Journal of Global Optimization (2020) 76:471–490
https://doi.org/10.1007/s10898-018-0686-2

Scheduling problems with controllable processing times and a common deadline to minimize maximum compression cost

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

Received: 28 October 2017 / Accepted: 28 June 2018 / Published online: 6 July 2018

Abstract
We consider a range of scheduling problems with controllable processing times, in which the jobs must be completed by a common deadline by compressing appropriately their processing times. The objective is to minimize the maximum compression cost. We present a number of algorithms based on common general principles adapted with a purpose of reducing the resulting running times.

Keywords Scheduling · Single machine · Parallel machines · Controllable processing times

Mathematics Subject Classification 90C27 · 90B35

1 Introduction

The area of *Scheduling with Controllable Processing Times* (*SCPT*) has been active since the 1980s; see surveys [21,26], as well as the review [31] that covers the most recent developments in the area. In particular, it is demonstrated in [31] that the SCPT models include those models that have been studied within another popular area of research known as *Scheduling with Imprecise Computation* (*SIC*), see [10,18] for the corresponding reviews.

The range of scheduling problems we address in this paper can be described as follows. The jobs of set $N = \{1, 2, \dots, n\}$ should be processed either on a single machine M_1 or on parallel machines $M_1, M_2, \dots, M_m$, where $m \geq 2$. Job $j \in N$ arrives at time $r(j)$ and must be completed by time $d(j)$. Processing of any job can be preempted and resumed later on the same or on a different machine. In classical scheduling models with fixed processing times, each job $j \in N$ is given a value $p(j)$ that represents its duration. In the SCPT models, an actual processing time $p(j)$ of job j is not known in advance, but has to be chosen from a

✉ Vitaly A. Strusevich
sv02@gre.ac.uk

1 Department of Industrial Engineering and Economics, Tokyo Institute of Technology, Tokyo 152-8550, Japan

2 School of Computing, University of Leeds, Leeds LS2 9JT, UK

3 Department of Mathematical Sciences, Old Royal Naval College, University of Greenwich, Park Row, London SE10 9LS, UK

given interval $[l(j), u(j)]$, where $l(j)$ and $u(j)$ are a given lower bound and a given upper bound on the processing time, respectively. The value $x(j) = u(j) - p(j)$ shows by how much the largest possible processing time is reduced and is called a *compression amount* of job $j \in N$.

In the case of the parallel machines, we distinguish between the *identical* machines and the *uniform* machines. In the former case, the machines have the same speed, so that for a job j with an actual processing time $p(j)$ the total length of the time intervals in which this job is processed in a feasible schedule is equal to $p(j)$. If the machines are uniform, then it is assumed that machine M_i has speed s_i, $1 \le i \le m$. If for job j the value $u(j)$ is compressed to $p(j)$ and this job is assigned to machine M_i alone then the duration of such processing is $p(j)/s_i$.

A popular range of the SCPT problems addresses issues of determining actual processing times of the jobs in such a way that there exists an optimal schedule in which no job $j \in N$ is processed outside the interval $[r(j), d(j)]$ and a certain objective Φ that depends on compressions $x(j)$ is minimized. Each job can be associated with a weight $w(j)$, which reflects its relative importance. The two following objective functions are among those that have received most attention:

- Total compression cost $\Phi_\Sigma = \sum_{j \in N} w(j)\, x(j)$, and
- Maximum compression cost $\Phi_{\max} = \max\{x(j)/w(j) \mid j \in N\}$.

The problems with the Φ_Σ objective are mainly addressed in the mainstream study of SCPT, while those with the $\Phi_{\max}$ objective are studied within the SIC literature. The most recent survey [31] indicates that in the most general setting, when the jobs have arbitrary release dates and arbitrary deadlines, algorithms developed for the problems with the Φ_Σ objective for all machine environments and for the $\Phi_{\max}$ objective on parallel machines are not-improvable, since they are as fast as the algorithms for the counterparts with fixed processing times. If the jobs have a common deadline, for the problems of minimizing Φ_Σ algorithms are known that perform faster than in the general case. In this paper, we explore the unstudied versions of the problems to minimize $\Phi_{\max}$, in which all jobs have a common deadline $d_{(j)} = d$.

The SIC models, although they have received considerable separate attention, should be viewed as one of possible interpretations of the SCPT models. In the SIC models, the jobs are seen as computational tasks to be processed with preemption in a computing system that consists either of one processor or of several parallel processors (machines). 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 the processing requirement $u(j)$ can be split into a mandatory part which takes $l(j)$ time, and an optional part that may take up to $u(j) - l(j)$ 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 $u(j)$ task j is executed for $p(j) = u(j) - x(j)$ time units, then computation is imprecise and $x(j)$ corresponds to the error of computation. In the SIC terminology the function Φ_Σ is called the total weighted error, while $\Phi_{\max}$ is known as the maximum weighted error.

In accordance with standard scheduling notation, a three-field string $\alpha|r(j),\ C(j) \le d(j),\ pmtn|-$ denotes a feasibility problem, in which it is required to verify whether a set N of jobs with given fixed processing times $p(j)$, $j \in N$, can be processed with preemption in such a way that each job j starts no earlier than its release date $r(j)$ and completes no later than its deadline $d(j)$. Here, in the first field we define a machine environment which could

Table 1 Running times for solving feasibility problems $\alpha|r(j), C(j) \le d(j), pmtn|-$ and problems $\alpha|r(j), p(j) = u(j) - x(j), C(j) \le d(j), pmtn|\Phi$

α	Feasibility	Total cost $\Phi = \Phi_\Sigma$	Maximum cost $\Phi = \Phi_{\max}$
1	$O(n \log n)$ [9,13]	$O(n \log n)$ [30]	$O\left(n^2\right)$ [11]
P	$O\left(n^3\right)$ [13]	$O\left(n^3\right)$ [19]	$O\left(n^3\right)$ [31]
Q	$O\left(mn^3\right)$ [4]	$O\left(mn^3\right)$ [19]	$O\left(mn^3\right)$ [31]

be a single machine ($\alpha = 1$), m parallel identical machines ($\alpha = P$), or m parallel uniform machines ($\alpha = Q$). In the middle field, the presence of $r(j)$ means that the jobs may have individual release date (otherwise, they have a common zero release date). The inequality $C(j) \le d(j)$ stresses that each job must complete by its deadline, while $pmtn$ points out that preemptive processing is allowed.

For problems with controllable processing times, the notation above is extended to become $\alpha|r(j), p(j) = u(j) - x(j), C(j) \le d(j), pmtn|\Phi$, where in the middle field we additionally write $p(j) = u(j) - x(j)$ to emphasize that the actual processing time of each job is obtained by reducing the largest possible processing time $u(j)$ by $x(j)$, while in the third field Φ is an objective function that depends on compressions $x(j)$, $j \in N$.

It is clear that the running time needed to solve a feasibility problem $\alpha|r(j), C(j) \le d(j), pmtn|-$ provides a lower bound on the running time that is required to solve the corresponding problem $\alpha|r(j), p(j) = u(j)-x(j), C(j) \le d(j), pmtn|\Phi$. Table 1 presents the best known running times required for solving the SCPT problems in the most general setting, i.e., with arbitrary release dates and deadlines, with an objective $\Phi \in \{\Phi_\Sigma, \Phi_{\max}\}$, compared with the running times needed to solve the corresponding feasibility problems. We see that the SCPT problems on parallel machines are no harder computationally than the associated feasibility problem. On a single machine, in the case of the $\Phi_{\max}$ objective there is still a complexity gap.

For completeness, we briefly review the algorithmic techniques employed in the methods listed in Table 1. The single machine feasibility problem $1|r(j), C(j) \le d(j), pmtn|-$ in principle cannot be solved faster than finding the sorted sequence of the release dates and deadlines. The best possible running time of $O(n \log n)$ for solving the problem is achieved by the Earliest Deadline First (EDF) scheduling policy designed by Horn [13]. Notice that the running time of $O(n \log n)$ remains the best even if either all release dates are zero or all jobs have a common deadline. For problem $1|r(j), p(j) = u(j)-x(j), C(j) \le d(j), pmtn|\Phi_\Sigma$, the running time of $O(n \log n)$, which matches that of the EDF algorithm, is obtained due to reformulating the problem as a linear programming problem over a submodular polyhedron and finding its solution by a novel decomposition algorithm [30].

The feasibility problems on parallel machines are handled by finding the maximum flow in the corresponding bipartite networks; see [4,13]. McCormick [19] reduces problems with the Φ_Σ objective to finding the maximum flow in networks in which capacities of some arcs linearly depend on a parameter, individual for each of these arcs. To deal with the $\Phi_{\max}$ objective, Shioura et al. [31] demonstrate that the corresponding SCPT problems can be reduced to solving lexicographic flow sharing problems in special networks. In these networks, capacities of some arcs depend on a common parameter and the required flow sharing problems can be solved by methods developed in [7], which results into the algorithms

Table 2 Running times for solving feasibility problems $\alpha|r(j), C(j) \leq d, pmtn|-$ and problems $\alpha|r(j), p(j) = u(j) - x(j), C(j) \leq d, pmtn|\Phi$

α	$r(j)$	Feasibility	Total cost $\Phi = \Phi_\Sigma$	Maximum cost $\Phi = \Phi_{\max}$
1	Arbitrary	$O(n \log n)$ [9,13]	$O(n \log n)$ [15,27]	$O(n \log n)$, Theorem 1
P	Zero	$O(n)$ [20]	$O(n)$ [16]	$O(n)$, Theorem 2
P	Arbitrary	$O(n \log n)$ [22]	$O(n \log n \log m)$ [29]	$O\left(n^2\right)$, Theorem 3
Q	Zero	$O(n + m \log m)$ [8]	$O(\min\{n \log n, n + m \log m \log n\})$ [29]	$O(n \log n + nm)$, Theorem 5
Q	Arbitrary	$O(nm + n \log n)$ [23]	$O(mn \log n)$ [29]	$O\left(mn^2\right)$, Theorem 6

that run faster than those reported for these problems in the SIC literature [10]. See [31], where the methodologies outlined above are discussed in detail.

It is quite often required that the jobs must be completed by a common deadline, rather than by individual deadlines. Besides, it is possible that the jobs do not have specific release dates but are available simultaneously at time zero. In such situations, there is a hope that a simplified problem can be solved faster than in the general case. The main purpose of this paper is to develop fast algorithms for the SCPT problems to minimize the maximum cost $\Phi_{\max}$ for these simplified problems. The summary of our results is presented in the last column of Table 2. For completeness, we list in that table the results for the relevant feasibility problems and the SCPT problems to minimize the total compression cost Φ_Σ.

Notice that for the problems to minimize the objective Φ_Σ, the results for parallel machine problems quoted in Table 2 are obtained by reformulating the corresponding problem as a linear programming problem over a submodular polyhedron and finding its solution by a novel decomposition algorithm [29]. The only exception is problem $P|p(j) = u(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$, which is essentially a continuous knapsack problem [16].

2 Problems with a common deadline: general principles

In this section, we outline a generic procedure aimed at solving problems $\alpha|r(j), p(j) = u(j) - x(j), C(j) \leq d, pmtn|\Phi_{\max}$ for $\alpha \in \{1, P, Q\}$. We start with some terminology, including that related to optimization with submodular constraints. For the latter purpose, we mainly follow a comprehensive monograph on submodular optimization by [6], see also [17,25].

For a positive integer n, let $N = \{1, 2, \ldots, n\}$ be a ground set, and let 2^N denote the family of all subsets of N. For a subset $Y \subseteq N$, let $\mathbb{R}^Y$ denote the set of all vectors $\mathbf{p}$ with real components $p(j)$, where $j \in Y$. For a vector $\mathbf{p} \in \mathbb{R}^N$, define $p(Y) = \sum_{j \in Y} p(j)$ for every set $Y \in 2^N$.

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

$$\varphi(Y) + \varphi(Z) \geq \varphi(Y \cup Z) + \varphi(Y \cap Z)$$

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

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

is the *submodular polyhedron.*

We now define submodular polyhedra for the problems under study. For a set of jobs $Y \subseteq N$, let $\varphi(Y)$ be a set-function that represents the total production capacity available for processing the jobs of set Y. If we ignore the machine speeds, then $\varphi(Y)$ is essentially equal to the length of all time intervals within which the jobs of set Y can be processed. This means that for a problem with fixed processing times a feasible schedule exists if and only if the inequality

$$p(Y) \le \varphi(Y) \tag{1}$$

holds for all sets $Y \subseteq N$.

The actual representation of function $\varphi(Y)$ is problem dependent. The capacity functions φ for the problems $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi$ are given in [28,29]. Notice that all these functions are submodular.

Given problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi$, we always assume that $r(j) + l(j) \le d$ for each job $j \in N$; otherwise the problem is infeasible. Let $x(j)$ be a compression amount of job $j \in N$ in some feasible schedule. Then in any feasible schedule $0 \le x(j) \le u(j) - l(j)$, or, equivalently, $l(j) \le p(j) \le u(j)$ for each job $j \in N$. A generic problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi$ associated with a capacity function φ can be formulated as the following problem with the decision variables $x(j)$, $j \in N$:

$$\begin{aligned} \min\ & \Phi(x(1), \ldots, x(n)) \\ \text{s.t.}\ & p(Y) = u(Y) - x(Y) \le \varphi(Y), \quad Y \in 2^N, \\ & 0 \le x(j) \le u(j) - l(j), \qquad j \in N. \end{aligned} \tag{2}$$

In the case of minimizing the total cost Φ_Σ, the objective function is $\sum w(j)x(j)$, and by switching to the decision variables $p(j) = u(j) - x(j)$, $j \in N$, we can reduce a problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_\Sigma$ to solving the linear programming problem of the following structure:

$$\begin{aligned} \max\ & \sum_{j \in N} w(j)p(j) \\ \text{s.t.}\ & p(Y) \le \varphi(Y), \qquad Y \in 2^N, \\ & l(j) \le p(j) \le u(j),\ j \in N. \end{aligned} \tag{3}$$

Notice that the feasible region of problem (3) is a submodular polyhedron (1) intersected with the box. For $\alpha \in \{P, Q\}$, fast algorithms developed in [29] for problems $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_\Sigma$ are based on reformulation of these problems in terms of (3). The optimal actual processing times can be found by solving (3), which can be done by a decomposition algorithm that consists of $O(\log n)$ iterations.

The maximum cost $\Phi_{\max}$ function is defined by $\max\{x(j)/w(j) \mid j \in N\}$, so that problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ can be reformulated as

$$\begin{aligned} \min\ & \max\{x(j)/w(j) \mid j \in N\} \\ \text{s.t.}\ & u(Y) - \varphi(Y) \le x(Y), \qquad Y \in 2^N, \\ & 0 \le x(j) \le u(j) - l(j), \quad j \in N. \end{aligned} \tag{4}$$

It is convenient to rewrite (4) in terms of new decision variables

$$\lambda(j) = \frac{x(j)}{w(j)},\ \ j \in N, \tag{5}$$

so that problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ reduces to

$$\begin{aligned} &\min\ \max\{\lambda(j)\,|j \in N\} \\ &\text{s.t.}\ \ u(Y) - \varphi(Y) \le \sum_{j \in Y} \lambda(j)\, w(j),\ Y \in 2^N, \\ &\qquad 0 \le \lambda(j) \le \lambda_j, \qquad\qquad\qquad j \in N. \end{aligned} \tag{6}$$

where

$$\lambda_j = \frac{u(j) - l(j)}{w(j)}. \tag{7}$$

A variable $\lambda(j)$ represents the weighted cost of compressing job $j \in N$. Since for job j the largest compression is equal to $u(j) - l(j)$, which occurs if the actual processing time of job j is $l(j)$, the value of λ_j represents the largest possible contribution that job j can make into the objective function $\Phi_{\max}$.

For simplicity of presentation, throughout this paper we assume that the values of λ_j defined by (7) are distinct and the jobs are numbered in such a way that

$$\lambda_1 = \min\left\{\lambda_j | j \in N\right\},\ \lambda_n = \max\left\{\lambda_j | j \in N\right\}. \tag{8}$$

Denote $\lambda_0 := 0$. Let Λ be the list of the λ -values $\lambda_0, \lambda_1, \ldots, \lambda_n$. Notice that, unless stated otherwise, we do not assume that the list Λ is sorted.

For a job $j \in N$, given a value of $\lambda \in [\lambda_0, \lambda_n]$ and λ_j defined by (7), define the values of the decision variables in (6) by

$$\lambda(j) = \begin{cases} \lambda_j, & \text{if } \lambda_j < \lambda \\ \lambda, & \text{if } \lambda_j \ge \lambda \end{cases} \tag{9}$$

and compute the associated compressions $x(j)$ by

$$x(j) = \lambda(j)\, w(j),\ \ j \in N. \tag{10}$$

Here, each of the jobs j with $\lambda_j < \lambda$ is fully compressed, i.e., $x(j) = u(j) - l(j)$ and its weighted compression is less than the value of λ. For each of the other jobs, the weighted compression is equal to λ.

A given $\lambda \in [\lambda_0, \lambda_n]$ is said to be *feasible* for problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$, if variables $\lambda(j)$, $j \in N$, defined by (9) deliver a feasible solution to problem (6). In scheduling terms, this means that there exists a feasible schedule with actual processing times equal to $u(j) - x(j)$, where compressions $x(j)$, $j \in N$, satisfy (10). For a particular problem, we can check whether λ is feasible either analytically (by verifying certain relations that define the feasible region of problem (6)) or by running an algorithm that checks the existence of a feasible schedule with constant processing times associated with the compressions $x(j)$, $j \in N$.

Below we outline a generic procedure for solving problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$.

Procedure A

Step 1. *Initialization*
Compute λ_j by (7). If necessary, renumber the jobs so that (8) holds and form the list Λ.

Step 2. *Check of a trivial solution*
If $\lambda = \lambda_0 = 0$ is feasible, then the optimal compressions are $x^* (j) = 0,\ 1 \le j \le n$, with the optimal value of the objective function $\Phi_{\max}$ equal to zero; otherwise go to Step 3.
Step 3. *Check for infeasibility*
If $\lambda = \lambda_n$ is infeasible, then the original problem does not have a feasible solution; otherwise go to Step 4.
Step 4. *Localization search*
Examining list Λ, find the largest infeasible value λ' and the smallest feasible value λ''.
Step 5. *Finding an optimal solution*
Examining the interval $\left[\lambda', \lambda''\right]$ find an optimal value λ^* as the smallest feasible value of $\lambda \in \left[\lambda', \lambda''\right]$. The optimal compressions $x^* (j)$ are equal to

$$x^* (j) = \begin{cases} \lambda_j w (j), & \text{if } \lambda_j < \lambda^* \\ \lambda^* w (j), & \text{if } \lambda_j \ge \lambda^*. \end{cases} \tag{11}$$

The implementation details of the feasibility checks and of Steps 4 and 5 differ for different problems, as presented in the remaining parts of this paper.

All problems considered below can be formulated using a generic problem (4). However, for all these problems there is no need to consider exponentially many inequalities that define the feasible region and the problems can be solved by fast polynomial-time algorithms.

3 Single machine: arbitrary release dates

In this section, we adapt Procedure A to solving problem $1|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$.

Assume that the values λ_j found by (7) are distinct and the jobs are numbered in accordance with

$$0 = \lambda_0 < \lambda_1 < \cdots < \lambda_n, \tag{12}$$

so that list $\Lambda = (\lambda_0, \lambda_1, \ldots, \lambda_n)$ is increasing.

Let $R^{(z)}$ denote the zth smallest release date, while $N^{(z)}$ be the set of jobs that arrive no earlier than time $R^{(z)}$, i.e.

$$N^{(z)} = \left\{ j \in N | r (j) \ge R^{(z)} \right\}, \tag{13}$$

where $1 \le z \le n$.

In order to complete all jobs $j \in N^{(z)}$ by the deadline d there must be a sufficient processing capability. If $p (j) = u (j) - x (j) \ge l (j)$ is an actual processing time of job $j \in N^{(z)}$, then the combined duration of the jobs from $N^{(z)}$ should not exceed the length of the interval $[r(j), d]$, i.e.,

$$p \left(N^{(z)}\right) = u \left(N^{(z)}\right) - x \left(N^{(z)}\right) \le d - R^{(z)}$$

must hold for each z, $1 \le z \le n$; see [9] where that condition is presented (in a different but equivalent form).

Thus, for problem $1|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ the generic formulation (4) simplifies, since we do not need to consider all 2^n inequalities that define the feasible region. The simplified formulation becomes

$$\begin{aligned} &\min \max \left\{ \frac{x(j)}{w(j)} | j \in N \right\} \\ &\text{s.t. } u\left(N^{(z)}\right) - d + R^{(z)} \le x\left(N^{(z)}\right), \ z = 1, \dots, n; \\ &\quad 0 \le x(j) \le u(j) - l(j), \qquad j \in N. \end{aligned} \tag{14}$$

Changing the decision variables to $\lambda(j)$, $j \in N$, defined by (5) the problem can be reformulated as

$$\begin{aligned} &\min \max \{\lambda(j) | j \in N\} \\ &\text{s.t. } u\left(N^{(z)}\right) - d + R^{(z)} \le \sum_{j \in N^{(z)}} \lambda(j) w(j), \ z = 1, \dots, n; \\ &\quad 0 \le \lambda(j) \le \lambda_j, \qquad j \in N. \end{aligned} \tag{15}$$

For the numbering (12), let us fix an interval $I = \left[\lambda_q, \lambda_{q+1}\right]$ between two consecutive elements of list Λ. For problem $1|r(j), p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$, in accordance with (9), a value of $\lambda \in I$ is feasible if

$$u\left(N^{(z)}\right) - d + R^{(z)} \le \sum_{j \in \{1,\dots,q\} \cap N^{(z)}} \lambda_j w(j) + \lambda \sum_{j \in \{q+1,\dots,n\} \cap N^{(z)}} w(j), \ 1 \le z \le n. \tag{16}$$

For each set $N^{(z)}$, $1 \le z \le n$, each of the jobs $1, \dots, q$ that belongs to that set is fully compressed. On other hand, each of the jobs $q+1, \dots, n$ is compressed in such a way that its compression cost is equal to λ, and none of these jobs, except job $q+1$, becomes fully compressed as long as λ changes within $\left[\lambda_q, \lambda_{q+1}\right]$. All feasibility checks for a given value of λ can be done in linear time.

Below we present an algorithm that is an adaptation of Procedure A to solving problem $1|r(j), p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$.

Algorithm A1

Step 1. Find the values $R^{(z)}$ and sets $N^{(z)}$ for $z = 1, 2, \dots, n$. Compute λ_j by (7), sort them in accordance with (12) and form list $\Lambda = (\lambda_0 = 0, \lambda_1, \dots, \lambda_n)$.

Step 2. Check if $\lambda = 0$ is feasible. If yes, then the optimal compressions are $x^*(j) = 0$, $1 \le j \le n$, with the optimal value of the objective function $\Phi_{\max}$ equal to zero; otherwise, set $\underline{q} = 0$ and go to Step 3.

Step 3. Check if $\lambda = \lambda_n$ is feasible. If not, then the original problem does not have a feasible solution; otherwise set $\overline{q} = n$ and go to Step 4.

Step 4. Repeat until $\overline{q} = \underline{q} + 1$:

(a) Compute $q = \left\lceil \left(\overline{q} + \underline{q}\right)/2 \right\rceil$.

(b) Define $\lambda = \lambda_q$. Compute $\lambda(j)$ by (9) and the compressions $x(j)$ by (10). Use (16) to check feasibility of λ. If λ is infeasible, then update $\underline{q} := q$; otherwise, λ is feasible and we update $\overline{q} = q$.

Step 5. Define $q = \underline{q}$. An optimal maximum compression cost λ^* is the smallest feasible value of $\lambda \in \left[\lambda_q, \lambda_{q+1}\right]$ found by

$$\lambda^* = \max_{1 \le z \le n} \left\{ \frac{u\left(N^{(z)}\right) - d + R^{(z)} - \sum_{j \in \{1,\dots,q\} \cap N^{(z)}} \lambda_j w(j)}{\sum_{j \in \{q+1,\dots,n\} \cap N^{(z)}} w(j)} \right\}. \tag{17}$$

The optimal compressions $x^*(j)$ are defined by (11).

In Algorithm A1, the loop performed in Step 4 is a binary search procedure that implements the localization search of Step 4 of Procedure A. We come to Step 5 having found an interval $\left[\lambda', \lambda''\right] = \left[\lambda_q, \lambda_{q+1}\right]$, such that λ_q is infeasible and λ_{q+1} is feasible.

Step 5 of Algorithm A1 corresponds to Step 5 of Procedure A. Here, all jobs $j \le q$ are fully compressed, while each of the remaining jobs $q+1, \dots, n$ is compressed in such a way that its weighted compression cost is equal to a common value λ. The formula (17) for an optimal value λ^* is derived from (16).

The binary search in Step 4 consists of $O(\log n)$ iterations, each iteration takes $O(n)$ time. Finding the optimal value of λ^* in Step 5 also takes $O(n)$ time. Thus, the following statement holds.

Theorem 1 *Problem* $1|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ *is solvable in* $O(n \log n)$ *time.*

An alternative algorithm for problem $1|r(j), p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$ of the same running time can be obtained by linking the problem of minimizing the maximum compression to the problem of minimizing a quadratic cost function over the same feasible region.

Based on the input data for problem $1|r(j), p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$, define the problem

$$\begin{aligned} &\min \sum_{j=1}^{n} \frac{1}{w(j)} x(j)^2 \\ &\text{s.t. } u\left(N^{(z)}\right) - \left(d - R^{(z)}\right) \le x\left(N^{(z)}\right), \quad z = 1, \dots, n; \\ &\qquad 0 \le x(j) \le u(j) - l(j), \qquad\qquad j \in N. \end{aligned}$$

It follows from a general result from [5] and from Sect. 9.2 of [6] on the equivalence of quadratic and lexicographic optimization with submodular constraints that any optimal solution to the problem above is also an optimal solution to problem (14). The former problem can be classified as the resource allocation problem with a separable quadratic function under nested constraints. Such a problem can be solved in $O(n \log n)$ time, as proved in [12]. A disadvantage of using this approach is that it does not deliver a solution in a closed form.

Notice that the running time established in Theorem 1 is faster than $O\left(n^2\right)$, the best known time for a more general problem with distinct release dates, see Table 1. On the other hand, it matches the running time for solving the feasibility problem $1|r(j), C(j) \le d, pmtn|-$ with constant processing times; see Table 2.

4 Identical parallel machines: zero release dates

In this section, we address problem $P|p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$, in which all jobs must be scheduled on m identical parallel machines in the time interval $[0, d]$. Therefore, in any feasible schedule the sum of actual processing times of all jobs $p(N)$ must not exceed md. Additionally, for each job j its actual processing time $p(j)$ may not exceed the deadline d.

As in Sect. 3, for problem $P|p(j)=u(j)-x(j),\, C(j)\le d,\, pmtn|\Phi_{\max}$ the generic formulation based on the compression variables $x\,(j)$, $j\in N$, can be written as

$$\begin{aligned}&\min\ \max\left\{\frac{x\,(j)}{w\,(j)}|j\in N\right\}\\ &\text{s.t.}\ \ u\,(N)-md\le x\,(N)\,,\\ &\qquad 0\le x(j)\le u(j)-l\,(j)\,,\ j\in N,\\ &\qquad u(j)-x(j)\le d,\ j\in N,\end{aligned} \tag{18}$$

see [9,13,27]. We may assume that the last inequality holds with $x(j)=u(j)-l(j)$, i.e., $l(j)\le d$, $j\in N$.

Let the values λ_j, $j\in N$, be found and the jobs be numbered in accordance with (12). Changing variables to $\lambda\,(j)=x\,(j)/w\,(j)$, $j\in N$, the problem can be reformulated as

$$\begin{aligned}&\min\ \max\left\{\lambda\,(j)\,|j\in N\right\}\\ &\text{s.t.}\ \ u\,(N)-md\le\sum_{j\in N}\lambda\,(j)\,w\,(j)\,,\\ &\qquad 0\le\lambda(j)\le\lambda_j,\ j\in N,\\ &\qquad u\,(j)-\lambda(j)w\,(j)\le d,\ j\in N.\end{aligned} \tag{19}$$

Similarly to Sect. 3, for problem $P|p(j)=u(j)-x(j),\, C(j)\le d,\, pmtn|\Phi_{\max}$ a value of $\lambda\in\left[\lambda_q,\lambda_{q+1}\right]$, $1\le q\le n-1$, is feasible if

$$\begin{aligned}u\,(N)-md&\le\sum_{j=1}^{q}\lambda_j w\,(j)+\lambda\sum_{j=q+1}^{n}w\,(j)\,;\\ \frac{u\,(j)-d}{w\,(j)}&\le\lambda,\ \ j=q+1,\ldots,n,\end{aligned} \tag{20}$$

where the second line guarantees that each job j, $q+1\le j\le n$, in the case of compression by $x\,(j)=\lambda w\,(j)$ will get the processing time at most d (jobs $1,\ldots,q$ are fully compressed).

Suppose that an interval $\left[\lambda_q,\lambda_{q+1}\right]$ is found such that λ_q is infeasible while λ_{q+1} is feasible. All jobs $j\le q$ are fully compressed and no other job, except for job $q+1$, may become fully compressed as long as λ changes within $\left[\lambda_q,\lambda_{q+1}\right]$. An optimal value λ^* is the smallest feasible value of $\lambda\in\left[\lambda_q,\lambda_{q+1}\right]$, which can be found by

$$\lambda^*=\max\left\{\frac{u\,(N)-md-\sum_{j=1}^{q}\lambda_j w\,(j)}{\sum_{j=q+1}^{n}w\,(j)},\ \max_{q+1\le j\le n}\left\{\frac{u\,(j)-d}{w\,(j)}\right\}\right\}. \tag{21}$$

Formula (21) is written under the assumption that the jobs are renumbered in such a way that the order of λ_j's is given by (12). Under this assumption, the search for an interval $\left[\lambda_q,\lambda_{q+1}\right]$ can be performed in $O\,(n\log n)$ time, by a straightforward adaptation of Algorithm A1.

However, as shown below, for problem $P|p(j)=u(j)-x(j),\, C(j)\le d,\, pmtn|\Phi_{\max}$ it is possible to perform the localization search in Step 4 of Procedure A in linear time without ordering the values λ_j, $1\le j\le n$, by (12) but rather keeping them in an unsorted list Λ. The corresponding algorithm is outlined below. Its running time reduces to $O\,(n)$.

The algorithm follows Procedure A and employs a search mechanism based on the median finding technique. This mechanism allows us splitting the set N of jobs into two subsets N' and N'' such that there exists an optimal schedule in which each job $j\in N'$ is fully compressed, while the weighted compression cost for each job of set N'' is equal to the optimum value λ^*. During the course of the algorithm, we keep track of set H of the candidate jobs which are not yet assigned either to N' or to N''. In each iteration for a λ equal to the median of the current list Λ set H is split into two subsets H' and H'' which are used for updating either set N' or set N''. To speed up the algorithm, we keep records of the total compression V' of the current jobs in N' and total weight W'' of the current jobs in N''.

Algorithm AP0

Step 1. Compute λ_j, $j \in N$, by (7). If necessary, renumber the jobs so that (8) holds and form an unsorted list Λ.

Step 2. Check if $\lambda_0 = 0$ is feasible. If yes, then the optimal compressions are $x^*(j) = 0$, $1 \le j \le n$, with the optimal value of the objective function $\Phi_{\max}$ equal to zero; otherwise, go to Step 3.

Step 3. Check if $\lambda = \lambda_n$ is feasible. If not, then the original problem does not have a feasible solution; otherwise define $N' := N'' := \varnothing$, $H := N$, $V' := W'' := 0$ and go to Step 4.

Step 4. Repeat until list Λ is empty:

(**a**) Find $\hat{\lambda}$, the median of the list Λ, and set

$$H' := \left\{ j \in H \mid \lambda_j \le \hat{\lambda} \right\}.$$

(**b**) Check the inequalities

$$\begin{aligned} u(N) - \Big(V' + \sum_{j \in H'} \lambda_j w(j) \Big) - \hat{\lambda} \Big(W'' + \sum_{j \in H \setminus H'} w(j) \Big) &\le md, \\ u(j) - \lambda_j w(j) &\le d, \quad j \in H', \\ u(j) - \hat{\lambda} w(j) &\le d, \quad j \in H \setminus H'. \end{aligned}$$

(**c**) If the above inequalities do not hold simultaneously (i.e., $\hat{\lambda}$ is infeasible), then update

$$V' := V' + \sum_{j \in H'} \lambda_j w(j), \quad N' := N' \cup H', \quad H := H \setminus H',$$

and remove λ_j with $j \in H'$ from the list Λ.

Otherwise (i.e., $\hat{\lambda}$ is feasible), find $H'' = \left\{ j \in H \mid \lambda_j \ge \hat{\lambda} \right\}$ and update

$$W'' := W'' + \sum_{j \in H''} w(j), \quad N'' := N'' \cup H'', \quad H := H \setminus H'',$$

and remove λ_j with $j \in H''$ from the list Λ.

Step 5. An optimal maximum compression cost λ^* is found by

$$\lambda^* = \max \left\{ \frac{u(N) - md - V'}{W''}, \max_{j \in N''} \left\{ \frac{u(j) - d}{w(j)} \right\} \right\}. \tag{22}$$

In each iteration of Step 4, let h be the number of elements in the current list Λ (or, equivalently, in set H). Then the feasibility check and the transition to the next iteration require $O(h)$ time. After each iteration, the number of jobs in the list Λ is halved. Thus, the total time needed for finding the required interval is

$$O(n) + O\left(\frac{n}{2}\right) + O\left(\frac{n}{4}\right) + \cdots = O(n).$$

Formula (22) is an adaptation of (21), so that in terms of Step 5 of Procedure A the last found infeasible λ will play the role of λ_q, while λ_{q+1} is the smallest element of the initial list Λ that is larger than λ. Finding the optimal value of λ^* in Step 5 takes $O(n)$ time.

Thus, the following statement holds.

Theorem 2 *Problem $P|p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ is solvable in $O(n)$ time.*

Similarly to Sect. 3, consider the problem of minimizing the function $\sum_{j\in N}(1/w(j))x(j)^2$ under the same constraints as in (18). Such a problem of quadratic programming is qualified as the simple resource allocation problem and can be solved in $O(n)$ time [12]. Since any optimal solution of that resource allocation problem is also an optimal solution to problem (18), we obtain an alternative linear time algorithm for problem $P|p(j) = u(j) - x(j),$ $C(j) \le d,\ pmtn|\Phi_{\max}$.

Notice that the established running time matches the running time needed for solving the feasibility problem $P|C(j) \le d,\ pmtn|-$ with constant processing times, as well as that needed for solving problem $P|p(j) = u(j) - x(j), C(j) \le d,\ pmtn|\Phi_{\Sigma}$ to minimize total compression cost; see Table 2.

5 Identical parallel machines: arbitrary release dates

The difference between problems $P|r(j),\ p(j) = u(j) - x(j), C(j) \le d,\ pmtn|\Phi_{\max}$ and $1|r(j),\ p(j) = u(j) - x(j), C(j) \le d,\ pmtn|\Phi_{\max}$ is that now we have m identical parallel machines. We can adapt and extend Algorithm A1 to handle the parallel machine problem.

Similarly to Sect. 3, given problem $P|r(j),\ p(j) = u(j) - x(j), C(j) \le d,\ pmtn|\Phi_{\max}$, let $R^{(z)}$ denote the zth smallest release date, $1 \le z \le n$.

Perform Steps 1–4 of Algorithm A1, where the feasibility check is done by running an algorithm by Sahni [22], which in $O(n \log n)$ time verifies feasibility of problem $P|r(j),$ $C(j) \le d,\ pmtn|-$ with fixed processing times. Since there are $\log n$ iterations in Step 4 of Algorithm A1, we arrive at Step 5 in $O\left(n \log^2 n\right)$ time, having found an interval $\left[\lambda_q, \lambda_{q+1}\right]$ such that λ_q is infeasible while λ_{q+1} is feasible, $0 \le q \le n-1$. Notice that unlike Algorithm AP0, for the problem with arbitrary release dates Step 4 cannot be implemented in linear time by the median-finding technique. Indeed, even if the processing times of some jobs are fixed checking the feasibility of a given λ_q requires time that is no less than linear in n.

We now explain how the optimal value of λ^* can be found in Step 5 and how such a search can be implemented in $O\left(n^2\right)$ time. Notice that even if a speed-up in Step 4 was possible it would not affect the overall running time of the algorithm.

To implement the search in Step 5 we cannot rely just on the feasibility check algorithm by Sahni [22]. We rather need to base our search on relevant analytical expressions. Recall that the necessary and sufficient conditions for the existence of a schedule that is feasible for problem $P|r(j), C(j) \le d,\ pmtn|-$ with fixed processing times $p(j)$, $j \in N$, are obtained by Horn [13]. For each job $j \in N$ and each z, $1 \le z \le n$, Horn [13] determines the value $g^{(z)}(j)$, which represents the minimum duration of the processing of job j after time $R^{(z)}$ in any feasible schedule. Such values can be found by

$$g^{(z)}(j) = \begin{cases} p(j), & \text{if } r(j) \ge R^{(z)}, \\ \max\left\{p(j) - \left(R^{(z)} - r(j)\right), 0\right\}, & \text{otherwise.} \end{cases}$$

It is proved in Theorem 2 of [13] that for problem $P|r(j), C(j) \le d, pmtn|-$ a feasible schedule exists if and only if the inequality

$$\sum_{j \in N} g^{(z)}(j) \le m\left(d - R^{(z)}\right) \tag{23}$$

holds for all z, $1 \le z \le n$, provided that

$$p(j) \le d - r(j),\ \ j \in N. \tag{24}$$

For problem $P|r(j), p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$, as λ changes within the found interval $\left[\lambda_q, \lambda_{q+1}\right]$, two types of jobs can be distinguished. First, each job j with $\lambda_j \le \lambda_q$ is fully compressed, i.e., its actual processing time $p(j)$ is equal to $l(j)$ and remains constant. Second, for the remaining jobs $p(j) = u(j) - \lambda w(j)$. Moreover, for each job j with $r(j) < R^{(z)}$, the value $g^{(z)}(j)$ computed with respect to the actual processing time $p(j)$ should be treated as a function $g^{(z)}(j, \lambda)$ which is piecewise linear in λ, with a break-point

$$\mu^{(z)}(j) = \frac{u(j) + r(j) - R^{(z)}}{w(j)}. \tag{25}$$

Let $\lambda \in \left[\lambda_q, \lambda_{q+1}\right]$ be feasible and $\lambda(j)$ be defined by (9). Then it follows from (24) that the inequality

$$\lambda(j)w(j) \ge u(j) - d + r(j)$$

holds for all j, $1 \le j \le n$. Additionally, (23) implies that the inequality

$$\sum_{j \in N} g^{(z)}(j, \lambda) \le m\left(d - R^{(z)}\right) \tag{26}$$

holds for all z, $1 \le z \le n$, where

$$g^{(z)}(j, \lambda) = \begin{cases} u(j) - \lambda(j)w(j), & \text{if } r(j) \ge R^{(z)}, \\ \max\left\{u(j) - \lambda(j)w(j) - \left(R^{(z)} - r(j)\right), 0\right\}, & \text{otherwise.} \end{cases} \tag{27}$$

Notice that in (27) the values $g^{(z)}(j, \lambda)$ depend on $\lambda(j)$, which means they depend on λ implicitly, due to (9).

Below we provide a possible implementation of the search for an optimal λ^*.

Step 5.

(a) Compute the break-points $\mu^{(z)}(j)$, $1 \le z \le n$, $1 \le j \le n$, by (25) and remove those which do not belong to the interval $\left[\lambda_q, \lambda_{q+1}\right]$. Let M be an unsorted list of the remaining μ-values.
(b) Repeat until list M contains two distinct elements:
Find $\hat{\mu}$, the median of the list M. If $\hat{\mu}$ is infeasible, then remove from list M all elements less than $\hat{\mu}$; otherwise, remove from list M all elements larger than $\hat{\mu}$.
(c) Let μ' and μ'' be the two remaining elements of list M. For $\lambda \in \left[\mu', \mu''\right]$, compute $\lambda(j)$ by (9) and $g^{(z)}(j, \lambda)$ by (27), $1 \le z \le n$, $1 \le j \le n$.
(d) Find an optimal λ^* as the smallest feasible $\lambda \in \left[\mu', \mu''\right]$. The optimal compressions $x^*(j)$ are defined by (11).

Let us analyze Step 5 described above. Computation in Step 5(a) requires $O\left(n^2\right)$ time. Step 5(b) is the median-based search similar to that employed in Algorithm AP0. The computation related to repeated median-finding in the list initially consisting of $O(n^2)$ elements takes $O(n^2)$ time for all iterations. Additionally, feasibility checks are performed for each trial μ-value by running the algorithm by Sahni [22]. Since after each iteration the number of elements in list M is halved, there are $\log\left(n^2\right) = O\left(\log n\right)$ iterations in this step to locate an interval $\left[\mu', \mu''\right] \subset \left[\lambda_q, \lambda_{q+1}\right]$ such that the value of μ' is infeasible, while the value of μ'' is feasible, so that all feasibility checks take $O\left(n \log^2 n\right)$ time. The latter estimate defines the overall time complexity of Step 5(b).

As λ changes within $\left[\mu', \mu''\right]$, each $g^{(z)}\left(j, \lambda\right)$ defined in Step 5(c) does not change its shape, i.e., remains either constant or linear in λ. All such $g^{(z)}\left(j, \lambda\right)$, $j \in N$, $1 \le z \le n$, can be found in $O\left(n^2\right)$ time. For each relevant z, $1 \le z \le n$, the sum $\sum_{j \in N} g^{(z)}\left(j, \lambda\right)$ involved in (26) is linear in λ and can be computed in $O\left(n\right)$ time as the sum of linear functions. Thus, an optimal value of λ that satisfies the condition (26) can be found in $O\left(n^2\right)$ time.

The overall running time of Step 5 is $O\left(n^2\right)$ and the following statement holds.

Theorem 3 *Problem $P|r\left(j\right), p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$ is solvable in $O\left(n^2\right)$ time.*

Notice that the established running time exceeds the running time of $O\left(n \log n\right)$ needed for solving the feasibility problem $P|r\left(j\right), C(j) \le d, pmtn|-$ with constant processing times, as well as the running time of $O\left(n \log n \log m\right)$, the best known time for solving problem $P|r\left(j\right), p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\Sigma}$ to minimize total compression cost; see Table 2.

6 Uniform parallel machines: zero release dates

Consider problem $Q|p(j) = u(j) - x(j), C(j) \le d, pmtn|\Phi_{\max}$ in which the jobs have no release dates and the machines are numbered in non-increasing order of their speeds, i.e.,

$$s_1 \ge s_2 \ge \cdots \ge s_m. \tag{28}$$

This problem can be solved by adapting the general scheme, in a similar manner as presented in Sect. 5.

Perform actions described in Steps 1–4 of Algorithm A1 (notice no finding of $R^{(z)}$ and $N^{(z)}$ is needed since all release dates are the same). The sorted list $\Lambda = \left(\lambda_0, \lambda_1, \ldots, \lambda_n\right)$ is searched until an interval $\left[\lambda_q, \lambda_{q+1}\right]$ if found, such that λ_q is infeasible while λ_{q+1} is feasible, $0 \le q \le n - 1$. The feasibility check for a trial value of λ is done by running an algorithm by Gonzalez and Sahni [8] which in $O\left(n + m \log m\right)$ time verifies feasibility of problem $Q|C(j) \le d, pmtn|-$ with fixed processing times.

In $O\left(n \log n + m \log m \log n\right)$ time we arrive at Step 5. We only need to explain how the optimal value of $\lambda^* \in \left[\lambda_q, \lambda_{q+1}\right]$ is to be found in Step 5. As in Sect. 5, for this stage of the search we need to explore the explicit feasibility conditions for the underlying problem with fixed processing times.

Such conditions for problem $Q|C(j) \le d, pmtn|-$ are well-known. Below, we present them following [2]. All jobs can be completed by time d, provided that

- Any job can be completed by time d if it is processed on the fastest machine M_1,
- For any v, $2 \le v \le m - 1$, any subset of v jobs can be completed by d on the v fastest machines $M_1, M_2, \ldots, M_v$;
- All jobs can be completed by d on all m machines.

For a taken $\lambda \in \left[\lambda_q, \lambda_{q+1}\right]$, define the compressions $x\left(j\right)$ by (10) and the actual processing times $p\left(j\right) = u\left(j\right) - x\left(j\right)$. For v, $1 \le v \le m - 1$, let $N^{[v]}\left(\lambda\right)$ be the set of v jobs with the longest processing times. Then λ is feasible if

$$p\left(N^{[v]}(\lambda)\right) \le d\sum_{i=1}^{v} s_i,\ 1 \le v \le m-1;$$
$$p(N) \le d\sum_{i=1}^{m} s_i.$$

As λ changes within $\left[\lambda_q, \lambda_{q+1}\right]$, we need to monitor (i) the sets $N^{[v]}(\lambda)$ of v longest jobs, $1 \le v \le m-1$, and (ii) the relative order of these jobs. In other words, as λ changes we need to keep the information regarding the $m-1$ longest jobs.

Consider a job $j \in N$ such that $\lambda_j > \lambda_q$, i.e., its compression $x(j) = \lambda_q w(j)$ is smaller than its largest possible compression $u(j) - l(j)$. For another job $k \in N$, also with $\lambda_k > \lambda_q$, the relative order of the actual processing times of the jobs j and k changes as λ passes the point that is the solution of the equation

$$u(j) - \lambda w(j) = u(k) - \lambda w(k).$$

On the other hand, a job $k \in N$ such that $\lambda_k \le \lambda_q$ is fully compressed and its actual time remains equal to $l(k)$ as λ changes. The relative order of the actual processing times of the jobs j and k changes as λ passes the point that is the solution of the equation

$$u(j) - \lambda w(j) = l(k).$$

Below we provide a possible implementation of the search for an optimal λ^*.

Step 5.

(a) For $j \in N$ with $\lambda_j > \lambda_q$, compute the break-points μ_{jk}, $1 \le k \le n$, $k \ne j$, by

$$\mu_{jk} = \begin{cases} \frac{u(j)-u(k)}{w(j)-w(k)} & \text{for } k \in N, k \ne j, \lambda_k > \lambda_q \\ \frac{u(j)-l(k)}{w(j)} & \text{for } k \in N, \lambda_k \le \lambda_q \end{cases} \tag{29}$$

and remove those μ-values which do not belong to the interval $\left[\lambda_q, \lambda_{q+1}\right]$. Let M be an unsorted list of the remaining μ-values.

(b) Repeat until list M contains two distinct elements:
Find $\hat{\mu}$, the median of the list M. If $\hat{\mu}$ is infeasible, then remove from list M all elements less than $\hat{\mu}$; otherwise, remove from list M all elements larger than $\hat{\mu}$.

(c) Let μ' and μ'' be the two remaining elements of list M. Find an optimal λ^* as the smallest feasible $\lambda \in \left[\mu', \mu''\right]$ by

$$\lambda^* = \max\left\{\tilde{\lambda}, \bar{\lambda}\right\},$$

where

$$\tilde{\lambda} := \max_{1\le v\le m-1}\left\{\frac{u\left(N^{(v)}(\tilde{\mu})\right) - d\sum_{i=1}^{v} s_i - \sum_{j\in\{1,\dots,q\}\cap N^{[v]}(\tilde{\mu})} \lambda_j w(j)}{\sum_{j\in\{q+1,\dots,n\}\cap N^{[v]}(\tilde{\mu})} w(j)}\right\},$$
$$\bar{\lambda} := \frac{u(N) - d\sum_{i=1}^{m} s_i - \sum_{j=1}^{q} \lambda_j w(j)}{\sum_{j=q+1}^{n} w(j)},$$

and $\tilde{\mu}$ is the midpoint of the interval $\left[\mu', \mu''\right]$. The optimal compressions $x^*(j)$ are defined by (11).

Step 5 described above is similar to Step 5 presented in Sect. 5. Computation in Step 5(a) requires $O\left(n^2\right)$ time. Step 5(b) is the median-based search, which requires $O(n^2)$ time for all iterations. The feasibility check of each trial μ-value is performed by running an algorithm by Gonzalez and Sahni [8]. Since there are $O(\log n)$ iterations, finding an interval $\left[\mu', \mu''\right] \subset \left[\lambda_q, \lambda_{q+1}\right]$ such that the value of μ' is infeasible, while the value of μ'' is feasible takes $O\left(n^2 + m\log m \log n\right)$ time.

As λ changes within the interval $\left(\mu', \mu''\right]$ the sets $N^{[v]}(\lambda)$ of v longest jobs do not change. In the formula for computing λ^* in Step 5(c) any value of $\mu \in \left(\mu', \mu''\right)$ can be used, and for certainty we have selected $\tilde{\mu}$ as the midpoint of the interval $\left[\mu', \mu''\right]$.

The overall running time of Step 5 is $O\left(n^2\right)$. Thus, the following statement holds.

Theorem 4 *Problem $Q|p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ is solvable in $O\left(n^2\right)$ time.*

The established running time can be reduced by making an observation that not all $O\left(n^2\right)$ breakpoints μ_{jk} computed in Step 5(a) are needed, but only those which affect a relative order of the v longest jobs for $1 \le v \le m-1$. Finding these relevant breakpoints can be reduced to the following problem of computational geometry. As observed earlier, after the interval $\left[\lambda_q, \lambda_{q+1}\right]$ is located, each job $j \in N$ can belong to one of the following categories: (i) incompressible with the actual processing time of $l(j)$, and (ii) compressible with the actual time of $u(j) - \lambda w(j)$. A job can be associated with a straight line on the plane, with λ as the independent variable. The relevant break-points are in fact those intersection points of these straight lines that at most $m-1$ straight lines are located above them. The problem of finding all such intersection points is known as the problem of determining $(\le k)$-levels for an arrangement of lines; see [1] and [3]. In particular, adapting the algorithm from [3] to our purposes, we can replace Step 5(a) above by finding the $(\le m-1)$-levels for the n lines associated with the jobs. As a result, we will obtain the list M of the relevant break-points of the form (29) in $O(n\log n + nm)$ time, the number of these break-points being $O(nm)$. The median-based search in Step 5(b) requires $O(nm)$ time. Thus, the overall running time that is needed for solving problem $Q|p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ reduces to $O(n\log n + nm)$.

Theorem 5 *Problem $Q|p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ is solvable in $O(n\log n + nm)$ time, provided that the method of determining $(\le k)$-levels is employed.*

The resulting running time exceeds the running time of $O(n + m\log m)$ needed for solving the feasibility problem $Q|C(j) \le d,\ pmtn|-$ with constant processing times, as well as the running time of $O(\min\{n\log n, n + m\log m\log n\})$, the best known time for solving problem $Q|p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\Sigma}$ to minimize total compression cost; see Table 2.

7 Uniform parallel machines: arbitrary release dates

In this section, we consider problem $Q|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ in which the jobs have release dates and the machines are numbered according to (28).

Below, we outline a rather universal approach to finding an optimal value λ^* of the objective function for a generic problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$. In principle, this approach can be used to solve any of the problems of the range under consideration. However, the SCPT problems from Sects. 3–6 due to their relative simplicity can be handled by faster purpose-built algorithms.

Consider problem $\alpha|r(j),\ p(j) = u(j) - x(j),\ C(j) \leq d,\ pmtn|\Phi_{\max}$ formulated by (6) with respect to an appropriate capacity function φ. Suppose that the actions described in Steps 1–4 of Procedure A have been performed and we have localized an interval $\left[\lambda', \lambda''\right]$ such that λ' is infeasible while λ'' is a feasible value of λ. For $\lambda = \lambda'$, let U denote the set of the incompressible jobs j with the actual processing time $l\,(j)$. The jobs of set $N \backslash U$ are compressible, and the actual processing time of job $j \in N\backslash U$ is equal to $u\,(j) - \lambda w\,(j)$.

For the sake of simplicity of the algorithm description, we redefine

$$w(j) = 0, \quad u(j) = l(j) \quad \text{for each incompressible job } j \in U, \tag{30}$$

so that $u\,(j) - \lambda w\,(j) = l(j)$ holds for $j \in U$ and $\lambda \in [\lambda', \lambda'']$. This also implies that $w\,(Y) = w(Y\backslash U)$ holds for any set $Y \subseteq N$. Therefore, $\lambda \in [\lambda', \lambda'']$ is feasible if and only if

$$\lambda w(Y) - u\,(Y) + \varphi\,(Y) \geq 0 \qquad \text{for any } Y \subseteq N. \tag{31}$$

The value λ^* (the optimal value of λ) can be defined as the minimum feasible value of $\lambda \in \left[\lambda', \lambda''\right]$ such that

$$\lambda^* = \min\left\{\lambda \in \left[\lambda', \lambda''\right] \mid \lambda w(Y) - u\,(Y) + \varphi\,(Y) \geq 0,\ Y \subseteq N\right\}.$$

In other words, λ^* is such a value within the interval $\left[\lambda', \lambda''\right]$ that

$$\lambda^* = \max_{Y \subseteq N} \frac{u(Y) - \varphi(Y)}{w(Y)}.$$

For a set $Y \subseteq N\backslash U$, introduce a parametric set-function

$$f\,(Y, \lambda) = \lambda w(Y) - u\,(Y) + \varphi\,(Y)\,, \quad Y \subseteq N,\ \lambda \in [\lambda', \lambda''].$$

It follows from (31) that $\lambda \in [\lambda', \lambda'']$ is feasible if and only if for each set $Y \subseteq N$ the inequality $f\,(Y, \lambda) \geq 0$ holds. Therefore, we need to find the smallest value $\lambda^* \in \left[\lambda', \lambda''\right]$ such that

$$\min_{Y \subseteq N} f\left(Y, \lambda^*\right) = 0.$$

The required λ^* can be found by a Newton-like search procedure, similar to that presented in [7].

Procedure Newton

Step 1. Take $\lambda^{(1)} = \lambda'$. Find set U of incompressible jobs, and redefine $w(j)$ and $u(j)$ for $j \in U$ by (30). Set $k = 1$.

Step 2. Find the minimal (with respect to inclusion) set-minimizer $Y^{(k)}$ such that

$$f(Y^{(k)}, \lambda^{(k)}) = \min\left\{\lambda^{(k)} w\,(Y) - u(Y) + \varphi\,(Y) \mid Y \subseteq N\right\}.$$

Step 3. If $f\left(Y^{(k)}, \lambda^{(k)}\right) = 0$ then output $\lambda^* = \lambda^{(k)}$ and stop. Otherwise, find

$$\lambda^{(k+1)} = \frac{u(Y^{(k)}) - \varphi(Y^{(k)})}{w(Y^{(k)})}$$

such that $f\left(Y^{(k)}, \lambda^{(k+1)}\right) = 0$. Set $k := k + 1$ and go back to Step 2.

Due to the minimality of the set-minimizers in Step 2, we have that $N \supseteq Y^{(k)} \supset Y^{(k+1)}$ for each $k \geq 1$, so that the procedure consists of at most n iterations. Since the capacity function $\varphi\,(Y)$ is submodular for all problems of the range under consideration, the search for a set-minimizer $Y^{(k)}$ in Step 2 reduces to minimizing a submodular function, which in general can be done in polynomial time, see [14,24].

We now describe how to implement this approach for solving problem $Q|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$, the most general problem from the range under consideration of the SCPT problems with a common deadline.

Run Steps 1–4 of Procedure A, or rather of Algorithm A1, where the feasibility check is done by running an algorithm by Sahni and Cho [23] which in $O(nm + n\log n)$ time verifies feasibility of problem $Q|r(j),\ C(j) \le d,\ pmtn|-$ with fixed processing times.

In $O\left(nm\log n + n\log^2 n\right)$ time we arrive at Step 5, having found an interval $\left[\lambda_q, \lambda_{q+1}\right]$ such that λ_q is infeasible while λ_{q+1} is feasible, $0 \le q \le n-1$. We only need to explain how the optimal value of λ^* is to be found in Step 5, where we use Procedure Newton.

As in Sects. 3 and 5, let $R^{(z)}$ denote the zth smallest release date. Extending this notation, for a set $Y \subseteq N$ define $R^{(z)}(Y)$ as the z th smallest release date among the jobs of set Y. For a set $Y \subseteq N$, denote $m_Y = \min\{m, |Y|\}$.

Let the machines be numbered by (28). As follows from [28,29], the capacity set-function that defines the largest capacity for preemptive processing a set Y of jobs on m uniform parallel machines can be written as

$$\varphi(Y) = \sum_{z=1}^{m_Y} s_z\left(d - R^{(z)}(Y)\right).$$

Let U be the set of incompressible jobs found with respect to $\lambda = \lambda_q$. In Step 2 of an iteration k in Procedure Newton, we need to minimize $\lambda^{(k)}w(Y) - u(Y) + \varphi(Y)$. Let each job $j \in N$ be associated with a real number

$$\gamma(j) = \lambda^{(k)}w(j) - u(j).$$

For a set $Y \subseteq N$, define a set-function $\psi(Y) = \gamma(Y) + \varphi(Y)$, which is submodular, since function φ is submodular. It is shown in [29] that a set-minimizer Y^* of the function ψ can be found in $O(nm)$ time, provided that the jobs are numbered in the non-decreasing order of their release dates. In order to guarantee minimality of the minimizer Y^*, instead of minimizing $\psi(Y)$ we minimize a function $\psi^{\varepsilon}(Y) := \psi(Y) + \varepsilon|Y|$ for an arbitrary small positive ε. A set-minimizer for function $\psi^{\varepsilon}(Y)$ is a minimal set-minimizer for function $\psi(Y)$. Such a modification does not affect the time needed for finding the set-minimizer. Time needed to find λ^* in Step 5 by employing Procedure Newton is $O\left(mn^2\right)$, and this determines the overall running time for solving problem $Q|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$.

Thus, the following statement holds.

Theorem 6 *Problem $Q|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\max}$ is solvable in $O\left(mn^2\right)$ time.*

Notice that the established running time exceeds the running time needed for solving the feasibility problem $Q|r(j),\ C(j) \le d,\ pmtn|-$ with constant processing times, as well as that needed for solving problem $Q|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\Sigma}$ to minimize the total compression cost; see Table 2. On the other hand, $O\left(mn^2\right)$ is less than the time needed for solving problem $Q|r(j),\ p(j) = u(j) - x(j),\ C(j) \le d(j),\ pmtn|\Phi_{\max}$ with arbitrary due dates; see Table 1.

8 Conclusions

In this paper we address a range of open problems arising in SCPT. The traditional stream of research on SCPT focuses on the Φ_Σ objectives. The research on the SIC models does consider $\Phi_{\max}$ but in the most general setting, when all jobs have arbitrary release dates and arbitrary deadlines. The subject of our study are the models with $\Phi_{\max}$ under an assumption that all jobs have a common deadline.

The general methodology we propose stems from the special representation of the solution region as a submodular polyhedron intersected with a box. It can be efficiently implemented for all versions of the problem under study, generically denoted by $\alpha|r(j), p(j) = u(j) - x(j), C(j) \leq d, pmtn|\Phi_{\max}$ for $\alpha \in \{1, P, Q\}$ (the models with a single machine, identical parallel machines and uniform machines).

For the two models, one with $\alpha = 1$ and arbitrary r_j and another with $\alpha = P$ and $r_j = 0$ for all jobs, our approach results in the fastest possible algorithms, with the same running times as the algorithms for the counterparts with fixed processing times, see the first two lines of Table 2. For the remaining three models the complexity gap is reduced, in comparison with the results from [10,18], but not closed. A similar phenomenon is observed for the problems of minimizing the the Φ_Σ objective. It remains to be seen whether further reduction of the complexity gap is possible. In particular, it is a challenging goal to verify whether any of the problems $P|r(j), p(j) = u(j) - x(j), C(j) \leq d, pmtn|\Phi_{\max}$ and $P|r(j), p(j) = u(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$ admits an $O(n \log n)$-time algorithm.

Acknowledgements 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 JSPS KAKENHI Grant Nos. 15K00030 and 15H00848.

References

1. Alon, N., Győri, E.: The number of small semispaces of a finite set of points in the plane. J. Comb. Optim. A **41**, 154–157 (1986)
2. Brucker, P.: Scheduling Algorithms, 5th edn. Springer, Berlin (2007)
3. Everett, H., Robert, J.-M., van Kreveld, M.: An optimal algorithm for computing ($\leq k$)-levels, with applications. Int. J. Comput. Geom. Appl. **6**, 247–261 (1996)
4. Federgruen, A., Groenevelt, H.: Preemptive scheduling of uniform machines by ordinary network flow techniques. Manag. Sci. **32**, 341–349 (1986)
5. Fujishige, S.: Lexicographically optimal base of a polymatroid with respect to a weight vector. Math. Oper. Res. **5**, 186–196 (1980)
6. Fujishige, S.: Submodular Functions and Optimization. Annals of Discrete Mathematics, vol. 58, 2nd edn. Elsevier, New York City (2005)
7. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**, 30–55 (1989)
8. Gonzales, T.F., Sahni, S.: Preemptive scheduling of uniform processor systems. J. ACM **25**, 92–101 (1978)
9. Gordon, V.S., Tanaev, V.S.: Deadlines in single-stage deterministic scheduling. In: Optimization of Systems for Collecting, Transfer and Processing of Analogous and Discrete Data in Local Information Computing Systems. Materials of the 1st Joint Soviet-Bulgarian Seminar (Institute of Engineering Cybernetics of Academy of Sciences of BSSR—Institute of Engineering Cybernetics of Bulgarian Academy of Sciences, Minsk), pp. 53–58 (1973) **(in Russian)**

10. Ho, K.I.-J.: Dual criteria optimization problems for imprecise computation tasks. In: Leung, J.Y.-T. (ed.) Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 35-1–35-26. CRC Press, Boca Raton (2004)
11. Ho, K.I.-J., Leung, J.Y.-T., Wei, W.-D.: Minimizing maximum weighted error for imprecise computation tasks. J. Algorithms **16**, 431–452 (1994)
12. Hochbaum, D.S., Hong, S.-P.: About strongly polynomial time algorithms for quadratic optimization over submodular constraints. Math. Program. A **69**, 269–309 (1995)
13. Horn, W.: Some simple scheduling algorithms. Naval Res. Logist. Quart. **21**, 177–185 (1974)
14. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. J. ACM **48**, 761–777 (2001)
15. Janiak, A., Kovalyov, M.Y.: Single machine scheduling with deadlines and resource dependent processing times. Eur. J. Oper. Res. **94**, 284–291 (1996)
16. Jansen, K., Mastrolilli, M.: Approximation schemes for parallel machine scheduling problems with controllable processing times. Comput. Oper. Res. **31**, 1565–1581 (2004)
17. 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)
18. Leung, J.Y.-T.: Minimizing total weighted error for imprecise computation tasks. In: Leung, J.Y.-T. (ed.) Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 34-1–34-16. CRC Press, Boca Raton (2004)
19. McCormick, S.T.: Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. Oper. Res. **47**, 744–756 (1999)
20. McNaughton, R.: Scheduling with deadlines and loss functions. Manag. Sci. **12**, 1–12 (1959)
21. Nowicki, E., Zdrzałka, S.: A survey of results for sequencing problems with controllable processing times. Discrete Appl. Math. **26**, 271–287 (1990)
22. Sahni, S.: Preemptive scheduling with due dates. Oper. Res. **27**, 925–934 (1979)
23. Sahni, S., Cho, Y.: Scheduling independent tasks with due times on a uniform processor system. J. ACM **27**, 550–563 (1980)
24. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theory B **80**, 346–355 (2000)
25. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003)
26. Shabtay, D., Steiner, G.: A survey of scheduling with controllable processing times. Discrete Appl. Math. **155**, 1643–1666 (2007)
27. Shakhlevich, N.V., Strusevich, V.A.: Pre-emptive scheduling problems with controllable processing times. J. Sched. **8**, 233–253 (2005)
28. 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)
29. Shioura, A., Shakhlevich, N.V., Strusevich, V.A.: Decomposition algorithms for submodular optimization with applications to parallel machine scheduling with controllable processing times. Math. Program. A **153**, 495–534 (2015)
30. Shioura, A., Shakhlevich, N.V., Strusevich, V.A.: Application of submodular optimization to single machine scheduling with controllable processing times subject to release dates and deadlines. INFORMS J. Comput. **28**, 148–161 (2016)
31. Shioura, A., Shakhlevich, N.V., Strusevich, V.A.: Preemptive models of scheduling with controllable processing times and of scheduling with imprecise computation: a review of solution approaches. Eur. J. Oper. Res. **266**, 795–818 (2018)