Math. Program., Ser. A (2015) 153:495–534
DOI 10.1007/s10107-014-0814-9

FULL LENGTH PAPER

Decomposition algorithms for submodular optimization with applications to parallel machine scheduling with controllable processing times

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

Received: 7 May 2013 / Accepted: 1 September 2014 / Published online: 17 September 2014

Abstract In this paper we present a decomposition algorithm for maximizing a linear function over a submodular polyhedron intersected with a box. Apart from this contribution to submodular optimization, our results extend the toolkit available in deterministic machine scheduling with controllable processing times. We demonstrate how this method can be applied to developing fast algorithms for minimizing total compression cost for preemptive schedules on parallel machines with respect to given release dates and a common deadline. Obtained scheduling algorithms are faster and easier to justify than those previously known in the scheduling literature.

Keywords Submodular optimization · Parallel machine scheduling · Controllable processing times · Decomposition

Mathematics Subject Classification 90C27 · 90B35 · 90C05

Supported by the EPSRC funded project EP/J019755/1 and partially 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.

A. Shioura
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
e-mail: shioura@dais.is.tohoku.ac.jp

N. V. Shakhlevich
School of Computing, University of Leeds, Leeds LS2 9JT, UK
e-mail: N.Shakhlevich@leeds.ac.uk

V. A. Strusevich (✉)
Department of Mathematical Sciences, Old Royal Naval College,
University of Greenwich, Park Row, London SE10 9LS, UK
e-mail: sv02@gre.ac.uk

1 Introduction

In scheduling with controllable processing times, the actual durations of the jobs are not fixed in advance, but have to be chosen from a given interval. This area of scheduling has been active since the 1980s, see surveys [16] and [22].

Normally, for a scheduling model with controllable processing times two types of decisions are required: (1) each job has to be assigned its actual processing time, and (2) a schedule has to be found that provides a required level of quality. There is a penalty for assigning shorter actual processing times, since the reduction in processing time is usually associated with an additional effort, e.g., allocation of additional resources or improving processing conditions. 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.

As established in [23,24], there is a close link between scheduling with controllable processing times and linear programming problems with submodular constraints. This allows us to use the achievements of submodular optimization [4,21] for design and justification of scheduling algorithms. On the other hand, formulation of scheduling problems in terms of submodular optimization leads to the necessity of studying novel models with submodular constraints. Our papers [25,27] can be viewed as convincing examples of such a positive mutual influence of scheduling and submodular optimization.

This paper, which builds up on [26], makes another contribution towards the development of solution procedures for problems of submodular optimization and their applications to scheduling models. We present a decomposition algorithm for maximizing a linear function over a submodular polyhedron intersected with a box. Apart from this contribution to submodular optimization, our results extend the toolkit available in deterministic machine scheduling. We demonstrate how this method can be applied to several scheduling problems, in which it is required to minimize the total penalty for choosing actual processing times, also known as total compression cost. The jobs have to be processed with preemption on several parallel machines, so that no job is processed after a common deadline. The jobs may have different release dates.

The paper is organized as follows. Section 2 gives a survey of the relevant results on scheduling with controllable processing times. In Sect. 3 we reformulate three scheduling problems in terms of linear programming problems over a submodular polyhedron intersected with a box. Section 4 outlines a recursive decomposition algorithm for solving maximization linear programming problems with submodular constraints. The applications of the developed decomposition algorithm to scheduling with controllable processing times are presented in Sect. 5. The concluding remarks are contained in Sect. 6.

2 Scheduling with controllable processing times: a review

In this section, we give a brief overview of the known results on the preemptive scheduling problems with controllable processing times to minimize the total compression cost for schedules that are feasible with respect to given release dates and a common deadline.

Formally, in the model under consideration the jobs of set $N = \{1, 2, \ldots, n\}$ have to be processed on parallel machines $M_1, M_2, \ldots, M_m$, where $m \geq 2$. 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 $\left[\underline{p}(j), \overline{p}(j)\right]$. That selection process can be seen as either *compressing* (also known as *crashing*) the longest processing time $\overline{p}(j)$ down to $p(j)$, or *decompressing* the shortest processing time $\underline{p}(j)$ up to $p(j)$. In the former case, the value $x(j) = \overline{p}(j) - p(j)$ is called the *compression amount* of job j, while in the latter case $z(j) = p(j) - \underline{p}(j)$ is called the *decompression 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 associated with a choice of the actual processing times is represented by the linear function $W = \sum_{j \in N} w(j)x(j)$.

Each job $j \in N$ is given a *release date* $r(j)$, before which it is not available, and a common *deadline* d, 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, i.e., the time at which the last portion of job j is finished on the corresponding machine. A schedule is called *feasible* if the processing of a job $j \in N$ takes place in the time interval $[r(j), d]$.

We distinguish between the *identical* parallel machines and the *uniform* parallel 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_h has speed s_h, $1 \leq h \leq m$. Without loss of generality, throughout this paper we assume that the machines are numbered in non-increasing order of their speeds, i.e.,

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

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

$$p(j) = \sum_{h=1}^{m} s_h q^h(j).$$

In all scheduling problems studied in this paper, we need to determine the values of actual processing times and find the corresponding feasible preemptive schedule so that all jobs complete before the deadline and total compression cost is minimized. Adapting standard notation for scheduling problems by Lawler et al. [11], we denote problems of this type by $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$. Here, in the first field α we write "P" in the case of $m \ge 2$ identical machines and "Q " in the case of $m \ge 2$ 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 "$C(j) \le d$" reflects the fact that in a feasible schedule the common deadline should be respected. 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 $W = \sum_{j \in N} w(j)x(j)$. Scheduling problems with controllable processing times have received considerable attention since the 1980s, see, e.g., surveys by Nowicki and Zdrzałka [16] and by Shabtay and Steiner [22].

If the processing times $p(j)$, $j \in N$, are fixed then the corresponding counterpart of problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ is denoted by $\alpha|r(j), pmtn|C_{\max}$. In the latter problem it is required to find a preemptive schedule that for the corresponding settings minimizes the makespan $C_{\max} = \max\{C(j)|j \in N\}$.

In the scheduling literature, there are several interpretations and formulations of scheduling models that are related to those with controllable processing times. Below we give a short overview of them, indicating the points of distinction and similarity with our definition of the model.

Janiak and Kovalyov [8] argue that the processing times are *resource-dependent*, so that the more units of a single additional resource is given to a job, the more it can be compressed. In their model, a job $j \in N$ has a 'normal' processing time $b(j)$ (no resource given), and its actual processing time becomes $p(j) = b(j) - a(j)u(j)$, provided that $u(j)$ units of the resource are allocated to the job, where $a(j)$ is interpreted as a compression rate. The amount of the resource to be allocated to a job is limited by $0 \le u(j) \le \tau(j)$, where $\tau(j)$ is a known job-dependent upper bound. The cost of using one unit of the resource for compressing job j is denoted by $v(j)$, and it is required to minimize the total cost of resource consumption. This interpretation of the controllable processing times is essentially equivalent to that adopted in this paper, which can be seen by setting

$$\begin{aligned}\overline{p}(j) = b(j), \quad \underline{p}(j) = b(j) - a(j)\tau(j), \quad x(j) = a(j)u(j),\\ w(j) = v(j)/a(j), \quad j \in N.\end{aligned}$$

A very similar model for scheduling with controllable processing times is due to Chen [2], later studied by McCormick [13]. For example, McCormick [13] gives algorithms for finding a preemptive schedule for parallel machines that is feasible with respect to arbitrary release dates and deadlines. The actual processing time of a job is determined by $p(j) = \max\{b(j) - a(j)\lambda(j), 0\}$ and the objective is to minimize the function $\sum_{j \in N} \lambda(j)$. This is also similar to our interpretation due to

$$\overline{p}(j) = b(j), \quad \underline{p}(j) = 0, \quad x(j) = a(j)\lambda(j), \quad w(j) = 1/a(j), \quad j \in N.$$

Another range of scheduling models relevant to our study belongs to the area of imprecise computation; see [12] for a recent review. In computing systems that support imprecise computation, some computations (image processing programs, implementations of heuristic algorithms) can be run partially, producing less precise results. In our notation, a task with processing requirement $\overline{p}(j)$ can be split into a mandatory part which takes $\underline{p}(j)$ time, and an optional part that may take up to $\overline{p}(j) - \underline{p}(j)$ additional time units. To produce a result of reasonable quality, the mandatory part must be completed in full, while an optional part improves the accuracy of the solution. If instead of an ideal computation time $\overline{p}(j)$ a task is executed for $p(j) = \overline{p}(j) - x(j)$ time units, 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 similarity between the models with controllable processing times and those of imprecise computation have not been noticed. Even the most recent survey by Shabtay and Steiner [22] makes no mention of the imprecise computation research.

Scheduling problems with controllable processing times can serve as mathematical models in make-or-buy decision-making; see, e.g., Shakhlevich et al. [25]. 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. Under this interpretation, the orders are the jobs and for each order $j \in N$, the value of $\overline{p}(j)$ is interpreted as the processing requirement, provided that the order is manufactured internally in full, while $\underline{p}(j)$ is a given mandatory limit on the internal production. Further, $p(j) = \overline{p}(j) - x(j)$ 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 find a deadline-feasible schedule for internally manufactured orders.

It is obvious that for scheduling problems with controllable processing times, minimizing the total compression cost W is equivalent to maximizing either the total decompression cost $\sum w(j)z(j)$ or total weighted processing time $\sum w(j)p(j)$. Most of the problems relevant to this study have been solved using a greedy approach. One way of implementing this approach is to start with a (possibly, infeasible) schedule in which all jobs are fully decompressed to their longest processing times $\overline{p}(j)$, scan the jobs in non-decreasing order of their weights $w(j)$ and compress each job by the smallest possible amount that guarantees a feasible processing of a job. Another approach, which is in some sense dual to the one described above, is to start with a feasible schedule in which all jobs are fully compressed to their smallest processing times $\underline{p}(j)$, scan the jobs in non-increasing order of their weights $w(j)$ and decompress each job by the largest possible amount.

Despite the similarity of these approaches, in early papers on this topic each problem is considered separately and a justification of the greedy approach is often lengthy and developed from the first principles. However, as established by later studies, the greedy nature of the solution approaches is due to the fact that many scheduling problems with controllable processing times can be reformulated in terms of linear programming problems over special regions such as submodular polyhedra, (generalized) polymatroids, base polyhedra, etc. See Sect. 3 for definitions and main concepts of submodular optimization.

Nemhauser and Wolsey [15] were among the first who noticed that scheduling with controllable processing times could be handled by methods of submodular optimization; see, e.g., Example 6 (Sect. 6 of Chapter III.3) of the book [15]. A systematic development of a general framework for solving scheduling problems with controllable processing times via submodular methods has been initiated by Shakhlevich and Strusevich [23,24] and further advanced by Shakhlevich et al. [25]. This paper makes another contribution in this direction.

Below we review the known results on the problems to be considered in this paper. Two aspects of the resulting algorithms are important: (1) finding the actual processing times and therefore the optimal value of the function, and (2) finding the corresponding optimal schedule. The second aspect is related to traditional scheduling to minimize the makespan with fixed processing times.

Zero release dates, common deadline The results for the models under these conditions are summarized in the second and third columns of Table 1. If the machines are identical, then solving problem $P|pmtn|C_{\max}$ with fixed processing times can be done by a linear-time algorithm that is due to McNaughton [14]. As shown by Jansen and Mastrolilli [9], problem $P|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|W$ reduces to a continuous generalized knapsack problem and can be solved in $O(n)$ time. Shakhlevich and Strusevich [23] consider the bicriteria problem $P|p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$, in which makespan $C_{\max}$ and the total compression cost $W = \sum w(j)x(j)$ have to be minimized simultaneously, in the Pareto sense; the running time of their algorithm is $O(n \log n)$.

In the case of uniform machines, the best known algorithm for solving problem $Q|pmtn|C_{\max}$ with fixed processing times is due to Gonzalez and Sahni [5]. For problem $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|W$ Nowicki and Zdrzałka [17] show how to find the actual processing times in $O(nm + n \log n)$ time. Shakhlevich and Strusevich [24] reduce the problem to maximizing a linear function over a generalized polymatroid; they give an algorithm that requires the same running time as that by Nowicki and Zdrzałka [17], but can be extended to solving a bicriteria problem $Q|p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$. The best running time for the bicriteria problem is $O(nm \log m)$, which is achieved in [27] by submodular optimization techniques.

Arbitrary release dates, common deadline The results for the models under these conditions are summarized in the fourth and fifth columns of Table 1. These models are symmetric to those with a common zero release date and arbitrary deadlines. Problem $P|r(j), pmtn|C_{\max}$ with fixed processing times on m identical parallel machines can

Table 1 Summary of the results

Problem	$r(j)=0$		Arbitrary $r(j)$	
	$\alpha = P$	$\alpha = Q$	$\alpha = P$	$\alpha = Q$
$\alpha \vert r(j), pmtn \vert C_{\max}$	$O(n)$ [14]	$O(m \log m + n)$ [5]	$O(n \log n)$ [18]	$O(nm + n \log n)$ [19]
$\alpha \vert r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \le d \vert W$				
Previously known	$O(n)$ [9]	$O(nm + n \log n)$ [17,24]	$O(n^2 \log m)$ [27]	$O(n^2 m)$ [27]
This paper	–	$O(\min\{n \log n, n+m \log m \log n\})$ Section 5.1	$O(n \log n \log m)$ Section 5.2	$O(nm \log n)$ Section 5.3
$\alpha \vert r(j), p(j) = \overline{p}_j - x(j), pmtn \vert (C_{\max}, W)$	$O(n \log n)$ [23]	$O(nm \log m)$ [27]	$O\left(n^2 \log m\right)$ [27]	$O\left(n^2 m\right)$ [27]

be solved in $O(n \log n)$ time (or in $O(n \log m)$ time if the jobs are pre-sorted) as proved by Sahni [18]. For the uniform machines, Sahni and Cho [19] give an algorithm for problem $Q|r(j), pmtn|C_{\max}$ that requires $O(mn + n \log n)$ time (or $O(mn)$ time if the jobs are pre-sorted).

Prior to our work on the links between submodular optimization and scheduling with controllable processing times [27], no purpose-built algorithms have been known for problems $\alpha|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \le d|W$ with $\alpha \in \{P, Q\}$. It is shown in [27] that the bicriteria 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. Since a solution to a single criterion problem $\alpha m|r(j), p(j) = \overline{p}(j) - x(j), pmtn, C(j) \le d|W$ is contained among the Pareto optimal solutions for the corresponding bicriteria problem $\alpha m|r(j), p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, W)$, the algorithms from [27] are quoted in Table 1 as the best previously known for the single criterion problems with controllable processing times.

The main purpose of this paper is to demonstrate that the single criterion scheduling problems with controllable processing times to minimize the total compression cost can be solved by faster algorithms that are based on reformulation of these problems in terms of a linear programming problem over a submodular polyhedron intersected with a box. For the latter generic problem, we develop a recursive decomposition algorithm and show that for the scheduling applications it can be implemented in a very efficient way.

3 Scheduling with controllable processing times: submodular reformulations

For completeness, we start this section with definitions related to submodular optimization. Unless stated otherwise, we follow a comprehensive monograph on this topic by Fujishige [4], see also [10,21]. In Sect. 3.1, we introduce a linear programming problem for which the set of constraints is a submodular polyhedron intersected with a box. Being quite general, the problem represents a range of scheduling models with controllable processing times. In Sect. 3.2 we give the details of the corresponding reformulations.

3.1 Preliminaries on submodular polyhedra

For a positive integer n, let $N = \{1, 2, \dots, n\}$ 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 $\mathbf{p} = (p(1), p(2), \dots, p(n)) \in \mathbb{R}^N$ and $\mathbf{q} = (q(1), q(2), \dots, q(n)) \in \mathbb{R}^N$, we write $\mathbf{p} \le \mathbf{q}$ if $p(j) \le q(j)$ for each $j \in N$. Given a set $X \subseteq \mathbb{R}^N$, a vector $\mathbf{p} \in X$ is called *maximal* in X if there exists no vector $\mathbf{q} \in X$ such that $\mathbf{p} \le \mathbf{q}$ and $\mathbf{p} \ne \mathbf{q}$. 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 $\varphi : 2^N \to \mathbb{R}$ is called *submodular* if the inequality

$$\varphi(X) + \varphi(Y) \ge \varphi(X \cup Y) + \varphi(X \cap Y)$$

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 the *rank function* of that system.

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

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

$$B(\varphi) = \left\{\mathbf{p} \in \mathbb{R}^N \mid \mathbf{p} \in P(\varphi), \quad p(N) = \varphi(N)\right\}, \tag{3}$$

called the *submodular polyhedron* and the *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 in this section is as follows:

$$\begin{array}{lll} \text{(LP):} \quad \text{Maximize} & \displaystyle\sum_{j \in N} w(j)p(j) & \\ \quad\quad\quad \text{subject to} & p(X) \le \varphi(X), & X \in 2^N, \\ & \underline{p}(j) \le p(j) \le \overline{p}(j), & j \in N, \end{array} \tag{4}$$

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. This problem serves as a mathematical model for many scheduling problems with controllable processing times. Problem (LP) can be classified as a problem of maximizing a linear function over a submodular polyhedron intersected with a box.

In our previous work [25], we have demonstrated that Problem (LP) can be reduced to optimization over a simpler structure, namely, over a base polyhedron. In fact, we have shown that a problem of maximizing a linear function over the intersection of a submodular polyhedron and a box is equivalent to maximizing the same objective function over a base polyhedron associated with another rank function.

Theorem 1 (cf. [25])

(i) *Problem (LP) has a feasible solution if and only if* $\underline{\mathbf{p}} \in P(\varphi)$ *and* $\underline{\mathbf{p}} \le \overline{\mathbf{p}}$.
(ii) *If Problem (LP) has a feasible solution, then the set of maximal feasible solutions of Problem (LP) 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*

$$\tilde{\varphi}(X) = \min_{Y \in 2^N} \left\{ \varphi(Y) + \overline{p}(X \backslash Y) - \underline{p}(Y \backslash X) \right\}. \tag{5}$$

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 computed in polynomial time by using any of the available algorithms for minimizing a submodular function [7,20]. However, the running time of known algorithms is fairly large. In many special cases of Problem (LP), including its applications to scheduling problems with controllable processing times, the value $\tilde{\varphi}(X)$ can be computed more efficiently without using the submodular function minimization, as shown later.

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 $\underline{\mathbf{p}} \le \overline{\mathbf{p}}$. Claim (ii) of Theorem 1 implies that Problem (LP) reduces to the following problem:

$$\begin{aligned} &\text{Maximize} \quad \sum_{j \in N} w(j) p(j) \\ &\text{subject to} \quad \mathbf{p} \in B(\tilde{\varphi}), \end{aligned} \tag{6}$$

where the rank function $\tilde{\varphi} : 2^N \to \mathbb{R}$ is given by (5).

An advantage of the reduction of Problem (LP) to a problem of the form (6) is that the solution vector can be obtained essentially in a closed form, as stated in the theorem below.

Theorem 2 (cf. [4]) *Let* $j_1, j_2, \ldots, j_n$ *be an ordering of elements in* N *that satisfies*

$$w(j_1) \ge w(j_2) \ge \cdots \ge w(j_n). \tag{7}$$

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

$$p^*(j_h) = \tilde{\varphi}(\{j_1, \ldots, j_{h-1}, j_h\}) - \tilde{\varphi}(\{j_1, \ldots, j_{h-1}\}), \quad h = 1, 2, \ldots, n, \tag{8}$$

is an optimal solution to the problem (6) *[and also to the problem* (4)*].*

This theorem immediately implies a simple algorithm for Problem (LP), which computes an optimal solution $\mathbf{p}^*$ by determining the value $\tilde{\varphi}(\{j_1, j_2, \ldots, j_h\})$ for each $h = 1, 2, \ldots, n$. In this paper, instead, we use a different algorithm based on decomposition approach to achieve better running times for special cases of Problem (LP), as explained in Sect. 4.

3.2 Rank functions for scheduling applications

In this subsection, we follow [27] and present reformulations of three scheduling problems on parallel machines with controllable processing times in terms of LP problems defined over a submodular polyhedron intersected with a box of the form (4). We assume that if the jobs have different release dates, they are numbered to satisfy

$$r(1) \le r(2) \le \cdots \le r(n). \tag{9}$$

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

$$S_0 = 0, \qquad S_k = s_1 + s_2 + \cdots + s_k, \quad 1 \le k \le m. \tag{10}$$

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

For each problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$, $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ and $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$, 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 due date d and the total compression cost $W = \sum_{j \in N} w(j)x(j)$ is minimized. In what follows, we present LP formulations of these problems with $p(j),\ j \in N$, being decision variables, and the objective function to be maximized being $\sum_{j \in N} w(j)p(j) = \sum_{j \in N} w(j)\left(\overline{p}(j) - x(j)\right)$. Since each decision variable $p(j)$ has a lower bound $\underline{p}(j)$ and an upper bound $\overline{p}(j)$, an LP formulation includes the box constraints of the form $\underline{p}(j) \le p(j) \le \overline{p}(j),\ j \in N$.

The derivations of the rank functions for the models under consideration can be justified by the conditions for the existence of a feasible schedule for a given common deadline d formulated, e.g., in [1]. Informally, these conditions state that for a given deadline d a feasible schedule exists if and only if

(i) for each k, $1 \le k \le m - 1$, 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.

We refer to [27] where the rank functions for the relevant problems are presented and discussed in more details. Below we present their definitions. In all scheduling applications a meaningful interpretation of $\varphi(X)$ is the largest capacity available for processing the jobs of set X.

For example, problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ reduces to Problem (LP) of the form (4) with the rank function

$$\varphi(X) = dS_{\min\{|X|, m\}} = \begin{cases} dS_{|X|}, & \text{if } |X| \le m - 1, \\ dS_m, & \text{if } |X| \ge m. \end{cases} \tag{11}$$

It is clear that the conditions $p(X) \le \varphi(X),\ X \in 2^N$, for the function $\varphi(X)$ defined by (11) correspond to the conditions (i) and (ii) above, provided that $|X| \le m-1$ and $|X| \ge m$, respectively. As proved in [24], function φ is submodular.

We then consider problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$. For a set of jobs $X \subseteq N$, we define $r_i(X)$ to be the i-th smallest release date in set $X \in 2^N,\ 1 \le i \le |X|$. 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

$$\varphi(X) = \begin{cases} dS_{|X|} - \sum_{i=1}^{|X|} s_i r_i(X), & \text{if } |X| \le m-1, \\ dS_m - \sum_{i=1}^{m} s_i r_i(X), & \text{if } |X| \ge m. \end{cases} \tag{12}$$

It can be verified that this function is submodular.

Problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ is a special case of problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$, where $s_1 = s_2 = \cdots = s_m = 1$. Hence, the corresponding rank function φ can be simplified as

$$\varphi(X) = \begin{cases} d|X| - \sum_{i=1}^{|X|} r_i(X), & \text{if } |X| \le m-1, \\ dm - \sum_{i=1}^{m} r_i(X), & \text{if } |X| \ge m. \end{cases} \tag{13}$$

4 Decomposition of LP problems with submodular constraints

In this section, we describe a decomposition algorithm for solving LP problems defined over a submodular polyhedron intersected with a box. In Sect. 4.1, we demonstrate that the linear programming problem under study can be recursively decomposed into subproblems of a smaller dimension, with some components of a solution vector fixed to one of their bounds. We provide an outline of an efficient recursive decomposition procedure in Sect. 4.2 and analyze its time complexity in Sect. 4.3. In Sect. 5 we present implementation details of the recursive decomposition procedure for the relevant scheduling models with controllable processing times.

4.1 Fundamental idea for decomposition

In this section, we show an important property, which makes the foundation of our decomposition algorithm for Problem (LP) of the form (4).

The lemma below demonstrates that some components of an optimal solution can be fixed either at their upper or lower bounds, while for some 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

$$\min_{j \in \hat{N}} w(j) \ge \max_{j \in N \setminus \hat{N}} w(j).$$

For completeness, we also regard the empty set as a heavy-element subset of N.

Given Problem (LP), in accordance with (5) define a set $Y_* \subseteq N$ such that the equality

$$\tilde{\varphi}(X) = \varphi(Y_*) + \overline{p}(X \backslash Y_*) - \underline{p}(Y_* \backslash X) \tag{14}$$

holds for a set $X \subseteq N$. Because of its special role, in the remainder of this paper 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*

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

Proof Since $\hat{N}$ is a heavy-element subset, there exists an ordering $j_1, j_2, \ldots, j_n$ of elements in N that satisfies (7) 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 (8) is optimal. In particular, this implies

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

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

$$p^*(Y_*) \le \varphi(Y_*), \quad p^*(j) \le \overline{p}(j),\ \ j \in \hat{N} \backslash Y_*, \quad -p^*(j) \le -\underline{p}(j), j \in Y_* \backslash \hat{N}. \tag{15}$$

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

$$p^*(\hat{N}) = \tilde{\varphi}(\hat{N}) = \varphi(Y_*) + \overline{p}(\hat{N} \backslash Y_*) - \underline{p}\left(Y_* \backslash \hat{N}\right),$$

which implies that each inequality of (15) 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 $\left(2^N, \varphi\right)$, as defined in [4, 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 $\varphi_A : 2^{N \backslash A} \to \mathbb{R}$ by

$$\varphi_A(X) = \varphi(X \cup A) - \varphi(A), \quad X \in 2^{N \backslash A}.$$

Then, $(2^{N\backslash A}, \varphi_A)$ is a submodular system on $N\backslash A$ and it is 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 $\left(2^N, \varphi\right)$ to A and contraction of $\left(2^N, \varphi\right)$ by A, respectively. These problems can be written as follows: for restriction as

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

and for contraction as

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

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 $\mathbf{p_2} \in \mathbb{R}^{N\backslash A}$, 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

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

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 $\mathbf{p_2} \in \mathbb{R}^{N\backslash A}$ 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).*

Proof The proof below is similar to that for Lemma 3.1 in [4]. We define vectors $\mathbf{q_1} \in \mathbb{R}^A$ and $\mathbf{q_2} \in \mathbb{R}^{N\backslash A}$ by

$$q_1(j) = q(j), j \in A, \qquad q_2(j) = q(j), j \in N\backslash A.$$

To prove (i), it suffices to show that $\mathbf{q_1}$ and $\mathbf{q_2}$ are feasible solutions of Problems (LP1) and (LP2), respectively. Since $\mathbf{q}$ is a feasible solution of Problem (LP), we have

$$q(X) \le \varphi(X), \qquad X \in 2^N, \tag{16}$$

$$\underline{p}(j) \le q(j) \le \overline{p}(j), \quad j \in N. \tag{17}$$

Then, (16) and (17) imply that $\mathbf{q_1} \in \mathbb{R}^A$ is a feasible solution of Problem (LP1). It follows from (16) and the equality $q(A) = \varphi(A)$ that

$$q(X) = q(X \cup A) - q(A) \leq \varphi(X \cup A) - \varphi(A), \quad X \in 2^{N\backslash A},$$

which, together with (17), implies that $\mathbf{q_2} \in \mathbb{R}^{N\backslash A}$ is a feasible solution of Problem (LP2). This concludes the proof of (i).

To prove (ii), we first show that $\mathbf{p}^*$ is a feasible solution of Problem (LP). Since $\mathbf{p_1}$ and $\mathbf{p_2}$ are feasible solutions of Problem (LP1) and Problem (LP2), respectively, we have

$$p^*(X) \leq \varphi(X), \qquad X \in 2^A, \tag{18}$$
$$p^*(X) \leq \varphi(X \cup A) - \varphi(A), \quad X \in 2^{N\backslash A}, \tag{19}$$
$$\underline{p}(j) \leq p^*(j) \leq \overline{p}(j), \qquad j \in N. \tag{20}$$

For any $X \in 2^N$, we derive

$$\begin{aligned} p^*(X) &= p^*(X \cap A) + p^*(X\backslash A) \\ &\leq \varphi(X \cap A) + \varphi((X\backslash A) \cup A) - \varphi(A) \\ &= \varphi(X \cap A) + \varphi(X \cup A) - \varphi(A) \\ &\leq \varphi(X), \end{aligned}$$

where the first inequality is by (18) and (19), and the second by the submodularity of φ. This inequality and (20) show that the vector $\mathbf{p}^*$ is a feasible solution of (LP).

To show optimality of $\mathbf{p}^*$, notice that by optimality of $\mathbf{p_1}$ and $\mathbf{p_2}$ we have

$$\sum_{j\in A} w(j)p_1(j) \geq \sum_{j\in A} w(j)q_1(j), \quad \sum_{j\in N\backslash A} w(j)p_2(j) \geq \sum_{j\in N\backslash A} w(j)q_2(j),$$

and due to the definition of $\mathbf{p}^*$ we obtain

$$\begin{aligned} \sum_{j\in N} w(j)p^*(j) &= \sum_{j\in A} w(j)p_1(j) + \sum_{j\in N\backslash A} w(j)p_2(j) \\ &\geq \sum_{j\in A} w(j)q_1(j) + \sum_{j\in N\backslash A} w(j)q_2(j) = \sum_{j\in N} w(j)q(j), \end{aligned}$$

so that, $\mathbf{p}^*$ is an optimal solution of (LP). □

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 $\mathbf{p_2} \in \mathbb{R}^{N\backslash Y^*}$ be optimal solutions of the linear programs (LPR) and (LPC), respectively, where (LPR) and (LPC) are given as*

$$
\begin{array}{llll}
\text{(LPR)}: & \text{Maximize} & \displaystyle\sum_{j\in Y_*} w(j)p(j) & \\
& \text{subject to} & p(X) \le \varphi(X), & X \in 2^{Y_*}, \\
& & \underline{p}(j) \le p(j) \le \overline{p}(j), & j \in Y_* \cap \hat{N}, \\
& & p(j) = \underline{p}(j), & j \in Y_* \backslash \hat{N} \\
\text{(LPC)}: & \text{Maximize} & \displaystyle\sum_{j\in N\backslash Y_*} w(j)p(j) & \\
& \text{subject to} & p(X) \le \varphi(X \cup Y_*) - \varphi(Y_*), & X \in 2^{N\backslash Y_*}, \\
& & \underline{p}(j) \le p(j) \le \overline{p}(j), & j \in (N\backslash Y_*) \backslash \left(\hat{N}\backslash Y_*\right), \\
& & p(j) = \overline{p}(j), & j \in \hat{N}\backslash Y_*.
\end{array}
$$

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 $p(j)$, $j \in Y_*\backslash\hat{N}$, 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 $p(j)$, $j \in \hat{N}\backslash Y_*$, 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, while 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 of set N but only among the non-fixed variables. Formally, if the set N of jobs is known to be partitioned as $N = Q \cup F$, where the variables of set Q are non-fixed and those of set F are fixed, then $\hat{Q} \subseteq Q$ is a *heavy-element subset with respect to the weight vector* $\mathbf{w}$ if it satisfies the condition

$$\min_{j\in\hat{Q}} w(j) \ge \max_{j\in Q\backslash\hat{Q}} w(j).$$

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:

$$\begin{array}{llll} \mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u}) & \text{Maximize} & \sum_{j \in H} w(j)p(j) & \\ & \text{subject to} & p(X) \leq \varphi_K^H(X) = \varphi(X \cup K) - \varphi(K), & X \in 2^H, \\ & & l(j) \leq p(j) \leq u(j), & j \in H \backslash F, \\ & & p(j) = u(j) = l(j), & j \in F, \end{array} \tag{21}$$

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 $j \in F$;
- $K \subseteq N \backslash H$ is the set that defines the rank function $\varphi_K^H : 2^H \to \mathbb{R}$ such that

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

- $\mathbf{l} = (l(j) \mid j \in H)$ and $\mathbf{u} = (u(j) \mid j \in H)$ are respectively the 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 $\overline{p}(j)$ from the original Problem (LP). Notice that $l(j) = u(j)$ for each $j \in F$.

Throughout this paper, we assume that each Problem 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 LP$(N, \emptyset, \emptyset, \underline{\mathbf{p}}, \overline{\mathbf{p}})$. 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 Problem 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 LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (21) contains at least one non-fixed variable, i.e., $|H \backslash F| > 0$. We define a function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ by

$$\widetilde{\varphi}_K^H(X) = \min_{Y \in 2^H} \{\varphi_K^H(Y) + u(X \backslash Y) - l(Y \backslash X)\}. \tag{22}$$

By Theorem 1 (ii), the set of maximal feasible solutions of Problem 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 $|H \backslash F| = 1$ and $H \backslash F = \{j'\}$, then an optimal solution $\mathbf{p}^* \in \mathbb{R}^H$ is given by

$$p^*(j) = \begin{cases} \tilde{\varphi}_K^H(\{j'\}), & j = j', \\ u(j), & j \in F, \end{cases} \tag{23}$$

Suppose that $|H \backslash F| \geq 2$. Then, we call a procedure Procedure DECOMP(H, F, K, $\mathbf{l}$, $\mathbf{u}$) explained below. Let $\hat{H} \subseteq H \backslash F$ 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.,

$$\widetilde{\varphi}_K^H(\hat{H}) = \varphi_K^H(Y_*) + u\left(\hat{H} \backslash Y_*\right) - l(Y_* \backslash \hat{H}). \tag{24}$$

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

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

and

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

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

We denote these subproblems by Problem LP(Y_*, F_1, K, $\mathbf{l_1}$, $\mathbf{u_1}$) and Problem 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 fixed variables F_1 and F_2 are given by

$$\begin{aligned} l_1(j) &= l(j), \ j \in Y_*, \\ u_1(j) &= \begin{cases} l(j), & j \in Y_* \backslash \hat{H}, \\ u(j), & j \in Y_* \cap \hat{H}, \end{cases} \\ F_1 &= Y_* \backslash \hat{H}, \end{aligned} \tag{25}$$

$$\begin{aligned} l_2(j) &= \begin{cases} u(j), & j \in \hat{H} \backslash Y_*, \\ l(j), & j \in H \backslash (Y_* \cup \hat{H}), \end{cases} \\ u_2(j) &= u(j), \ j \in H \backslash Y_*, \\ F_2 &= (\hat{H} \cup (H \cap F)) \backslash Y_*. \end{aligned} \tag{26}$$

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$(H \backslash Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$ 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$(H \backslash Y_*, F_2, K, \mathbf{l_2}, \mathbf{u_2})$ 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 (23) 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 (24).

Step 3. Define the vectors $\mathbf{l_1}, \mathbf{u_1} \in \mathbf{R}^{Y_*}$ and set F_1 by (25).
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 $\mathbf{l_2}, \mathbf{u_2} \in \mathbf{R}^{H \backslash Y_*}$ and set F_2 by (26).
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$(N, \emptyset, \emptyset, \underline{\mathbf{p}}, \overline{\mathbf{p}})$. 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_*.

4.3 Analysis of time complexity

We analyze the time complexity of Procedure DECOMP. 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 the number of non-fixed 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 $|H \backslash F| > 1$ a heavy-element subset $\hat{H} \subseteq H \backslash F$ in Step 2 is chosen to contain $\lceil |H \backslash F|/2 \rceil$ non-fixed variables, then the number of non-fixed* *variables in each of the two subproblems that emerge as a result of decomposition is either* $\lceil |H\backslash F|/2\rceil$ *or* $\lfloor |H\backslash F|/2\rfloor$.

Proof For Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$, let $g = |H\backslash F|$ denote the number of the non-fixed variables. In Step 2 Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$ selects 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 $|Y_* \cap \hat{H}| \leq \lceil g/2\rceil$.

Due to (26), 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

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

□

This lemma implies that the overall depth of recursion of Procedure Decomp applied to Problem LP$(N, \emptyset, \emptyset, \underline{\mathbf{p}}, \overline{\mathbf{p}})$ is $O(\log n)$.

Let us analyze the running time of Procedure Decomp applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$. We denote by $T_{\mathrm{LP}}(h, g)$ the time complexity of Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$, where $h = |H|$ and $g = |H\backslash F|$. 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_* that minimizes the right-hand side of the Eq. (22). 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 there exist $g_1 \leq \min\{h_1, \lceil g/2\rceil\}$ non-fixed variables, and the other one with $h_2 = h - h_1$ variables, among which there exist $g_2 \leq \min\{h_2, \lfloor g/2\rfloor\}$ non-fixed variables. Let $T_{\mathrm{Split}}(h)$ denote the time complexity of such a decomposition, i.e., for setting up the instances of the two subproblems. A 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:

$$T_{\mathrm{LP}}(h, g) = \begin{cases} O(1), & \text{if } g = 0,\\ T_{Y_*}(h), & \text{if } g = 1,\\ T_{Y_*}(h) + T_{\mathrm{Split}}(h) + T_{\mathrm{LP}}(h_1, g_1) + T_{\mathrm{LP}}(h_2, g_2), & \text{if } g > 1. \end{cases}$$

By solving the recursive equation under an assumption that both functions $T_{Y_*}(h)$ and $T_{\mathrm{Split}}(h)$ are non-decreasing and convex, we obtain

$$T_{\mathrm{LP}}(n, n) = O(\big(T_{Y_*}(n) + T_{\mathrm{Split}}(n)\big) \log n).$$

Thus, the findings of this section can be summarized as the following statement.

Theorem 4 *Problem (LP) can be solved by Procedure* Decomp *in* $O((T_{Y_*}(n) + T_{\mathrm{Split}}(n)) \log n)$ *time.*

In the forthcoming discussion of three scheduling applications of the results of this section, we pay special attention to designing fast algorithms that could find the required set Y_* in all levels of the recursive Procedure DECOMP. We develop fast algorithms that compute the value $\widetilde{\varphi}(\hat{H})$ and find a set Y_* in accordance with its definition; see Sect. 5.

4.4 Comparison with decomposition algorithm for maximizing a concave separable function

In this subsection, we refer to our decomposition algorithm for Problem (LP) defined over a submodular polyhedron intersected with a box as Algorithm SSS-Decomp. Below, we compare that algorithm with a known decomposition algorithm that is applicable for maximizing a separable concave function over a submodular polyhedron; see [3], [4, Sect. 8.2] and [6].

Consider the problem of maximizing a separable concave function over a submodular polyhedron:

$$\text{(SCFM)} \quad \begin{array}{ll} \text{Maximize} & \displaystyle\sum_{j \in N} f_j(p(j)) \\ \text{subject to} & p(X) \le \varphi(X), \quad X \in 2^N, \end{array}$$

where $f_j : \mathbb{R} \to \mathbb{R}$ is a univariate concave function for $j \in N$ and $\varphi : 2^N \to \mathbb{R}$ is a submodular function with $\varphi(\emptyset) = 0$.

The decomposition algorithm for Problem (SCFM) was first proposed by Fujishige [3] for the special case where each f_j is quadratic and φ is a polymatroid rank function. Groenevelt [6] then generalized the decomposition algorithm for the case where each f_j is a general concave function and φ is a polymatroid rank function. Later, it was pointed out by Fujishige [4, Sect. 8.2] that the decomposition algorithm in [6] can be further generalized to the case where φ is a general submodular function. We refer to that algorithm as Algorithm FG-Decomp.

For simplicity of presentation, in the description of Algorithm FG-Decomp we assume that each f_j is monotone increasing; the general case with non-monotone f_j can be dealt with by an appropriate modification of the algorithm; see [6].

Algorithm FG-Decomp

Step 1. Find an optimal solution $\mathbf{q} \in \mathbb{R}^N$ of the following "relaxed" problem with a single constraint:

$$\begin{array}{ll} \text{Maximize} & \displaystyle\sum_{j \in N} f_j(p(j)) \\ \text{subject to} & p(N) \le \varphi(N). \end{array}$$

Note: since f_j is monotone it follows that $q(N) = \varphi(N)$.

Step 2. Find a maximal vector $\mathbf{q}' \in \mathbb{R}^N$ satisfying the following condition:

$$q'(X) \le \varphi(X), \; X \in 2^N, \qquad q'(j) \le q(j), \; j \in N.$$

Step 3. Find a (unique) maximal set $Y_* \subseteq N$ such that $\varphi(Y_*) = q'(Y_*)$.

Step 4. If $Y_* = N$, then output the vector $\mathbf{q}'$ and stop. Otherwise, go to Step 5.
Step 5. Find an optimal solution $\mathbf{p_1} \in \mathbb{R}^{Y^*}$ of the following problem:

$$\begin{array}{ll} \text{Maximize} & \sum_{j \in Y_*} f_j(p(j)) \\ \text{subject to} & p(X) \leq \varphi(X), X \in 2^{Y_*}. \end{array}$$

Step 6. Find an optimal solution $\mathbf{p_2} \in \mathbb{R}^{N \setminus Y^*}$ of the following problem:

$$\begin{array}{ll} \text{Maximize} & \sum_{j \in N \setminus Y_*} f_j(p(j)) \\ \text{subject to} & p(X) \leq \varphi(X \cup Y_*) - \varphi(Y_*), \quad X \in 2^{N \setminus Y_*}. \end{array}$$

Step 7. Output the direct sum $\mathbf{p}^* = \mathbf{p_1} \oplus \mathbf{p_2} \in \mathbb{R}^N$ and stop.

Notice that for the set Y_* chosen in Step 3, there exists some optimal solution $\mathbf{p}^*$ of Problem (SCFM) such that $\varphi(Y_*) = p^*(Y_*)$; see [4, Sect. 8.2], [6].

It is easy to see that Problem (LP) can be reduced to Problem (SCFM) by setting the functions f_j as

$$f_j(\alpha) = \begin{cases} w(j)\underline{p}(j) + M(\alpha - \underline{p}(j)), & \text{if } \alpha < \underline{p}(j); \\ w(j)\alpha, & \text{if } \underline{p}(j) \leq \alpha \leq \overline{p}(j); \\ w(j)\overline{p}(j) - M(\alpha - \overline{p}(j)), & \text{if } \alpha > \overline{p}(j) \end{cases} \tag{27}$$

with a sufficiently large positive number M. Thus, Algorithm FG-Decomp (appropriately adjusted to deal with non-monotone functions f_j) can be applied to solving Problem (LP).

For Problem (LP), Algorithm FG-Decomp is quite similar to Algorithm SSS-Decomp. Indeed, both algorithms recursively find a set Y_* and decompose a problem into two subproblems by using restriction to Y_* and contraction by Y_*.

The difference of the two decomposition algorithms is in the selection rule of a set Y_*. In fact, a numerical example can be provided that demonstrates that for the same instance of Problem (LP) the two decomposition algorithms may find different sets Y_* in the same iteration.

In addition, Algorithm SSS-Decomp fixes some variables in the subproblems so that the number of non-fixed variables in each subproblem is at most the half of the non-fixed variables in the original problem; this is an important feature of our algorithm which is not enjoyed by Algorithm FG-Decomp. This difference affects the efficiency of the two decomposition algorithms; indeed, for Problem (LP) the height of the decomposition tree can be $\Theta(n)$ if Algorithm FG-Decomp is used, while it is $O(\log n)$ in our Algorithm SSS-Decomp.

Thus, despite certain similarity between the two decomposition algorithms, our algorithm cannot be seen as a straightforward adaptation of Algorithm FG-Decomp designed for solving problems of non-linear optimization with submodular constraints to a less general problem of linear programming.

On the other hand, assume that the feasible region for Problem (SCFM) is additionally restricted by imposing the box constraints, similar to those used in Problem (LP). Theorem 1 can be used to reduce the resulting problem to Problem (SCFM) with a feasible region being the base polyhedron with a modified rank function. Although the obtained problem can be solved by Algorithm FG-Decomp, this approach is computationally inefficient, since it requires multiple calls to a procedure for minimizing a submodular function. It is more efficient not to rely on Theorem 1, but to handle the additional box constraints by adapting the objective function, similarly to (27), and then to use Algorithm FG-Decomp.

5 Application to parallel machine scheduling problems

In this section, we show how the decomposition algorithm based on Procedure DECOMP can be adapted for solving problems with parallel machines efficiently. Before considering implementation details that are individual for each scheduling problem under consideration, we start this section with a discussion that addresses the matters that are common to all three problems.

Recall that each scheduling problem we study in this paper can be formulated as Problem (LP) of the form (4) with an appropriate rank function. Thus, each of these problems can be solved by the decomposition algorithm described in Sect. 4.2 applied to Problem LP$(N, \emptyset, \emptyset, \mathbf{l}, \mathbf{u})$, where $\mathbf{l} = \underline{\mathbf{p}}$ and $\mathbf{u} = \overline{\mathbf{p}}$.

For an initial Problem LP$(N, \emptyset, \emptyset, \mathbf{l}, \mathbf{u})$, we assume that the following preprocessing is done before calling Procedure DECOMP$(N, \emptyset, \emptyset, \mathbf{l}, \mathbf{u})$:

1. If required, the jobs are numbered in non-decreasing order of their release dates in accordance with (9).
2. If required, 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 (10).
3. 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.

The required preprocessing takes $O(n \log n)$ time.

To adapt the generic Procedure DECOMP to solving a particular scheduling problem, we only need to provide the implementation details for Procedure DECOMP$(H, F, K, \mathbf{l}, \mathbf{u})$ that emerges at a certain level of recursion. To be precise, we need to explain how to compute for each particular problem the function $\widetilde{\varphi}_K^H(X)$ for a chosen set $X \in 2^H$ and how to find for a current heavy-element set an instrumental set Y_* defined by (22), which determines the pair of problems into which the current problem is decomposed.

Given Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (21) define $h = |H|$ and $k = |K|$. Recall that $K, H \subseteq N$ are sets with $K \cap H = \emptyset$. For $v = 0, 1, \dots, h$, define

$$\mathcal{H}_v = \{Y \subseteq H \mid |H| = v\} \tag{28}$$

Introduce

$$\hat{h} = \min\{h, m-k-1\}. \tag{29}$$

Since $\varphi_K^H(Y) = \varphi(Y \cup K) - \varphi(K)$ for $Y \in 2^H$, it follows that for a given set $X \subseteq H$ the function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ can be computed as follows:

$$\begin{aligned}\widetilde{\varphi}_K^H(X) &= \min_{Y \in 2^H}\left\{\varphi_K^H(Y) + u(X\backslash Y) - l(Y\backslash X)\right\}\\ &= u(X) - \varphi(K) + \min_{Y \in 2^H}\{\varphi(Y \cup K) - u(Y \cap X) - l(Y\backslash X)\}\\ &= u(X) - \varphi(K) + \min_{Y \in 2^H}\{\varphi(Y \cup K) - \lambda(Y)\},\end{aligned} \tag{30}$$

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

$$\lambda(j) = \begin{cases} u(j), & \text{if } j \in X,\\ l(j), & \text{if } j \in H\backslash X.\end{cases} \tag{31}$$

Notice that if the minimum in the left-hand side of (30) is achieved for $Y = Y_*$, then Y_* is an instrumental set for set X.

5.1 Uniform machines, equal release dates

In this subsection, we show that problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ can be solved in $O(n \log n)$ time by the decomposition algorithm. To achieve this, we consider Problem LP(H, F, K, $\mathbf{l}$, $\mathbf{u}$) that arises at some level of recursion of Procedure Decomp and present a procedure for computing the function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ given by (22). We show that for an arbitrary set $X \subseteq H$ the value $\widetilde{\varphi}_K^H(X)$ can be computed in $O(h)$ time. For a heavy-element set $\hat{H} \subseteq H\backslash F$, finding a set Y_* that is instrumental for set $\hat{H}$ also requires $O(h)$ time.

Recall that for problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ the rank function $\varphi : 2^N \to \mathbb{R}$ is defined by (11), i.e.,

$$\varphi(X) = dS_{\min\{m,|X|\}}, \qquad X \in 2^N.$$

This, together with (30), implies

$$\widetilde{\varphi}_K^H(X) = u(X) - dS_{\min\{m,k\}} + \min_{Y \in 2^H}\left\{dS_{\min\{m,|Y|+k\}} - \lambda(Y)\right\}. \tag{32}$$

The computation of the minimum in the last term in (32) is done differently for the sets $Y \subseteq H$ with $|Y| \le \hat{h}$ and with $|Y| > \hat{h}$, where $\hat{h}$ is defined by (29), provided that the corresponding sets exist. With $\mathcal{H}_v$, $0 \le v \le h$, defined by (28), introduce

$$\Phi' = \begin{cases} \min\limits_{0 \le v \le \hat{h}} \left\{dS_{v+k} - \max\limits_{Y \in \mathcal{H}_v} \lambda(Y)\right\}, & \text{if } m > k, \\ +\infty, & \text{if } m \le k, \end{cases} \tag{33}$$

and

$$\Phi'' = \begin{cases} dS_m - \max\{\lambda(Y) \mid Y \in 2^H,\ |Y| > \hat{h}\}, & \text{if } h > m - k - 1, \\ +\infty, & \text{if } h \le m - k - 1. \end{cases} \tag{34}$$

Then, we can rewrite the last term in (32) as

$$\min_{Y \in 2^H} \{dS_{\min\{m,|Y|+k\}} - \lambda(Y)\} = \min\left\{\Phi', \Phi''\right\}.$$

Notice that $\Phi' = +\infty$ corresponds to the case that the set $Y \in \mathcal{H}_v$ does not exist for $0 \le v \le \hat{h}$ (this happens if $m \le k$ or equivalently $\hat{h} < 0$); $\Phi'' = +\infty$ corresponds to the case that the set $Y \in \mathcal{H}_v$ does not exist for $v > \hat{h}$ (this happens if $h \le m - k - 1$ or equivalently $\hat{h} = h$).

Assume $m > k$, and let λ_v be the v-th largest value in the list $(\lambda(j) \mid j \in H)$ for $v = 1, 2, \dots, \hat{h}$. It follows that

$$\Phi' = \min_{0 \le v \le \hat{h}} \left\{dS_{v+k} - \sum_{i=1}^{v} \lambda_i\right\}. \tag{35}$$

We then assume $h > m - k - 1$. Since $\lambda(j) \ge 0$ for $j \in H$, the maximum in the right-hand side of the top line of (34) is achieved for $Y = H$, i.e.,

$$\Phi'' = dS_m - \lambda(H). \tag{36}$$

Below we describe the procedure that uses Eqs. (35) and (36) for computing the values Φ' and Φ''. Since the procedure will be used as a subroutine within the recursive Procedure DECOMP, here we present it for computing $\widetilde{\varphi}_K^H(X)$ with X being a heavy-element set $\hat{H}$. Besides, its output contains set Y_*, an instrumental set for set $\hat{H}$.

Procedure CompQr0

INPUT: Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$, a heavy-element set $\hat{H} \subseteq H \backslash F$, the values of h, k and $\hat{h}$ defined by (29), and the list $(\lambda(j) \mid j \in H)$ computed by (31) with respect to $X = \hat{H}$.
OUTPUT: the value of function $\widetilde{\varphi}_K^H(X)$ and an instrumental set Y_* for set $X = \hat{H}$.

Step 1. If $k \ge m$, then set $\Phi' := +\infty$ and go to Step 3.
Step 2. Do the following:
  Step 2-1. For $v = 1, 2, \dots, \hat{h}$, compute the v-th largest value λ_v among the numbers $\lambda(j)$, $j \in H$.
  Step 2-2. Compute the value Φ' by using (35). If $\Phi' = dS_{v+k} - \sum_{i=1}^{v} \lambda_i$ for some v, $0 \le v \le \hat{h}$, then define Y' to be the set of jobs in H that correspond to the values $\lambda_1, \lambda_2, \dots, \lambda_v$.

Step 3. If $h + k < m$, then set $\Phi'' := +\infty$; otherwise, set $\Phi'' := dS_m - \lambda(H)$ and $Y'' = H$.

Step 4. Compute the value $\widetilde{\varphi}_K^H(X) = u(X) - dS_{\min\{m,k\}} + \min\left\{\Phi', \Phi''\right\}$, applied to $X = \hat{H}$. If $\Phi' < \Phi''$, define $Y_* := Y'$; otherwise, define $Y_* := Y''$.

Let us analyze the time complexity of Procedure CompQr0. In Step 2, the values $\lambda_1, \lambda_2, \ldots, \lambda_{\hat{h}}$ can be found in $O(h)$ time by using the list $(\lambda(j) \mid j \in H)$, so that the value Φ' and set Y' can be computed in $O(h)$ time. It is easy to see that Φ'' and Y'' can be obtained in $O(h)$ time as well. Hence, the value $\widetilde{\varphi}_K^H(X)$ and set Y_* can be found in $O(h)$ time.

Theorem 5 *Problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ can be solved either in $O(n \log n)$ time or in $O(n + m \log m \log n)$ time.*

Proof Here, we only present the proof of the running time $O(n \log n)$, that is derived if in each level of recursion of Procedure Decomp we use Procedure CompQr0; the proof of the running time $O(n + m \log m \log n)$ is given in "Appendix".

As proved above, Procedure CompQr0 applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ takes $O(h)$ time. In terms of Theorem 4 on the running time of Procedure Decomp, this implies that $T_{Y_*}(h) = O(h)$.

In the analysis of the time complexity of Procedure CompQr0, we assume that certain information is given as part of the input. This assumption can be satisfied by an appropriate preprocessing. In particular, when we decompose a problem with a set of job H at a certain level of recursion into two subproblems, we may create the sorted lists $(u(j) \mid j \in H)$ and $(l(j) \mid j \in H)$. This can be done in $O(h)$ time, since the sorted lists $(u(j) \mid j \in N)$ and $(l(j) \mid j \in N)$ are available as a result of the initial preprocessing. Thus, we have that $T_{\text{Split}}(h) = O(h)$. Hence, the theorem follows from Theorem 4. □

5.2 Identical machines, different release dates

In this subsection, we show that problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ can be solved in $O(n \log m \log n)$ time by the decomposition algorithm. To achieve this, we consider Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ that arises at some level of recursion of Procedure Decomp and present a procedure for computing the function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ given by (22). We show that for an arbitrary set $X \subseteq H$ the value $\widetilde{\varphi}_K^H(X)$ can be computed in $O(h \log m)$ time. For a heavy-element set $\hat{H} \subseteq H \backslash F$, finding a set Y_* that is instrumental for set $\hat{H}$ also requires $O(h \log m)$ time.

Recall that for problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ the rank function $\varphi : 2^N \to \mathbb{R}$ is defined by (13), i.e.,

$$\varphi(X) = d \cdot \min\{m, |X|\} - \sum_{i=1}^{\min\{m,|X|\}} r_i(X), \qquad X \in 2^N,$$

where $r_i(X)$ denotes the i-th smallest release dates among the jobs of set X.
This, together with (30), implies that

$$\widetilde{\varphi}_K^H(X) = u(X) - \left(d \cdot \min\{m, k\} - \sum_{i=1}^{\min\{m,k\}} r_i(K) \right) + \min_{Y \in 2^H} \Bigg\{ d \cdot \min\{m, |Y| + k\} - \sum_{i=1}^{\min\{m,|Y|+k\}} r_i(Y \cup K) - \lambda(Y) \Bigg\}, \tag{37}$$

where $\lambda(j),\ j \in H$, are given by (31).

Let $\hat{h}$ be defined by (29). Computation of the minimum in the last term in (37) is done differently for sets $Y \subseteq H$ with $|Y| \le \hat{h}$ and $|Y| > \hat{h}$. With $\mathcal{H}_v$, $0 \le v \le h$, defined by (28), introduce

$$\Phi' = \begin{cases} \min\limits_{0 \le v \le \hat{h}} \left\{ d \cdot (v + k) - \max\limits_{Y \in \mathcal{H}_v} \left\{ \sum\limits_{i=1}^{v+k} r_i(Y \cup K) + \lambda(Y) \right\} \right\}, & \text{if } m > k, \\ +\infty, & \text{if } m \le k, \end{cases} \tag{38}$$

and

$$\Phi'' = \begin{cases} dm - \max \left\{ \sum\limits_{i=1}^{m} r_i(Y \cup K) + \lambda(Y) \;\middle|\; Y \in 2^H,\ |Y| > \hat{h} \right\}, & \\ \qquad \text{if } h > m - k - 1, & \\ +\infty, \qquad \text{if } h \le m - k - 1. & \end{cases} \tag{39}$$

Similarly to Sect. 5.1, the values Φ' and Φ'' are responsible for computing the minimum in the last term in (37) over the sets $Y \subseteq H$ with $|Y| \le \hat{h}$ and with $|Y| > \hat{h}$, respectively, provided that the corresponding sets exist. Thus, (37) can be rewritten as

$$\widetilde{\varphi}_K^H(X) = u(X) - \left(d \cdot \min\{m, k\} - \sum_{i=1}^{\min\{m,k\}} r_i(K) \right) + \min\left\{\Phi', \Phi''\right\}. \tag{40}$$

We now explain how to compute the values Φ' and Φ''. From the list $(\widetilde{\lambda}(j) \mid j \in H)$, where

$$\widetilde{\lambda}(j) = r(j) + \lambda(j), \qquad j \in H. \tag{41}$$

Suppose that $m > k$. Computing of Φ' can be done in a similar manner as in Sect. 5.1. The top line of the formula (38) can be rewritten as

$$\begin{aligned}\Phi' &= \min_{0\le v\le \hat{h}} \left\{ d\cdot(v+k) - \max_{Y\in\mathcal{H}_v} \left\{ r(Y) + r(K) + \lambda(Y) \right\} \right\} \\ &= -r(K) + \min_{0\le v\le \hat{h}} \left\{ d\cdot(v+k) - \max_{Y\in\mathcal{H}_v} \widetilde{\lambda}(Y) \right\}.\end{aligned}$$

For $v,\ 1\le v\le \hat{h}$, let $\widetilde{\lambda}_v$ be the v-th largest value among the numbers $\widetilde{\lambda}(j),\ j\in H$. Then, we have

$$\Phi' = -r(K) + \min_{0\le v\le \hat{h}} \left\{ d\cdot(v+k) - \sum_{i=1}^{v} \widetilde{\lambda}_i \right\}. \tag{42}$$

We now turn to computing the value Φ''. We may assume $\hat{h} < h$, i.e., $h > m-k-1$, since otherwise $\Phi'' = +\infty$. For simplicity of the description, we assume, without loss of generality, that the jobs of set $H\cup K$ are renumbered in such a way that

$$H\cup K = \{1,2,\ldots,h+k\}, \qquad r(1)\le r(2)\le \cdots \le r(h+k). \tag{43}$$

For $t = m, m+1, \ldots, h+k$, introduce

$$\begin{aligned}K[t] &= \{j\in K \mid j\le t\},\\ \mathcal{H}^z[t] &= \left\{ Y\in 2^H \mid Y\subseteq \{1,2,\ldots,t\},\ |Y| + |K[t]| = z \right\}, \quad |K[t]|\le z\le m.\end{aligned} \tag{44}$$

We define $\bar{t}$ to be the minimum t with $|K[t]| = m$ if $k\ge m$; otherwise, let $\bar{t} = h+k$. Note that $\bar{t}\ge m$, and $\mathcal{H}^m[t]\ne\emptyset$ if $m\le t\le \bar{t}$.

The following lemma is useful for computing the value Φ'' efficiently.

Lemma 4 *Let $Y''\in 2^H$ be a set satisfying $|Y''| > \hat{h}$ and*

$$\sum_{i=1}^{m} r_i(Y''\cup K) + \lambda(Y'') = \max\left\{ \sum_{i=1}^{m} r_i(Y\cup K) + \lambda(Y) \;\middle|\; Y\in 2^H,\ |Y| > \hat{h} \right\}. \tag{45}$$

Let $t_*\in H\cup K$ be a job such that $m\le t_*\le \bar{t}$ and the set $\{j\in Y''\cup K \mid j\le t_*\}$ contains exactly m elements. Define the sets $Y''_1 = \{j\in Y'' \mid j\le t_*\}$ and $Y''_2 = \{j\in Y'' \mid j > t_*\}$. Then the following properties hold:

$$\begin{aligned}&\text{(i)} \quad \sum_{i=1}^{m} r_i(Y''\cup K) + \lambda(Y'') = \widetilde{\lambda}(Y''_1) + r(K[t_*]) + \lambda(Y''_2),\\ &\text{(ii)} \quad Y''_1\in\mathcal{H}^m[t_*] \textit{ and } \widetilde{\lambda}(Y''_1) = \max\{\widetilde{\lambda}(Y) \mid Y\in\mathcal{H}^m[t_*]\},\\ &\text{(iii)} \quad Y''_2 = \{j\in H \mid j > t_*\}.\end{aligned}$$

Proof First, notice that set $Y'' \cup K$ contains at least $\hat{h} + 1 + k \geq m$ jobs, so that job t_* exists and $m \leq t_* \leq h + k$. Notice that job t_* might belong to set $H \backslash Y''$, and that job t_* is not necessarily unique. Indeed, if, e.g., job $t_* + 1 \in H \backslash Y''$, then $\{j \in Y'' \cup K \mid j \leq t_*\} = \{j \in Y'' \cup K \mid j \leq t_* + 1\}$.

We need to show that there exists a t_* that satisfies $t_* \leq \bar{t}$. To prove this, we only need to consider the case that $k \geq m$, since otherwise by definition $\bar{t} = h + k$. For $k \geq m$, let t_* be the smallest value of t such that the equality $|\{j \in Y'' \cup K \mid j \leq t\}| = m$ holds. Since $|\{j \in K \mid j \leq t_*\}| \leq m$, we have $t_* \leq \bar{t}$ by the definition of $\bar{t}$.

Take a t_* that satisfies the lemma conditions. For an arbitrarily chosen set $Z_1 \in \mathcal{H}^m[t_*]$, define set $Z \in 2^H$ as $Z = Z_1 \cup Y_2''$. Notice that $\{j \in Z \cup K \mid j \leq t_*\} = Z_1 \cup K[t_*]$. This implies

$$\begin{aligned}\sum_{i=1}^{m} r_i(Z \cup K) + \lambda(Z) &= r(Z_1) + r(K[t_*]) + \lambda(Z_1) + \lambda(Y_2'') \\ &= \widetilde{\lambda}(Z_1) + r(K[t_*]) + \lambda(Y_2''). \end{aligned} \tag{46}$$

Since $\{j \in Y'' \cup K \mid j \leq t_*\} = Y_1'' \cup K[t_*]$ and $|\{j \in Y'' \cup K \mid j \leq t_*\}| = m$, we have $Y_1'' \in \mathcal{H}^m[t_*]$. Applying (46) with $Z_1 = Y_1''$, we obtain

$$\sum_{i=1}^{m} r_i(Y'' \cup K) + \lambda(Y'') = \widetilde{\lambda}(Y_1'') + r(K[t_*]) + \lambda(Y_2''),$$

i.e., property (i) holds.

Since the maximum in (45) is achieved for $Y = Y''$, the inequality

$$\sum_{i=1}^{m} r_i(Y'' \cup K) + \lambda(Y'') \geq \sum_{i=1}^{m} r_i(Z \cup K) + \lambda(Z)$$

holds for any set $Z = Z_1 \cup Y_2''$ with $Z_1 \in \mathcal{H}^m[t_*]$. Then (46) and property (i) imply that $\widetilde{\lambda}(Y_1'') \geq \widetilde{\lambda}(Z_1)$. Hence, property (ii) holds.

Since $\lambda(j) \geq 0$ for $j \in H$, we should include all jobs $j \in H$ with $j > t_*$ into set Y_2'' to achieve the maximum in (45), i.e., property (iii) holds. □

For each t, $m \leq t \leq \bar{t}$, define

$$\eta_1[t] = \max_{Y \in \mathcal{H}^m[t]} \widetilde{\lambda}(Y), \qquad \rho[t] = r(K[t]), \qquad \eta_2[t] = \sum_{j \in H,\, j > t} \lambda(j).$$

We see from Lemma 4 that

$$\Phi'' = dm - \max_{m \leq t \leq \bar{t}} \{\eta_1[t] + \rho[t] + \eta_2[t]\} \tag{47}$$

holds. We now show how to compute the values $\eta_1[t]$, $\rho[t]$, and $\eta_2[t]$ efficiently.

For $t = m$, define

$$Q_m = \{j \in H \mid j \le m\}. \tag{48}$$

Notice that

$$\max\{\widetilde{\lambda}(Y) \mid Y \in \mathcal{H}^m[m]\} = \max\left\{\widetilde{\lambda}(Y) \mid Y \subseteq Q_m,\ |Y| + |K[m]| = m\right\} = \widetilde{\lambda}(Q_m).$$

Thus, we have

$$\eta_1[m] = \widetilde{\lambda}(Q_m), \qquad \rho[m] = r(K[m]), \qquad \eta_2[m] = \sum_{j \in H,\ j > m} \lambda(j). \tag{49}$$

Lemma 5 *Let t be an integer with $m < t \le \bar{t}$.*

(i) *Given the values $\rho[t-1]$ and $\eta_2[t-1]$, $\rho[t]$ and $\eta_2[t]$ can be obtained as*

$$\rho[t] = \begin{cases} \rho[t-1], & \text{if } t \in H, \\ \rho[t-1] + r(t), & \text{if } t \in K, \end{cases} \qquad \eta_2[t] = \begin{cases} \eta_2[t-1] - \lambda(t), & \text{if } t \in H, \\ \eta_2[t-1], & \text{if } t \in K. \end{cases} \tag{50}$$

(ii) *Given a set $Q \in \mathcal{H}^m[t-1]$ with $\eta_1[t-1] = \widetilde{\lambda}(Q)$, the value $\eta_1[t-1]$ and job $z \in Q$ such that $\widetilde{\lambda}(z) = \min_{j \in Q} \widetilde{\lambda}(j)$, the value $\eta_1[t]$ can be obtained as*

$$\eta_1[t] = \begin{cases} \eta_1[t-1], & \text{if } t \in H,\ \widetilde{\lambda}(z) \ge \widetilde{\lambda}(t), \\ \eta_1[t-1] - \widetilde{\lambda}(z) + \widetilde{\lambda}(t), & \text{if } t \in H,\ \widetilde{\lambda}(z) < \widetilde{\lambda}(t), \\ \eta_1[t-1] - \widetilde{\lambda}(z), & \text{if } t \in K. \end{cases} \tag{51}$$

Proof We have $K[t] = K[t-1]$ if $t \in H$ and $K[t] = K[t-1] \cup \{t\}$ if $t \in K$. Hence, the first equation in (50) follows. The second equation in (50) is immediate from the definition of η_2. The Eq. (51) follows from the observation that $\eta_1[t]$ is equal to the sum of $m - |K[t]|$ largest numbers in the list $\left(\widetilde{\lambda}(j) \mid j \in H,\ j \le t\right)$. □

Below we describe the procedure that uses Eqs. (42) and (47) for computing the values Φ' and Φ''. As in Sect. 5.1, the procedure outputs $\widetilde{\varphi}_K^H(X)$ for $X = \hat{H}$ and an instrumental set Y_* for set $\hat{H}$.

Procedure CompPrj

Input: Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$, a heavy-element set $\hat{H} \subseteq H \backslash F$, the values of h, k and $\hat{h}$ defined by (29), the lists $(\lambda(j) \mid j \in H)$ and $\left(\widetilde{\lambda}(j) \mid j \in H\right)$ computed by (31) with respect to $X = \hat{H}$ and by (41), and the non-decreasing lists $(r(j) \mid j \in H)$ and $(r(j) \mid j \in K)$.
Output: the value of function $\widetilde{\varphi}_K^H(X)$ and an instrumental set Y_* for set $X = \hat{H}$.

Step 1. If $k \ge m$, then set $\Phi' := +\infty$ and go to Step 3.

Step 2. Do the following:

Step 2-1. For $v = 1, 2, \ldots, \hat{h}$, find the v-th largest value $\widetilde{\lambda}_v$ in the list $\big(\widetilde{\lambda}(j) \mid j \in H\big)$.

Step 2-2. Compute the value Φ' by using (42). If $\Phi' = -r(K) + d \cdot (v + k) - \sum_{i=1}^{v} \widetilde{\lambda}_i$ for some v, $0 \leq v \leq \hat{h}$, then define Y' to be the set of jobs in H that correspond to the values $\widetilde{\lambda}_1, \widetilde{\lambda}_2, \ldots, \widetilde{\lambda}_v$.

Step 3. If $h + k < m$ then set $\Phi'' := +\infty$ and go to Step 5.

Step 4. Do the following:

Step 4-1. Find a non-decreasing list $(r(j) \mid j \in H \cup K)$ and renumber the jobs in $H \cup K$ so that they satisfy (43). Compute $\bar{t}$.

Step 4-2. Define $\rho[m]$ and $\eta_2[m]$ in accordance with (49). For $t = m + 1, m + 2, \ldots, \bar{t}$, compute $\rho[t]$ and $\eta_2[t]$ by (50).

Step 4-3. Set $Q := Q_m$, where Q_m is given by (48) and define $\eta_1[m]$ by (49). For $t = m + 1, m + 2, \ldots, \bar{t}$ do:

```
  Find $z \in Q$ such that $\widetilde{\lambda}(z) = \min\{\widetilde{\lambda}(j) \mid j \in Q\}$.
  Case 1: $t \in H$ and $\widetilde{\lambda}(z) < \widetilde{\lambda}(t)$ .
          Set $Q := (Q \backslash \{z\}) \cup \{t\}$ and $\eta_1[t] := \eta_1[t-1] - \widetilde{\lambda}(z) + \widetilde{\lambda}(t)$.
  Case 2: $t \in H$ and $\widetilde{\lambda}(z) \geq \widetilde{\lambda}(t)$.
          Set $\eta_1[t] := \eta_1[t-1]$.
  Case 3: $t \in K$.
          Set $Q := Q \backslash \{z\}$ and $\eta_1[t] := \eta_1[t-1] - \widetilde{\lambda}(z)$.
```

Step 4-4. Find the integer t_* such that

$$\eta_1[t_*] + \rho[t_*] + \eta_2[t_*] = \max_{m \leq t \leq \bar{t}} \{\eta_1[t] + \rho[t] + \eta_2[t]\}.$$

Compute the value Φ'' by using (47).

Step 4-5. Perform Step 4-3 again, breaking the loop after t exceeds t_*, i.e., after the value $\eta_1[t_*]$ is computed. With the found set Q, define the sets $Y''_1 := Q$, $Y''_2 := \{j \in H \mid j > t_*\}$ and $Y'' := Y''_1 \cup Y''_2$.

Step 5. Compute the value $\widetilde{\varphi}^H_K(X)$ by (40) applied to $X = \hat{H}$. If $\Phi' < \Phi''$, define $Y_* := Y'$; otherwise, define $Y_* := Y''$.

Now we analyze the running time of this procedure. In Steps 1 and 2 we compute the value Φ' and find set Y'. Step 1 can be done in constant time. Step 2-1 can be done by selecting $\hat{h}$ largest numbers in the list $(\widetilde{\lambda}(j) \mid j \in H)$ in $O(h)$ time and then sorting them in $O(\hat{h} \log \hat{h})$ time. Since Step 2-2 can be done in $O(k + \hat{h})$ time, Step 2 requires $O(k + h + \hat{h} \log \hat{h}) = O(k + h \log \hat{h}) = O(k + h \log m)$ time in total.

In Steps 3 and 4 we compute the value Φ'' and find set Y''. Step 3 can be also done in constant time. We assume that both $(r(j) \mid j \in H)$ and $(r(j) \mid j \in K)$ are given as sorted lists; this can be easily satisfied by appropriate preprocessing. Then, Step 4-1 can be done in $O(h + k)$ time by using merge sort. Step 4-2 can be done in $O(h + k)$ time. In Step 4-3, we implement Q as a heap for computational efficiency. Initially $Q = Q_m$ consists of at most m elements, and to initialize the heap Q takes $O(h + m \log m)$ time. The number of elements in the heap does not increase, so that each iteration in Step 4-3 can be done in $O(\log m)$ time, which implies that Step 4-3 requires $O((h+k)\log m)$ time. Step 4-4 can be done in $O(h+k)$ time. Step 4-5 is needed for finding the set Y'' and is implemented as a partial rerun of Step 4-3 in $O((h+k)\log m)$ time.

Finally, we compute the value $\widetilde{\varphi}_K^H(X)$ in Step 5. We may assume that the value $u(X)$ in Step 5 is given in advance. The value $\sum_{i=1}^{\min\{m,k\}} r_i(K)$ can be computed in $O(k)$ time, since a sorted list $(r(j) \mid j \in K)$ is available. Hence, Step 5 can be done in $O(k)$ time. In total, Procedure CompPrj requires $O((h+k)\log m)$ time. In particular, the procedure runs in $O(h \log m)$ time if $h \geq k$.

In the rest of this subsection, we show that a slightly modified version of Procedure CompPrj can also be run in $O(h \log m)$ time for $h < k$.

First, consider the case that $h \geq m$. Then, we have $k > h \geq m$. Let K_m be a set of m jobs in K with m smallest release dates. It is easy to see that the jobs in $K \backslash K_m$ do not affect the values $r_i(K)$ and $r_i(Y \cup K)$, i.e., it holds that

$$r_i(K) = r_i(K_m), \quad r_i(Y \cup K) = r_i(Y \cup K_m), \qquad i = 1, 2, \ldots, m, \quad Y \in 2^H.$$

It follows that in the formula (37) for $\widetilde{\varphi}_K^H(X)$, the value in the right-hand side remains the same even if we replace K and k with K_m and m, respectively. Making the same replacement in Procedure CompPrj, we deduce that it will run in $O((h+m)\log m) = O(h \log m)$ time, provided that set K_m is given in advance.

We finally consider the case that $h < m$. From the discussion above, we may assume that $k \leq m$. For any $Y \in 2^H$, the contribution of the release dates into the right-hand side of (37) is equal to $\sum_{i=1}^{k} r_i(K) - \sum_{i=1}^{\min\{m,|Y|+k\}} r_i(Y \cup K)$. Let $k' = m - h$ and K' be the set of jobs in K with k' smallest release dates among $r(j),\ j \in K$. Since $|Y| \leq h < m$, each of the values $r(j),\ j \in K'$, contributes to the sum $\sum_{i=1}^{\min\{m,|Y|+k\}} r_i(Y \cup K)$. Hence, it follows that

$$\begin{aligned}\sum_{i=1}^{k} r_i(K) - \sum_{i=1}^{\min\{m,|Y|+k\}} r_i(Y \cup K) = &\sum_{i=1}^{k-k'} r_i\left(K \backslash K'\right) \\ &- \sum_{i=1}^{\min\{m,|Y|+(k-k')\}} r_i\left(Y \cup (K \backslash K')\right).\end{aligned}$$

Thus, in formula (37), the value in the right-hand side remains the same if we replace K and k with $K \backslash K'$ and $k - k'$, respectively. Making the same replacement in Procedure CompPrj, we deduce that it will run in $O((h+k-k')\log m)$ time, provided that the set $K \backslash K'$ is given in advance. Since $k - k' = k - (m-h) \leq h$ holds for $k \leq m$, the running time of Procedure CompPrj is $O(h \log m)$.

We are now ready to prove the main statement regarding problem $P|r(j),\ p(j) = \overline{p}(j) - x(j), C(j) \leq d,\ pmtn|W$.

Theorem 6 *Problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$ can be solved in $O(n \log m \log n)$ time.*

Proof As proved above, Procedure CompPrj applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ takes $O(h \log m)$ time. In terms of Theorem 4 on the running time of Procedure Decomp, we have proved that $T_{Y_*}(h) = O(h \log m)$.

In the analysis of the time complexity of Procedure CompPrj, we assume that certain information is given as part of the input. This assumption can be satisfied by an appropriate preprocessing, when we decompose a problem at a certain level of recursion into two subproblems, based on the found set Y_*. It is not hard to see that this can be done in $O(h \log m)$ time, i.e., we have $T_{\text{Split}}(h) = O(h \log m)$. Hence, the theorem follows from Theorem 4. □

5.3 Uniform machines, different release dates

In this subsection, we show that problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ can be solved in $O(nm \log n)$ time by the decomposition algorithm. To achieve this, we consider Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ that arises at some level of recursion of Procedure Decomp and present a procedure for computing the function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ given by (22). We show that for an arbitrary set $X \subseteq H$ the value $\widetilde{\varphi}_K^H(X)$ can be computed in $O(hm)$ time. For a heavy-element set $\hat{H} \subseteq H \backslash F$, finding a set Y_* that is instrumental for set $\hat{H}$ also requires $O(hm)$ time.

Recall that for problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ the rank function $\varphi : 2^N \to \mathbb{R}$ is defined by (12), i.e.,

$$\varphi(X) = dS_{\min\{m,|X|\}} - \sum_{i=1}^{\min\{m,|X|\}} s_i r_i(X),$$

where $r_i(X)$ denotes the i-th smallest release dates among the jobs of set X. This, together with (30), implies that

$$\widetilde{\varphi}_K^H(X) = u(X) - \left(dS_{\min\{m,k\}} - \sum_{i=1}^{\min\{m,k\}} s_i r_i(K) \right) + \min_{Y \in 2^H} \left\{ \left(dS_{\min\{m,|Y|+k\}} - \sum_{i=1}^{\min\{m,|Y|+k\}} s_i r_i(Y \cup K) \right) - \lambda(Y) \right\}, \quad (52)$$

where $\lambda(j),\ j \in H$, are given by (31).

Let $\hat{h}$ be defined by (29). Computation of the minimum in the last term in (52) is done differently for sets $Y \subseteq H$ with $|Y| \le \hat{h}$ and $|Y| > \hat{h}$. With $\mathcal{H}_v$, $0 \le v \le h$, defined by (28), introduce

$$\Phi' = \begin{cases} \min\limits_{0 \le v \le \hat{h}} \left\{ dS_{v+k} - \max\limits_{Y \in \mathcal{H}_v} \left\{ \sum_{i=1}^{v+k} s_i r_i(Y \cup K) + \lambda(Y) \right\} \right\}, & \text{if } m > k, \\ +\infty, & \text{if } m \le k, \end{cases} \tag{53}$$

and

$$\Phi'' = \begin{cases} dS_m - \max \left\{ \sum_{i=1}^{m} s_i r_i(Y \cup K) + \lambda(Y) \;\middle|\; Y \in 2^H,\ |Y| > \hat{h} \right\}, \\ \qquad\qquad \text{if } h > m - k - 1, \\ +\infty, \qquad\quad \text{if } h \le m - k - 1. \end{cases} \tag{54}$$

Thus, (52) can be rewritten as

$$\widetilde{\varphi}_K^H(X) = u(X) - \left(dS_{\min\{m,k\}} - \sum_{i=1}^{\min\{m,k\}} s_i r_i(K) \right) + \min\left\{ \Phi', \Phi'' \right\}. \tag{55}$$

We explain how to compute the values Φ' and Φ''. As in Sect. 5.2, for simplicity of the description, we assume, without loss of generality, that the jobs are renumbered so that (43) holds.

In order to compute Φ', for v and t such that $0 \le v \le \hat{h}$ and $1 \le t \le h + k$, define

$$\begin{aligned} \mathcal{H}_v[t] &= \{ Y \in \mathcal{H}_v \mid Y \subseteq \{1, 2, \dots, t\} \}, \\ \xi_v[t] &= \max_{Y \in \mathcal{H}_v[t]} \left\{ \sum_{i=1}^{v+k} s_i r_i(Y \cup K) + \lambda(Y) \right\}, \end{aligned} \tag{56}$$

where $\xi_v[t]$ is set to $-\infty$ if $\mathcal{H}_v[t] = \emptyset$. Then, we have

$$\Phi' = \max_{0 \le v \le \hat{h}} \{ dS_{v+k} - \xi_v[h+k] \}. \tag{57}$$

Notice that all k jobs of set K and v jobs of set $Y \in \mathcal{H}_v[t]$ contribute into $\sum_{i=1}^{v+k} s_i r_i(Y \cup K)$. The required values $\xi_v[t]$ can be computed by a dynamic programming algorithm. Assume that for the current numbering of the jobs in $H \cup K$, the jobs in set K get the numbers $j_1, j_2, \dots, j_k$, so that $r(j_1) \le \cdots \le r(j_k)$.

For $v = 0$, notice that $\mathcal{H}_0[t] = \{\emptyset\}$, so that in accordance with (56) we compute

$$\xi_0[t] = \sum_{i=1}^{k} s_i r(j_i), \qquad t = 1, \dots, h + k. \tag{58}$$

If job 1 belongs to set H, then $\mathcal{H}_1[1] = \{\{1\}\}$; otherwise $\mathcal{H}_1[1] = \emptyset$. Besides, $\mathcal{H}_v[1] = \emptyset$ for each $v \ge 2$. Suppose that for some value of t, $1 \le t \le h + k$, the sets $\mathcal{H}_v[\tau]$ have been identified for all v and τ, $0 \le v \le \hat{h}$, $1 \le \tau \le t - 1$. Then

$$\mathcal{H}_v[t] = \begin{cases} \mathcal{H}_v[t-1] \cup \{Y \cup \{t\} \mid Y \in \mathcal{H}_{v-1}[t-1]\}, & \text{if } t \in H, \\ \mathcal{H}_v[t-1] & \text{if } t \in K. \end{cases} \tag{59}$$

Given a job $t \in H$, let us determine the position of job t relative to the jobs of set K. If $r(t) > r(j_k)$, then define $\ell_t = k+1$; otherwise, set ℓ_t to be equal to ℓ such that for job $j_\ell \in K$ we have that $j_{\ell-1} < t < j_\ell$. The values of ℓ_t can be found for all $t \in H$ in $O(h+k)$ time by scanning the sorted sequence of jobs of set $H \cup K$.

For some $t \in H$ and v, $1 \le v \le \hat{h}$, assume that we have found the value

$$\xi_{v-1}[t-1] = \sum_{i=1}^{v+k-1} s_i r_i \left(\bar{Y} \cup K\right) + \lambda(\bar{Y}),$$

where $\bar{Y} \in \mathcal{H}_{v-1}[t-1]$. Take $\ell = \ell_t$.

If $\ell = k+1$, then job t has the largest release date among the jobs of set $\bar{Y} \cup K \cup \{t\}$, so that

$$\begin{aligned} \xi_v[t] &= \max\{\xi_v[t-1], \xi_{v-1}[t-1] + s_{k+v} r(t) + \lambda(t)\} \\ &= \max\{\xi_v[t-1], \xi_{v-1}[t-1] + s_{\ell+v-1} r(t) + \lambda(t)\}. \end{aligned}$$

If $\ell \le k$, then among jobs $j \in \bar{Y} \cup K$ such that $j \le j_\ell$, there are $v-1$ jobs of set H and ℓ jobs of set K, i.e., job j_ℓ has the $(\ell+v-1)$ −th smallest release date in $\bar{Y} \cup K$. We deduce that the total contribution of the jobs $j_\ell, j_{\ell+1}, \dots, j_k$ into $\sum_{i=1}^{v+k-1} s_i r_i \left(\bar{Y} \cup K\right)$ is equal to

$$\beta(\ell, v-1) = \sum_{i=\ell}^{k} s_{v+i-1} r(j_i).$$

For computing $\xi_v[t]$, we need to find a set $\bar{Y}_+ \in \mathcal{H}_v[t]$ such that

$$\xi_v[t] = \sum_{i=1}^{v+k} s_i r_i \left(\bar{Y}_+ \cup K\right) + \lambda(\bar{Y}_+).$$

According to (59), if $\bar{Y}_+$ is sought in set $\mathcal{H}_v[t-1]$, then $\xi_v[t] = \xi_v[t-1]$. Otherwise, it is sought in the sets obtained from sets of $\mathcal{H}_{v-1}[t-1]$ by including job t. In the latter case, set $\bar{Y}_+$ can be found based on set $\bar{Y}$ and on those changes that are caused by the insertion of job t. As a result of this insertion, job t has the $(\ell+v-1)$ −th smallest release date in $\bar{Y} \cup K \cup \{t\}$, so that it will contribute $s_{\ell+v-1} r(t) + \lambda(t)$ into $\xi_v[t]$. Notice that all jobs of set K continue making contributions, since $v < m-k$. The new joint contribution of jobs $j_\ell, j_{\ell+1}, \dots, j_k$ becomes

$$\beta(\ell, v) = \sum_{i=\ell}^{k} s_{v+i} r(j_i).$$

Therefore, we deduce:

$$\xi_v[t] = \max\left\{\xi_v[t-1], \xi_{v-1}[t-1] + \beta(\ell, v) - \beta(\ell, v-1) + s_{\ell+v-1} r(t) + \lambda(t)\right\}. \tag{60}$$

All required partial sums $\beta(\ell, v)$ can be found at the preprocessing stage by computing

$$\beta(k+1, v) = 0, \qquad v = 0, \dots, \hat{h}, \tag{61}$$

followed by computing all $\beta(\ell, v)$ for $v,\ 0 \le v \le \hat{h}$ and $\ell,\ \ell = k-1, k-2, \dots, 1$ by

$$\beta(\ell, v) = \beta(\ell+1, v) + s_{v+\ell} r(j_\ell). \tag{62}$$

Notice that for $\ell = k+1$ both $\beta(\ell, v) = \beta(\ell, v-1) = 0$, so that the recursive formula (60) is valid for $\ell = k+1$ as well.

Applying (60) for $t,\ 1 \le t \le h+k$, and $v,\ 1 \le v \le m-k$ with the initial condition (58), we may find all values $\xi_v[t]$ needed for computing Φ' by (57).

We now consider the value Φ''. It is assumed that $\hat{h} < h$, i.e., $h + k \ge m$. Suppose that we know the set $Y'' \in 2^H$ such that $|Y''| > \hat{h}$ and

$$\sum_{i=1}^{m} s_i r_i(Y'' \cup K) + \lambda(Y'') = \max\left\{\sum_{i=1}^{m} s_i r_i(Y \cup K) + \lambda(Y) \;\middle|\; Y \in 2^H,\ |Y| > \hat{h}\right\}. \tag{63}$$

Similarly to Sect. 5.2, for $t,\ 1 \le t \le h+k$, introduce sets $K[t]$ and $\mathcal{H}^z[t]$ of the form (44). Let $t_* \in H \cup K$ be the job such that the set $\{j \in Y'' \cup K \mid j \le t_*\}$ contains exactly m elements. Since the jobs are numbered in non-decreasing order of the release dates, the set $\{j \in Y'' \cup K \mid j \le t_*\}$ contains the jobs in $Y'' \cup K$ with m smallest release dates.

Putting $Y_1'' = \{j \in Y'' \mid j \le t_*\} \in \mathcal{H}^m[t_*]$, we have

$$\sum_{i=1}^{m} s_i r_i\left(Y'' \cup K\right) = \sum_{i=1}^{m} s_i r_i\left(Y_1'' \cup K[t_*]\right).$$

Putting $Y_2'' = Y'' \backslash Y_1'' = \{j \in Y'' \mid j > t_*\}$, we have

$$\sum_{i=1}^{m} s_i r_i(Y'' \cup K) + \lambda(Y'') = \sum_{i=1}^{m} s_i r_i(Y_1'' \cup K[t_*]) + \lambda(Y_1'') + \lambda\left(Y_2''\right).$$

Thus, we should include all jobs $j \in H$ with $j > t_*$ into set Y_2'' to achieve the maximum in (63), i.e., we may assume $Y_2'' = \{j \in H \mid j > t_*\}$. We also have

$$\sum_{i=1}^{m} s_i r_i (Y_1'' \cup K) + \lambda(Y_1'') = \max_{Y \in \mathcal{H}^m[t_*]} \left\{ \sum_{i=1}^{m} s_i r_i (Y \cup K[t_*]) + \lambda(Y) \right\}.$$

For z and t, $1 \le z \le m$, $1 \le t \le h + k$, define

$$\zeta_z[t] = \begin{cases} \max\limits_{Y \in \mathcal{H}^z[t]} \left\{ \sum\limits_{i=1}^{z} s_i r_i (Y \cup K[t]) + \lambda(Y) \right\}, & \text{if } z \ge |K[t]|, \\ -\infty, & \text{otherwise.} \end{cases} \tag{64}$$

Provided that these values are known, we can compute Φ'' by

$$\Phi'' = dS_m - \max_{m \le t \le h+k} \left\{ \zeta_m[t] + \sum_{j \in H,\ j > t} \lambda(j) \right\}. \tag{65}$$

Notice that for a given t, $t \ge m$, the term $\sum_{j \in H,\ j>t} \lambda(j)$ is identical to $\eta_2[t]$ used in Sect. 5.2 and for its computation we can use the formulae (50) with the initial condition (49).

For convenience, define $\lambda(j) = 0$ for $j \in K$. The required values of $\zeta_z[t]$ can be found recursively by

$$\zeta_z[t] = \max \left\{ \zeta_z[t-1],\ \zeta_{z-1}[t-1] + s_z r(t) + \lambda(t) \right\},\ 1 \le z \le m,\ 1 \le t \le h + k \tag{66}$$

with the initial conditions

$$\zeta_0[t] = 0,\ 0 \le t \le h + k; \quad \zeta_z[0] = -\infty,\ 1 \le z \le m. \tag{67}$$

To see why the recursion (66) works, notice that if in the expression for $\zeta_z[t]$ job $t \in H$ does not belong to set Y that delivers the maximum in (64), then $\zeta_z[t] = \zeta_z[t-1]$. Otherwise, job $t \in H$, as the job with the largest release date, will be matched with the smallest multiplier s_z and will make an additional contribution of $\lambda(t)$, so that $\zeta_z[t] = \zeta_{z-1}[t-1] + s_z r(t) + \lambda(t)$. The latter situation also occurs if $t \in K$, since in this case $t \in K[t]$.

Now we are ready to present the procedure that outputs $\widetilde{\varphi}_K^H(X)$ for $X = \hat{H}$ and an instrumental set Y_* for set $\hat{H}$.

Procedure CompQrj

Input: Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$, a heavy-element set $\hat{H} \subseteq H \backslash F$, the values of h, k and $\hat{h}$ defined by (29), the values $\lambda(j)$, $j \in H$, computed by (31) with respect to $X = \hat{H}$, and the non-decreasing lists $(r(j) \mid j \in H)$ and $(r(j) \mid j \in K) = (r(j_1), \ldots, r(j_k))$.
Output: the value of function $\widetilde{\varphi}_K^H(X)$ and an instrumental set Y_* for set $X = \hat{H}$.

Step 1. If $k \ge m$, then set $\Phi' := +\infty$ and go to Step 8.

Step 2. Compute the values $\beta(\ell, v)$ by (61) and (62).
Step 3. For $v = 0$ compute the values of $\xi_0[t]$, $t = 1, \ldots, h + k$, by (58).
Step 4. Take $t = 1$ and define $\xi_v[1] = -\infty$, $v = 2, \ldots, \hat{h}$. For $v = 1$, if job 1 belongs to set H, compute

$$\xi_1[1] = \xi_0[0] + \beta(1, 1) - \beta(1, 0) + s_1 r(1) + \lambda(1);$$

otherwise, i.e., if job 1 belongs to set K, define $\xi_1[1] = -\infty$.
Step 5. For each $t \in H$ find the value of ℓ_t.
Step 6. For $v = 0, \ldots, \hat{h}$ do

```
        For $t = 2, \ldots, h + k$ do
            If $t \in H$, then with $\ell = \ell_t$ compute $\xi_v[t]$ by (60);
            otherwise, define $\xi_v[t] := \xi_v[t - 1]$.
```

Step 7. Compute Φ' by (57). If $\Phi' = dS_{v_*+k} - \xi_{v_*}[t_*]$ for some v_* and t_*, $0 \le v_* \le \hat{h}$, $1 \le t_* \le h + k$, then perform backtracking to determine the set $Y' \in \mathcal{H}_{v_*}[t_*]$ such that $\xi_{v_*}[t_*] = \sum_{i=1}^{v_*+k} s_i r_i(Y' \cup K) + \lambda(Y')$.
Step 8. If $h + k < m$, define $\Phi'' := +\infty$ and go to Step 12.
Step 9. Compute $\eta_2[m]$ by (49). For $t \in H$, $t \ge m$, compute $\eta_2[t]$ by (50).
Step 10. For $v = 1, \ldots, m$

```
        For $t = 1, \ldots, h + k$ do
            Compute all values $\zeta_v[t]$ by (66) with the initial conditions (67).
```

Step 11. Compute Φ'' by (65). If $\Phi'' = dS_m - \zeta_m[t_*] - \eta_2[t_*]$ for some $t_* \in H$, $m \le t_*$, then perform backtracking to determine the set $Y_1'' \in \mathcal{H}[t_*]$ such that $\zeta_m[t_*] = \sum_{i=1}^{m} s_i r_i(Y_1'' \cup K[t_*]) + \eta_2[t_*]$. Define the sets $Y_2'' := \{j \in H \mid j > t_*\}$ and $Y'' := Y_1'' \cup Y_2''$.
Step 12. Compute the value $\widetilde{\varphi}_K^H(X)$ by (55) applied to $X = \hat{H}$. If $\Phi' < \Phi''$, define $Y_* := Y'$; otherwise, define $Y_* := Y''$.

The most time consuming parts of the procedure are the double loops is Steps 6 and 10, which require $O\left(\hat{h}(h + k)\right)$ time and $O(m(h + k))$ time, respectively. Thus, the overall time complexity of Procedure CompQrj is $O(m(h + k))$.

For $h \ge k$, the time complexity becomes $O(mh)$. We can show that the bound $O(mh)$ also applies to the case that $h < k$; this can be done by an approach similar to that used in Sect. 5.2. Hence, the next theorem follows from Theorem 4.

Theorem 7 *Problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ can be solved in $O(nm \log n)$ time.*

6 Conclusions

In this paper, we develop a decomposition recursive algorithm for maximizing a linear function over a submodular polyhedron intersected with a box. We illustrate the power of our approach by adapting the algorithm to solving three scheduling problems with controllable processing times. In these problems, it is required to find a preemptive schedule that is feasible with respect to a given deadline and minimizes total compression cost. The resulting algorithms run faster than previously known.

We intend to extend this approach to other scheduling models with controllable processing times, e.g., to a single machine with distinct release dates and deadlines. It will be interesting to identify problems, including those outside the area of scheduling, for which an adaptation of our approach is beneficial.

Although throughout the paper we assume that the processing times are real numbers from intervals $\left[\underline{p}(j), \overline{p}(j)\right]$, the formulated approach is applicable to the case where the processing times may only take integer values in the interval. Indeed, if all the input numbers, except for costs $w(j)$, are given by integers, then the submodular rank function takes integer values, and the optimal solution $p(j),\ j \in N$, found by Procedure Decomp is integral.

Acknowledgments 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, grants 24500002 and 25106503.

Appendix: Towards the Proof of Theorem 5

We show that a modified version of the solution procedure for problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|W$ can be made run in $O(n + m \log m \log n)$ time, which is better than the previously proved time $O(n \log n)$, provided that $n > m \log m$. One of the reasons for the running time $O(n \log n)$ is that Procedure CompQr0 uses the sorted lists U and L of length n each, created at the preprocessing stage. Another reason is that in the previous implementation, for Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ at each level of recursion we have that $T_{Y_*}(h) = T_{\text{Split}}(h) = O(h)$. Thus, to achieve the overall running time of $O(n + m \log m \log n)$ we should not use the sorted lists of more than m elements, and try to reduce $T_{Y_*}(h)$ and $T_{\text{Split}}(h)$.

First, we show that for Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ Procedure CompQr0 can be made run in $O(g + m \log m)$ time, where as in Sect. 4.3, integer g denotes the number of non-fixed variables in set H, i.e., $g = |H \backslash F|$.

Before calling Procedure Decomp$(N, \emptyset, \emptyset, \underline{\mathbf{p}}, \overline{\mathbf{p}})$, there is no need in creating the sorted lists U and L. As part of the input of the modified Procedure CompQr0 we use an unsorted list Λ of the values $\lambda(j),\ j \in H \backslash F$, computed for the non-fixed variables with respect to a chosen heavy-element set $\hat{H}$. Additionally, the input includes an unsorted list Z that contains $\min\{|F|, m-1\}$ largest elements $\lambda(j) = l(j) = u(j),\ j \in F$; besides, we also keep the value $\lambda(F)$.

In Step 2, in order to compute Φ' and Y' we need to find the values $\lambda_1, \lambda_2, \ldots, \lambda_{\hat{h}}$, and their partial sums that are used in (35). It follows that

$$\sum_{i=1}^{v} \lambda_v = \max_{Y \in \mathcal{H}_v} \{\lambda(Y \backslash F) + \lambda(F \cap Y)\}, \qquad v = 1, \ldots, \hat{h},$$

which implies that the values $\lambda_1, \lambda_2, \ldots, \lambda_{\hat{h}}$ are the largest values in the merger of the lists Λ and Z. In order to merge these lists in $O(g + \hat{h} \log \hat{h})$ time, we find the $\hat{h}-$th largest element and find the sorted sequence of $\hat{h}$ largest elements in these two lists. After that we compute the partial sums $\sum_{i=1}^{v} \lambda_v$, $1 \leq v \leq \hat{h}$, in $O(\hat{h})$ time. To compute Φ' and determine set Y', we perform Step 2-2, which takes $O(\hat{h})$ time. Since $\hat{h} < m$, we deduce that Step 2 of Procedure CompQr0 can be made to run in $O(g + m \log m)$ time.

In Step 3, we need to compute $\lambda(H)$ which contributes to Φ''. Notice that $\lambda(H) = \lambda(H \backslash F) + \lambda(F)$, where $\lambda(F)$ is known as part of the input. Thus, Step 3 requires $O(g)$ time. In Step 4, to compute the rank function $\widetilde{\varphi}_K^H(X)$ for $X = \hat{H}$, we need the value $u(\hat{H})$, which can be found in $O(g)$ time, since the heavy-element set $\hat{H}$ is chosen from the non-fixed variables only.

Thus, in terms of Theorem 4, the described modifications imply that $T_{Y_*}(h) = O(g + m \log m)$.

In accordance with Procedure Decomp, Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ has to be decomposed into two subproblems with respect to a found set Y_*, which is either $|Y_*| \leq m - 1$ or $Y_* = H$. However, if either $Y_* = \emptyset$ or $Y_* = H$, then for one of the two emerging subproblems the set of variables will be empty. Besides, if $|Y_*| < m$, one of the subproblems will have at most m variables and can be recursively solved in $O(m \log m)$ time by a straightforward application of the method described in Sect. 5.1. Thus, in any case we are left with exactly one non-trivial subproblem to be solved at each level of recursion. Let us show that the instance of that subproblem together with the accompanying information can be derived in $O(g + m)$ time; in other words, that $T_{\text{Split}}(h) = O(g + m)$. Without loss of generality, we assume that the parameters of the generated problem are defined by (26), i.e., we deal with Problem LP$\left(H \backslash Y_*, \left(\hat{H} \cup F\right) \backslash Y_*, K \cup Y_*, \mathbf{l}_2, \mathbf{u}_2\right)$. Recall that the variables of set $\hat{H} \backslash Y_*$ become fixed and can be excluded from the list Λ in $O(g)$ time. We will also need $O(g)$ time to update the sum of the processing times of the fixed variables. To obtain a new list Z we add the values $u(j)$, $j \in \hat{H} \backslash Y_*$, to the old list Z, find the $(m-1)$-th largest element in the resulting list and keep the elements that do not exceed that element. The new list Z will be found in $O(g + m)$ time.

As implied by Lemma 3, the heavy-element $\hat{H}$ is selected in such a way that the number of non-fixed variables is reduced by half in each new level of recursion, i.e., it is n in the initial level 0, at most $n/2$ in the next level 1, at most $n/4$ in level 2, etc. Thus, the running time of the modified algorithm for solving problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|W$ is at most

$$\sum_{u=0}^{\log n} \left(O(n/2^u) + O(m \log m)\right) = O(n + m \log m \log n).$$

References

1. Brucker, P.: Scheduling Algorithms, 5th edn. Springer, Berlin (2007)
2. Chen, Y.L.: Scheduling jobs to minimize total cost. Eur. J. Oper. Res. **74**, 111–119 (1994)
3. Fujishige, S.: Lexicographically optimal base of a polymatroid with respect to a weight factor. Math. Oper. Res. **5**, 186–196 (1980)
4. Fujishige, S.: Submodular Functions and Optimization. Annals of Discrete Mathematics, vol. 58, 2nd edn. Elsevier, Amsterdam (2005)
5. Gonzales, T.F., Sahni, S.: Preemptive scheduling of uniform processor systems. J. ACM **25**, 92–101 (1978)
6. Groenevelt, H.: Two algorithms for maximizing a separable concave function over a polymatroid feasible region. Eur. J. Oper. Res. **54**, 227–236 (1991)
7. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. J. ACM **48**, 761–777 (2001)
8. Janiak, A., Kovalyov, M.Y.: Single machine scheduling with deadlines and resource dependent processing times. Eur. J. Oper. Res. **94**, 284–291 (1996)
9. Jansen, K., Mastrolilli, M.: Approximation schemes for parallel machine scheduling problems with controllable processing times. Comput. Oper. Res. **31**, 1565–1581 (2004)
10. 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)
11. 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. Logistics of Production and Inventory, vol. 4, pp. 445–522. Elsevier, Amsterdam (1993)
12. Leung, J.Y.-T.: Minimizing total weighted error for imprecise computation tasks. In: Leung, J.Y.-T. (eds.) Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 34-1–34-16. Chapman & Hall/CRC, London (2004)
13. McCormick, S.T.: Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. Oper. Res. **47**, 744–756 (1999)
14. McNaughton, R.: Scheduling with deadlines and loss functions. Manag. Sci. **12**, 1–12 (1959)
15. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
16. Nowicki, E., Zdrzałka, S.: A survey of results for sequencing problems with controllable processing times. Discrete Appl. Math. **26**, 271–287 (1990)
17. Nowicki, E., Zdrzałka, S.: A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times. Discrete Appl. Math. **63**, 237–256 (1995)
18. Sahni, S.: Preemptive scheduling with due dates. Oper. Res. **27**, 925–934 (1979)
19. Sahni, S., Cho, Y.: Scheduling independent tasks with due times on a uniform processor system. J. ACM **27**, 550–563 (1980)
20. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theory B **80**, 346–355 (2000)
21. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003)
22. Shabtay, D., Steiner, G.: A survey of scheduling with controllable processing times. Discrete Appl. Math. **155**, 1643–1666 (2007)
23. Shakhlevich, N.V., Strusevich, V.A.: Pre-emptive scheduling problems with controllable processing times. J. Sched. **8**, 233–253 (2005)
24. Shakhlevich, N.V., Strusevich, V.A.: Preemptive scheduling on uniform parallel machines with controllable job processing times. Algorithmica **51**, 451–473 (2008)
25. 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)
26. Shakhlevich, N.V., Shioura, A., Strusevich, V.A.: Fast divide-and-conquer algorithms for preemptive scheduling problems with controllable processing times—a polymatroidal approach. In: Halperin, D., Mehlhorn, K. (eds.) Lecture Notes Computer Science 5193, ESA 2008, pp. 756–767. Springer, Berlin (2008)
27. 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. Discrete Math. **27**, 186–204 (2013)