Proceedings Paper:
Shioura, A, Chakhlevitch, NV and Strusevich, VA (2016) Handling Scheduling Problems with Controllable Parameters by Methods of Submodular Optimization. In: Kochetov, Y, Khachay, M, Beresnev, V, Nurminski, E and Pardalos, P, (eds.) Discrete Optimization and Operations Research. 9th International Conference, DOOR, 19-23 Sep 2016, Vladivostok, Russia. Lecture Notes in Computer Science (9869). Springer International Publishing , Cham, Switzerland , pp. 74-90. ISBN 978-3-319-44913-5

https://doi.org/10.1007/978-3-319-44914-2_7

Handling Scheduling Problems with Controllable Parameters by Methods of Submodular Optimization

Vitaly A. Strusevich[1], Akiyoshi Shioura[2], and Natalia V. Shakhlevich[3]

[1] Univeristy of Greenwich, London, U.K.
[2] Tokyo Institute of Technology, Tokyo, Japan
[3] University of Leeds, Leeds, U.K.

Abstract. In this paper, we demonstrate how scheduling problems with controllable processing times can be reformulated as maximization linear programming problems over a submodular polyhedron intersected with a box. We explain a decomposition algorithm for solving the latter problem and discuss its implications for the relevant problems of preemptive scheduling on a single machine and parallel machines.

1 Introduction

In *Scheduling with Controllable Processing Times* (*SCPT*), the actual durations of the jobs are not fixed in advance, but have to be chosen from a given interval. For a SCPT model, two types of decisions are required: (i) each job has to be assigned its actual processing time, and (ii) a schedule has to be found that provides a required level of quality. A penalty is applied for assigning shorter actual processing times. The quality of the resulting schedule is measured with respect to the cost of assigning the actual processing times that guarantee a certain scheduling performance. This area of scheduling has been active since the 1980s, see surveys [21] and [27].

Nemhauser and Wolsey were among the first who noticed that the SCPT models could be handled by methods of *Submodular Optimization* (*SO*); see, e.g., Example 6.1 (Section 6 of Chapter III.3) of their book [20]. A systematic development of a general framework for solving the SCPT problems via submodular methods has been initiated by Shakhlevich and Strusevich [28, 29] and further advanced in [30]. As a result, a powerful toolkit of the SO techniques [5, 26] can be used for design and justification of algorithms for solving a wide range of the SCPT problems. In this paper, we present convincing examples of a positive mutual influence of scheduling and submodular optimization, mainly based on our recent work [32]-[35].

2 Scheduling with Controllable Processing Times

The jobs of set $N = \{1, 2, \ldots, n\}$ have to be processed either on a single machine M_1 or on parallel machines $M_1, M_2, \ldots, M_m$, where $m \geq 2$. For each job $j \in N$,

its processing time $p(j)$ is not given in advance but has to be chosen from a given interval $\left[\underline{p}(j), \overline{p}(j)\right]$. That selection process is often seen as *compressing* the longest processing time $\overline{p}(j)$ down to $p(j)$. The value $x(j) = \overline{p}(j) - p(j)$ is called the *compression amount* of job j. Compression may decrease the completion time of each job j but incurs additional cost $w(j)x(j)$, where $w(j)$ is a given non-negative unit compression cost. The total cost is represented by the linear function $W = \sum_{j \in N} w(j)x(j)$.

Each job $j \in N$ is given a *release date* $r(j)$, before which it is not available, and a *deadline* $d(j)$, by which its processing must be completed. In the processing of any job, *preemption* is allowed, so that the processing can be interrupted on any machine at any time and resumed later, 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. Given a schedule, let $C(j)$ denote the completion time of job j. A schedule is called *feasible* if the processing of a job $j \in N$ takes place in the time interval $[r(j), d(j)]$.

We distinguish between the *identical* parallel machines and the *uniform* parallel machines. In the former case, the machines have the same speed. If the machines are uniform, then it is assumed that machine M_h has speed s_h, $1 \leq h \leq m$. Throughout this paper we assume that the machines are numbered in non-increasing order of their speeds, i.e.,

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

Adapting standard notation for scheduling problems [15], we denote a generic problem of our primary concern by $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|W$. Here, in the first field α we write "1" for a single machine, "P" in the case of $m \geq 2$ identical machines and "Q" in the case of $m \geq 2$ uniform machines. 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 "$p(j) = \overline{p}(j) - x(j)$" to indicate that the processing times are controllable and $x(j)$ is the compression amount to be found. The condition "$C(j) \leq d(j)$" reflects the fact that in a feasible schedule the deadlines should be respected; we write "$C(j) \leq d$", if all jobs have a common deadline d. The abbreviation "*pmtn*" is used to point out that preemption is allowed. Finally, in the third field we write the objective function to be minimized, which is the total compression cost $W = \sum w(j)x(j)$.

If the processing times $p(j)$, $j \in N$, are fixed then the corresponding counterpart of problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|W$ is denoted by $\alpha|r(j), C(j) \leq d(j), pmtn|\circ$. In the latter problem, it is required to verify whether a feasible preemptive schedule exists. If the deadlines are equal, then the counterpart of problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$ with fixed processing times can be denoted by $\alpha|r(j), pmtn|C_{\max}$, so that it is required to find a preemptive schedule that for the corresponding settings minimizes the makespan $C_{\max} = \max\{C(j) | j \in N\}$: if the optimal makespan is larger than d the required feasible schedule does not exist; otherwise, it exists.

Below we give examples of two most popular interpretations of the SCPT models. Alternative interpretations can be found, e.g., in [12] and [18].

In computing systems that support imprecise computation [16], some computations can be run partially, producing less precise results. In our notation, to produce a result of reasonable quality, the mandatory part of each task j must be completed, and this takes $\underline{p}(j)$ time units. If instead of an ideal computation time $\overline{p}(j)$ a task is executed for $p(j) = \overline{p}(j) - x(j)$ time, then computation is imprecise, and $x(j)$ corresponds to the error of computation. The objective function $W = \sum w(j)x(j)$ is interpreted as the total weighted error.

The SCPT problems serve as mathematical models in make-or-buy decision-making [30], where it is required to determine which part of each order j is manufactured internally and which is subcontracted. For this model, $p(j) = \overline{p}(j) - x(j)$ is understood as the chosen actual time for internal manufacturing, where $x(j)$ shows how much of the order is subcontracted and $w(j)x(j)$ is the cost of this subcontracting. Thus, we need to minimize the total subcontracting cost and to find a deadline-feasible schedule for internally manufactured orders.

These and other versions of the SCPT problems can be formulated as SO models and handled by SO methods.

3 Submodular Polyhedra and Decomposition Algorithm

In this section, we present some basic facts related to submodular optimization. Unless stated otherwise, we follow a comprehensive monograph on this topic by Fujishige [5], see also [14, 26]. We also describe a decomposition algorithm for solving a linear programming problem subject to submodular constraints.

For a positive integer n, let $N = \{1, 2, \ldots, n\}$ be a ground set, and 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 two vectors $\mathbf{p} = (p(1), p(2), \ldots, p(n)) \in \mathbb{R}^N$ and $\mathbf{q} = (q(1), q(2), \ldots, q(n)) \in \mathbb{R}^N$, we write $\mathbf{p} \leq \mathbf{q}$ if $p(j) \leq q(j)$ for each $j \in N$. For a vector $\mathbf{p} \in \mathbb{R}^N$, define $p(X) = \sum_{j \in X} p(j)$ for every set $X \in 2^N$.

A set function $\varphi : 2^N \to \mathbb{R}$ is called *submodular* if the inequality $\varphi(X) + \varphi(Y) \geq \varphi(X \cup Y) + \varphi(X \cap Y)$ holds for all sets $X, Y \in 2^N$. For a submodular function φ defined on 2^N such that $\varphi(\emptyset) = 0$, the pair $(2^N, \varphi)$ is called a *submodular system* on N, while φ is referred to as its *rank function*.

For a submodular system $(2^N, \varphi)$, define two polyhedra $P(\varphi) = \{\mathbf{p} \in \mathbb{R}^N \mid p(X) \leq \varphi(X), \quad X \in 2^N\}$ and $B(\varphi) = \{\mathbf{p} \in \mathbb{R}^N \mid \mathbf{p} \in P(\varphi), \quad p(N) = \varphi(N)\}$, called the *submodular polyhedron* and the *base polyhedron*, respectively, associated with the submodular system. The main problem of our interest is as follows:

$$\begin{aligned} (\mathrm{LP}) : \max \ & \sum_{j \in N} w(j)p(j) \\ \text{s.t.} \ & p(X) \leq \varphi(X), \qquad X \in 2^N, \\ & \underline{p}(j) \leq p(j) \leq \overline{p}(j), \, j \in N, \end{aligned} \tag{2}$$

where $\varphi : 2^N \to \mathbb{R}$ is a submodular function with $\varphi(\emptyset) = 0$, $\mathbf{w} \in \mathbb{R}^N_+$ is a nonnegative weight vector, and $\overline{\mathbf{p}}, \underline{\mathbf{p}} \in \mathbb{R}^N$ are upper and lower bound vectors, respectively. We refer to (2) as *Problem (LP)*. This problem serves as a mathematical model for many SCPT problems, as demonstrated in Sections 4-5.

Problem (LP) can be classified as a problem of maximizing a linear function over a submodular polyhedron intersected with a box. As shown in [30], Problem (LP) can be reduced to optimization over a base polyhedron.

Theorem 1 (cf. [30]). *If Problem (LP) has a feasible solution, then the set of its maximal feasible solutions is a base polyhedron $B(\tilde{\varphi})$ associated with the submodular system $(2^N, \tilde{\varphi})$, where the rank function $\tilde{\varphi} : 2^N \to \mathbb{R}$ is given by*

$$\tilde{\varphi}(X) = \min_{Y \in 2^N} \{\varphi(Y) + \overline{p}(X \setminus Y) - \underline{p}(Y \setminus X)\}. \tag{3}$$

Notice that the computation of the value $\tilde{\varphi}(X)$ for a given $X \in 2^N$ reduces to minimization of a submodular function, which can be done in polynomial time [11, 25]. However, the running time of known general algorithms is fairly large. In many special cases of Problem (LP), including its applications to the SCPT problems, the value $\tilde{\varphi}(X)$ can be computed more efficiently, as shown later.

Throughout this paper, we assume that Problem (LP) has a feasible solution, which is equivalent to the conditions $\underline{\mathbf{p}} \in P(\varphi)$ and $\underline{\mathbf{p}} \leq \overline{\mathbf{p}}$. Theorem 1 implies that Problem (LP) reduces to the following problem:

$$\begin{aligned} &\max \sum_{j \in N} w(j)p(j) \\ &\text{s.t. } \mathbf{p} \in B(\tilde{\varphi}), \end{aligned} \tag{4}$$

where the rank function $\tilde{\varphi} : 2^N \to \mathbb{R}$ is given by (3).

An advantage of the reduction of Problem (LP) to a problem of the form (4) is that the solution vector can be obtained essentially in a closed form by a greedy algorithm. To determine an optimal vector $\mathbf{p}^*$, the algorithm starts with $\mathbf{p}^* = \underline{\mathbf{p}}$, considers the components of the current $\mathbf{p}^*$ in non-increasing order of their weights and gives the current component the largest possible increment that keeps the vector feasible.

Introduce the sequence $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ such that $w(\sigma(1)) \geq w(\sigma(2)) \geq \cdots \geq w(\sigma(n))$ and define $N_t(\sigma) = \{\sigma(1), \ldots, \sigma(t)\}$, $1 \leq t \leq n$, where, for completeness, $N_0(\sigma) = \emptyset$.

Theorem 2 (cf. [5]). *Vector $\mathbf{p}^* \in \mathbb{R}^N$ given by*

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

is an optimal solution to problem (4) (and also to the problem (2)).

For certain problems of type (2) it is possible to get a faster algorithm in comparison with the greedy algorithm by decomposing Problem (LP) into subproblems and solving them recursively. We say that a subset $\hat{N} \subseteq N$ is a *heavy-element subset of N with respect to the weight vector* $\mathbf{w}$ if it satisfies the condition $\min_{j \in \hat{N}} w(j) \geq \max_{j \in N \setminus \hat{N}} w(j)$. For completeness, we also regard the empty set as a heavy-element subset of N. For a given set $X \subseteq N$, in accordance with (3) define a set $Y_* \subseteq N$ such that the equality

$$\tilde{\varphi}(X) = \varphi(Y_*) + \overline{p}(X \setminus Y_*) - \underline{p}(Y_* \setminus X) \tag{5}$$

holds. In the remainder of this paper, we call Y_* an *instrumental* set for set X.

Lemma 1 (cf. [33, 35]). *Let $\hat{N} \subseteq N$ be a heavy-element subset of N with respect to $\mathbf{w}$, and $Y_* \subseteq N$ be an instrumental set for set $\hat{N}$. Then, there exists an optimal solution $\mathbf{p}^*$ of Problem (LP) such that*
(a) $p^*(Y_*) = \varphi(Y_*)$, (b) $p^*(j) = \overline{p}(j)$, $j \in \hat{N} \setminus Y_*$, (c) $p^*(j) = \underline{p}(j)$, $j \in Y_* \setminus \hat{N}$.

In what follows, we use two fundamental operations on a submodular system $\left(2^N, \varphi\right)$, as defined in [5, Section 3.1]. For a set $A \in 2^N$, define a set function $\varphi^A : 2^A \to \mathbb{R}$ by $\varphi^A(X) = \varphi(X), \quad X \in 2^A$. Then, $(2^A, \varphi^A)$ is a submodular system on A and it is called a *restriction of* $(2^N, \varphi)$ *to* A. On the other hand, for a set $A \in 2^N$ define a set function $\varphi_A : 2^{N \setminus A} \to \mathbb{R}$ by $\varphi_A(X) = \varphi(X \cup A) - \varphi(A), \quad X \in 2^{N \setminus A}$. Then, $(2^{N \setminus A}, \varphi_A)$ is a submodular system on $N \setminus A$ and it is called a *contraction of* $(2^N, \varphi)$ *by* A.

Theorem 3 (cf. [33, 35]). *Let $\hat{N} \subseteq N$ be a heavy-element subset of N with respect to $\mathbf{w}$, and Y_* be an instrumental set for set $\hat{N}$. Let $\mathbf{p_1} \in \mathbb{R}^{Y^*}$ and $\mathbf{p_2} \in \mathbb{R}^{N \setminus Y^*}$ be optimal solutions of the linear programs (LPR) and (LPC), respectively, given by*

$$
\begin{aligned}
(\mathrm{LPR}) : \max \quad & \sum_{j \in Y_*} w(j) p(j) \\
\text{s.t.} \quad & p(X) \le \varphi(X), && X \in 2^{Y_*}, \\
& \underline{p}(j) \le p(j) \le \overline{p}(j), && j \in Y_* \cap \hat{N}, \\
& p(j) = \underline{p}(j), && j \in Y_* \setminus \hat{N}.
\end{aligned}
$$

$$
\begin{aligned}
(\mathrm{LPC}) : \max \quad & \sum_{j \in N \setminus Y_*} w(j) p(j) \\
\text{s.t.} \quad & p(X) \le \varphi(X \cup Y_*) - \varphi(Y_*), && X \in 2^{N \setminus Y_*}, \\
& \underline{p}(j) \le p(j) \le \overline{p}(j), && j \in (N \setminus Y_*) \setminus \left(\hat{N} \setminus Y_*\right), \\
& p(j) = \overline{p}(j), && j \in \hat{N} \setminus Y_*.
\end{aligned}
$$

Then, the vector $\mathbf{p}^* \in \mathbb{R}^N$ given by the direct sum $\mathbf{p}^* = \mathbf{p_1} \oplus \mathbf{p_2}$, where

$$
(p_1 \oplus p_2)(j) = \begin{cases} p_1(j), \text{ if } j \in Y_*, \\ p_2(j), \text{ if } j \in N \setminus Y_*. \end{cases}
$$

is an optimal solution of Problem (LP).

Notice that Problem (LPR) is obtained from Problem (LP) as a result of restriction to Y_* and the values of components $p(j), j \in Y_* \setminus \hat{N}$, are fixed to their lower bounds in accordance with Property (c) of Lemma 1. Similarly, Problem (LPC) is obtained from Problem (LP) as a result of contraction by Y_* and the values of components $p(j)$, $j \in \hat{N} \setminus Y_*$, are fixed to their upper bounds in accordance with Property (b) of Lemma 1.

Now we explain how the original Problem (LP) can be decomposed recursively based on Theorem 3, until we obtain a collection of trivially solvable problems with no non-fixed variables. As described in [33, 35], in each stage of the recursive procedure, we need to solve a subproblem that can be written in the following generic form:

$$\begin{aligned} \mathrm{LP}(H,F,K,\mathbf{l},\mathbf{u}) : \max \ & \sum_{j\in H} w(j)p(j) && (6)\\ \text{s.t.} \ & p(X) \le \varphi_K^H(X) = \varphi(X\cup K) - \varphi(K),\ X\in 2^H,\\ & l(j) \le p(j) \le u(j), && j\in H\setminus F,\\ & p(j) = u(j) = l(j), && j\in F, \end{aligned}$$

where $H \subseteq N$ is the index set of components of vector $\mathbf{p}$; $\mathbf{l} = (l(j) \mid j \in H)$ and $\mathbf{u} = (u(j) \mid j \in H)$ are, respectively, the current vectors of the lower and upper bounds on variables $p(j), j \in H$; $F \subseteq H$ is the index set of fixed components, i.e., $l(j) = u(j)$ holds for each $j \in F$; $K \subseteq N \setminus H$ is the set that defines the rank function $\varphi_K^H : 2^H \to \mathbb{R}$ such that $\varphi_K^H(X) = \varphi(X \cup K) - \varphi(K)$, $X \in 2^H$.

Suppose that Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (6) contains at least one non-fixed variable, i.e., $|H \setminus F| > 0$. We define a function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ by

$$\widetilde{\varphi}_K^H(X) = \min_{Y\in 2^H} \{\varphi_K^H(Y) + u(X \setminus Y) - l(Y \setminus X)\}. \tag{7}$$

By Theorem 1, the set of maximal feasible solutions of Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ is given as a base polyhedron $B(\widetilde{\varphi}_K^H)$ associated with the function $\widetilde{\varphi}_K^H$. Therefore, if $|H \setminus F| = 1$ and $H \setminus F = \{j'\}$, then an optimal solution $\mathbf{p}^* \in \mathbb{R}^H$ is given by

$$p^*(j) = \begin{cases} \tilde{\varphi}_K^H(\{j'\}), & j = j',\\ u(j), & j \in F. \end{cases} \tag{8}$$

Suppose that $|H \setminus F| \ge 2$. Then, we call a recursive Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$ explained below. Let $\hat{H} \subseteq H$ be a heavy-element subset of H with respect to the vector $(w(j) \mid j \in H)$, and $Y_* \subseteq H$ be an instrumental set for set $\hat{H}$, i.e.,

$$\widetilde{\varphi}_K^H(\hat{H}) = \varphi_K^H(Y_*) + u(\hat{H} \setminus Y_*) - l(Y_* \setminus \hat{H}). \tag{9}$$

Without going into implementation details, we follow [33, 35] and give a formal description of the recursive procedure. For the current problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$, we compute optimal solutions $\mathbf{p_1} \in \mathbb{R}^{Y_*}$ and $\mathbf{p_2} \in \mathbb{R}^{H\setminus Y_*}$ of the two subproblems by calling Procedures Decomp$(Y_*, F_1, K, \mathbf{l_1}, \mathbf{u_1})$ and Decomp$(H \setminus Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$. By Theorem 3, the direct sum $\mathbf{p}^* = \mathbf{p}_1 \oplus \mathbf{p_2}$ is an optimal solution of Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$, which is the output of Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$.

Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$

Step 1. If $|H \setminus F| = 0$, then output the vector $\mathbf{p}^* = \mathbf{u} \in \mathbb{R}^H$ and return.
If $|H \setminus F| = 1$ and $H \setminus F = \{j'\}$, then compute the value $\tilde{\varphi}_K^H(\{j'\})$, output the vector $\mathbf{p}^*$ given by (8) and return.

Step 2. Select a heavy-element subset $\hat{H}$ of $H \setminus F$ with respect to $\mathbf{w}$, and determine an instrumental set $Y_* \subseteq H$ for set $\hat{H}$ satisfying (9).

Step 3. Define the vectors $\mathbf{l_1}, \mathbf{u_1} \in \mathbf{R}^{Y_*}$ and set F_1 by

$$l_1(j) = l(j), j \in Y_*, \quad u_1(j) = \begin{cases} l(j), & j \in Y_* \setminus \hat{H},\\ u(j), & j \in Y_* \cap \hat{H}, \end{cases}; \quad F_1 = Y_* \setminus \hat{H},$$

Call Procedure DECOMP$(Y_*, F_1, K, \mathbf{l_1}, \mathbf{u_1})$ to obtain an optimal solution $\mathbf{p_1} \in \mathbb{R}^{Y_*}$ of Problem LP$(Y_*, F_1, K, \mathbf{l_1}, \mathbf{u_1})$.

Step 4. Define the vectors $\mathbf{l_2}, \mathbf{u_2} \in \mathbf{R}^{H \setminus Y_*}$ and set F_2 by

$$l_2(j) = \begin{cases} u(j), \ j \in \hat{H} \setminus Y_*, \\ l(j), \ j \in H \setminus (Y_* \cup \hat{H}), \end{cases} \quad u_2(j) = u(j), j \in H \setminus Y_*;$$

$$F_2 = (\hat{H} \cup (H \cap F)) \setminus Y_*.$$

Call Procedure DECOMP$(H \setminus Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$ to obtain an optimal solution $\mathbf{p_2} \in \mathbb{R}^{H \setminus Y_*}$ of Problem LP$(H \setminus Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$.

Step 5. Output the direct sum $\mathbf{p}^* = \mathbf{p_1} \oplus \mathbf{p_2} \in \mathbb{R}^H$ and return.

The original Problem (LP) is solved by calling Procedure DECOMP$(N, \emptyset, \emptyset, \underline{\mathbf{p}}, \overline{\mathbf{p}})$. Its actual running time depends on the choice of a heavy-element subset $\hat{H}$ in Step 2 and on the time complexity of finding an instrumental set Y_*. As proved in [33], if at each level of recursion a heavy-element set is chosen to contain roughly a half of the non-fixed variables, then the overall depth of recursion of Procedure DECOMP applied to Problem LP$(N, \emptyset, \emptyset, \underline{\mathbf{p}}, \overline{\mathbf{p}})$ is $O(\log n)$.

For a typical iteration of Procedure DECOMP applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ with $|H| = h$ and $|H \setminus F| = g$, let $T_{Y_*}(h)$ denote the running time for computing the value $\widetilde{\varphi}_K^H(\hat{H})$ for a given set $\hat{H} \subseteq H$ and finding an instrumental set Y_* in Step 2. In Steps 3 and 4, Procedure DECOMP splits Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ into two subproblems: one with h_1 variables among which $g_1 \leq \min\{h_1, \lceil g/2 \rceil\}$ variables are not fixed, and the other one with $h_2 = h - h_1$ variables, among which $g_2 \leq \min\{h_2, \lfloor g/2 \rfloor\}$ variables are not fixed. Let $T_{\text{Split}}(h)$ denote the time complexity for setting up the instances of these two subproblems. It is shown in [33, 35] that Problem (LP) can be solved by Procedure DECOMP in $O((T_{Y_*}(n) + T_{\text{Split}}(n)) \log n)$ time.

4 SCPT Problems with a Common Deadline

In this section, we review the results on the SCPT problems $\alpha|r(j), pmtn, C(j) \leq d|W$, where $\alpha \in \{1, P, Q\}$, provided that the jobs have a common deadline d. We also report the results on the makespan minimization versions $\alpha|r(j), pmtn|C_{\max}$ and the bicriteria versions $\alpha|r(j), pmtn, C(j) \leq d|\,(C_{\max}, W)$. For the latter type of problems, it is required to minimize both objective functions $C_{\max}$ and W simultaneously, in the Pareto sense, so that the solution is delivered in the form of an efficiency frontier.

First, assume that the release dates are equal to zero, so that the problems with a single machine are trivial. Solving problem $P|pmtn|C_{\max}$ with fixed processing times can be done by a linear-time algorithm [19]. As shown in [13], problem $P|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|W$ reduces to a continuous knapsack problem and can be solved in $O(n)$ time. The bicriteria problem $P|p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$ is solved in [28] by an $O(n \log n)$-time algorithm, which is the best possible.

In the case of uniform machines, the best known algorithm for solving problem $Q|pmtn|C_{\max}$ with fixed processing times is due to [7]. For problem $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|W$, it is shown in [22] how to find the actual processing times in $O(nm + n \log n)$ time. For the latter problem, Shakhlevich and Strusevich [29] use the SO reasoning to design an algorithm of the same running time and extend it to solving a bicriteria problem $Q|p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$. The best known algorithms for solving problems $Q|p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$ and $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|W$ are discussed in Section 4.2 and in Section 4.3, respectively; their time complexity is $O(nm \log m)$ and $O(n \log n)$.

For the models with different release dates, problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d\,(j)\,|W$ is one of the most studied SCPT problems. The first algorithm that requires $O(n \log n)$ time and provides all implementation details is developed in [28].

Problem $P|r(j), pmtn|C_{\max}$ with fixed processing times on m identical parallel machines can be solved in $O(n \log n)$ time, as proved in [23]. For problem $Q|r(j), pmtn|C_{\max}$, an algorithm for that requires $O(mn + n \log n)$ time is due to [24]. Prior to work of our team on the links between the SCPT problems and SO [32], no purpose-built algorithms had been known for problems $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|W$ and $\alpha|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|W$ with $\alpha \in \{P, Q\}$, as well as for their bicriteria counterparts. We consider these problems in Section 4.3 and Section 4.2, respectively.

4.1 Production Capacity Rank Functions

In this subsection, we present reformulations of the SCPT problems with a common deadline in terms of Problem (LP) with appropriately defined rank functions.

We assume that if the jobs have different release dates, they are numbered to satisfy

$$r(1) \leq r(2) \leq \ldots \leq r(n). \tag{10}$$

If the machines are uniform, they are numbered in accordance with (1). We denote

$$S_0 = 0, \qquad S_k = s_1 + s_2 + \cdots + s_k, \quad 1 \leq k \leq m. \tag{11}$$

S_k represents the total speed of k fastest machines; if the machines are identical, $S_k = k$ holds.

For each problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$, $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$ and $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$, we need to find the actual processing times $p(j) = \overline{p}(j) - x(j)$, $j \in N$, such that all jobs can be completed by a common deadline d and the total compression cost $W = \sum_{j \in N} w(j)x(j)$ is minimized. Each of these problems can be formulated as Problem (LP) with $p(j)$, $j \in N$, being decision variables, and the objective function to be maximized being $\sum_{j \in N} w(j)p(j) = \sum_{j \in N} w(j)\,(\overline{p}(j) - x(j))$. Since each decision variable $p(j)$ has a lower bound $\underline{p}(j)$ and an upper bound $\overline{p}(j)$,

the set of constraints of Problem (LP) includes the box constraints of the form $\underline{p}(j) \leq p(j) \leq \overline{p}(j)$, $j \in N$. Besides, the inequality

$$p\,(X) \leq \varphi(X) \tag{12}$$

should hold for each subset $X \subseteq N$ of jobs, where a meaningful interpretation of a rank function $\varphi(X)$ is the largest capacity available for processing the jobs of set X.

To determine $\varphi(X)$ for each of these SCPT problems, an important generic condition is available, which, according to [2] can be stated as follows: for a given deadline d a feasible schedule exists if and only if: (i) for each $k, 1 \leq k \leq m-1$, k longest jobs can be processed on k fastest machines by time d, and (ii) all n jobs can be completed on all m machines by time d.

For example, problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$, in which all jobs are simultaneously available from time zero, reduces to Problem (LP) of the form (2) with the rank function

$$\varphi(X) = dS_{\min\{|X|,m\}} = \begin{cases} dS_{|X|}, \text{ if } |X| \leq m-1, \\ dS_m, \quad \text{if } |X| \geq m. \end{cases} \tag{13}$$

It is clear that the conditions $p(X) \leq \varphi(X)$, $X \in 2^N$, for the function $\varphi(X)$ defined by (13) correspond to the conditions (i) and (ii) above, provided that $|X| \leq m-1$ and $|X| \geq m$, respectively. As proved in [29], function φ is submodular.

Given problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$, for a set of jobs $X \subseteq N$, we define $r_i(X)$ to be the i-th smallest release date in set $X \in 2^N$, $1 \leq i \leq |X|$. Then, for a non-empty 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 two fastest machines M_1 and M_2 is $s_1\,(d - r_1(X)) + s_2\,(d - r_2(X))$, etc. We deduce that

$$\varphi(X) = \begin{cases} dS_{|X|} - \sum_{i=1}^{|X|} s_i r_i(X), \text{ if } |X| \leq m-1, \\ dS_m - \sum_{i=1}^{m} s_i r_i(X), \quad \text{if } |X| \geq m, \end{cases} \tag{14}$$

which in the case of problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$ simplifies to

$$\varphi(X) = \begin{cases} d|X| - \sum_{i=1}^{|X|} r_i(X), \text{ if } |X| \leq m-1, \\ dm - \sum_{i=1}^{m} r_i(X), \quad \text{if } |X| \geq m. \end{cases} \tag{15}$$

It can be verified that functions (14) and (15) are submodular.

Thus, each of the three SCPT problems above reduces to Problem (LP) and in principle can be solved by the greedy algorithm discussed in Theorem 2. Similar reductions can be provided for other SCPT problems [30]. In what follows, we show that using the decomposition algorithm from Section 3, these problems can be solved faster.

Notice that the greedy reasoning has always been the main tool for solving the SCPT problems. However, in early papers on this topic, each individual

problem was considered separately and a justification of the greedy approach was often lengthy and developed from the first principles. In fact, as seen from above, the greedy nature of the solution approaches is due to the fact that many SCPT problems can be reformulated in terms of linear programming problems with submodular constraints.

4.2 Solving Bicriteria Problems by Submodular Methods

Theorem 2 provides the foundation to an approach that finds the efficiency frontier of the bicriteria scheduling problems $Q|p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$ and $\alpha m|r(j), p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$ with $\alpha \in \{P, Q\}$ in a closed form [32].

Given an instance of the problem, let $S^*(d)$ denote a schedule with a makespan $C_{\max} = d$ that minimizes the total compression cost. The solution to the 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 (13), (14) and (15), the rank function $\varphi(X)$ as well as the function $\tilde{\varphi}(X)$ of the form (3) are functions of d; therefore in this paper we may write $\varphi(X, d)$ and $\tilde{\varphi}(X, d)$ 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

$$\psi_t(d) = \tilde{\varphi}(N_t(\sigma), d), \quad 1 \le t \le n, \tag{17}$$

computed for this value of d. By (5),

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

For all scheduling problems under consideration, due to (13), (14) and (15), there are m expressions for $\varphi(X)$, depending on whether $|X| \le m-1$ or $|X| \ge m$, and $\psi_t(d)$ can be represented as a piecewise-linear function of the form of an envelope with $m + 1$ pieces.

Setting for completeness $w\left(\sigma(n+1)\right) = 0$, we deduce from Theorem 2 that

$$W(d) = \sum_{t=1}^{n} w\left(\sigma(t)\right)\left(\psi_t(d) - \psi_{t-1}(d)\right) = \sum_{t=1}^{n} \left(w\left(\sigma(t)\right) - w\left(\sigma(t+1)\right)\right)\psi_t(d). \tag{18}$$

Thus, in order to be able to compute the (piecewise-linear) function $W(d)$, we first have to compute the functions $\psi_t(d)$, $1 \le t \le n$, for all relevant values of d. It

is shown in [32], that after the functions $\psi_t(d)$, $1 \le t \le n$, for all relevant values of d are found, their weighted sum by (18) can be computed in $O(nm \log n)$ time. This (piecewise-linear) function $W(d)$ fully defines the efficiency frontier for the corresponding bicriteria scheduling problem.

Adapting this general framework to problem $Q|p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, W)$, it can be proved that all required functions $\psi_t(d)$, $1 \le t \le n$, can be found in $O(n \log n + nm)$ time, and the overall problem is solvable in $O(nm \log m)$ time, while problems $\alpha m|r(j), p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, W)$ can be solved in $O\left(n^2 \log m\right)$ time and in $O(n^2 m)$ time for $\alpha = P$ and $\alpha = Q$, respectively.

4.3 Solving Single Criterion Problems by Decomposition

We now show that problems $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \le d|W$ and $\alpha m|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \le d|W$ with $\alpha \in \{P, Q\}$ can be solved faster than is guaranteed by the algorithms for the respective bicriteria problems considered in Section 4.2. This is achieved by adapting the decomposition algorithm based on Procedure DECOMP presented in Section 3. The crucial issue here is the computation of the instrumental set Y_* in each iteration.

For an initial Problem LP$(N, \emptyset, \emptyset, \mathbf{l}, \mathbf{u})$, associated with one the of three scheduling problems above, assume that the following preprocessing is done in $O(n \log n)$ time before calling Procedure DECOMP$(N, \emptyset, \emptyset, \mathbf{l}, \mathbf{u})$: the jobs are numbered in non-decreasing order of their release dates in accordance with (10); the machines are numbered in non-increasing order of their speeds in accordance with (1), and the partial sums S_v are computed for all v, $0 \le v \le m$, by (11); the lists $(l(j) \mid j \in N)$ and $(u(j) \mid j \in N)$ are formed and their elements are sorted in non-decreasing order.

In a typical iteration of Procedure DECOMP applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (6) related to the rank function $\varphi_K^H(Y) = \varphi(Y \cup K) - \varphi(K)$, it is shown in [33] that for a given set $X \subseteq H$ the function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ can be computed as

$$\widetilde{\varphi}_K^H(X) = u(X) - \varphi(K) + \min_{Y \in 2^H} \{\varphi(Y \cup K) - b(Y)\}, \tag{19}$$

where φ is the initial rank function associated with the scheduling problem under consideration, and

$$b(j) = \begin{cases} u(j), \text{ if } j \in X, \\ l(j), \text{ if } j \in H \setminus X. \end{cases} \tag{20}$$

Notice that if the minimum in the right-hand side of (19) is achieved for $Y = Y_*$, then Y_* is an instrumental set for set X.

For Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ associated with problem $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \le d|W$ due to (13) and (19) we deduce that

$$\widetilde{\varphi}_K^H(X) = u(X) - dS_{\min\{m,k\}} + \min\{\Phi', \Phi''\}. \tag{21}$$

Here, $\Phi' = +\infty$ if $h > m - k - 1$; otherwise

$$\Phi' = \min_{0 \le v \le m-k-1} \{dS_{v+k} - \sum_{i=1}^{v} b_i\}, \tag{22}$$

where b_i is the i-th largest value in the list $(b(j) \mid j \in H)$, while $\Phi'' = +\infty$ if $h \le m - k - 1$; otherwise $\Phi'' = dS_m - b(H)$. In any case, in terms of the notions introduced in Section 3 we deduce that $T_{Y_*}(h) = T_{\mathrm{Split}}(h) = O(h)$, so that the overall running time needed to solve problem $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \le d|W$ by the decomposition algorithm based on recursive applications of Procedure DECOMP is $O(n \log n)$. An alternative implementation of the same approach, also presented in [33], does not involve a full preprocessing and requires $O(n + m \log m \log n)$ time.

Problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ and Problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ can be solved by the decomposition algorithm in $O(n \log m \log n)$ time and in $O(nm \log n)$ time, respectively.

5 SCPT Problems with Distinct Deadlines

We start with a brief review of the feasibility problems $\alpha|r(j), pmtn, C(j) \le d(j)|\circ$, where $\alpha \in \{1, P, Q\}$, in which for each job $j \in N$ the processing time $p(j)$ is fixed and the task is to verify the existence of a deadline-feasible preemptive schedule.

Divide the interval $[\min_{j \in N} r(j), \max_{j \in N} d(j)]$ into subintervals by using the release dates $r(j)$ and the deadlines $d(j)$ for $j \in N$. Let $T = (\tau_0, \tau_1, \ldots, \tau_\gamma)$, where $1 \le \gamma \le 2n - 1$, be the increasing sequence of distinct numbers in the list $(r(j), d(j) \mid j \in N)$. Introduce the intervals $I_k = [\tau_{k-1}, \tau_k]$, $1 \le k \le \gamma$. Denote the length of interval I_k by $\Delta_k = \tau_k - \tau_{k-1}$.

For a set of jobs $X \subseteq N$, let $\varphi(X)$ be a set function that represents the total production capacity available for the feasible processing of the jobs of set X.

For a particular problem, the function $\varphi(X)$ can be suitably defined. Interval I_k is *available* for processing job j if $r(j) \le \tau_{k-1}$ and $d(j) \ge \tau_k$. For a job j, denote the set of the available intervals by $\Gamma(j)$. For a set of jobs $X \subseteq N$, introduce the set function

$$\varphi_1(X) = \sum_{I_k \in \cup_{j \in X} \Gamma(j)} \Delta_k. \tag{23}$$

Then for problem $1|r(j), pmtn, C(j) \le d(j)|\circ$ a feasible schedule exists if and only if inequality (12) holds for all sets $X \subseteq N$ for $\varphi(X) = \varphi_1(X)$. Such a statement (in different terms) was first formulated in [8] and [10]. For the problems on parallel machines, the corresponding representation of the total processing capacity in the form of a set function is defined in [29]. For all versions of the problem, with a single or parallel machines, the set function φ is submodular.

The single machine feasibility problem $1|r(j), pmtn, C(j) \le d(j)|\circ$ in principle cannot be solved faster than finding the ordered sequence $T = (\tau_0, \tau_1, \ldots, \tau_\gamma)$

of the release dates and deadlines. The best possible running time $O(n \log n)$ for solving problem $1|r(j), pmtn, C(j) \leq d(j)|\circ$ is achieved by the EDF (Earliest Deadline First) algorithm designed in [10].

For parallel machine problems, it is efficient to reformulate the problem of checking the inequalities (12) in terms of finding the maximum flow in a special bipartite network; see, e.g., [4]. Using an algorithm from [1], such a network problem can be solved in $O\left(n^3\right)$ time and in $O\left(mn^3\right)$ time, for problem $P|r(j), pmtn, C(j) \leq d(j)|\circ$ and problem $Q|r(j), pmtn, C(j) \leq d(j)|\circ$, respectively.

Most studies on the SCPT problems $\alpha|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$, where $\alpha \in \{1, P, Q\}$ have been conducted within the body of research on imprecise computation scheduling [16]; however, the best known algorithms have been produced by alternative methods.

For the SCPT problems on parallel machines, the most efficient algorithms are based on reductions to the parametric max-flow problems in bipartite networks. McCormick [18] develops an extension of the parametric flow algorithm in [6] and this approach gives the running times of $O\left(n^3\right)$ for problem $P|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$ and of $O\left(mn^3\right)$ for problem $Q|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$, matching the best known times for the corresponding problems with fixed processing times.

Notice that in most papers on imprecise computation scheduling it is claimed that $P|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$ can be solved faster, in $O\left(n^2 \log^2 n\right)$ time, by reducing it to the min-cost flow problem in a special network; see [16]. However, as demonstrated in [34], such a representation, although possible for a single machine problem, cannot be extended to the parallel machines models, so that the best known running time for solving problem $P|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$, as well as its counterpart with fixed processing times, is $O\left(n^3\right)$.

Problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$, for many years has been an object of intensive study, mainly within the body of research on imprecise computation. The history of studies on this problem is a race for developing an $O(n \log n)$-time algorithm, matching the best possible estimate achieved for a simpler feasibility problem $1|r(j), pmtn, C(j) \leq d(j)|\circ$.

Hochbaum and Shamir [9] present two algorithms, one solves problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$ in $O(n^2)$ time and the other solves its counterpart $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|\sum x(j)$ with the unweighted objective function in $O(n \log n)$ time. An algorithm for problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$ developed in [17] requires $O(n \log n + \kappa n)$ time, where κ is the number of distinct weights $w(j)$, while an algorithm in [31] takes $O\left(n \log^2 n\right)$ time, provided that the numbers $\overline{p}(j), \underline{p}(j)$, $r(j), d(j)$ are integers.

5.1 Solving Single Machine Problem by Decomposition

The time complexity of problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$ is finally settled in [35], where an $O(n \log n)$-time algorithm is given. The al-

gorithm is based on a decomposition algorithm for Problem (LP) and uses an algorithm from [9] as a subroutine.

The efficient implementation of the decomposition algorithm developed in [35] is based on the following statement.

Theorem 4 (cf. [5, Corollary 3.4]). *For a submodular system* $(2^H, \varphi)$ *and a vector* $\mathbf{b} \in \mathbb{R}^H$*, the equality*

$$\min_{Y \in 2^H} \{\varphi(Y) + b(H \setminus Y)\} = \max\{p(H) \mid \mathbf{p} \in P(\varphi),\ \mathbf{p} \leq \mathbf{b}\}$$

holds. In particular, if $\mathbf{b} \geq \mathbf{0}$ *and* $\varphi(X) \geq 0$ *for all* $X \subseteq N$ *then the right-hand side is equal to* $\max\{p(H) \mid \mathbf{p} \in P(\varphi),\ \mathbf{0} \leq \mathbf{p} \leq \mathbf{b}\}$.

Given Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (6), for a set $X \subseteq H$ define the vector $\mathbf{b} \in \mathbb{R}^H$ by (20), and for a set $X \subseteq H$ represent $\tilde{\varphi}_K^H(X)$ in the form

$$\begin{aligned}\tilde{\varphi}_K^H(X) = \min_{Y \in 2^H} \{\varphi_K^H(Y) + u(X \setminus Y) - l(Y \setminus X)\} = -l(H \setminus X) \\ + \min_{Y \in 2^H} \{\varphi_K^H(Y) + b(H \setminus Y)\}.\end{aligned}$$

Since $-l(H \setminus X)$ is a constant, in order to find an instrumental set Y_* that defines $\tilde{\varphi}_K^H(X)$ it suffices to find the set-minimizer for $\min_{Y \in 2^H} \{\varphi_K^H(Y) + b(H \setminus Y)\}$. By Theorem 4, the latter minimization problem is equivalent to the following auxiliary problem:

$$\begin{aligned}(\mathrm{AuxLP}) : \max \sum_{j \in H} q(j) \\ \text{s.t.}\quad q(Y) \leq \varphi_K^H(Y),\ Y \in 2^H; \\ 0 \leq q(j) \leq b(j),\ j \in H.\end{aligned} \tag{24}$$

Let $\mathbf{q}_* \in \mathbb{R}^H$ be an optimal solution to Problem (AuxLP) with the values $b(j)$ defined with respect to a set $X \subseteq H$. It is proved in [35] that a set Y_* is the required instrumental set for Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (6) if and only if

$$q_*(Y_*) = \varphi_K^H(Y_*); \qquad q(j) = b(j),\ j \in H \setminus Y_*.$$

Problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|\sum w(j)x(j)$ reduces to Problem (LP) with the rank function $\varphi = \varphi_1$ defined by (23). Consider a typical iteration of Procedure Decomp applied to Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (6) related to the rank function $\varphi_K^H(Y) = \varphi(Y \cup K) - \varphi(K)$. For a set $X \subseteq H$ of jobs, a meaningful interpretation of $\varphi_K^H(X)$ is the total length of the time intervals originally available for processing the jobs of set $X \cup K$ after the intervals for processing the jobs of set K have been completely used up.

Select a heavy-element set $\hat{H}$ and define the values $b(j)$ by (20) applied to $X = \hat{H}$. Our goal is to find an instrumental set Y_* for set $\hat{H}$. As described above, for this purpose we may solve the auxiliary Problem (ULP)

$$\begin{aligned}(\mathrm{ULP}) : \max \sum_{j \in H} q(j) \\ \text{s.t.}\quad q(X) \leq \psi(X),\ X \in 2^H, \\ 0 \leq q(j) \leq b(j),\ j \in H.\end{aligned} \tag{25}$$

Problem (ULP) can be seen as a version of a scheduling problem $1|r(j), q(j) = b(j) - x(j), pmtn, C(j) \leq d(j)| \sum x(j)$, in which it is required to determine the actual processing times $q(j)$ of jobs of set H to maximize the total (unweighted) actual processing time, provided that $0 \leq q(j) \leq b(j)$ for each $j \in H$. It can be solved by an algorithm developed by Hochbaum and Shamir [9], which uses the UNION-FIND technique and guarantees that the actual processing times of all jobs and the corresponding optimal schedule are found in $O(h)$ time, provided that the jobs are renumbered in non-increasing order of their release dates. The algorithm is based on the latest-release-date-first rule. Informally, the jobs are taken one by one in the order of their numbering and each job $j \in H$ is placed into the current partial schedule to fill the available time intervals consecutively, from right to left, starting from the right-most available interval. The assignment of a job j is complete either if its actual processing time $q(j)$ reaches its upper bound $b(j)$ or if no available interval is left. Only a slight modification of the Hochbaum-Shamir algorithm is required to find not only the optimal values $q_*(j)$ of the processing times, but also an associated instrumental set. The running time of modified algorithm is still $O(h)$.

In terms of the notions introduced in Section 3 we deduce that $T_{Y_*}(h) = T_{\text{Split}}(h) = O(h)$, so that the overall running time needed to solve problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d(j)|W$ by the decomposition algorithm based on recursive applications of Procedure DECOMP is $O(n \log n)$.

6 Conclusions

In this paper, we demonstrate how the SCPT problems on parallel machines can be solved efficiently by applying methods of submodular optimization. For single criterion SCPT problems to minimize the total compression costs a developed decomposition recursive algorithm for maximizing a linear function over a submodular polyhedron intersected with a box is especially useful, since it leads to fast algorithms, some of which are the best possible. Another area of applications of submodular reformulations of the SCPT problems includes bicriteria problems, for which either faster than previously known algorithms are obtained or first polynomial algorithms are designed.

We intend to extend this approach to other scheduling models with controllable processing parameters, in particular to speed scaling problems. It will be interesting to identify problems, including those outside the area of scheduling, for which an adaptation of our approach is beneficial.

Acknowledgement

This research was supported by the EPSRC funded project EP/J019755/1 "Submodular Optimisation Techniques for Scheduling with Controllable Parameters". The first 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.

References

1. Ahuja, R.K., Orlin, J.B., Stein, C., Tarjan, R.E.: Improved algorithms for bipartite network flow. SIAM J Comput. 23, 906–933 (1994)
2. Brucker, P.: Scheduling Algorithms. 5th edition, Springer, Berlin (2007)
3. Chen, Y.L.: Scheduling jobs to minimize total cost. Eur. J. Oper. Res. 74, 111–119 (1994)
4. Federgruen, A., Groenevelt, H.: Preemptive scheduling of uniform machines by ordinary network flow techniques. Manag. Sci. 32, 341–349 (1986)
5. Fujishige, S.: Submodular Functions and Optimization. 2nd Edition, Ann. Discr. Math. 58, Elsevier (2005)
6. Gallo, G., Grigoriadis, M.D., Tarjan, R.E. A fast parametric maximum flow algorithm and applications. SIAM J Comput. 18, 30–55 (1989)
7. Gonzales, T.F., Sahni, S., Preemptive scheduling of uniform processor systems. J. ACM 25, 92–101 (1978)
8. Gordon, V.S., Tanaev, V.S.: Deadlines in single-stage deterministic scheduling. Optimization of Systems for Collecting, Transfer and Processing of Analogous and Dscirete Data in Local Information Computing Systems. Materials of the 1st Joint Soviet-Bulgarian seminar (Institute of Engineering Cybernetics of Academy of Sciences of BSSR – Institute of Engineering Cybernetics of Bulgarian Academy of Sciences, Minsk), 53–58 (in Russian) (1973)
9. Hochbaum, D.S., Shamir, R.: Minimizing the number of tardy job unit under release time constraints. Discr. Appl. Math. 28, 45–57 (1990)
10. Horn. W.: Some simple scheduling algorithms. Naval Res. Logist. Quart. 21, 177–185 (1974)
11. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. J. ACM 48, 761–777 (2001)
12. Janiak, A., Kovalyov, M.Y., Single machine scheduling with deadlines and resource dependent processing times. Eur. J. Oper. Res. 94, 284–291 (1996)
13. Jansen, K., Mastrolilli, M.: Approximation schemes for parallel machine scheduling problems with controllable processing times. Comput. Oper. Res. 31, 1565–1581 (2004)
14. Katoh, N., Ibaraki, T.: Resource allocation problems. In Du, D.-Z., Pardalos, P.M. (eds) Handbook of Combinatorial Optimization, Vol. 2, pp. 159–260. Kluwer, Dordrecht (1998)
15. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: algorithms and complexity. In Graves, S.C., Rinnooy Kan, A.H.G. Zipkin, P.H. (eds.) Handbooks in Operations Research and Management Science, Vol. 4, Logistics of Production and Inventory, pp. 445–522. Elsevier, North–Holland, Amsterdam (1993)
16. Leung, J.Y.-T.: Minimizing total weighted error for imprecise computation tasks. In Leung, J.Y.-T. (ed) Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 34-1 – 34-16. Chapman & Hall/CRC (2004)
17. Leung, J.Y.-T., Yu, V.K.M., Wei, W.-D.: Minimizing the weighted number of tardy task units. Discr. Appl. Math. 51: 307–316 (1994)
18. McCormick, S. T.: Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. Oper. Res. 47, 744–756 (1999)
19. McNaughton, R.: Scheduling with deadlines and loss functions. Manage. Sci. 12, 1–12 (1959)

20. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley (1988)
21. Nowicki, E., Zdrzałka, S.: A survey of results for sequencing problems with controllable processing times. Discr. Appl. Math. 26, 271–287 (1990)
22. Nowicki, E., Zdrzałka, S.: A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times. Discr. Appl. Math. 63, 237–256 (1995)
23. Sahni, S.: Preemptive scheduling with due dates. Oper. Res. 27, 925–934 (1979)
24. Sahni, S., Cho, Y. Scheduling independent tasks with due times on a uniform processor system. J. ACM 27, 550–563 (1980)
25. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theory B 80, 346–355 (2000)
26. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003)
27. Shabtay, D., Steiner, G.: A survey of scheduling with controllable processing times. Discr. Appl. Math. 155, 1643–1666 (2007)
28. Shakhlevich, N. V., Strusevich, V. A.: Pre-emptive scheduling problems with controllable processing times. J. Sched. 8, 233–253 (2005)
29. Shakhlevich, N.V. Strusevich, V.A.: Preemptive scheduling on uniform parallel machines with controllable job processing times. Algorithmica 51, 451–473 (2008)
30. Shakhlevich, N.V., Shioura, A., Strusevich, V.A.: Single machine scheduling with controllable processing times by submodular optimization. Int. J. Found. Comput. Sci. 20, 247–269 (2009)
31. Shih, W.-K., Lee, C.-R., Tang, C.H.: A fast algorithm for scheduling imprecise computations with timing constraints to minimize weighted error. Proc. 21th IEEE Real-Time Systems Symposium (RTSS2000), 305–310 (2000)
32. Shioura, A., Shakhlevich, N.V., Strusevich, V.A.: A submodular optimization approach to bicriteria scheduling problems with controllable processing times on parallel machines. SIAM J. Discr. Math. 27, 186–204 (2013)
33. Shioura, A., Shakhlevich, N.V., Strusevich, V.A.: Decomposition algorithms for submodular optimization with applications to parallel machine scheduling with controllable processing times. Math. Progr. A 153: 495–534 (2015)
34. Shioura, A., Shakhlevich, N.V., Strusevich, V.A.: Scheduling imprecise computation tasks on parallel machines to minimize linear and non-linear error penalties: Reviews, links and improvements. University of Greenwich, London, Report SORG-04-2015 (2015)
35. Shioura, A., Shakhlevich, N.V., Strusevich, V.A.: Application of submodular optimization to single machine scheduling with controllable processing times subject to release dates and deadlines. INFORMS J. Comp. 28, 148–161 (2016)