INFORMS Journal on Computing

Application of Submodular Optimization to Single Machine Scheduling with Controllable Processing Times Subject to Release Dates and Deadlines

Akiyoshi Shioura, Natalia V. Shakhlevich, Vitaly A. Strusevich

INFORMS Journal on Computing
Vol. 28, No. 1, Winter 2016, pp. 148–161
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2015.0660
© 2016 INFORMS

Application of Submodular Optimization to Single Machine Scheduling with Controllable Processing Times Subject to Release Dates and Deadlines

Akiyoshi Shioura
Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan, shioura@dais.is.tohoku.ac.jp

Natalia V. Shakhlevich
School of Computing, University of Leeds, Leeds LS2 9JT, United Kingdom, n.shakhlevich@leeds.ac.uk

Vitaly A. Strusevich
Department of Mathematical Sciences, University of Greenwich, London SE10 9LS, United Kingdom, v.strusevich@greenwich.ac.uk

In this paper, we study a scheduling problem on a single machine, provided that the jobs have individual release dates and deadlines, and the processing times are controllable. The objective is to find a feasible schedule that minimizes the total cost of reducing the processing times. We reformulate the problem in terms of maximizing a linear function over a submodular polyhedron intersected with a box. For the latter problem of submodular optimization, we develop a recursive decomposition algorithm and apply it to solving the single machine scheduling problem to achieve the best possible running time.

Keywords: programming: linear; production scheduling: deterministic, single machine; analysis of algorithms: computational complexity
History: Accepted by Karen Aardal, Area Editor for Design and Analysis on Algorithms; received February 2014; revised February 2015; accepted May 2015. Published online February 3, 2016.

1. Introduction

We consider a scheduling problem on a single machine, provided that the processing times of the jobs are controllable and each job has a release date and a deadline. The objective is to determine the actual durations of jobs from given intervals and to find a feasible preemptive schedule such that the total cost of compressing the processing times is minimized. This area of scheduling has been active since the 1980s, see surveys by Nowicki and Zdrzałka (1990) and Shabtay and Steiner (2007). The corresponding models are applicable to production, make-or-buy decision making, supply chain management, and imprecise computation.

In this paper, we study the general version of the single machine model in which the jobs are available at arbitrary, nonequal release times and should be completed by their individual deadlines, which can also be different for different jobs. The main outcome of our study implies that this problem with controllable processing times is no harder in terms of its computational complexity than its counterpart with fixed processing times. In the latter problem it is required to verify whether there exists a feasible schedule meeting the deadline and release time constraints.

Our approach is based on submodular optimization techniques. It continues the line of research initiated by Shakhlevich and Strusevich (2005, 2008) and explores a close link between scheduling with controllable processing times and linear programming problems with submodular constraints. Our papers Shakhlevich et al. (2009) and Shioura et al. (2013) can be viewed as convincing examples of a positive mutual influence of scheduling and submodular optimization. This paper, which builds upon Shakhlevich et al. (2008), makes another contribution toward the development of solution procedures for problems of submodular optimization and their applications to scheduling models. The efficiency of the proposed approach is a result of the following two factors: (i) the decomposition approach that breaks down the problem into smaller subproblems, and (ii) a new special technique for finding so-called instrumental sets in each stage of the decomposition approach, which serves as the basis for defining the subproblems. As a result, we arrive at an algorithm that solves the problem with n jobs in $O(n \log n)$ time, which is best possible.

The paper is organized as follows. Section 2 gives a formal description of the scheduling model under consideration. For completeness, we include a review of the relevant results for more general models, with identical and uniform machines, with fixed and controllable processing times. We stress that for the parallel machines, the problems with controllable processing times are no harder in terms of their computational complexity than their counterparts with fixed processing times, whereas there is a time complexity gap between the running times of the best known algorithms for the single-machine version of the problem. To close this gap, we reformulate the problem with controllable processing times in terms of submodular optimization and develop a novel decomposition algorithm for solving problems of this type. Section 3 provides the necessary information on submodular optimization and establishes its link with the single machine problem under consideration. Section 4 studies a linear programming problem over a submodular polyhedron intersected with a box and develops a recursive decomposition algorithm for its solution. The application of the developed decomposition algorithm to the single machine problem to minimize total compression cost is discussed in §5. The concluding remarks are contained in §6.

2. Review of Scheduling with Fixed and Controllable Processing Times

In this section, we present a formal description of the scheduling problems under consideration. We provide their meaningful interpretations and give a brief review of the results for the problems of finding a feasible preemptive schedule, provided that the processing times are fixed, as well as for problems with controllable processing times to minimize total compression cost.

Formally, in the main scheduling model under consideration, the jobs of set $N = \{1, 2, \ldots, n\}$ have to be processed on a single machine M_1. For completeness, in this section we also review the models in which the jobs of set N are processed 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 by the decision maker from a given interval $[\underline{p}(j), \bar{p}(j)]$. Such a decision results in *compression* of the longest processing time $\bar{p}(j)$ down to $p(j)$, and the value $x(j) = \bar{p}(j) - p(j)$ 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 costs associated with a choice of the actual processing times is represented by the linear function $\sum_{j \in N} w(j)x(j)$.

Notice that the described model is historically the first scheduling model with controllable processing times that traces back to the 1980s; see Nowicki and Zdrzałka (1990) for a review of earlier results on these models, other than the model studied in this paper. A popular alternative model that emerged after 2000 assumes that the actual processing time of a job is expressed as a ratio of the "normal" processing time to a (possibly, nonlinear convex) function that depends on an amount of a non-renewable resource allocated to the job. The survey by Shabtay and Steiner (2007) reviews the body of research on both mentioned models with controllable times. The latter model is also closely related to models of power-aware scheduling, in which the speed of a processor is affected by the amount of provided energy; see, e.g., Bansal et al. (2009) and Bunde (2009).

In the problem studied in this paper, 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 at any time and resumed later. It is not allowed to process more than one job at a time.

In what follows, we only discuss scheduling problems with distinct release dates and deadlines. A review on problems with a common deadline or equal release dates can be found in Shioura et al. (2013).

Provided that the processing time of a job $j \in N$ is equal to $p(j)$, a *feasible* schedule guarantees that no job is processed outside the time interval $[r(j), d(j)]$. 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.

If there are m parallel machines, we distinguish between the *identical* machines and the *uniform* machines. In the former case, the machines have the same speed, so that for a job j with an actual processing time $p(j)$ the total length of the time intervals in which this job is processed in a feasible schedule is equal to $p(j)$. If the machines are uniform, then it is assumed that machine M_i has speed s_i, $1 \leq i \leq m$, which defines the speed-up factor for jobs (or parts of jobs) allocated to machine M_i.

For a given machine environment, the problem of our primal concern is to determine the values of actual processing times and to find the corresponding feasible preemptive schedule so that all jobs meet their deadlines and total weighted compression cost is minimized. Adapting standard notation for scheduling problems by Lawler et al. (1993), we denote problems of this type by $\alpha \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \leq d(j) \mid \sum w(j)x(j)$. Here, in the first field α we write 1 in the case of 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. We write $p(j) = \bar{p}(j) - x(j)$ to indicate that the processing times are controllable and x_j is the compression amount to be found. The abbreviation *pmtn* is used to point out that preemption is allowed. The condition $C(j) \le d(j)$ reflects the fact that in a feasible schedule the deadlines should be respected. Finally, in the third field we write the objective function to be minimized, which is the total compression cost.

Next, we use problem $1 \mid r(j),\ p(j) = \bar{p}(j) - x(j),\ pmtn,\ C(j) \le d(j) \mid \sum w(j)x(j)$ as an example to illustrate different interpretations and applications of scheduling with controllable processing times.

A range of scheduling models relevant to our study belongs to the area of imprecise computation; see Leung (2004) for a recent review. In computing systems that support imprecise computation, some computations (image processing programs, implementations of heuristic algorithms, etc.) can be run partially, producing less precise results. In our notation, a task with a processing requirement $\bar{p}(j)$ can be split into a mandatory part that takes $\underline{p}(j)$ time, and an optional part that may take up to $\bar{p}(j) - \underline{p}(j)$ additional time units. To produce a result of reasonable quality, the mandatory part must be completed in full, whereas an optional part improves the accuracy of the output. If instead of an ideal computation time $\bar{p}(j)$ a task is executed for $p(j) = \bar{p}(j) - x(j)$ time, then computation is imprecise and $x(j)$ corresponds to the error of computation. Typically, the problems of imprecise computation are those of finding a deadline feasible preemptive schedule either on a single machine or on parallel machines. A popular objective function is $\sum w(j)x(j)$, which is interpreted here as the total weighted error. It is surprising that until very recently, the common underlying model for problems with controllable processing times and those of imprecise computation has not been noticed. Even the most recent survey by Shabtay and Steiner (2007) makes no mention of the imprecise computation research.

Scheduling problems with controllable processing times also serve as mathematical models in make-or-buy decision making; see, e.g., Shakhlevich et al. (2009). In manufacturing, it is often the case that either the existing production capabilities are insufficient to fulfill all orders internally in time or the cost of work in process of an order exceeds a desirable amount. Such an order can be partly subcontracted. Subcontracting incurs additional cost but that can be either compensated by quoting realistic deadlines for all jobs or balanced by a reduction in internal production expenses. The make-or-buy decisions should be taken to determine which part of each order is manufactured internally and which is subcontracted. For instance, problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ admits the following interpretation. The internal production facility is the machine, and the orders are the jobs that arrive at their release dates $r(j)$. For each order $j \in N$, the value of $\bar{p}(j)$ is interpreted as the processing requirement, provided that the order is manufactured internally in full, whereas $\underline{p}(j)$ is a given mandatory limit on the internal production. Further, $p(j) = \bar{p}(j) - x(j)$ is 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, the problem is to minimize the total subcontracting cost and to find a deadline-feasible schedule for internally manufactured orders.

Each problem with controllable processing times can be seen as an extension of the corresponding problem, in which the processing times of all jobs are fixed, i.e., equal to given values $p(j), 1 \le j \le n$. We generically denote problems with fixed processing times by $\alpha \mid r(j), pmtn, C(j) \le d(j) \mid \circ$, where $\alpha \in \{1, P, Q\}$. These are essentially feasibility problems. To solve such a problem means either to find a feasible schedule if it exists or to report that a schedule does not exist.

Now we review the results on these feasibility problems. For a problem $\alpha \mid r(j), pmtn, C(j) \le d(j) \mid \circ$, 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 $I_k = [\tau_{k-1}, \tau_k]$, $1 \le k \le \gamma$, and define the set of all intervals $I = \{I_k \mid 1 \le k \le \gamma\}$. 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. Then, a feasible schedule exists if and only if the inequality

$$\sum_{j \in X} p_j \le \varphi(X) \qquad (1)$$

holds for all sets $X \subseteq N$.

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

$$\Gamma(j) = \{I_k \in I \mid I_k \subseteq [r(j), d(j)]\}. \qquad (2)$$

For a set of jobs $X \subseteq N$, introduce set-functions

$$\varphi_1(X) = \sum_{I_k \in \bigcup_{j \in X} \Gamma(j)} \Delta_k; \qquad (3)$$

$$\varphi_P(X) = m \sum_{I_k \in \bigcup_{j \in X} \Gamma(j)} \Delta_k. \qquad (4)$$

Then for problem $1 \mid r(j), pmtn, C(j) \le d(j) \mid \circ$ (or problem $P \mid r(j), pmtn, C(j) \le d(j) \mid \circ$) a feasible schedule exists if and only if inequality (1) holds for all sets $X \subseteq N$ for $\varphi(X) = \varphi_1(X)$ (respectively, $\varphi(X) = \varphi_P(X)$). Such a statement (in different terms) was first formulated by Gordon and Tanaev (1973) and Horn (1974). For the uniform machines, the corresponding representation of the total processing capacity in the form of a set-function $\varphi_Q(X)$ is defined by Shakhlevich and Strusevich (2008).

The single machine feasibility problem $1 \mid r(j), pmtn, C(j) \le d(j) \mid \circ$ in principle cannot be solved faster than finding the 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 $1 \mid r(j), pmtn, C(j) \le d(j) \mid \circ$ is achieved by an algorithm designed by Horn (1974). This algorithm employs the EDF (earliest deadline first) scheduling policy, i.e., at any time it schedules the job (or part of the job) that has the smallest deadline among all available jobs. In fact, finding sequence T is the most time-consuming part of the algorithm; if sequence T is available, the remaining steps of the algorithm can be implemented in $O(n)$ time.

For parallel machine problems, it is efficient to reformulate the problem of checking the inequalities (1) in terms of finding the maximum flow in a special bipartite network; see, e.g., Federgruen and Groenevelt (1986). Using an algorithm by Ahuja et al. (1994), such a network problem can be solved in $O(n^3)$ time and in $O(mn^3)$ time, for problem $P \mid r(j), pmtn, C(j) \le d(j) \mid \circ$ and problem $Q \mid r(j), pmtn, C(j) \le d(j) \mid \circ$, respectively.

We now discuss known algorithms for solving problems $\alpha \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$, where $\alpha \in \{1, P, Q\}$.

First, notice that for parallel machines the corresponding problems can be reduced to either min-cost flow problems or to parametric max-flow problems in bipartite networks. The most efficient algorithms are due to McCormick (1999), who develops an extension of the parametric flow algorithm by Gallo et al. (1989), initially developed for arc capacities dependent on a single common parameter, to the case of several parameters. The approach of McCormick gives the running times of $O(n^3)$ for problem $P \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ and of $O(mn^3)$ for problem $Q \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$. Notice that the algorithms by Gallo et al. (1989) and McCormick (1999) are developed for the flow problems with zero lower bounds on the arc capacities; in scheduling terms that means zero lower bounds on processing times, $\underline{p}(j) = 0$, $j \in N$. Still, the algorithms can be extended to deal with non-zero lower bounds $\underline{p}(j)$, $j \in N$, by standard network flow techniques.

The single machine problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$, for many years has been an object of intensive study, mainly within the body of research on imprecise computation. Problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ with controllable processing times is no easier than problem $1 \mid r(j), pmtn, C(j) \le d(j) \mid \circ$, a feasibility problem with fixed processing times. Thus, $O(n \log n)$, the best possible time for solving the latter problem is an obvious lower bound on the running time required to solve the former problem. The history of studies on $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ is a race for developing an $O(n \log n)$-time algorithm.

Notice that McCormick's approach, although applicable in principle, does not lead to an algorithm faster than $O(n^3)$ (or $O(n^2 \log^2 n)$ if a network suggested by Chung et al. 1989 and Shih et al. 1989 is used). Therefore, scheduling reasoning is used in all known faster algorithms.

Hochbaum and Shamir (1990) present two algorithms for the problem with zero lower bounds on the processing times. One of their algorithms solves problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ in $O(n^2)$ time and the other solves its counterpart $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum x(j)$ with the unweighted objective function in $O(n \log n)$ time (or in $O(n)$ time if the sorted sequence of release dates and deadlines is known). Hochbaum and Shamir interpret the objective function as the total (weighted) number of late (rejected) units of the jobs. Their algorithm for the unweighted problem is of particular importance for this study, because we use its extended form as a subroutine in §5.

Shih et al. (1991) study the general case of $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ and $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum x(j)$ with nonzero lower bounds on the processing times; the proposed algorithms have the same time complexity as those by Hochbaum and Shamir (1990). Notice that it is fairly easy to incorporate nonzero lower bounds if a network flow technique is used, whereas in an algorithm that relies on scheduling reasoning a move from zero lower bounds to nonzero ones requires additional effort.

Leung et al. (1994) give an algorithm for problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ that requires $O(n \log n + \kappa n)$ time, where κ is the number of distinct weights $w(j)$.

Shih et al. (2000) develop an $O(n \log^2 n)$-time algorithm, provided that the numbers $\bar{p}(j)$, $\underline{p}(j)$, $r(j)$, $d(j)$ are integers. The integrality assumption is essential for the algorithm since one of its steps determines what the authors call a "jam set," and for a purpose of its finding they modify some input values making them noninteger and verify that any combination of the processing times of jobs scheduled in a particular interval is not integer; see Shih et al. (2000, Theorem 2).

Thus, unlike for the models on parallel machines, for a single machine model there is a computational complexity gap: the feasibility problem $1 \mid r(j)$,

$pmtn$, $C(j) \le d(j) \mid \circ$ is solvable in $O(n \log n)$ time, but for problem $1 \mid r(j)$, $p(j) = \bar{p}(j) - x(j)$, $pmtn$, $C(j) \le d(j) \mid \sum w(j)x(j)$ with controllable processing times no $O(n \log n)$-time algorithm is known.

The main purpose of the remainder of this paper is to design an algorithm that solves problem $1 \mid r(j)$, $p(j) = \bar{p}(j) - x(j)$, $pmtn$, $C(j) \le d(j) \mid \sum w(j)x(j)$ in $O(n \log n)$ time. The algorithm handles all instances, with integer and real data. This time is the best possible since it matches the running time of the EDF algorithm.

3. Review of Submodular Optimization Concepts

To make this paper self-contained, in this section we briefly describe the necessary concepts related to submodular optimization and establish its links to the scheduling problems of interest. Unless stated otherwise, we follow the comprehensive monographs by Fujishige (2005) and Schrijver (2003).

Let $N = \{1, 2, \ldots, n\}$ be a ground set, where n is a positive integer, and 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 $\mathbf{p} \le \mathbf{q}$ if $p(j) \le q(j)$ for each $j \in N$, and write $\mathbf{p} < \mathbf{q}$ if $\mathbf{p} \le \mathbf{q}$ and $p(j) < q(j)$ for some $j \in N$. Given a set $U \subseteq \mathbb{R}^N$, a vector $\mathbf{p} \in U$ is called *maximal* in U if there exists no vector $\mathbf{q} \in U$ such that $\mathbf{p} < \mathbf{q}$. 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 φ: $2^N \to \mathbb{R}$ is called *submodular* if the inequality

$$\varphi(X \cup Y) + \varphi(X \cap Y) \le \varphi(X) + \varphi(Y) \tag{5}$$

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

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

$$P(\varphi) = \{\mathbf{p} \in \mathbb{R}^N \mid p(X) \le \varphi(X),\ X \in 2^N\}, \tag{6}$$

$$B(\varphi) = \{\mathbf{p} \in \mathbb{R}^N \mid \mathbf{p} \in P(\varphi),\ p(N) = \varphi(N)\}, \tag{7}$$

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

The main problem that we consider is as follows:

$$\begin{aligned}
\text{(LP):}\quad & \text{Maximize} && \sum_{j \in N} w(j)p(j) \\
& \text{subject to} && p(X) \le \varphi(X), \quad X \in 2^N; \\
& && \underline{p}(j) \le p(j) \le \bar{p}(j), \quad j \in N,
\end{aligned} \tag{8}$$

where φ: $2^N \to \mathbb{R}$ is a submodular function with $\varphi(\varnothing) = 0$, $\mathbf{w} \in \mathbb{R}^N_+$ is a nonnegative weight vector, and $\bar{\mathbf{p}}$, $\underline{\mathbf{p}} \in \mathbb{R}^N$ are upper and lower bound vectors such that $\mathbf{0} \le \underline{\mathbf{p}} \le \bar{\mathbf{p}}$.

Problem (LP) can be classified as a problem of maximizing a linear function over a submodular polyhedron intersected with a box.

Any problem $\alpha \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ can be formulated as Problem (LP), provided that function $\varphi(X)$ is the total processing capacity available for processing jobs of set X; see §2. It is clear that a scheduling problem with controllable processing times to minimize the total compression cost $\sum w(j)x(j)$ is equivalent to that of maximizing the weighted sum $\sum w(j)p(j)$ of actual processing times.

In particular, problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn$, $C(j) \le d(j) \mid \sum w(j)x(j)$ reduces to Problem (LP) with $\varphi(X) = \varphi_1(X)$. Notice that the set function $\varphi_1(X)$ of the form (3) is submodular, which can be proved directly, as is done, e.g., in Shakhlevich and Strusevich (2008).

In our previous work Shakhlevich et al. (2009), we show that Problem (LP) can be reduced to an LP problem defined over a base polyhedron. This fact has been of great importance for solving scheduling problems via submodular methods, see Shakhlevich et al. (2009) and Shioura et al. (2013). It is an essential component of our reasoning in this paper as well.

For the submodular polyhedron $P(\varphi)$ associated with a submodular system $(2^N, \varphi)$ and vectors $\mathbf{l}$, $\mathbf{u} \in \mathbb{R}^N$, we define polyhedrons

$$\begin{aligned}
P(\varphi)^u &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \in P(\varphi),\ \mathbf{x} \ge \mathbf{u}\}; \\
P(\varphi)_l &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \in P(\varphi),\ \mathbf{x} \ge \mathbf{l}\}; \\
P(\varphi)_l^u &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \in P(\varphi),\ \mathbf{l} \le \mathbf{x} \le \mathbf{u}\}.
\end{aligned}$$

It is known (see, e.g., Fujishige 2005, Theorem 3.3) that the set of maximal vectors of $P(\varphi)^u$ forms a base polyhedron $B(\varphi^u)$ associated with $(2^N, \varphi^u)$, where φ^u is a rank function such that $\varphi^u(\varnothing) = 0$ and $\varphi^u(X) = \min_{Y \in 2^N,\, Y \subseteq X}\{\varphi(Y) + u(X \backslash Y)\}$ for a nonempty set $X \in 2^N$. Similarly, the set of maximal vectors of $P(\varphi)_l$ is a base polyhedron $B(\varphi_l)$ associated with $(2^N, \varphi_l)$, where φ_l is a rank function such that $\varphi_l(\varnothing) = 0$ and $\varphi_l(X) = \min_{Z \in 2^N,\, X \subseteq Z}\{\varphi(Z) - l(Z \backslash X)\}$ for a nonempty set $X \in 2^N$; see, e.g., Fujishige (2005), Corollary 3.5.

For the polyhedron $P(\varphi)_l^u$, which is the intersection of a submodular polyhedron with a box, the following statement holds.

Theorem 1 (cf. Shakhlevich et al. 2009). (i) *Polyhedron $P(\varphi)_l^u$ is nonempty if and only if $\mathbf{l} \in P(\varphi)$ and $\mathbf{l} \le \mathbf{u}$.*

(ii) *If $P(\varphi)_l^u$ is nonempty, then the set of maximal vectors in $P(\varphi)_l^u$ is a base polyhedron $B(\tilde{\varphi})$ associated with the* *submodular system* $(2^N, \tilde{\varphi})$, *where the submodular rank function* $\tilde{\varphi}$: $2^N \to \mathbb{R}$ *is given by*

$$\tilde{\varphi}(X) = \min_{Y \in 2^N}\{\varphi(Y) + u(X\setminus Y) - l(Y\setminus X)\}, \quad X \in 2^N. \quad (9)$$

Proof. Expressing the rank function $\tilde{\varphi}$ in terms of function φ^u, we obtain that for any nonempty set $X \in 2^N$ the equality

$$\tilde{\varphi}(X) = \min_{Z\in 2^N,\, X\subseteq Z}\{\varphi^u(Z) - l(Z\setminus X)\}$$

holds. In turn, writing out $\varphi^u(Z)$, we deduce that

$$\tilde{\varphi}(X) = \min_{Z\in 2^N,\, X\subseteq Z}\left\{\min_{Y\in 2^N,\, Y\subseteq Z}\{\varphi(Y) + u(Z\setminus Y)\} - l(Z\setminus X)\right\}.$$

Observe that for $X \subseteq Z$ and $Y \subseteq Z$,

$$l(Z\setminus X) = l(Y\setminus X) + l(Z\setminus(X\cup Y));$$
$$u(Z\setminus Y) = u(X\setminus Y) + u(Z\setminus(X\cup Y)),$$

which allows us to rewrite

$$\tilde{\varphi}(X) = \min_{Y,Z\in 2^N,\, X\cup Y\subseteq Z}\{(\varphi(Y) + u(X\setminus Y) - l(Y\setminus X)) + u(Z\setminus(X\cup Y)) - l(Z\setminus(X\cup Y))\}.$$

Since $u(Z\setminus(X\cup Y)) - l(Z\setminus(X\cup Y)) \geq 0$ and $X\cup Y \subseteq Z$, the above minimum is achieved by $Z = X \cup Y$, so that Z can be removed, which yields (9). □

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

$$\begin{aligned} &\text{Maximize } \sum_{j\in N} w(j)p(j) \\ &\text{subject to } \mathbf{p} \in B(\tilde{\varphi}), \end{aligned} \quad (10)$$

where the rank function $\tilde{\varphi}$: $2^N \to \mathbb{R}$ is given by (9) with $\mathbf{l} = \underline{\mathbf{p}}$ and $\mathbf{u} = \bar{\mathbf{p}}$.

The benefit gained by such a reduction is that we can use a well-known result of submodular optimization, which states that a solution to the problem of maximizing a linear function over a base polyhedron can be obtained essentially in closed form, as presented in the following theorem.

Theorem 2 (cf. Fujishige 2005). *Let* $j_1, j_2, \ldots, j_n$ *be an ordering of elements in* N *that satisfies*

$$w(j_1) \geq w(j_2) \geq \cdots \geq w(j_n). \quad (11)$$

Then, vector $\mathbf{p}^* \in \mathbb{R}^N$ *given by*

$$p^*(j_h) = \tilde{\varphi}(\{j_1, \ldots, j_{h-1}, j_h\}) - \tilde{\varphi}(\{j_1, \ldots, j_{h-1}\}), \quad h = 1, 2, \ldots, n \quad (12)$$

is an optimal solution to the problem (10) (*and also to the problem* (8)).

We use the obtained representation (10) of Problem (LP) in our decomposition algorithm presented in §4.

4. Decomposition of LP problems with Submodular Constraints

In this section, we describe a decomposition algorithm for solving Problem (LP), i.e., an LP problem defined over a submodular polyhedron intersected with a box. In §4.1, we demonstrate that Problem (LP) can be recursively decomposed into subproblems of smaller dimensions, with some components of a solution vector fixed to one of their bounds. In §4.2, we provide an outline of an efficient recursive decomposition procedure. Important implementation details of that procedure are presented in §4.3.

4.1. Fundamental Idea for Decomposition

In this section, we establish an important property, which lays the foundation of our decomposition algorithm for Problem (LP) of the form (8).

Lemma 1 demonstrates that some components of an optimal solution can be fixed either at their upper or lower bounds, whereas for some other components their sum is fixed. Given a subset $\hat{N}$ of N, we say that $\hat{N}$ 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.

Given Problem (LP) and a set $X \subseteq N$, in accordance with (9) define a set $Y_* \subseteq N$ such that the equality

$$\tilde{\varphi}(X) = \varphi(Y_*) + \bar{p}(X\setminus Y_*) - \underline{p}(Y_*\setminus X), \quad (13)$$

holds. We call Y_* an *instrumental* set for set X.

Lemma 1. *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*

$$\text{(a) } p^*(Y_*) = \varphi(Y_*), \quad \text{(b) } p^*(j) = \bar{p}(j), \quad j \in \hat{N}\setminus Y_*,$$
$$\text{(c) } p^*(j) = \underline{p}(j), \quad j \in Y_*\setminus\hat{N}.$$

Proof. Because $\hat{N}$ is a heavy-element subset, there exists an ordering $j_1, j_2, \ldots, j_n$ of elements in N that satisfies (11) and $\hat{N} = \{j_1, j_2, \ldots, j_k\}$, where $k = |\hat{N}|$. Theorems 1 and 2 guarantee that the solution $\mathbf{p}^*$ given by (12) is optimal. In particular, this implies

$$p^*(\hat{N}) = \tilde{\varphi}(j_1) + \sum_{i=2}^{k}(\tilde{\varphi}(\{j_1, j_2, \ldots, j_i\}) - \tilde{\varphi}(\{j_1, j_2, \ldots, j_{i-1}\}))$$
$$= \tilde{\varphi}(\{j_1, j_2, \ldots, j_k\}) = \tilde{\varphi}(\hat{N}).$$

Since $\mathbf{p}^*$ is a feasible solution of Problem (LP), the following conditions

$$p^*(Y_*) \leq \varphi(Y_*), \; p^*(j) \leq \bar{p}(j), \quad j \in \hat{N}\setminus Y_*,$$
$$-p^*(j) \leq -\underline{p}(j), \quad j \in Y_*\setminus\hat{N} \quad (14)$$

hold simultaneously. On the other hand, due to the choice of set Y_*, we have

$$p^*(\hat{N}) = \tilde{\varphi}(\hat{N}) = \varphi(Y_*) + \bar{p}(\hat{N}\setminus Y_*) - \underline{p}(Y_*\setminus\hat{N}),$$

which implies that each inequality of (14) must hold as equality, and that is equivalent to the properties (a), (b), and (c) in the lemma. □

In what follows, we use two fundamental operations on a submodular system $(2^N, \varphi)$, as defined in Fujishige (2005, §3.1). For a set $A \in 2^N$, define a set-function φ^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 called a *restriction of* $(2^N, \varphi)$ *to* A. On the other hand, for a set $A \in 2^N$ define a set-function φ_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 called a *contraction of* $(2^N, \varphi)$ by A.

For an arbitrary set $A \in 2^N$, Problem (LP) can be decomposed into two subproblems of a similar structure by performing restriction of $(2^N, \varphi)$ to A and contraction of $(2^N, \varphi)$ by A, respectively. These problems can be written as follows: for restriction as

$$\begin{aligned} \text{(LP1):}\quad & \text{Maximize} \sum_{j\in A} w(j)p(j) \\ & \text{subject to}\ \ p(X) \le \varphi^A(X) = \varphi(X), \quad X \in 2^A, \\ & \qquad\qquad\quad \underline{p}(j) \le p(j) \le \bar{p}(j), \quad j \in A, \end{aligned}$$

and for contraction as

$$\begin{aligned} \text{(LP2):}\quad & \text{Maximize} \sum_{j\in N\setminus A} w(j)p(j) \\ & \text{subject to}\ \ p(X) \le \varphi_A(X) = \varphi(X\cup A) - \varphi(A), \\ & \qquad\qquad\qquad\qquad\qquad\qquad X \in 2^{N\setminus A}, \\ & \qquad\qquad\quad \underline{p}(j) \le p(j) \le \bar{p}(j), \quad j \in N\setminus A. \end{aligned}$$

We show that an optimal solution of the original Problem (LP) can be easily restored from the optimal solutions of these two subproblems. For every subset $A \subseteq N$ and vectors $\mathbf{p_1} \in \mathbb{R}^A$ and $\mathbf{p_2} \in \mathbb{R}^{N\setminus A}$, the *direct sum* $\mathbf{p_1} \oplus \mathbf{p_2} \in \mathbb{R}^N$ of $\mathbf{p_1}$ and $\mathbf{p_2}$ is defined by

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

Lemma 2. *Let* $A \in 2^N$, *and suppose that* $q(A) = \varphi(A)$ *holds for some optimal solution* $\mathbf{q} \in \mathbb{R}^N$ *of Problem* (LP). *Then,*

(i) *Each of problems* (LP1) *and* (LP2) *has a feasible solution.*

(ii) *If a vector* $\mathbf{p_1} \in \mathbb{R}^A$ *is an optimal solution of Problem* (LP1) *and a vector* $\mathbf{p_2} \in \mathbb{R}^{N\setminus A}$ *is an optimal solution of Problem* (LP2), *then the direct sum* $\mathbf{p}^* = \mathbf{p_1} \oplus \mathbf{p_2} \in \mathbb{R}^N$ *of* $\mathbf{p_1}$ *and* $\mathbf{p_2}$ *is an optimal solution of Problem* (LP).

The proof of this lemma is similar to that for Lemma 3.1 in Fujishige (2005), and is therefore omitted.

From Lemmas 1 and 2, we obtain the following property, which is used recursively in our decomposition algorithm.

Theorem 3. *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, where* (LPR) *and* (LPC) *are given as*

$$\begin{aligned} \text{(LPR):}\quad & \text{Maximize} \sum_{j\in Y_*} w(j)p(j) \\ & \text{subject to}\ \ p(X) \le \varphi(X), \quad X \in 2^{Y_*}, \\ & \qquad\qquad\quad \underline{p}(j) \le p(j) \le \bar{p}(j), \quad j \in Y_*, \\ & \qquad\qquad\quad p(j) = \underline{p}(j), \quad j \in Y_*\setminus\hat{N}. \end{aligned}$$

$$\begin{aligned} \text{(LPC):}\quad & \text{Maximize} \sum_{j\in N\setminus Y_*} w(j)p(j) \\ & \text{subject to}\ \ p(X) \le \varphi(X \cup Y_*) - \varphi(Y_*), \\ & \qquad\qquad\qquad\qquad\qquad\qquad X \in 2^{N\setminus Y_*}, \\ & \qquad\qquad\quad \underline{p}(j) \le p(j) \le \bar{p}(j), \quad j \in N\setminus Y_*, \\ & \qquad\qquad\quad p(j) = \bar{p}(j), \quad j \in \hat{N}\setminus Y_*. \end{aligned}$$

Then, the vector $\mathbf{p}^* \in \mathbb{R}^N$ *given by the direct sum* $\mathbf{p}^* = \mathbf{p_1} \oplus \mathbf{p_2}$ *is an optimal solution of* (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.

4.2. Recursive Decomposition Procedure

In this subsection, we describe 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. In each stage of this process, the current LP problem is decomposed into two subproblems, each with a reduced set of variables, whereas some of the original variables receive fixed values and stay fixed until the end.

Remark 1. The definition of a heavy-element set can be revised to take into account the fact that some variables may become fixed during the solution process. The fixed variables make a fixed contribution into the objective function so that the values of their weights become irrelevant for further consideration and can therefore be made, e.g., zero. This means that a heavy-element set can be selected not among all variables $p(j)$, $j \in N$, but only among the nonfixed variables. Formally, if the ground set N is known to be partitioned as $N = L \cup F$, where the variables $p(j) \in L$ are nonfixed and the variable $p(j) \in F$ are fixed, then $\hat{L} \subseteq L$ is a *heavy-element subset with respect to the weight vector* $\mathbf{w}$ if it satisfies the condition

$$\min_{j \in \hat{L}} w(j) \geq \max_{j \in L \setminus \hat{L}} w(j).$$

Notice that for this refined definition of a heavy-element subset, Lemma 1 and Theorem 3 can be appropriately adjusted.

In each stage of the recursive procedure, we need to solve a subproblem that can be written in the following generic form:

$$\begin{aligned} &\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})\\ &\quad \text{Maximize} \quad \sum_{j \in H} w(j)p(j)\\ &\quad \text{subject to} \quad p(Y) \leq \varphi_K^H(Y) = \varphi(Y \cup K) - \varphi(K), \quad Y \in 2^H,\\ &\qquad\qquad\qquad l(j) \leq p(j) \leq u(j), \quad j \in H \setminus F,\\ &\qquad\qquad\qquad p(j) = u(j) = l(j), \quad j \in F, \end{aligned} \tag{15}$$

where

- $H \subseteq N$ is the index set of components of vector $\mathbf{p}$;
- $F \subseteq H$ 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 $\varphi_K^H \colon 2^H \to \mathbb{R}$ such that

$$\varphi_K^H(Y) = \varphi(Y \cup K) - \varphi(K), \quad Y \in 2^H;$$

- $\mathbf{l} = (l(j) \mid j \in H)$ and $\mathbf{u} = (u(j) \mid j \in H)$ are, respectively, the current vectors of the lower and upper bounds on variables $p(j)$, $j \in H$. For $j \in N$, each of $l(j)$ and $u(j)$ either takes the value of $\underline{p}(j)$ or that of $\bar{p}(j)$ from the original Problem (LP).

Throughout this paper, we assume that each Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ is feasible. This is guaranteed by Lemma 2 if the initial Problem (LP) is feasible.

The original Problem (LP) is represented as Problem $\mathrm{LP}(N, \varnothing, \varnothing, \underline{\mathbf{p}}, \bar{\mathbf{p}})$. For $j \in H$, we say that the variable $p(j)$ is a *non-fixed variable* if $l(j) < u(j)$ holds, and a *fixed variable* if $l(j) = u(j)$ holds. If all the variables in $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ are fixed, i.e., $l(j) = u(j)$ holds for all $j \in H$, then an optimal solution is uniquely determined by the vector $\mathbf{u} \in \mathbb{R}^H$.

Consider a general case that Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (15) contains at least one nonfixed variable, i.e., $|H \setminus F| > 0$. In accordance with (9) applied to function φ_K^H, we define a function $\tilde{\varphi}_K^H \colon 2^H \to \mathbb{R}$ by

$$\tilde{\varphi}_K^H(X) = \min_{Y \in 2^H} \{\varphi_K^H(Y) + u(X \setminus Y) - l(Y \setminus X)\}. \tag{16}$$

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

$$p^*(j) = \begin{cases} \tilde{\varphi}_K^H(\{j'\}), & j = j';\\ u(j), & j \in F. \end{cases} \tag{17}$$

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

$$\tilde{\varphi}_K^H(\hat{H}) = \varphi_K^H(Y_*) + u(\hat{H} \setminus Y_*) - l(Y_* \setminus \hat{H}). \tag{18}$$

Theorem 3, when applied to Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$, implies that the problem is decomposed into two subproblems

$$\begin{aligned} &\text{Maximize} \quad \sum_{j \in Y_*} w(j)p(j)\\ &\text{subject to} \quad p(X) \leq \varphi_K^{Y_*}(X) = \varphi(X \cup K) - \varphi(K), \quad X \in 2^{Y_*},\\ &\qquad\qquad l(j) \leq p(j) \leq l(j), \quad j \in Y_* \setminus \hat{H},\\ &\qquad\qquad l(j) \leq p(j) \leq u(j), \quad j \in Y_* \cap \hat{H}, \end{aligned} \tag{19}$$

and

$$\begin{aligned} &\text{Maximize} \quad \sum_{j \in H \setminus Y_*} w(j)p(j)\\ &\text{subject to} \quad p(X) \leq \varphi_{K \cup Y_*}^{H \setminus Y_*}(X) = \varphi(X \cup K \cup Y_*) - \varphi(K \cup Y_*), \quad X \in 2^{H \setminus Y_*},\\ &\qquad\qquad u(j) \leq p(j) \leq u(j), \quad j \in \hat{H} \setminus Y_*,\\ &\qquad\qquad l(j) \leq p(j) \leq u(j), \quad j \in (H \setminus Y_*) \setminus (\hat{H} \setminus Y_*). \end{aligned} \tag{20}$$

The first of these subproblems corresponds to Problem (LPR), and in that problem the values of components $p(j)$, $j \in Y_* \setminus \hat{H}$, are fixed to their lower bounds. The second subproblem corresponds to Problem (LPC), and in that problem the values of components $p(j)$, $j \in \hat{H} \setminus Y_*$, are fixed to their upper bounds.

We denote these subproblems by LP$(Y_*, F_1, K, \mathbf{l}_1, \mathbf{u}_1)$ and LP$(H\backslash Y_*, F_2, K \cup Y_*, \mathbf{l}_2, \mathbf{u}_2)$, respectively, where the vectors $\mathbf{l}_1, \mathbf{u}_1 \in \mathbb{R}^{Y_*}$ and $\mathbf{l}_2, \mathbf{u}_2 \in \mathbb{R}^{H\backslash Y_*}$, and the updated sets of the fixed variables F_1 and F_2 are given by

$$l_1(j) = l(j),\ j \in Y_*,$$

$$u_1(j) = \begin{cases} l(j), & j \in Y_* \backslash \hat{H}, \\ u(j), & j \in Y_* \cap \hat{H}, \end{cases} \qquad F_1 = Y_* \backslash \hat{H}, \tag{21}$$

$$l_2(j) = \begin{cases} u(j), & j \in \hat{H} \backslash Y_*, \\ l(j), & j \in H \backslash (Y_* \cup \hat{H}), \end{cases} \tag{22}$$

$$u_2(j) = u(j),\ j \in H \backslash Y_*, \quad F_2 = (\hat{H} \cup (H \cap F)) \backslash Y_*.$$

Notice that Problem LP$(Y_*, F_1, K, \mathbf{l}_1, \mathbf{u}_1)$ inherits the set of fixed variables $Y_* \cap F$ from the problem of a higher level, and additionally the variables of set $Y_* \backslash \hat{H}$ become fixed. However, since $\hat{H}$ contains only non-fixed variables, we deduce that $Y_* \backslash \hat{H} \supseteq Y_* \cap F$, so that the complete description of the set F_1 of fixed variables in Problem LP$(Y_*, F_1, K, \mathbf{l}_1, \mathbf{u}_1)$ is given by $Y_* \backslash \hat{H}$.

Problem LP$(Y_*, F_2, K \cup Y_*, \mathbf{l}_2, \mathbf{u}_2)$ inherits the set of fixed variables $(H \backslash Y_*) \cap F$ from the problem of a higher level, and additionally the variables of set $\hat{H} \backslash Y_*$ become fixed. These two sets are disjoint. Thus, the complete description of the set F_2 of fixed variables in Problem LP$(Y_*, F_2, K, \mathbf{l}_2, \mathbf{u}_2)$ is given by $(\hat{H} \cup (H \cap F)) \backslash Y_*$.

Without going into implementation details, we now give a formal description of the recursive procedure, that takes Remark 1 into account. For the current Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$, we compute optimal solutions $\mathbf{p_1} \in \mathbb{R}^{Y_*}$ and $\mathbf{p_2} \in \mathbb{R}^{H\backslash Y_*}$ of the two subproblems by calling procedures Decomp$(Y_*, F_1, K, \mathbf{l}_1, \mathbf{u}_1)$ and Decomp$(H \backslash Y_*, F_2, K \cup Y_*, \mathbf{l}_2, \mathbf{u}_2)$. By Theorem 3, the direct sum $\mathbf{p}^* = \mathbf{p}_1 \oplus \mathbf{p}_2$ is an optimal solution of Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$, which is the output of the Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$.

```
Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$
Step 1. If $|H\backslash F| = 0$, then output the vector $\mathbf{p}^* = \mathbf{u} \in \mathbb{R}^H$ and return.
    If $|H\backslash F| = 1$ and $H \backslash F = \{j'\}$, then compute the value $\tilde{\varphi}_K^H(\{j'\})$, and output the vector $\mathbf{p}^*$ given by (17) and return.
Step 2. Select a heavy-element subset $\hat{H}$ of $H\backslash F$ with respect to $\mathbf{w}$, and determine an instrumental set $Y_* \subseteq H$ for set $\hat{H}$, satisfying (18).
Step 3. Define the vectors $\mathbf{l_1}, \mathbf{u}_1 \in \mathbb{R}^{Y_*}$ and set $F_1$ by (21).
    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 $\mathbf{l}_2, \mathbf{u}_2 \in \mathbb{R}^{H\backslash Y_*}$ and set $F_2$ by (22).
    Call Procedure Decomp$(H\backslash Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$ to obtain an optimal solution $\mathbf{p_2} \in \mathbb{R}^{H\backslash Y_*}$ of Problem LP$(H\backslash Y_*, F_2, K \cup Y_*, \mathbf{l}_2, \mathbf{u}_2)$.
Step 5. Output the direct sum $\mathbf{p}^* = \mathbf{p}_1 \oplus \mathbf{p}_2 \in \mathbb{R}^H$ and return.
```

Recall that the original Problem (LP) is solved by calling Procedure Decomp$(N, \varnothing, \varnothing, \underline{\mathbf{p}}, \bar{\mathbf{p}})$. 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 set Y_*.

To reduce the depth of recursion of the procedure, it makes sense to perform decomposition in such a way that the number of non-fixed variables in each of the two emerging subproblems is roughly a half of $g = |H\backslash F|$, the number of nonfixed variables in the current Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$.

Lemma 3. *If at each level of recursion of Procedure* Decomp *for Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ with $g = |H\backslash F| > 1$ a heavy-element subset $\hat{H} \subseteq H\backslash F$ in Step 2 is chosen to contain $\lceil g/2 \rceil$ non-fixed variables, then the number of nonfixed variables in each of the two subproblems that emerge as a result of decomposition is either $\lceil g/2 \rceil$ or $\lfloor g/2 \rfloor$.*

Proof. In Step 2 of Procedure *Decomp*$(H, F, K, \mathbf{l}, \mathbf{u})$ select a heavy-element subset $\hat{H} \subset H\backslash F$ that contains $\lceil g/2 \rceil$ non-fixed variables, i.e., $|\hat{H}| = \lceil g/2 \rceil$. Then, the number of the non-fixed variables in Problem LP$(Y_*, F_1, K, \mathbf{l}_1, \mathbf{u}_1)$ considered in Step 3 satisfies $|Y_* \cap \hat{H}| \le \lceil g/2 \rceil$. Because of (22), the number of non-fixed variables in Problem LP$(H\backslash Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$ considered in Step 4 satisfies

$$|H \backslash (\hat{H} \cup F \cup Y_*)| \le |H \backslash \hat{H}| = \left\lfloor \frac{g}{2} \right\rfloor. \quad \square$$

Lemma 3 implies that the overall depth of recursion of Procedure Decomp applied to the initial Problem LP$(N, \varnothing, \varnothing, \underline{\mathbf{p}}, \bar{\mathbf{p}})$ is $O(\log n)$.

4.3. Finding an Instrumental Set

In each iteration of Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$, for the current heavy-element set $\hat{H}$ we need to find an instrumental set Y_* defined by (13) with $X = \hat{H}$ that determines the pair of problems into which the current problem is decomposed.

Most of the presented reasoning holds for any subset of H. Throughout this section, $X \subseteq H$ denotes an arbitrary set.

Formally speaking, for a given set X, the instrumental set Y_* can be found by minimizing the submodular function $\varphi_K^H(Y) + u(X\backslash Y) - l(Y\backslash X)$ over all subsets Y of set H. Thus, Y_* can be computed in polynomial time by using any of the available algorithms for minimizing a submodular function, see, e.g., Schrijver (2000) or Iwata et al. (2001). However, the running time of known algorithms is fairly large.

In many special cases of Problem (LP), including those related to scheduling applications, the value $\tilde{\varphi}(X)$ can be computed more efficiently without a direct use of the submodular function minimization; see, e.g., Shioura et al. (2013, 2015), where we present algorithms for minimizing functions similar to $\varphi_K^H(Y) + u(X\backslash Y) - l(Y\backslash X)$ that arise in solving scheduling problems with controllable processing times on parallel machines by submodular methods. In this paper, we describe another, rather universal approach that is based on the following statement.

Theorem 4 (cf. Fujishige 2005, Corollary 3.4). *For a submodular system $(2^H, \varphi)$ and a vector $\mathbf{b} \in \mathbb{R}^H$, the equality*

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

holds. In particular, if φ is a polymatroid rank function and $\mathbf{b} \ge \mathbf{0}$, then the right-hand side is equal to $\max\{p(H) \mid \mathbf{p} \in P(\varphi),\ \mathbf{0} \le \mathbf{p} \le \mathbf{b}\}$.

Given Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (15), for a set $X \subseteq H$ define the vector $\mathbf{b} \in \mathbb{R}^H$ by

$$b(j) = \begin{cases} u(j), & \text{if } j \in X; \\ l(j), & \text{if } j \in H\backslash X. \end{cases} \tag{23}$$

Starting from (16), for a set $X \subseteq H$ transform

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

Since $-l(H\backslash X)$ is a constant, to find an instrumental set Y_* that defines $\tilde{\varphi}_K^H(X)$ it suffices to find the set minimizer for $\min_{Y \in 2^H}\{\varphi_K^H(Y) + b(H\backslash Y)\}$, where the values $b(j)$ depend on X, as seen from (23). By Theorem 4, the latter minimization problem is equivalent to the following auxiliary problem:

$$\begin{aligned}\text{(AuxLP):}\quad \text{Maximize}\quad & \sum_{j \in H} q(j)\\ \text{subject to}\quad & q(Y) \le \varphi_K^H(Y), \quad Y \in 2^H;\\ & 0 \le q(j) \le b(j), \quad j \in H.\end{aligned} \tag{24}$$

The following property is useful.

Lemma 4. *For a set $X \subseteq H$, let $\mathbf{q}_* \in \mathbb{R}^H$ be an optimal solution to the auxiliary linear program (24). Then a set Y_* is the required instrumental set for Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (15) with $X \subseteq H$ if and only if*

$$q_*(Y_*) = \varphi_K^H(Y_*); \quad q_*(j) = b(j),\ j \in H\backslash Y_*,$$

where the values $b(j)$ are defined with respect to X.

Proof. By Theorem 4, Y_* is the set minimizer for $\min_{Y \in 2^H}\{\varphi_K^H(Y) + b(H\backslash Y)\}$ if and only if

$$\varphi_K^H(Y_*) + b(H\backslash Y_*) = q_*(H) = q_*(Y_*) + q_*(H\backslash Y_*).$$

Since vector $\mathbf{q}_* \in \mathbb{R}^H$ is a feasible solution to the auxiliary linear program (24), we have

$$q_*(Y_*) \le \varphi_K^H(Y_*); \quad q_*(j) \le b(j),\ j \in H\backslash Y_*.$$

Hence, Y_* is an instrumental set for set X if and only if each inequality displayed above holds as equality. □

Notice that Lemma 4 holds for any set $X \subseteq H$, but in a particular iteration of Procedure Decomp the search for an instrumental set is performed for the current heavy-element set $\hat{H}$.

Lemma 4 implies that once an optimal solution $\mathbf{q}_* \in \mathbb{R}^H$ to the auxiliary LP problem (24) is obtained, a required set Y_* can be found by partitioning the ground set H into two sets Y_* and $H\backslash Y_*$ so that $q_*(Y_*) = \varphi_K^H(Y_*)$ and $q_*(j) = b(j)$ for $j \in H\backslash Y_*$.

Observe that Problem (AuxLP) has a structure similar to that of Problem (LP); in fact, for $H = N$ Problem (AuxLP) is a special case of Problem (LP) with the following points of difference:

(i) the objective function is the sum of the decision variables, with all weights $w(j)$ equal to 1;

(ii) each decision variable has no lower bound (or zero lower bound if in Problem (LP) φ is a nonnegative rank function).

As follows from Lemma 4, for given Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (15), to find an instrumental set Y_* for a chosen heavy-element set $\hat{H}$, we may find an optimal solution $\mathbf{q}_* \in \mathbb{R}^H$ of Problem (AuxLP), in which the values $b(j)$ are defined by (23) with respect to $X = \hat{H}$, and then partition the original set H into two sets Y_* and $H\backslash Y_*$ such that

$$q(Y_*) = \varphi_K^H(Y_*); \quad q_*(j) = b(j), \quad j \in H\backslash Y_*. \tag{25}$$

In scheduling applications, solving Problem (AuxLP) can be understood as solving a relaxed version of the initial problem to minimize the total compression cost, in which the unit compression costs are all equal and the processing times have only upper bounds. This is illustrated for problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ in §5.

5. Solving Single Machine Problem via Decomposition

As demonstrated in §3, problem

$$1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \Big| \sum w(j)x(j)$$

can be reformulated as Problem (LP) with a nonnegative rank function $\varphi = \varphi_1$ given by (3). Section 4 presents a generic recursive decomposition Procedure Decomp for solving Problem (LP). To adapt that procedure for solving Problem (LP) related to problem $1|r(j),\ p(j)=\bar{p}(j)-x(j),\ pmtn,\ C(j)\le d(j)|\sum w(j)x(j)$ we only need to present implementation details of Steps 2–4 of a typical iteration of Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$.

We start our consideration with Step 2, in which an instrumental set Y_* has to be found. Given Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (15), for simplicity of exposition, define $\psi = \varphi_K^H$. For a set X of jobs, a meaningful interpretation of $\psi(X)$ 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. Renumber the jobs of set H by the integers $1, 2, \ldots, h$ in nonincreasing order of their release dates, i.e.,

$$r(1) \ge r(2) \ge \cdots \ge r(h); \tag{26}$$

additionally, we assume that if $r(j) = r(j+1)$ for some $j \in H$ then $d(j) \le d(j+1)$ holds.

Recall that in §2, we have introduced a set I of γ time intervals initially available for processing the jobs of set N. Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ corresponds to a single machine scheduling problem in which the jobs of set K have already been scheduled, and the jobs of set H must be scheduled in the remaining available time intervals. Denote the set of these available intervals by I_A, and assume that it consists of the intervals

$$[\tau_{\pi(1)-1}, \tau_{\pi(1)}], [\tau_{\pi(2)-1}, \tau_{\pi(2)}], \ldots, [\tau_{\pi(\nu)-1}, \tau_{\pi(\nu)}],$$

where $1 \le \pi(1) < \pi(2) < \cdots < \pi(\nu) \le h$. Comparing the sets I and I_A, we see that the machine is busy during the intervals

$$\left[\min_{j\in H} r(j), \tau_{\pi(1)-1}\right], [\tau_{\pi(1)}, \tau_{\pi(2)-1}], [\tau_{\pi(2)}, \tau_{\pi(3)-1}], \ldots,$$
$$[\tau_{\pi(\nu-1)}, \tau_{\pi(\nu)-1}], \left[\tau_{\pi(\nu)}, \max_{j\in H} d(j)\right].$$

We denote the set of these busy intervals by I_B.

Select a heavy-element set $\hat{H}$ and define the values $b(j)$ by (23) applied to $X = \hat{H}$. Our goal is to find an instrumental set Y_* for set $\hat{H}$. As described in §4.3, for this purpose we may solve the auxiliary Problem (ULP)

$$\text{(ULP):}\quad \text{Maximize} \sum_{j\in H} q(j)$$
$$\text{subject to}\quad q(X) \le \psi(X), \quad X \in 2^H,$$
$$0 \le q(j) \le b(j), \quad j \in H. \tag{27}$$

Problem (ULP) can be seen as a version of a scheduling problem $1|r(j),\ q(j) = b(j) - x(j),\ pmtn,\ C(j) \le d(j)|\sum x(j)$, 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 these jobs can only be processed in the intervals of set I_A, and $0 \le q(j) \le b(j)$ for each $j \in H$. Each job $j \in H$ should be assigned an actual processing time $q(j)$, where $0 \le q(j) \le b(j)$, so that all jobs are scheduled within the ν intervals of set I_A, no job j is scheduled outside the interval $[r(j), d(j)]$ and the total processing time $\sum_{j\in H} q(j)$ is maximized. No confusion arises if we speak about an LP problem (27) using its scheduling interpretation.

Problem (27) is related to the problem of minimizing the total (unweighted) compression with zero lower bounds on actual processing times. It can be solved by algorithms developed by Hochbaum and Shamir (1990) and Shih et al. (1991). In particular, the algorithm by Hochbaum and Shamir (1990) 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 numbered in accordance with (26). 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 scheduled in a "backward" manner: 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 within the interval $[r(j), d(j)]$ is left (recall that the intervals of set I_B are seen as busy).

For our purposes, however, we not only need the optimal values $q_*(j)$ of the processing times, but also a set of jobs $Y_* \subseteq H$ such that

$$q_*(Y_*) = \psi(Y_*), \quad q_*(j) = b(j), \quad j \in H \setminus Y_*. \tag{28}$$

We call a set Y^* that satisfies $q_*(Y_*) = \psi(Y_*)$ in (28) *tight*. In scheduling terms, for Problem (ULP) the jobs of a tight set completely use all intervals available for their processing. A job that belongs to some tight set is called *critical*. The length of a critical job cannot be extended (even ignoring its upper bound) without compromising feasibility of the schedule. If a job j is not critical, then the job does not use the whole interval even if its processing time is fully extended (and could have been extended further if we had ignored the upper bound $b(j)$).

Only a slight modification of the Hochbaum-Shamir algorithm, which does not affect its linear running time, leads to finding the required tight set Y_*. In the description of the algorithm the jobs are assumed to be numbered in accordance with (26).

For a schedule that is feasible for Problem (ULP) under consideration, an interval during which the machine is permanently busy, possibly including the intervals from I_B, is called a *block*. Recall that a schedule delivered by the Hochbaum-Shamir algorithm can be seen as a collection of blocks separated by idle intervals.

Algorithm HSY

```
*Step* 1. Set $Y_*^0 := \varnothing$.
*Step* 2. For each job $k$ from 1 to $h$ do
    (a) Schedule job $k$ in accordance with the algorithm by Hochbaum and Shamir (1990).
    (b) If in the current schedule the interval $[r(k), d(k)]$ has no idle time, then find a block $B^k$ in which job $k$ completes and determine the set $Y^k$ of all jobs that complete in the same block; define $Y_*^k := Y_*^{k-1} \cup Y^k$. Otherwise (i.e., if in the current schedule the interval $[r(k), d(k)]$ has an idle time), define $Y_*^k := Y_*^{k-1}$.
*Step* 3. Output $Y_* := Y_*^h$ and stop.
```

In what follows we formulate the statements that show that Algorithm HSY finds the set Y_* correctly. Each of the following lemmas is applied to schedule S_k, which is the schedule found in Step 2(a) for the jobs $1, \dots, k$.

Lemma 5. *In schedule S_k any job $j \le k$ starts and finishes in one block.*

Proof. Suppose $[t_1, t_2]$ and $[t_3, t_4]$, where $t_1 < t_2 < t_3 < t_4$, are two consecutive blocks in S_k such that job j, $j \le k$, is processed in each of these blocks. Because of the feasibility of schedule S_k, we have that $r(j) < t_2 < t_3 < d(j)$, i.e., the interval $[t_2, t_3]$ could be used for processing job j, but is left idle. This contradicts to the way the Hochbaum-Shamir algorithm operates. □

Lemma 6. *If the interval $[r(k), d(k)]$ has no idle time in schedule S_k, then Y^k is a tight set.*

Proof. Lemma 5 implies that in Step 2(b) of Algorithm HSY, Y^k is the set of jobs that start and complete in block B^k. Since job k has the smallest release date among all jobs in schedule S_k and the interval $[r(k), d(k)]$ has no idle time, it follows that block B^k contains the interval $\hat{I} = [r(k), t]$, where $t = \max\{d(j) \mid j \in Y^k\}$. Let δ denote the total length of all intervals of set I_B within the interval $\hat{I}$. Then $q_*(Y^k) = t - r(k) - \delta$. On the other hand, no job of set Y^k can start before time $r(k)$, complete after time t and be assigned to the intervals of set I_B, so that $\psi(Y^k) = t - r(k) - \delta$. Thus, $q_*(Y^k) = \psi(Y^k)$. □

Note that the set Y_*, which is output in Step 3, is given as the union of sets $Y^1, Y^2, \dots, Y^h$, and each Y^k ($k = 1, 2, \dots, h$) is a tight set by Lemma 6. Since the union of tight sets is again a tight set, the equality $q_*(Y_*) = \psi(Y_*)$ holds.

For $k \in H$, if $[r(k), d(k)]$ has no idle time in S_k, then k is included in Y^k, and therefore $k \in Y^k \subseteq Y_*$ holds. Hence, if $k \in H \backslash Y_*$, then $[r(k), d(k)]$ has idle time in S_k, implying that $q_*(k) = b(k)$. Thus, set Y_* found by the algorithm satisfies the condition (28). This guarantees that Algorithm HSY is correct, and set Y_* is the required instrumental set as a result of Lemma 4.

Recall that the Hochbaum-Shamir algorithm manipulates the intervals of machine availability organized in sets of contiguous intervals. In particular, it uses the FIND function to determine the set that contains any given original interval by retrieving the first interval in that set. Moreover, it uses the procedure UNION to merge two sets of intervals into a new set. Because the Hochbaum-Shamir algorithm actually determines the length of processing of each job k in the original intervals of availability, the required block B^k (the set of intervals that contains the latest interval for processing job k) will be found (see Step 2(b)). To determine the set Y^k of the jobs in block B^k, we assume that for each block (or a set of intervals) the list of jobs assigned to be processed in this block is maintained. Once the jobs of set Y^k are added to set Y_*^k, the corresponding block together with its list of jobs is deleted. When two sets of intervals merge (a larger block is formed), the corresponding lists of jobs are linked. Thus, the running time of the original algorithm by Hochbaum and Shamir is not affected.

Thus, we have proved that Algorithm HSY solves Problem (ULP) of the form (27) by scheduling h jobs of set H in the ν intervals of set I_A and finds the corresponding set Y_* that satisfies (28) in $O(h+\nu)$ time. By Lemma 4, the found set Y_* is an instrumental set.

Now we describe how Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ is decomposed in Steps 3 and 4 of Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$, provided that a heavy-element subset $\hat{H}$ is selected and the corresponding instrumental set Y_* is found. Determine

$$I' = \{[\tau_{k-1}, \tau_k] \mid k \in \Gamma(j) \text{ for some } j \in Y_*\},$$

where $\Gamma(j)$ is defined by (2). For problem (19) of processing the jobs of set Y_*, we update $I_A := I'$ and let $\nu_1 = |I'|$. For problem (20) of processing the jobs of set $H \backslash Y_*$, we update $I_A := I_A \backslash I'$ and let $\nu_2 = \nu - \nu_1$. The other required updates are carried out in accordance with (21) and (22).

Now we pass to analyzing the time complexity of the algorithm. Recall that the release dates and deadlines are assumed sorted, which requires $O(n \log n)$.

Let us estimate the running time of Procedure Decomp applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$. We denote by $T_{\mathrm{LP}}(h, g, \nu)$ the time complexity of Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$, where

$$h = |H|, \quad g = |H \backslash F|, \quad \nu = |I_A|.$$

Let $T_{Y_*}(h, g, \nu)$ denote the running time for computing the value $\tilde{\varphi}_K^H(\hat{H})$ for a given set $\hat{H} \subseteq H$

and for finding the corresponding instrumental set Y_*. From the previous discussion, we have $T_{Y_*}(h, g, \nu) = O(h+\nu)$. 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 variables are nonfixed, and ν_1 available intervals, and the other subproblem with $h_2 = h - h_1$ variables, among which g_2 variables are nonfixed, and ν_2 available intervals. By Lemma 3, we have

$$g_1 \le \min\{h_1, \lceil g/2 \rceil\}, \quad g_2 \le \min\{h_2, \lfloor g/2 \rfloor\}.$$

The required heavy-element set can be found in $O(h)$ time by using a linear-time median-finding algorithm.

Then, we obtain a recursive equation

$$T_{\mathrm{LP}}(h, g, \nu) = \begin{cases} O(1), & \text{if } g = 0; \\ O(h+\nu), & \text{if } g = 1; \\ O(h+\nu) + T_{\mathrm{LP}}(h_1, g_1, \nu_1) \\ \quad + T_{\mathrm{LP}}(h_2, g_2, \nu_2), & \text{if } g > 1. \end{cases} \tag{29}$$

Initially, we have $h = n$, $g = n$, and $\nu = 2n$. Hence, the overall running time of the decomposition algorithm is defined by $T_{\mathrm{LP}}(n, n, 2n)$. Taking into account that the depth of the recursion is $O(\log n)$, we deduce from the recursive equation that

$$T_{\mathrm{LP}}(n, n, 2n) = O(n \log n).$$

Theorem 5. *Problem $1 \mid r(j), p(j) = \bar{p}(j) - x(j), pmtn, C(j) \le d(j) \mid \sum w(j)x(j)$ can be solved by the decomposition algorithm in $O(n \log n)$ time.*

Notice that the running time of $O(n \log n)$ is the best possible and matches the time required for solving the feasibility problem $1 \mid r(j), pmtn, C(j) \le d(j) \mid \circ$.

6. Conclusions

It is known that for scheduling problems on parallel machines with controllable processing times, the running time of an algorithm that minimizes the total compression cost matches the running time needed for checking the existence of a feasible schedule with fixed processing times: $O(n^3)$ for identical parallel machines, and $O(mn^3)$ for uniform parallel machines. In this paper, we give an $O(n \log n)$-time algorithm for solving a single-machine scheduling problem with controllable processing times to minimize the total compression cost, thereby removing the complexity gap between the earlier known running times for this problem and its feasibility counterpart with fixed processing times. Moreover, this running time cannot be further reduced.

To achieve the best possible running time for the single machine model with controllable processing times, we use a submodular optimization approach, whose power has been demonstrated in our earlier work. Here, we develop a decomposition algorithm for solving an LP problem over a submodular polyhedron intersected with a box. This algorithm contributes to a toolkit of submodular optimization techniques and can be applied to problems from other areas that allow an appropriate submodular reformulation.

An important feature of the algorithm described in this paper is that the search for an instrumental set that is required for performing a decomposition step relies on solving an auxiliary problem; see §§4.3 and 5. In our recent paper Shioura et al. (2015) we use the decomposition algorithm to develop improved solution methods for three more scheduling problems with controllable processing times; however, in that paper we use different principles for finding an instrumental set.

Acknowledgments

This research was supported by the EPSRC funded project [EP/J019755/1] "Submodular Optimisation Techniques for Scheduling with Controllable Parameters." The first author is partially supported by the Humboldt Research Fellowship of the Alexander von Humboldt Foundation and by JSPS/MEXT KAKENHI [Grants 21740060, 24500002].

References

Ahuja RK, Orlin JB, Stein C, Tarjan RE (1994) Improved algorithms for bipartite network flow. *SIAM J. Comput.* 23:906–933.

Bansal N, Pruhs K, Stein C (2009) Speed scaling for weighted flow time. *SIAM J. Comput.* 39:1294–1308.

Bunde DP (2009) Power-aware scheduling for makespan and flow. *J. Sched.* 12:489–500.

Chung JY, Shih W-K, Liu JWS, Gillies DW (1989) Scheduling imprecise computations to minimize total error. *Microproc. Microprogram.* 27:767–774.

Federgruen A, Groenevelt H (1986) Preemptive scheduling of uniform machines by ordinary network flow techniques. *Management Sci.* 32:341–349.

Fujishige S (2005) *Submodular Functions and Optimization*, 2nd ed., Annals of Discrete Mathematics, Vol. 58 (Elsevier, Amsterdam).

Gallo G, Grigoriadis MD, Tarjan RE (1989) A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* 18: 30–55.

Gordon VS, Tanaev VS (1973) Deadlines in single-stage deterministic scheduling. *Optim. Systems Collecting, Transfer Processing Analogous Discrete Data Local Information Comput. Systems. Materials 1st Joint Soviet-Bulgarian Seminar* (Institute of Engineering Cybernetics of Bulgarian Academy of Sciences/Institute of Engineering Cybernetics of Belarussian Academy of Sciences, Minsk), 53–58. (in Russian).

Hochbaum DS, Shamir R (1990) Minimizing the number of tardy job unit under release time constraints. *Discr. Appl. Math.* 28: 45–57.

Horn W (1974) Some simple scheduling algorithms. *Naval Res. Logist. Quart.* 21:177–185.

Iwata S, Fleischer L, Fujishige S (2001) A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. *J. ACM* 48:761–777.

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: Algorithms and complexity. Graves CS, Rinnooy Kan AHG, Zipkin PH, eds. *Handbooks in Operations Research and Management Science*, Vol. 4, Logistics of Production and Inventory (North–Holland, Amsterdam), 445–522.

Leung JY-T (2004) Minimizing total weighted error for imprecise computation tasks. Leung JY-T, ed. *Handbook of Scheduling: Algorithms, Models and Performance Analysis* (Chapman & Hall/CRC, Boca Raton, FL), 34-1–34-16.

Leung JY-T, Yu VKM, Wei W-D (1994) Minimizing the weighted number of tardy task units. *Discr. Appl. Math.* 51:307–316.

McCormick ST (1999) Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. *Oper. Res.* 47:744–756.

Nowicki E, Zdrzałka S (1990) A survey of results for sequencing problems with controllable processing times. *Discr. Appl. Math.* 26:271–287.

Schrijver A (2000) A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory Ser. B* 80:346–355.

Schrijver A (2003) *Combinatorial Optimization: Polyhedra and Efficiency* (Springer, Berlin).

Shabtay D, Steiner G (2007) A survey of scheduling with controllable processing times. *Discr. Appl. Math.* 155:1643–1666.

Shakhlevich NV, Strusevich VA (2005) Pre-emptive scheduling problems with controllable processing times. *J. Sched.* 8: 233–253.

Shakhlevich NV, Strusevich VA (2008) Preemptive scheduling on uniform parallel machines with controllable job processing times. *Algorithmica* 51:451–473.

Shakhlevich NV, Shioura A, Strusevich VA (2008) Fast divide-and-conquer algorithms for preemptive scheduling problems with controllable processing times—A polymatroidal approach. Halperin D, Mehlhorn K, eds. *Euro. Symposium Algorithms 2008*, Lecture Notes in Computer Science, Vol. 5193 (Springer, Berlin), 756–767.

Shakhlevich NV, Shioura A, Strusevich VA (2009) Single machine scheduling with controllable processing times by submodular optimization. *Internat. J. Found. Comput. Sci.* 20:247–269.

Shih W-K, Lee C-R, Tang CH (2000) A fast algorithm for scheduling imprecise computations with timing constraints to minimize weighted error. *Proc. 21st IEEE Real-Time Systems Sympos.* (*RTSS*2000), *Orlando, FL*, 305–310.

Shih W-K, Liu JWS, Chung J-Y (1991) Algorithms for scheduling imprecise computations with timing constraints. *SIAM J. Comput.* 20:537–552.

Shih W-K, Liu JWS, Chung J-Y, Gillies DW (1989) Scheduling tasks with ready times and deadlines to minimize average error. *ACM SIGOPS Oper. Syst. Rev.* 23:14–28.

Shioura A, Shakhlevich NV, Strusevich VA (2013) A submodular optimization approach to bicriteria scheduling problems with controllable processing times on parallel machines. *SIAM J. Discr. Math.* 27:186–204.

Shioura A, Shakhlevich NV, Strusevich VA (2015) Decomposition algorithms for submodular optimization with applications to parallel machine scheduling with controllable processing times. *Math. Program., Ser. A.* 153:495–534.