\varphi : 2^N \to \mathbb{R}$ is called *submodular* if the inequality

$$\varphi(X \cup Y) + \varphi(X \cap Y) \leq \varphi(X) + \varphi(Y) \tag{2}$$

holds for all sets $X, Y \in 2^N$. For a submodular function $\varphi$ defined on $2^N$ such that $\varphi(\emptyset) = 0$, the pair $(2^N, \varphi)$ is called a *submodular system* on $N$, while $\varphi$ is referred to as the *rank function* of that system.

For a submodular system $(2^N, \varphi)$, define two polyhedra

$$P(\varphi) = \{\mathbf{p} \in \mathbb{R}^N \mid p(X) \leq \varphi(X), X \in 2^N\}, \tag{3}$$

$$B(\varphi) = \{\mathbf{p} \in \mathbb{R}^N \mid \mathbf{p} \in P(\varphi),\ p(N) = \varphi(N)\}, \tag{4}$$

called a *submodular polyhedron* and a *base polyhedron*, respectively, associated with the submodular system. Notice that $B(\varphi)$ represents the set of all maximal vectors in $P(\varphi)$.

For a scheduling problem under consideration, below we explain that the set of feasible schedules $S(d)$ can be described as a polyhedron of the form

$$P(\varphi)_l^u = \{\mathbf{p} \in \mathbb{R}^N \mid p(X) \leq \varphi(X), X \in 2^N; l(j) \leq p(j) \leq u(j),\ j \in N\}, \tag{5}$$

where $\varphi : 2^N \to \mathbb{R}$ is a set function and $\mathbf{l}, \mathbf{u} \in \mathbb{R}^N$ are vectors of lower and upper bounds on the processing times, respectively. For a set of jobs $X \subseteq N$ the value $p(X)$ is the total processing requirement for the jobs of set $X$ with respect to their actual processing times, while function $\varphi(X)$ represents the total largest processing capacity available for these jobs. Notice that if function $\varphi$ is submodular, then the polyhedron $P(\varphi)_l^u$ is a submodular polyhedron $P(\varphi)$ of the form (3) intersected with a box.

All problems under consideration share the same necessary and sufficient conditions for the existence of a feasible schedule with a given common deadline $d$, as formulated, e.g., in [1]. Informally, these conditions state that for a given deadline $d$ a feasible schedule exists if and only if

(i) for each $v, 1 \leq v \leq m-1$, $v$ longest jobs can be processed on $v$ fastest machines by time $d$, and

(ii) all $n$ jobs can be completed on all $m$ machines by time $d$.

Thus, for a problem at hand, we need to find an expression for the largest processing capacity available to process any subset $X$ of jobs. Such expressions are presented below in the form of a set function $\varphi(X)$, which in all cases appears to be submodular. As a result, checking the existence of a feasible schedule $S(d)$ reduces to determining a feasible point in the polyhedron $P(\varphi)_l^u$ associated with the relevant rank function $\varphi$.

Consider problem $Q|u(j) - x(j), pmtn|(C_{\max}, W)$ in which all jobs are simultaneously available at time zero, i.e., $r(j) = 0$. The machines are numbered in accordance with (1). Define

$$S_0 = 0,\ S_v = \sum_{i=1}^{v} s_i,\ 1 \leq v \leq m, \tag{6}$$

where $S_v$ is the total speed of $v$ fastest machines.

To complete before time $d$, for any set that contains more than $m$ jobs, including the whole set of jobs $N$, the total processing capacity $dS_m$ should be enough, while for any set $X$ with less than $m$ jobs it should be sufficient to use the $|X|$ fastest machines during the time interval $[0, d]$. Therefore, problem $Q|u(j) - x(j), pmtn|(C_{\max}, W)$ is associated with the polyhedron $P(\varphi)_l^u$ with the rank function $\varphi$ of the form

$$\varphi(X) = \begin{cases} dS_{|X|} & \text{if } |X| \le m-1, \\ dS_m & \text{otherwise.} \end{cases} \tag{7}$$

The conditions $p(X) \le \varphi(X)$, $X \in 2^N$, for the function $\varphi(X)$ of the form (7) correspond to conditions (i) and (ii) above. As proved in [15], function $\varphi$ is submodular.

Consider now problem $Q|r(j), u(j)-x(j), pmtn|(C_{\max}, W)$ in which the jobs have individual release dates. To derive the rank function for the polyhedron $P(\varphi)_l^u$ associated with checking the existence of a feasible schedule $S(d)$ observe the following. For a feasible solution vector $\mathbf{p}$, the following conditions should be satisfied: any job can be completed by time $d$ if it is processed by the fastest machine, any pair of jobs can be completed by $d$ if they are processed on the two fastest machines, etc., any subset of at most $m-1$ jobs can be completed by $d$ on $m-1$ fastest machines, and finally, all jobs can be completed by $d$ on all $m$ machines. In order to take into account the release dates, assume that the jobs are numbered in such a way that

$$r(1) \le r(2) \le \cdots \le r(n), \tag{8}$$

and for a set $X \in 2^N$ define $r_i(X)$ as the $i$th smallest release date in set $X$, $1 \le i \le |X|$. Similarly to the piece of notation $p(X)$, we denote the sum of the release dates of the jobs of set $X$ by $r(X)$.

For a nonempty set $X$ of jobs, the largest processing capacity available on the fastest machine $M_1$ is $s_1(d - r_1(X))$, the total largest processing capacity on the two fastest machines $M_1$ and $M_2$ is equal to $s_1(d - r_1(X)) + s_2(d - r_2(X))$, etc. Thus, we deduce that

$$\varphi(X) = \begin{cases} dS_{|X|} - \sum_{i=1}^{|X|} s_i r_i(X) & \text{if } |X| \le m-1, \\ dS_m - \sum_{i=1}^{m} s_i r_i(X) & \text{otherwise.} \end{cases} \tag{9}$$

This formula is shown in [8, 15] in a different (but equivalent) form. Function $\varphi(X)$ can be proved to be submodular as in [15].

If the machines are identical, then for the resulting problem $P|r(j), u(j) - x(j), pmtn|\ (C_{\max}, W)$ function (9) can be simplified. For a set of jobs $X \subseteq N$, let $R(X)$ denote the sum of $\min\{|X|, m\}$ smallest release dates for the jobs of set $X$. Notice that if $|X| \le m-1$, then $R(X) = r(X)$. For completeness, define $R(\emptyset) = 0$. Then

$$\varphi(X) = \begin{cases} dS_{|X|} - r(X) & \text{if } |X| \le m-1, \\ dS_m - R(X) & \text{otherwise.} \end{cases} \tag{10}$$

This formula is also shown in [17], where the roles of the release dates and the deadlines are exchanged. Observe that $S_{|X|} = |X|$ and $S_m = m$ for identical machines, assuming that the speed of any machine is 1.

In our previous work [16], we demonstrated that a linear programming problem over $P(\varphi)_l^u$ can be reduced to optimization over a simpler structure, namely, over a base polyhedron. In fact, we have shown that a problem of maximizing a linear

function over the intersection of a submodular polyhedron and a box is equivalent to maximizing the same objective function over a base polyhedron associated with another rank function.

THEOREM 1 (cf. [16]). *Polyhedron $P(\varphi)_l^u$ is nonempty if and only if $\mathbf{l} \in P(\varphi)$ and $\mathbf{l} \leq \mathbf{u}$. If $P(\varphi)_l^u$ is nonempty, then the set of maximal vectors in $P(\varphi)_l^u$ is a base polyhedron $B(\psi)$ associated with the submodular system $(2^N, \psi)$, where the submodular rank function $\psi : 2^N \to \mathbb{R}$ is given by*

$$\psi(X) = \min_{Y \in 2^N} \{\varphi(Y) + u(X \setminus Y) - l(Y \setminus X)\},\ X \in 2^N. \tag{11}$$

A detailed proof of Theorem 1 is given in [16]. Notice that Theorem 1 can also be derived from Proposition II.2.11 of [2], which addresses the truncation operation for generalized polymatroids.

Since $u(X \setminus Y) = u(X) - u(X \cap Y)$ for $X, Y \subseteq N$, we may rewrite (11) to obtain

$$\psi(X) = u(X) + \min_{Y \in 2^N} \{\varphi(Y) - u(X \cap Y) - l(Y \setminus X)\}. \tag{12}$$

For the problems under consideration, finding a schedule with the makespan $C_{\max} = d$ and the minimum total compression cost $W$ reduces to the problem of maximizing the function $\sum_{j \in N} w(j)p(j)$ over $P(\varphi)_l^u$. In turn, due to Theorem 1, the latter problem reduces to

$$\begin{aligned} &\text{Maximize} \quad \sum_{j \in N} w(j)p(j) \\ &\text{subject to} \quad \mathbf{p} \in B(\psi). \end{aligned} \tag{13}$$

The benefit gained by such a reduction is the possibility of using the most well-known result of submodular optimization that guarantees that a solution to the problem of maximizing a linear function over a base polyhedron can be found by a greedy algorithm. Informally, to determine an optimal vector $\mathbf{p}^*$ such an algorithm starts with $\mathbf{p}^* = \mathbf{l}$ and considers the components of the current $\mathbf{p}^*$ in the sequence $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ such that

$$w(\sigma(1)) \geq w(\sigma(2)) \geq \cdots \geq w(\sigma(n)), \tag{14}$$

giving the current component the largest possible increment that keeps the vector feasible.

Another advantage of the reduction to a problem of the form (13) is that the solution vector $\mathbf{p}^*$ can be obtained essentially in a closed form, as stated in the theorem below. Define

$$N_t(\sigma) = \{\sigma(1), \ldots, \sigma(t)\},\ 1 \leq t \leq n; \tag{15}$$

for completeness, define $N_0(\sigma) = \emptyset$.

THEOREM 2 (cf. [3]). *Given an LP problem of the form* (13), *let $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be an ordering of elements in $N$ that satisfies* (14). *Then, vector $\mathbf{p}^* \in \mathbb{R}^N$ given by*

$$p^*(\sigma(t)) = \psi(N_t(\sigma)) - \psi(N_{t-1}(\sigma)),\ t = 1, 2, \ldots, n,$$

*is an optimal solution to problem* (13).

Theorem 2 provides the foundation for our new approach that finds the efficiency frontier of the bicriteria scheduling problems in a closed form.

Let $S^*(d)$ denote a schedule with a makespan $C_{\max} = d$ that minimizes the total compression cost. The solution to a bicriteria problem will be delivered as a collection of break points of the efficiency frontier $(d, W(d))$, where $d$ is a value of the makespan of schedule $S^*(d)$ and $W(d)$ is a (piecewise-linear in $d$) function that represents the total optimal compression cost. Let also $p^*(j,d)$ denote the optimal value of the actual processing time of job $j$ in schedule $S^*(d)$. It follows that

$$W(d) = \sum_{t=1}^{n} w\left(\sigma(t)\right) p^*\left(\sigma(t), d\right). \tag{16}$$

For the problems under consideration, due to (7), (9), and (10), the rank function $\varphi(X)$ as well as the function $\psi(X)$ are functions of $d$; therefore in this paper we may write $\varphi(X,d)$ and $\psi(X,d)$ whenever we want to stress that dependence.

Given a value of $d$ such that all jobs can be completed by time $d$, define a function

$$\psi_t(d) = \psi(N_t(\sigma), d), \qquad 1 \le t \le n, \tag{17}$$

computed for this value of $d$. By (12),

$$\psi_t(d) = u(N_t(\sigma)) + \min_{Y \in 2^N} \left\{\varphi(Y) - u(N_t(\sigma) \cap Y) - l(Y \setminus N_t(\sigma))\right\}.$$

For all scheduling problems under consideration, due to (7), (9), and (10), there are $m$ expressions for $\varphi(Y)$, depending on $|Y| \in \{1, 2, \ldots, m-1\}$ or $|Y| \ge m$. As we show in the following sections, finding the minimum in the above formula results in $\psi_t(d)$ represented as a piecewise-linear function of the form of an envelope

$$\psi_t(d) = u(N_t(\sigma)) + \begin{cases} dS_m + A_t^m & \text{for } \underline{d} < d \le d_t^m, \\ dS_{m-1} + A_t^{m-1} & \text{for } d_t^m < d \le d_t^{m-1}, \\ \vdots & \\ dS_1 + A_t^1 & \text{for } d_t^2 < d \le d_t^1, \\ 0 & \text{for } d_t^1 < d < +\infty, \end{cases} \tag{18}$$

where $\underline{d}$ denotes the smallest deadline $d$ for which there exists a feasible schedule $S(d)$, while the values $A_t^v$, $1 \le v \le m$, are appropriately determined problem-dependent constants. Their calculation will be explained in the subsequent sections. Notice that if for some $i$, $0 \le i \le m-1$, the function $u(N_t(\sigma)) + dS_{m-i} + A_t^{m-i}$ does not contribute into $\psi_t(d)$ as a piece, the break point $d_t^{m-i}$ is set equal $d_t^{m-i-1}$.

The value of $\underline{d}$ can be found for each problem in advance, since it is equal to the minimum makespan, provided that the processing times are equal to their lower bounds $l(j)$. For the model with uniform machines this takes $O(n + m \log m)$ time if the release dates are equal [4] and $O(n \log n + nm)$ time if the release dates are different [12]. For the model with identical machines and different release dates we can use the algorithm from [11] that requires $O(n \log m)$ time, provided that the jobs are numbered in accordance with (8).

Recall that by Theorem 2

$$W(d) = \sum_{t=1}^{n} w\left(\sigma(t)\right) p^*\left(\sigma(t), d\right) = \sum_{t=1}^{n} w\left(\sigma(t)\right) \left(\psi_t(d) - \psi_{t-1}(d)\right).$$

For completeness, define $w(\sigma(n+1)) = 0$ and rewrite

$$W(d) = \sum_{t=1}^{n} w(\sigma(t))(\psi_t(d) - \psi_{t-1}(d)) = \sum_{t=1}^{n} (w(\sigma(t)) - w(\sigma(t+1)))\psi_t(d). \tag{19}$$

Thus, in order to be able to compute the function $W(d)$, we first have to compute the functions $\psi_t(d)$, $t = 1, 2, \ldots, n$, for all relevant values of $d$ and then compute their weighted sum by (19). This function fully defines the efficiency frontier for the corresponding bicriteria scheduling problem.

Theorem 2 implies that we need a procedure that computes the value of a submodular function $\psi(X)$ for a given $X \in 2^N$. In the following sections, we explain how to compute function $\psi(X)$ for the scheduling problems under consideration and how to adapt the corresponding procedures for computing the functions $\psi_t(d)$ of the form (18).

Another feature of our approach that is common for all scheduling problems under consideration is related to computing function $W(d)$. Assume that for a scheduling problem piecewise-linear functions $\psi_t(d)$ of the form (18) are found. We can organize the break points of these functions as an $n \times m$ matrix

$$D = \begin{pmatrix} d_1^m & d_1^{m-1} & \cdots & d_1^1 \\ d_2^m & d_2^{m-1} & \cdots & d_2^1 \\ \vdots & & \ddots & \vdots \\ d_n^m & d_n^{m-1} & \cdots & d_n^1 \end{pmatrix}, \tag{20}$$

where each row is a nondecreasing array. Then, the weighted sum $W(d)$ given by (19) can be found by merging the arrays of the break points, obtaining a nondecreasing sequence of $O(nm)$ potential break points of function $W(d)$. It is straightforward to compute the value of $W(d)$ between any two consecutive break points.

It is well known that merging $k$ sorted arrays of $l$ elements each into a single sorted array requires $O(kl \log k)$ time; see, e.g., section 5.4.1 of [6]. Merging $k = n$ rows of $l = m$ elements of matrix $D$ would take $O(nm \log n)$ time. Thus, having found the functions $\psi_t(d)$, $1 \le t \le n$, the function $W(d)$ of the form (19) can be computed in $O(nm \log n)$ time.

On the other hand, if the columns of matrix $D$ are known to be sorted, then the list of the break points of $W(d)$ can be obtained in $O(nm \log m)$ time by merging $k = m$ columns of $l = n$ elements each, which is an improvement over $O(nm \log n)$ for $n \ge m$.

Below we present a sufficient condition for the columns of matrix $D$ to be ordered. The following lemma holds for all scheduling problems under consideration, provided the rank functions $\psi_t(d)$ satisfy (18).

LEMMA 1. *Let $D$ be the matrix of the break points of the piecewise-linear functions $\psi_t(d)$, $1 \le t \le n$, of the form* (18). *Then, for $i$, $1 \le i \le m$, and $t$, $1 \le t \le n-1$, the inequality*

$$A_t^{i-1} - A_t^i \le A_{t+1}^{i-1} - A_{t+1}^i \tag{21}$$

*implies that*

$$d_t^i \le d_{t+1}^i. \tag{22}$$

*Proof.* For simplicity, we present the proof assuming that the intervals $(d_t^{i-1}, d_t^i]$ and $(d_{t+1}^{i-1}, d_{t+1}^i]$ are both nonempty. The proof can be appropriately adjusted to handle the case that at least one of these intervals is empty.

The break point $d_t^i$ is the solution of the equation

$$u(N_t(\sigma)) + dS_{i-1} + A_t^{i-1} = u(N_t(\sigma)) + dS_i + A_t^i,$$

i.e., $d_t^i = (A_t^{i-1} - A_t^i)/(S_i - S_{i-1}) = (A_t^{i-1} - A_t^i)/s_i$, where $s_i$ is the speed of machine $M_i$. Similarly, $d_{t+1}^i = (A_{t+1}^{i-1} - A_{t+1}^i)/s_i$. Then (21) implies (22), as required. ∎

As discussed in section 3, for one of the scheduling problems due to its special structure the running time for computing function $W(d)$ can be reduced to $O(nm \log m)$. For the other problems, such a reduction, even if possible, will not reduce the overall running time; see sections 4 and 5.

For all scheduling problems studied below, the corresponding rank functions $\varphi(X)$ given by (7), (9), and (10) depend on the cardinality of set $X$. This is why in the subsequent consideration it is convenient to use the sets

$$\mathcal{Y}_v = \{Y \in 2^N \mid |Y| = v\},\ 1 \le v \le n, \tag{23}$$

which contain all subsets of the ground set with exactly $v$ elements; for completeness we define $\mathcal{Y}_0 = \{\emptyset\}$.

**3. Uniform machines, common release date: Rank function computation.** In this section, we consider problem $Q|u(j)-x(j), pmtn|(C_{\max}, W)$ and present a procedure for computing the rank function $\psi(X)$ for a given deadline $d$ and set $X \subseteq N$. Then we show how that procedure can be adapted for finding all functions $\psi_t(d) = \psi(N_t(\sigma), d)$ for all $t \in N$ as piecewise-linear functions of $d$.

We assume that the machines are numbered in accordance with (1) and the values $S_0, S_1, \ldots, S_m$ are defined by (6).

For given $d$ and set $X$, due to (7) and (12) we may write $\psi(X) = \min\{\psi'(X), \psi''(X)\}$, where

$$\psi'(X) = u(X) + \min_{0 \le v \le m-1} \left\{ dS_v - \max_{Y \in \mathcal{Y}_v} \{u(X \cap Y) + l(Y \backslash X)\} \right\}; \tag{24}$$

$$\psi''(X) = u(X) + dS_m - \max_{v \ge m} \left\{ \max_{Y \in \mathcal{Y}_v} \{u(X \cap Y) + l(Y \backslash X)\} \right\}; \tag{25}$$

recall the definition of $\mathcal{Y}_v$ in (23).

LEMMA 2. *For problem $Q|u(j)-x(j), pmtn|(C_{\max}, W)$, given a deadline $d$ and a set $X$, define $U$ to be a list of values $(u(j)|j \in X)$ and define $L$ to be a list of values $(l(j)|j \in N \backslash X)$; let $\beta_z$ denote the $z$th largest element in the merger of these lists. Then*

$$\psi'(X) = u(X) + \min_{0 \le v \le m-1} \left\{ dS_v - \sum_{j=1}^{v} \beta_j \right\}, \tag{26}$$

$$\psi''(X) = dS_m - l(N \backslash X). \tag{27}$$

*Proof.* Observe that for each $v$, $0 \le v \le m-1$, the equality

$$\max_{Y \in \mathcal{Y}_v} \{u(X \cap Y) + l(Y \backslash X)\} = \sum_{j=1}^{v} \beta_j$$

holds, and (26) immediately follows from (24).

To compute $\psi''(X)$, we need to determine $\max_{Y\in\mathcal{Y}_v, v\geq m}\{u(X\cap Y)+l(Y\setminus X)\}$. Since each job in $Y$ contributes either the lower bound or the upper bound on its processing time, it follows that

$$\max_{Y\in\mathcal{Y}_v, v\geq m}\{u(X\cap Y)+l(Y\setminus X)\} = u(X)+l\left(N\backslash X\right),$$

so that (25) becomes $\psi''(X)=u(X)+dS_m-u(X)-l(N\backslash X)$ and (27) is valid. □

The value $\psi(X)$ can be found by the procedure below.

PROCEDURE PSICOMPQR0.

INPUT: An instance of problem $Q|u(j)-x(j), pmtn|(C_{\max}, W)$, a deadline $d$, and a set $X\subseteq N$

OUTPUT: The value $\psi(X)$

**Step 1.** Determine a list $U$ of values $(u(j)|j\in X)$ and a list $L$ of values $(l(j)|j\in N\backslash X)$. Determine the values $\beta_1, \beta_2, \ldots, \beta_{m-1}$, where $\beta_z$ is the $z$th largest element in the merger of the lists $U$ and $L$.

**Step 2.** Compute $\psi'(X)$ and $\psi''(X)$ by (26) and (27), respectively.

**Step 3.** Output $\psi(X)=\min\{\psi'(X), \psi''(X)\}$.

The most time-consuming part is Step 1, where choosing the $m-1$ largest elements in the merger of the lists $U$ and $L$ can be done in $O(n)$ time by using the median finding technique, and sorting the $m-1$ largest elements can be done in $O(m\log m)$ time. Hence, Procedure PsiCompQr0 requires $O(n+m\log m)$ time.

Procedure PsiCompQr0 will be the basis of our algorithm for solving the bicriteria problem $Q|u(j)-x(j), pmtn|(C_{\max}, W)$ presented below. Recall that we need to find the functions $\psi_t(d)=\psi(N_t(\sigma), d)$ computed for all values of $d$ and all $t\in N$, where the sets $N_t(\sigma)$ are defined by (15). As mentioned in section 2, the value of $\underline{d}$, the smallest deadline $d$ for which a feasible schedule exists, can be found in $O(n+m\log m)$ time by an algorithm from [4].

Given a $t$, $1\leq t\leq n$, define $\beta_{t,z}, 1\leq z\leq m-1$, as the $z$th largest element in the merger of the lists $U_t$ of the values $(u(j)|j\in N_t(\sigma))$ and $L_t$ of the values $(l(j)|j\in N\backslash N_t(\sigma))$. It follows from Lemma 2 that

$$\psi_t(d) = u(N_t(\sigma)) + \min_{0\leq v\leq m}\left\{dS_v + A_t^v\right\}, \tag{28}$$

where

$$A_t^v = \begin{cases} 0 & \text{for } v=0,\\ -\sum_{z=1}^{v}\beta_{t,z} & \text{for } 1\leq v\leq m-1,\\ -l(N\backslash N_t(\sigma)) - u(N_t(\sigma)) & \text{for } v=m. \end{cases} \tag{29}$$

Notice that once all values $A_t^v$ are found for some $t$, $1\leq t\leq n$, determining $\psi_t(d)$ in the form (18) or (28) is equivalent to the problem of finding the lower envelope of $m$ linear functions given in increasing order of their slopes $S_v$. Exactly this problem has been studied in [20] and has been shown to be solvable in $O(m)$ time. We use that method as part of our algorithm presented below. Our algorithm is based on Procedure PsiCompQr0 applied to $X=N_t(\sigma)$ for all $t=1,\ldots,n$. For each $t$, $0\leq t\leq n$, it maintains a sorted list $V_t$ of $m-1$ largest values among $\{u(j)\mid j\in N_t(\sigma)\}\cup\{l(j)\mid j\in N\backslash N_t(\sigma)\}$.

ALGORITHM ALLPSIQR0.

INPUT: An instance of problem $Q|u(j)-x(j),pmtn|(C_{\max},W)$

OUTPUT: A collection of functions $\psi_t(d)$, $1\le t\le n$, each in a piecewise-linear form

**Step 1.** Find a sequence $\sigma$ defined by (14). If required, renumber the machines so that (1) holds and compute the values $S_v$, $1\le v\le m$, by (6). Compute $\underline{d}$ by running an algorithm from [4] applied to $p(j)=l(j)$, $j\in N$.

**Step 2.** Create list $V_0$ that contains $m-1$ largest values $l(j)$, $j\in N$, sorted in nonincreasing order. Define $N_0(\sigma):=\emptyset$ and set $u(N_0(\sigma)):=0$ and $l(N\backslash N_0(\sigma)):=l(N)$.

**Step 3.** For $t$ from 1 to $n$ do:

**(a)** Set $N_t(\sigma):=N_{t-1}(\sigma)\cup\{\sigma(t)\}$, $u(N_t(\sigma)):=u(N_{t-1}(\sigma))+u(\sigma(t))$, and $l(N\backslash N_t(\sigma)):=l(N\backslash N_{t-1}(\sigma))-l(\sigma(t))$.
If $u(\sigma(t))$ is less than the smallest element of $V_{t-1}$, rename $V_{t-1}$ as $V_t$ without updating it and go to Step 3(b).
Else delete from $V_{t-1}$ the element $l(\sigma(t))$, if it belongs to $V_{t-1}$, or its smallest element, otherwise. Insert $u(\sigma(t))$ in that list keeping the resulting list in nonincreasing order. Call the resulting list $V_t$.

**(b)** Taking the elements in list $V_t$ in the order of appearance, rename them by $\beta_{t,z}$, $1\le z\le m-1$. Scanning the values $\beta_{t,z}$ in the order of their numbering, compute the sums $\sum_{z=1}^{v}\beta_{t,z}$, $1\le v\le m-1$, and thereby find the values $A_t^v$ by (29).

**(c)** Compute $A_t^m:=-l(N\backslash N_t(\sigma))-u(N_t(\sigma))$.

**(d)** Use the algorithm from [20] to determine function $\psi_t(d)$ in the form (18), as a lower envelope given by its break points $\underline{d}\le d_t^m\le d_t^{m-1}\le\cdots\le d_t^1$.

Let us estimate the running time of Algorithm AllPsiQr0. Step 1 is the preprocessing stage that requires $O(n\log n)$ time. Step 2 can be implemented in $O(n+m\log m)$ time, since choosing the $m-1$ largest values takes $O(n)$ time and sorting them requires $O(m\log m)$ time. For a typical iteration $t$ of the loop in Step 3 the updates in Step 3(a) require $O(m)$ time. Since the list $V_t$ is kept sorted, the values $\beta_{t,z}$, $1\le z\le m-1$, and all their partial sums can be found in $O(m)$ time. Thus, all values $A_t^v$, $1\le v\le m$, can be found in $O(m)$ time. The algorithm from [20] employed in Step 3(d) also needs $O(m)$ time. Thus, the following statement holds.

LEMMA 3. *For problem $Q|u(j)-x(j),pmtn|(C_{\max},W)$ the functions $\psi_t(d)$ for all $t$, $1\le t\le n$, can be computed in $O(n\log n+nm)$ time.*

As mentioned in section 2, computing function $W(d)$ of the form (19) that determines the efficiency frontier for the original bicriteria scheduling problem additionally requires $O(nm\log n)$ time. However, for the problem under consideration this running time can be reduced due to the following statement.

LEMMA 4. *Let $D$ be the matrix of the break points of the piecewise-linear functions $\psi_t(d)$, $1\le t\le n$, of the form* (18) *computed for problem $Q|u(j)-x(j),pmtn|(C_{\max},W)$. Then, for all $i$, $1\le i\le m$ and all $t$, $1\le t\le n-1$, the inequality* (21) *holds.*

*Proof.* Take an arbitrary $t$, $1\le t\le n-1$. First, consider the case that $1\le i\le m-1$, so that in accordance with (29)

$$A_t^{i-1}=-\sum_{z=1}^{i-1}\beta_{t,z},\ A_t^i=-\sum_{z=1}^{i}\beta_{t,z},$$

and $A_t^{i-1}-A_t^i=\beta_{t,i}$, where $\beta_{t,i}$ is the $i$th largest element in list $V_t$. Similarly, $A_{t+1}^{i-1}-A_{t+1}^i=\beta_{t+1,i}$. Recall that list $V_{t+1}$ either coincides with $V_t$ or is obtained

from list $V_t$ by replacing an element $x$ of list $V_t$ by $u(\sigma(t+1))$, where

$$l\left(\sigma\left(t+1\right)\right) \leq x \leq u\left(\sigma\left(t+1\right)\right); \tag{30}$$

see Step 3(a) of Algorithm AllPsiQr0. As the result, the $i$th largest element in list $V_{t+1}$ is no smaller than in list $V_t$, i.e.,

$$A_t^{i-1} - A_t^i = \beta_{t,i} \leq \beta_{t+1,i} = A_{t+1}^{i-1} - A_{t+1}^i,$$

and (21) holds.

Now we look at the case $i = m$. It follows from (29) that

$$A_t^{m-1} = -\sum_{z=1}^{m-1} \beta_{t,z},\ A_{t+1}^{m-1} = -\sum_{z=1}^{m-1} \beta_{t+1,z}.$$

Also (29) implies that

$$\begin{aligned} A_{t+1}^m &= -l(N\backslash N_{t+1}(\sigma)) - u(N_{t+1}(\sigma)) \\ &= -l(N\backslash N_t(\sigma)) + l\left(\sigma\left(t+1\right)\right) - u(N_t(\sigma)) - u\left(\sigma\left(t+1\right)\right) \\ &= A_t^m + l\left(\sigma\left(t+1\right)\right) - u\left(\sigma\left(t+1\right)\right) \leq A_t^m. \end{aligned}$$

If the lists $V_t$ and $V_{t+1}$ coincide, then $A_t^{m-1} = A_{t+1}^{m-1} = -\sum_{z=1}^{m-1} \beta_{t,z}$ and

$$A_t^{m-1} - A_t^m = A_{t+1}^{m-1} - A_t^m \leq A_{t+1}^{m-1} - A_{t+1}^m,$$

as required.

If the two lists are different, then, as explained above, this is due to the fact that in Step 3(a) of Algorithm AllPsiQr0 an element $x$ of list $V_t$ is replaced by $u(\sigma(t+1))$, so that (30) holds. This implies that

$$\begin{aligned} A_{t+1}^{m-1} - A_{t+1}^m &= \left(-\sum_{z=1}^{m-1} \beta_{t,z} + x - u\left(\sigma\left(t+1\right)\right)\right) - \left(A_t^m + l\left(\sigma\left(t+1\right)\right) - u\left(\sigma\left(t+1\right)\right)\right) \\ &= A_t^{m-1} - A_t^m + x - l\left(\sigma\left(t+1\right)\right) \geq A_t^{m-1} - A_t^m, \end{aligned}$$

which completes the proof. □

Lemma 4 allows us to give the following estimate of the running time required to solve problem $Q|u(j) - x(j), pmtn|(C_{\max}, W)$.

THEOREM 3. *Problem $Q|u(j) - x(j), pmtn|(C_{\max}, W)$ is solvable in $O(nm \log m)$ time.*

**4. Uniform machines, different release dates: Rank function computation.** In this section, we consider problem $Q|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$ and present a procedure for computing the rank function $\psi(X)$ for a given deadline $d$ and set $X \subseteq N$. Then we show how that procedure can be adapted for finding all functions $\psi_t(d) = \psi(N_t(\sigma), d)$ for all $t \in N$ as piecewise-linear functions of $d$.

We assume that the jobs are numbered in nondecreasing order of the release dates, i.e., in accordance with (8). As above, the machines are numbered in accordance with (1), and the values $S_0, S_1, \ldots, S_m$ are defined by (6). As in section 2, let $r_i(X)$ denote the $i$th smallest release date for the jobs of set $X$.

Due to (9) and (12) we represent the function $\psi(X)$ as $\psi(X) = \min\{\psi'(X), \psi''(X)\}$, where

$$\psi'(X) = u(X) + \min_{0\le v\le m-1}\left\{dS_v - \max_{Y\in\mathcal{Y}_v}\left\{\sum_{i=1}^{v} s_i r_i(Y) + u(X\cap Y) + l\left(Y\backslash X\right)\right\}\right\}, \tag{31}$$

$$\psi''(X) = u(X) + dS_m - \max_{Y\in\mathcal{Y}_v, v\ge m}\left\{\sum_{i=1}^{m} s_i r_i(Y) + u(X\cap Y) + l\left(Y\backslash X\right)\right\}. \tag{32}$$

We describe a dynamic programming procedure that computes the function $\psi(X)$ for given $d$ and $X$. To compute $\psi'(X)$ we define

$$\lambda(j) = \begin{cases} u(j) & \text{if } j \in X, \\ l(j) & \text{if } j \notin X \end{cases} \tag{33}$$

and introduce

$$z_v(Y) = \sum_{i=1}^{v} s_i r_i(Y) + \lambda(Y),$$

so that (31) can be rewritten as

$$\psi'(X) = u(X) + \min_{0\le v\le m-1}\left\{dS_v - \max_{Y\in\mathcal{Y}_v}\left\{z_v(Y)\right\}\right\}.$$

For some $v$, $1 \le v \le m-1$, consider job $k$, $v \le k \le n$, and the subsets in $\mathcal{Y}_v$ that contain no jobs $j > k$. Denote the set of such subsets by $\mathcal{Y}_v[k]$. Let $z_v[k]$ be a real number given by

$$z_v[k] = \max\left\{z_v(Y) \mid Y \in \mathcal{Y}_v\left[k\right]\right\}.$$

We note that $\mathcal{Y}_v[n] = \mathcal{Y}_v$, and therefore in order to compute $\psi'(X)$ we need to determine the value

$$z_v[n] = \max\left\{z_v(Y) \mid Y \in \mathcal{Y}_v\right\}.$$

The lemma below shows that the values $z_v[j]$ can be computed by a dynamic programming approach.

LEMMA 5. *For each $v = 1, 2, \ldots, m-1$, we have*

$$z_v[v] = z_{v-1}[v-1] + s_v r(v) + \lambda(v), \tag{34}$$

$$z_v[j] = \max\{z_v[j-1], z_{v-1}[j-1] + s_v r(j) + \lambda(j)\} \quad \textit{for } j = v+1, \ldots, n, \tag{35}$$

*where it is assumed that $z_0[j] = 0$ for $j = 0, 1, 2, \ldots, n$.*

*Proof.* For each $v = 1, 2, \ldots, m-1$, the set family $\mathcal{Y}_v[v]$ contains only one set $\{1, 2, \ldots, v\}$. Therefore, we obtain (34) as follows:

$$z_v[v] = z_v(\{1, 2, \ldots, v\}) = \sum_{j=1}^{v}\{s_j r(j) + \lambda(j)\} = z_{v-1}[v-1] + s_v r(v) + \lambda(v).$$

We then suppose that $1 \leq v \leq m-1$ and $2 \leq j \leq n$. We have

$$\mathcal{Y}_v[j] = \mathcal{Y}_v[j-1] \cup \{Y \cup \{j\} \mid Y \in \mathcal{Y}_{v-1}[j-1]\}.$$

For $Y \in \mathcal{Y}_{v-1}[j-1]$, the equality $z_v(Y \cup \{j\}) = z_{v-1}(Y) + s_v r(j) + \lambda(j)$ holds, since the jobs are numbered in nondecreasing order of the release dates. Hence, (35) can be obtained as follows:

$$\begin{aligned}
&\max\{z_v(Y) \mid Y \in \mathcal{Y}_v[j]\} \\
&\quad = \max\{\max\{z_v(Y) \mid Y \in \mathcal{Y}_v[j-1]\}, \max\{z_v(Y \cup \{j\}) \mid Y \in \mathcal{Y}_{v-1}[j-1]\}\} \\
&\quad = \max\{z_v[j-1], z_{v-1}[j-1] + s_v r(j) + \lambda(j)\}. \qquad \square
\end{aligned}$$

We now explain how to compute function $\psi''(X)$. Suppose that we know the set $Y''$ such that $|Y''| \geq m$ and

$$\sum_{i=1}^{m} s_i r_i(Y'') + \lambda(Y'') = \max\left\{ \sum_{i=1}^{m} s_i r_i(Y) + \lambda(Y) \middle| Y \in \mathcal{Y}_v, v \geq m \right\}. \tag{36}$$

Let $k \in N$ be the job such that the set $\{j \in Y'' \mid j \leq k\}$ contains exactly $m$ elements. It should be noted that $k$ itself need not be an element of $Y''$. Since the jobs are numbered in nondecreasing order of the release dates, it follows that the jobs of set $Y''$ with $m$ smallest release dates are contained in the set $Y_1'' = \{j \in Y'' \mid j \leq k\}$, so that

$$\sum_{i=1}^{m} s_i r_i(Y'') = \sum_{i=1}^{m} s_i r_i(Y_1'').$$

Denote $Y_2'' = \{j \in Y'' \mid j > k\}$, so that $Y'' = Y_1'' \cup Y_2''$. Since

$$\sum_{i=1}^{m} s_i r_i(Y'') + \lambda(Y'') = \sum_{i=1}^{m} s_i r_i(Y_1'') + \lambda(Y_1'') + \lambda(Y_2''),$$

we may include all jobs $j > k$ into set $Y_2''$ to achieve the maximum in (36). Thus, for a chosen $k$, we define $Y_2'' = \{k+1, k+2, \ldots, n\}$.

In order to find set $Y''$, it suffices to determine set $Y_1''$, which can be done by performing a systematic search among all $m$-element subsets of jobs from $N$. In our computation of $\psi'(X)$ we use the values $z_v[k]$ and the sets $\mathcal{Y}_v[k]$ defined for $1 \leq v \leq m-1$ and $v \leq k \leq n$. We now extend our definitions to the case of $v = m$. In other words,

$$z_m[k] = \max_{Y \in \mathcal{Y}_m[k]} \left\{ \sum_{i=1}^{m} s_i r_i(Y) + \lambda(Y) \right\}, \tag{37}$$

where $\mathcal{Y}_m[k]$ denotes the set of all $m$-element subsets with no jobs $j > k$. For finding the values $z_m[k]$, we simply extend the procedure outlined in Lemma 5 for $v = m$. Let $Y_1[k]$ be a set in $\mathcal{Y}_m[k]$ such that

$$\sum_{i=1}^{m} s_i r_i(Y_1[k]) + \lambda(Y_1[k]) = z_m[k].$$

We also define

$$Y_2[k] = \{j \in N \mid j > k\}.$$

Then, $\psi''(X)$ can be represented as

$$\psi''(X) = u(X) + dS_m - \max_{m \le k \le n} \left\{ \sum_{i=1}^{m} s_i r_i(Y_1[k]) + \lambda(Y_1[k]) + \lambda(Y_2[k]) \right\}$$
$$= u(X) + dS_m - \max_{m \le k \le n} \left\{ z_m[k] + \sum_{j=k+1}^{n} \lambda(j) \right\}.$$

The value $\psi(X)$ can be found by the procedure below.

PROCEDURE PSICOMPQRJ.

INPUT: An instance of problem $Q|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$, a deadline $d$, and a set $X \subseteq N$

OUTPUT: The value $\psi(X)$

**Step 1.** For $k = 0, 1, 2, \ldots, n$, set $z_0[k] = 0$. For $v = 1, 2, \ldots, m$ and $k = v, v+1, \ldots, n$, compute the value $z_v[k]$ by using the recursive formulas (34) and (35). Compute $\Lambda(k) = \sum_{j=k+1}^{n} \lambda(j)$ for $k \in N$.

**Step 2.** Compute

$$\psi'(X) = u(X) + \min_{0 \le v \le m-1} \{dS_v - z_v[n]\},$$
$$\psi''(X) = u(X) + dS_m - \max_{m \le k \le n} \{z_m[k] + \Lambda(k)\}. \tag{38}$$

**Step 3** Output $\psi(X) = \min\{\psi'(X), \psi''(X)\}$.

The dynamic programming computation in Step 1 requires $O(nm)$ time, and Step 2 can be done in $O(m+n)$ time. Thus, the overall time complexity of Procedure PsiCompQrj is $O(nm)$.

In order to obtain the functions $\psi_t(d)$ of the form (18), we run Procedure PsiCompQrj for $X = N(\sigma(t))$, $1 \le t \le n$, and define

$$A_t^v = \begin{cases} 0 & \text{for } v = 0, \\ -z_v[n] & \text{for } 1 \le v \le m-1, \\ -\max_{m \le k \le n}\{z_m[k] + \Lambda(k)\} & \text{for } v = m. \end{cases} \tag{39}$$

Having found these values we find the representation of the function $\psi_t(d)$ in the form of a lower envelope, as in Step 3(d) of Algorithm AllPsiQr0.

As the preprocessing stage, we need to determine the value of $\underline{d}$ and the sequence $\sigma$, which takes no more than $O(n \log n + nm)$ time. All functions $\psi_t(d)$ for all $t$, $1 \le t \le n$, will be found in $O(n^2 m)$ time. As mentioned in section 2, computing function $W(d)$ of the form (19) that determines the efficiency frontier for the original bicriteria scheduling problem additionally requires $O(nm \log n)$ time. Thus, the following statement holds.

THEOREM 4. *For problem $Q|u(j) - x(j), pmtn|(C_{\max}, W)$, finding the functions $\psi_t(d)$ for all $t$, $1 \le t \le n$, followed by the computation of function $W(d)$ requires $O(n^2 m)$ time.*

**5. Identical machines, different release dates: Rank function computation.** The approach from the previous section is of course applicable to problem

$P|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$ with identical parallel machines. However, for the latter problem a more efficient procedure for computing the rank function $\psi(X)$ for a given deadline $d$ and set $X \subseteq N$ can be developed. We present such a procedure below and show how that procedure can be adapted for finding all functions $\psi_t(d) = \psi(N_t(\sigma), d)$ for all $t \in N$ as piecewise-linear functions of $d$.

As above, we assume that the jobs are numbered in nondecreasing order of the release dates, i.e., in accordance with (8). As in section 2, for the jobs of set $X$, let $R(X)$ denote the sum of $m$ smallest release dates (provided that $|X| \geq m$), while $r(X)$ denotes the sum of all release dates.

We have $\psi(X) = \min\{\psi'(X), \psi''(X)\}$, where $\psi'(X)$ and $\psi''(X)$ are given by (31) and (32), respectively. Since we consider identical parallel machines, (31) can be simplified as

$$\begin{aligned}\psi'(X) &= u(X) + \min_{0 \leq v \leq m-1} \left\{ dS_v - \max_{Y \in \mathcal{Y}_v} \{r(Y) + u(X \cap Y) + l(Y \backslash X)\} \right\} \\ &= u(X) + \min_{0 \leq v \leq m-1} \left\{ dS_v - \max_{Y \in \mathcal{Y}_v} \{\tilde{u}(X \cap Y) + \tilde{l}(Y \setminus X)\} \right\},\end{aligned}$$

where $\tilde{u}(j) = r(j) + u(j)$ for $j \in X$ and $\tilde{l}(j) = r(j) + l(j)$ for $j \in N \setminus X$. This formula is the same as formula (24) in section 3, except that $u(j)$ and $l(j)$ in (24) are replaced with $\tilde{u}(j)$ and $\tilde{l}(j)$, respectively. Therefore, for a given set $X$ and a deadline $d$, the value $\psi'(X)$ can be computed by a slight modification of Procedure PsiCompQr0, which requires $O(n + m \log m)$ time.

We now show that the computation of $\psi''(X)$ in Procedure PsiCompQrj of section 4 can be made faster. The most time-consuming part in Procedure PsiCompQrj is to compute the values $z_m[m], z_m[m+1], \ldots, z_m[n]$ in formula (38), and below we show that this can be done more efficiently in $O(n \log m)$ time, instead of $O(nm)$ time.

For $j \in N$, define $\lambda(j)$ by (33). The values $z_m[k]$ defined by (37) can be computed as described in the procedure below.

Procedure Psi2CompPrj.

Input: An instance of problem $P|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$, a deadline $d$, and a set $X \subseteq N$

Output: The value $\psi''(X)$

**Step 1.** Define the values $\gamma(j) := r(j) + \lambda(j)$ for $j \in N$.

**Step 2.** Define $Q := \{\gamma(1), \ldots, \gamma(m)\}$ and $z_m[m] := \sum_{j \in Q} \gamma(j)$.

**Step 3.** For $j$ from $m+1$ to $n$ do:

Find $\gamma(z)$, the smallest value in $Q$. If $\gamma(z) \geq \gamma(j)$, define $z_m[j] := z_m[j-1]$; otherwise define

$$Q := (Q \setminus \{\gamma(z)\}) \cup \{\gamma(j)\}, \qquad z_m[j] := z_m[j-1] - \gamma(z) + \gamma(j).$$

**Step 4.** Compute $\psi''(X)$ by (38).

Notice that in each iteration $j$ of the loop in Step 3 of Procedure Psi2CompPrj, set $Q$ consists of $m$ largest values in $\{\gamma(i) \mid i = 1, 2, \ldots, j\}$. Since we need to get access to the smallest element in $Q$, we implement $Q$ as a heap.

Step 1 takes $O(n)$ time. Forming a heap for set $Q$ in Step 2 requires $O(m \log m)$ time. For each $j$ in the loop in Step 3 the required updates can be done in $O(\log m)$ time. Step 4 needs $O(n)$ time. Thus, the overall time complexity for computing $\psi''(X)$ is $O(n \log m)$.

Algorithm AllPsiQr0 can be modified in order to obtain the functions $\psi_t(d)$ of the form (18). In particular, the values $A_t^v$ for $t = 1, 2, \ldots, n$ and $v = 0, 1, \ldots, m-1$ can be computed quite similarly to Algorithm AllPsiQr0 in $O(n \log n + nm)$ time. The most time-consuming part is to compute the value $A_t^m$ for $t = 1, 2, \ldots, n$, each of which can be computed in $O(n \log m)$ time by Procedure Psi2CompPrj. Thus, finding all functions $\psi_t(d)$, $1 \le t \le n$, takes $O(n^2 \log m)$ time. As mentioned in section 2, computing function $W(d)$ of the form (19) that determines the efficiency frontier for the original bicriteria scheduling problem additionally requires $O(nm \log n)$ time. It can be verified that the function $n/\log n$ is nondecreasing as long as $n \ge e$. Thus, the inequality $n \log m \ge m \log n$ holds for all $m \le n$, except for $n = 3$ and $m = 2$. This implies that for all nontrivial values of $n$ and $m$ the equality $O(n(n \log m + m \log n)) = O(n^2 \log m)$ is valid. Thus, the following statement holds.

THEOREM 5. *For problem $P|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$, finding the functions $\psi_t(d)$ for all $t$, $1 \le t \le n$, followed by the computation of function $W(d)$ requires $O(n^2 \log m)$ time.*

**6. Conclusion.** The main contribution of this paper is a new methodology for solving bicriteria scheduling problems with controllable processing times. It is based on the reduction to optimization problems over submodular polyhedra and base polyhedra. The main stages of the new approach can be described as follows.

*Stage* 1. Derive an algorithm to compute a submodular rank function $\psi(X)$ of the form (11) for an arbitrary set $X$.

*Stage* 2. For each subset of jobs $N_t(\sigma)$, $1 \le t \le n$, defined by (15), compute parametric functions $\psi_t(d)$ of the form (17). Each $\psi_t(d)$ is a function of the parameter $d$ which represents the range of makespan values. Being a piecewise-linear function of the form (18), $\psi_t(d)$ is given by the set of its break points, slope values, and intercepts.

*Stage* 3. Find the efficiency frontier in the space of the makespan $C_{\max}$ and compression cost $W$. The cost function $W(d)$ is defined by Theorem 2, in accordance with either (16) or (19), as a parametric function of makespan values $d$.

Stage 1 is problem-specific. It is implemented by Procedures PsiCompQr0 and PsiCompQrj for problems $Q|u(j) - x(j), pmtn|(C_{\max}, W)$ and $Q|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$, respectively, while Procedure Psi2CompPrj is used for problem $P|r(j), u(j) - x(j), pmtn|(C_{\max}, W)$.

Stage 2 is problem-independent, but its efficient implementation may use specific features of function $\psi(X)$. For problem $Q|u(j) - x(j), pmtn|(C_{\max}, W)$, this stage is implemented as Algorithm AllPsiQr0 in section 3, and that algorithm can be easily modified for the other two problems, as described in sections 4 and 5.

Stage 3 is common for all problems and can be done in a straightforward manner, as described in section 2.

Our approach to bicriteria preemptive scheduling problems with controllable processing times is essentially different from earlier used approaches based on finding the break points of the efficiency frontier by tracking the changes in a schedule. Due to the close link with a problem formulation, any intermediate solution generated by traditional nonsubmodular approaches can easily be visualized in the form of a Gantt chart and its feasibility can easily be verified. Despite this minor advantage, early research in the area shows that the methods of such nature are not always effective and their justification may be overcomplicated. By contrast, our new methodology deals with optimization over various polyhedra defined by submodular constraints, and the actual meaning of an intermediate solution may be difficult to interpret in scheduling terms. However, due to the powerful techniques of submodular optimization, the so-

lution vector can be obtained essentially in a closed form and the efficiency frontier can be found directly as a parametric function. As a result, the new algorithms either are faster in comparison with the known methods or solve problems with no prior history of study.

**Acknowledgments.** The authors are grateful to two anonymous referees and an associate editor for their recommendations aimed at improving the content of this paper.

## REFERENCES

[1] P. Brucker, *Scheduling Algorithms*, 5th ed., Springer, Berlin, 2007.
[2] A. Frank and E. Tardos, *Generalized polymatroids and submodular flows*, Math. Program., 42 (1988), pp. 489–563.
[3] S. Fujishige, *Submodular Functions and Optimization*, 2nd ed., Ann. Discrete Math. 58, Elsevier, Dordrecht, Netherlands, 2005.
[4] T. F. Gonzales and S. Sahni, *Preemptive scheduling of uniform processor systems*, J. ACM, 25 (1978), pp. 92–101.
[5] H. Hoogeveen and G. J. Woeginger, *Some comments on sequencing with controllable processing times*, Computing, 68 (2001), pp. 181–192.
[6] D. Knuth, *The Art of Computer Programming, Vol.* 3: *Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.
[7] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *Sequencing and scheduling: Algorithms and complexity*, in Handbooks in Operations Research and Management Science, Logistics of Production and Inventory 4, S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, eds., North-Holland, Amsterdam, 1993, pp. 445–522.
[8] C. Martel, *Preemptive scheduling with release times, deadlines, and due dates*, J. ACM, 29 (1982), pp. 812–829.
[9] E. Nowicki and S. Zdrzałka, *A survey of results for sequencing problems with controllable processing times*, Discrete Appl. Math., 26 (1990), pp. 271–287.
[10] E. Nowicki and S. Zdrzałka, *A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times*, Discrete Appl. Math., 63 (1995), pp. 237–256.
[11] S. Sahni, *Preemptive scheduling with due dates*, Oper. Res., 27 (1979), pp. 925–934.
[12] S. Sahni and Y. Cho, *Scheduling independent tasks with due times on a uniform processor system*, J. ACM, 27 (1980), pp. 550–563.
[13] D. Shabtay and G. Steiner, *A survey of scheduling with controllable processing times*, Discrete Appl. Math., 155 (2007), pp. 1643–1666.
[14] N. V. Shakhlevich and V. A. Strusevich, *Pre-emptive scheduling problems with controllable processing times*, J. Sched., 8 (2005), pp. 233–253.
[15] N. V. Shakhlevich and V. A. Strusevich, *Preemptive scheduling on uniform parallel machines with controllable job processing times*, Algorithmica, 51 (2008), pp. 451–473.
[16] N. V. Shakhlevich, A. Shioura, and V. A. Strusevich, *Single machine scheduling with controllable processing times by submodular optimization*, Internat. J. Found. Comput. Sci., 20 (2009), pp. 247–269.
[17] A. Shioura, N. V. Shakhlevich, and V. A. Strusevich, *Decomposition Algorithms for Polymatroidal Optimization with Applications to Preemptive Scheduling with Controllable Processing Times*, Report 2010.01, School of Computing, University of Leeds, 2010.
[18] V. T'kindt and J.-C. Billaut, *Multicriteria Scheduling: Theory, Models and Algorithms*, Springer, Berlin, 2006.
[19] A. V. Tuzikov, *A bicriterion problem of the scheduling theory taking the variation of servicing time into account*, USSR Comput. Math. Math. Phys., 24 (1984), pp. 191–194.
[20] S. van Hoesel, A. Wagelmans, and B. Moerman, *Using geometric techniques to improve dynamic programming algorithms for the economic lot-sizing problem and extensions*, European J. Oper. Res., 75 (1994), pp. 312–331.
[21] L. N. Van Wassenhove and K. R. Baker, *A bicriterion approach to time/cost tradeoffs in sequencing*, European J. Oper. Res., 11 (1982), pp. 48–54.