Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Applied Mathematics 154 (2006) 2178–2199

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Single machine scheduling with controllable release and processing parameters

Natalia V. Shakhlevich[a], Vitaly A. Strusevich[b]

[a]*School of Computing, University of Leeds, Leeds LS2 9JT, UK*

[b]*School of Computing and Mathematical Sciences, University of Greenwich, Old Royal Naval College, Park Row, London SE10 9LS, UK*

Received 24 December 2002; received in revised form 17 September 2004; accepted 21 April 2005

Available online 23 March 2006

Abstract

This paper considers single machine scheduling problems in which the job processing times and/or their release dates are controllable. Possible changes to the controllable parameters are either individual or done by controlling the relevant processing or release rate. The objective is to minimize the sum of the makespan plus the cost for changing the parameters. For the problems of this type, we provide a number of polynomial-time algorithms and give a fairly complete complexity classification.

MSC: Primary 90B35; secondary 90B30;90B06

Keywords: Single machine scheduling; Controllable processing times; Controllable processing speeds; Controllable release dates; Controllable release speeds

1. Introduction

We consider a series of single machine scheduling problems that may serve as mathematical models that arise in supply chain scheduling. Various aspects of supply chain scheduling have been systematically studied by Hall and his coauthors, see [1,4,8,9]. These papers address the issues of coordination and cooperation between the elements of the supply chain and analyze operational decisions taken in deterministic machine environment (an assembly model, a manufacturer with several suppliers, etc.).

Our generic model concentrates on a segment of the supply chain that includes three participants: a *supplier* at the top of the chain, a *manufacturer* in the middle and a *customer* at the bottom. The manufacturer's production programme includes the processing of many jobs which become available after they are released by the supplier. The programme has to be completed in full in order to satisfy the customer's demand. See Fig. 1.

A straightforward approach to the manufacturer's problem leads to the following single machine scheduling problem. Each job j of set $N = \{1, 2, \ldots, n\}$ becomes available at time r_j and is processed without preemption, and this processing takes p_j time units. The manufacturer's goal is to complete all jobs as early as possible, i.e., to minimize the maximum completion time or the *makespan* $C_{\max}$. Following standard scheduling notation [16], we denote this basic

E-mail addresses: ns@comp.leeds.ac.uk (N.V. Shakhlevich), V.Strusevich@greenwich.ac.uk (V.A. Strusevich).

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.04.014

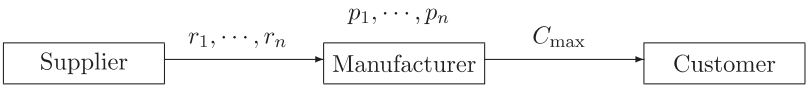

Fig. 1. Supply chain and controllable parameters.

problem by $1|r_j|C_{\max}$. Let the jobs be numbered in non-decreasing order of their release dates, i.e.,

$$r_1 \leqslant r_2 \leqslant \cdots \leqslant r_n. \tag{1}$$

It is well-known that an optimal schedule S^* is associated with the job sequence $(1, 2, \ldots, n)$ and the optimal makespan $C_{\max}(S^*)$ is given by

$$C_{\max}(S^*) = \max_{1 \leqslant u \leqslant n} \left\{ r_u + \sum_{j=u}^{n} p_j \right\}. \tag{2}$$

Thus, problem $1|r_j|C_{\max}$ is solvable in $O(n \log n)$ time. A job for which the maximum in the right-hand side of (2) is achieved is called *critical*. A schedule may contain several critical jobs. A critical job starts processing as soon as it is released, and that processing cannot be delayed without increasing the overall makespan.

From the point of view of the supply chain in Fig. 1, problem $1|r_j|C_{\max}$ arises when the manufacturer (i) accepts the job release dates $r_1, \ldots, r_n$ established by the supplier, and (ii) does not alter the processing times $p_1, \ldots, p_n$. If the customer is willing to receive the completed jobs from the manufacturer earlier, i.e., is interested in reducing the makespan $C_{\max}$, he may want to encourage the manufacturer to revise the parameters of the production programme. If the manufacturer is ready to cooperate, the customer is prepared to meet or to share the incurred expenses to create a so-called "win–win" situation. The options open to the manufacturer in revising his programme can be characterized as internal and external.

Internally, the manufacturer may put additional effort in order to reduce the job processing times. In this paper, we consider two ways of changing processing times: to alter the times simultaneously for all jobs by means of speeding up the machine or to amend the processing times individually, on a job-by-job basis. Anyway, these changes require consumption of additional resources (energy, workforce, advanced equipment or materials, etc.) allocated either to the machine or to individual jobs, and this incurs additional cost.

It may appear profitable to try external alterations to the production programme. This means that the manufacturer may want to negotiate possible changes of job release dates with the supplier. Similarly, the release dates can be modified either simultaneously for all jobs by controlling the release speed or individually, job by job. Note that in the latter case not only the values of job release dates are subject to change but also the sequence in which the jobs become available to the manufacturer may become different.

We now formally describe how the parameters can be controlled and define the associated cost functions:

- *Controllable processing speed*: For the manufacturer, if the processing speed of the machine is equal to v_p then the processing time of job $j \in N$ equals $w_p p_j$, where p_j is a given 'standard' value of the processing time and $w_p = 1/v_p$ is the processing rate. The associated cost is given by the polynomial $c_p v_p^q$, where c_p is a given positive constant, while q is a given positive integer.
- *Controllable processing times*: For the manufacturer, we are given 'standard' processing times $\bar{p}_j$ that can be crashed down to the minimum value $\underline{p}_j$, where $\underline{p}_j \leqslant \bar{p}_j$. Crashing $\bar{p}_j$ to some actual processing time p_j, $\underline{p}_j \leqslant p_j \leqslant \bar{p}_j$, may decrease the makespan but incurs additional cost $\alpha_j x_j$, where $x_j = \bar{p}_j - p_j$ is the compression amount of job j. The associated cost is given by the linear function $\sum_{j \in N} \alpha_j x_j$.
- *Controllable release speed*: If the speed at which the supplier releases the jobs is equal to v_r, then the release date of job $j \in N$ equals $w_r r_j$, where r_j is a given 'standard' value of the job release date and $w_r = 1/v_r$ is the release rate. The associated cost is given by $c_r v_r^q$, where c_r is a given positive constant, while q is a given positive integer.
- *Controllable release dates*: Under 'standard' conditions, the supplier releases job $j \in N$ at time $\bar{r}_j$ which can be reduced to the minimum value $\underline{r}_j$, where $\underline{r}_j \leqslant \bar{r}_j$. Reducing $\bar{r}_j$ to some actual value r_j, $\underline{r}_j \leqslant r_j \leqslant \bar{r}_j$, may decrease

the makespan for the manufacturer but incurs additional cost $\beta_j y_j$, where $y_j = \bar{r}_j - r_j$ is the compression amount of the corresponding release date. The associated cost is given by the linear function $\sum_{j\in N} \beta_j y_j$.

The resulting makespan and the associated costs are the components that can be used to measure the quality of decisions taken by all participants of the chain. Since efficacy of supply chain planning and scheduling is strengthened by information sharing and coordination, it is reasonable to minimize the total cost represented by the sum of all these components.

Note that the models studied in this paper combine both simultaneous and individual changes of the processing and release parameters. So far, in scheduling literature these types of control have been treated separately. The problems with controllable machine speeds have been considered mostly for the two-machine shop scheduling problems, see [11,12,22,23]. The survey of scheduling problems with controllable processing times is given in [20]. A more general problem with controllable processing times and fixed release times and delivery times is studied in [19]. Scheduling problems with controllable release dates considered in [13,17]. The single-machine problem in which both processing times and release dates are controllable is studied in [5].

2. Controllable release speed

In this section, we assume that the speed v_r at which the jobs are released can be controlled. In terms of the supply chain in Fig. 1, this problem corresponds to the situation that the manufacturer wishes to reduce the makespan by encouraging the supplier to speed up the job release.

2.1. Controllable release and processing speeds

Additionally, assume that the manufacturer is prepared to change the processing speed v_p. For fixed release speed v_r and fixed processing speed v_p, let $C_{\max}(v_r, v_p)$ denote the optimal makespan. Consider the single machine problem of finding the values of speeds v_r^* and v_p^* that minimize the function

$$F(v_r, v_p) = c_0(C_{\max}(v_r, v_p))^{q_1} + c_r v_r^{q_2} + c_p v_p^{q_2}, \tag{3}$$

where c_0, c_r and c_p are given positive constants, while q_1 and q_2 are given positive integers. In what follows, we assume that $c_0 = 1$; otherwise, the coefficients could be appropriately normalized. The last two components of function (3) can be seen as the costs for speeding up the job release and selecting the speed of the manufacturer's machine. Function (3) takes into account both the advantage of decreasing the makespan due to increasing speeds and the costs at which this is achieved. The cost components $c_r v_r^{q_2}$ and $c_p v_p^{q_2}$ are used as negotiation tools to encourage the relevant participants of the chain to alter the standard release and/or processing conditions.

Provided that the jobs are numbered according to (1), by introducing the release rate $w_r = 1/v_r$ and the production rate $w_p = 1/v_p$, we define $t(w_r, w_p) = C_{\max}(v_r, v_p)$, so that

$$t(w_r, w_p) = \max_{1\leqslant u\leqslant n} \left\{ w_r r_u + \sum_{j=u}^{n} w_p p_j \right\}.$$

To minimize function (3), we use the approach first developed by Ishii and Nishida [12] for the two-machine open shop scheduling problem with controllable machine speeds, and later successfully applied to other problems of this type, see [11,22,23].

Define $\gamma = w_r/w_p = v_p/v_r$ and rewrite $t(w_r, w_p)$ as $w_p t(\gamma)$, where

$$t(\gamma) = \max_{1\leqslant u\leqslant n} \left\{ \gamma r_u + \sum_{j=u}^{n} p_j \right\}. \tag{4}$$

Function $t(\gamma)$ is a piece-wise linear function. Let $\gamma_0, \gamma_1, \ldots, \gamma_m$ be the sequence of its breakpoints, such that $0 = \gamma_0 < \gamma_1 < \cdots < \gamma_m < +\infty$ and for each k, $1 \leqslant k \leqslant m+1$, the same job $u = u(k)$ is critical for all $\gamma \in (\gamma_{k-1}, \gamma_k]$. The breakpoints $\gamma_0, \gamma_1, \ldots, \gamma_m$ can be found in $O(n \log n)$ time by the algorithm developed by Meggido [18] for finding an

upper convex envelope of a piece-wise linear function. Additionally, define $\gamma_{m+1} = W$, where $W > \gamma_m$ is a sufficiently large number. For each k, $1 \leqslant k \leqslant m+1$, we have that

$$t(\gamma) = \gamma R(k) + P(k),$$

where $R(k)$ is the release date of the critical job $u(k)$, and $P(k)$ is the partial sum of all processing times of the jobs from $u(k)$ till n, inclusive.

For each k, $1 \leqslant k \leqslant m+1$, introduce the function

$$G_k(\gamma, w_p) = w_p^{q_1}(\gamma R(k) + P(k))^{q_1} + w_p^{-q_2}(c_r\gamma^{-q_2} + c_p).$$

The original problem of minimizing function (3) is reduced to the sequence of subproblems P_k for $k=1, 2, \ldots, m+1$, where subproblem P_k consists in minimizing the function $G_k(\gamma, w_p)$ for $\gamma \in (\gamma_{k-1}, \gamma_k]$ and $w_p > 0$. Since function $t(\gamma)$ is continuous at each point γ_k for $k > 1$, and $G_1(\gamma, w_p) \to \infty$ as γ approaches zero, we may conclude that in subproblem P_k function $G_k(\gamma, w_p)$ can be minimized over the closed interval $[\gamma_{k-1}, \gamma_k]$, rather than over the semi-open interval $(\gamma_{k-1}, \gamma_k]$.

Let $\gamma^*(k)$ and $w_p^*(k)$ be the values of γ and w_p, respectively, that minimize function $G_k(\gamma, w_p)$ over $[\gamma_{k-1}, \gamma_k]$. To find these values, we follow the idea of Ishii and Nishida [12] who base their analysis on the classical inequality of the arithmetic and geometric means:

$$\frac{1}{h}\sum_{i=1}^{h} z_i \geqslant \left(\prod_{i=1}^{h} z_i\right)^{1/h},$$

where all z_i are non-negative, and the equality is reached if and only if all z_i are equal.

Split the term $w_p^{q_1}(\gamma R(k) + P(k))^{q_1}$ into q_2 equal summands, and split the other term $w_p^{-q_2}(c_r\gamma^{-q_2} + c_p)$ into q_1 equal summands. We obtain that

$$G_k(\gamma, w_p) \geqslant (q_1+q_2)\left[\left(\frac{1}{q_2}\right)^{q_2}(\gamma R(k) + P(k))^{q_1 q_2}\left(\frac{1}{q_1}\right)^{q_1}(c_r\gamma^{-q_2} + c_p)^{q_1}\right]^{1/(q_1+q_2)}. \tag{5}$$

To find the optimal value $\gamma^*(k)$ we minimize the right-hand side of the above inequality. The optimal value $w_p^*(k)$ of w_p is found as the root of the equation

$$\frac{w_p^{q_1}(\gamma^*(k)R(k) + P(k))^{q_1}}{q_2} = \frac{w_p^{-q_2}[c_r(\gamma^*(k))^{-q_2} + c_p]}{q_1}$$

to guarantee the equality in (5).

It can be seen from (5) that the value $\gamma^*(k)$ can be found by minimizing the function $(\gamma R(k) + P(k))^{q_2}(c_r\gamma^{-q_2} + c_p)$ over $[\gamma_{k-1}, \gamma_k]$. The required minimum is either at one of the endpoints of the interval or at the internal stationary point. Thus, we may apply the following procedure.

Procedure Min(G_k):

1. Compute
$$Q = \left(\frac{c_r P(k)}{c_p R(k)}\right)^{1/(q_2+1)}.$$
2. Find
$$\gamma^*(k) = \begin{cases} \gamma_{k-1} & \text{if } \gamma_{k-1} \geqslant Q, \\ \gamma_k & \text{if } \gamma_k \leqslant Q, \\ Q & \text{if } \gamma_{k-1} < Q < \gamma_k. \end{cases}$$
3. Compute
$$w_p^*(k) = \left[\frac{q_2(c_r\gamma^*(k)^{-q_2} + c_p)}{q_1(\gamma^*(k)R(k) + P(k))^{q_1}}\right]^{1/(q_1+q_2)}$$
and stop.

We are now ready to present the complete algorithm for solving the single machine problem with controllable release and processing speeds.

Algorithm CRS/CPS.

1. If necessary, renumber the jobs so that (1) holds.
2. Run the algorithm by Meggido [18] and find all breakpoints $\gamma_0, \gamma_1, \ldots, \gamma_m$ of function $t(\gamma)$. Define $\gamma_{m+1} = W$, where $W > \gamma_m$ is a sufficiently large number. For each k, $1 \leqslant k \leqslant m+1$, determine the values $R(k)$ and $P(k)$.
3. For each k, $1 \leqslant k \leqslant m+1$, run Procedure Min$(G_k)$ and find the values $\gamma^*(k)$ and $w_p^*(k)$. Compute $G_k(\gamma^*(k), w_p^*(k))$.
4. Find the values γ^* and w_p^* such that $\gamma^* = \gamma^*(l)$ and $w_p^* = w_p^*(l)$ for some l, $1 \leqslant l \leqslant m+1$, where

$$G_l(\gamma^*(l), w_p^*(l)) = \min_{1 \leqslant k \leqslant m+1} G_k(\gamma^*(k), w_p^*(k)).$$

5. Define $v_p^* = 1/w_p^*$ and $v_r^* = v_p^*/\gamma^*$.

Step 1 of Algorithm CRS/CPS requires at most $O(n \log n)$ time. Finding the breakpoints in Step 2 takes $O(n \log n)$ time, while the values $R(k)$ and $P(k)$ for all k, $1 \leqslant k \leqslant m+1$, can be found in $O(n)$ time. In Step 3, for each k, $1 \leqslant k \leqslant m+1$, finding the values $\gamma^*(k)$, $w_p^*(k)$ and $G_k(\gamma^*(k), w_p^*(k))$ requires constant time, provided that the power and root operations can be implemented in constant time. Thus, Step 3 takes $O(m) = O(n)$ time, and so does Step 4. Therefore, the overall time complexity of Algorithm CRS/CPS is $O(n \log n)$, which cannot be improved, since solving the problem of minimizing the makespan with constant speeds needs that much time.

2.2. Controllable release speed and controllable job processing times

In this subsection we consider the situation that the supplier has agreed to increase the release speed, while the manufacturer is prepared to reduce individual job processing times. The objective is to determine the optimum release speed v_r and the job compression amounts $x_1, \ldots, x_n$ that minimize the function

$$F(v_r, x_1, \ldots, x_n) = C_{\max}(v_r, x_1, \ldots, x_n) + c_r v_r^q + \sum_{j=1}^{n} \alpha_j x_j, \tag{6}$$

where $C_{\max}(v_r, x_1, \ldots, x_n)$ is the makespan, c_r is a given positive constant, and q is a given positive integer. It is easy to verify that in an associated optimal schedule the jobs follow their numbering given by (1). Note that in the case of a constant release speed the problem can be solved in $O(n^2)$ time; see Van Wassenhove and Baker [24], Nowicki and Zdrzałka [20].

Define the release rate as $w = 1/v_r$. Using this new variable, introduce the function

$$t(w, x_1, \ldots, x_n) = C_{\max}(v_r, x_1, \ldots, x_n).$$

It follows that

$$t(w, x_1, \ldots, x_n) = \max_{1 \leqslant i \leqslant n} \left\{ w r_i + \sum_{j=i}^{n} (\bar{p}_j - x_j) \right\}.$$

Recall that job u such that

$$t(w, x_1, \ldots, x_n) = w r_u + \sum_{j=u}^{n} (\bar{p}_j - x_j)$$

is called critical.

A critical job need not be unique. For a given release rate w and the actual processing times $p_j = \bar{p}_j - x_j$, every critical job u starts exactly at its release date $w r_u$.

Fix a value of w, and determine the first critical job u_0, provided that all jobs remain uncrashed. The makespan of the corresponding schedule is equal to

$$t(w, 0, \ldots, 0) = wr_{u_0} + \sum_{j=u_0}^{n} \bar{p}_j.$$

For various values of w a critical job (or a set of critical jobs) may change. However, if w varies within certain limits so that the first critical job u_0 remains the same, then the jobs $1, \ldots, u_0 - 1$ have no influence on the value of makespan and they remain uncrashed in an optimal schedule. Moreover, as u_0 remains critical, the jobs $u_0, u_0 + 1, \ldots, n$ are processed in an optimal schedule as a block, without intermediate idle time. Note that jobs $u_0, u_0 + 1, \ldots, n$ may be subject to compression. For a fixed value of w we can determine the compression amounts x_j for $j = u_0, \ldots, n$ such that job u_0 still remains critical, i.e., the makespan of the corresponding schedule is given by

$$t(w, x_1, \ldots, x_n) = wr_{u_0} + \sum_{j=u_0}^{n} (\bar{p}_j - x_j).$$

Define the function

$$\Phi(w, x_1, \ldots, x_n) = \left(wr_{u_0} + \sum_{j=u_0}^{n} (\bar{p}_j - x_j) \right) + c_r w^{-q} + \sum_{j=u_0}^{n} \alpha_j x_j. \tag{7}$$

We are interested in minimizing $\Phi(w, x_1, \ldots, x_n)$, provided that w and x_j are subject to change while job u_0 remains critical. Since in this case $x_1 = \cdots = x_{u_0-1} = 0$, we can rewrite

$$\begin{aligned} \Phi(w, x_1, \ldots, x_n) &= \left(wr_{u_0} + \sum_{j=u_0}^{n} \bar{p}_j \right) + c_r w^{-q} - \sum_{j=u_0}^{n} (1 - \alpha_j) x_j \\ &= t(w, 0, \ldots, 0) + c_r w^{-q} - \sum_{j=1}^{n} (1 - \alpha_j) x_j. \end{aligned}$$

Job u_0 remains critical for a specific range of w-values. Decreasing w beyond that range makes one of the earlier jobs critical. This means that the function

$$t(w, 0, \ldots, 0) = \max_{1 \leqslant u \leqslant n} \left\{ wr_u + \sum_{j=u}^{n} \bar{p}_j \right\}$$

is a piece-wise linear function similar to function $t(\gamma)$ of form (4) introduced in Section 2.1. Its breakpoints $w_0, w_1, \ldots, w_m$ can be found in $O(n \log n)$ time due to Meggido [18], and $w_{m+1} = W > w_m$ can be additionally defined. For all $w \in (w_{k-1}, w_k]$ the first critical job does not change, so that

$$t(w, 0, \ldots, 0) = wR(k) + P(k),$$

where $R(k)$ is the release date of the current first critical job, and $P(k)$ is the partial sum of all processing times $\bar{p}_j$ of the jobs that follow the critical job.

For each k, $1 \leqslant k \leqslant m + 1$, introduce the function

$$\Phi_k(w, x_1, \ldots, x_n) = (wR(k) + P(k)) + c_r w^{-q} - \sum_{j=1}^{n} (1 - \alpha_j) x_j. \tag{8}$$

Analogously to Section 2.1, the original problem of minimizing function (6) is reduced to the sequence of subproblems P_k for $k = 1, 2, \ldots, m + 1$, where subproblem P_k consists in minimizing the function $\Phi_k(w, x_1, \ldots, x_n)$ for $w \in [w_{k-1}, w_k]$.

For any fixed value of w, the optimal compression amounts $x_1(w), \ldots, x_n(w)$ can be constructed by the algorithm formulated in [20,24]. In order to find an optimal value $w^*(k) \in [w_{k-1}, w_k]$, we derive analytical formulas for

$x_1(w), \ldots, x_n(w)$ and use them to represent function $\Phi_k(w, x_1, \ldots, x_n)$ as a single-variable function $\Phi_k(w)$ that can be minimized by the standard technique of finding stationary points.

In what follows we state formally the algorithm from [20,24] for constructing an optimal schedule with controllable processing times and a fixed value of w and describe the structure of an optimal schedule (see Section 2.2.1). To simplify our discussion, we study in Section 2.2.2 the component

$$\varphi_k(w, x_1, \ldots, x_n) = \left(wr_{u_0} + \sum_{j=u_0}^{n} \bar{p}_j \right) - \sum_{j=1}^{n} (1 - \alpha_j) x_j$$

of function $\Phi_k(w, x_1, \ldots, x_n)$. We describe how an interval $[w_{k-1}, w_k]$ can be split into a number of smaller intervals $[w_{k-1}, \xi_1] \cup [\xi_1, \xi_2] \cup \cdots \cup [\xi_{l-1}, \xi_l] \cup \cdots \cup [\xi_{d-1}, w_k]$ such that for any $w \in [\xi_{l-1}, \xi_l]$ the optimal schedules have a similar structure. For each subinterval $[\xi_{l-1}, \xi_l]$ we determine the optimal compression amounts $x_1(w), \ldots, x_n(w)$ as linear functions of the release rate w and represent function $\varphi_k(w, x_1, \ldots, x_n)$ in the form $\varphi_k(w) = Hw + D$, where H and D are constants and their values depend on the subinterval $[\xi_{l-1}, \xi_l]$. In Section 2.2.3 we describe how the function $\Phi_k(w) = \varphi_k(w) + c_r w^{-q}$ can be minimized in each subinterval $[\xi_{l-1}, \xi_l]$ and thus how the minimum in $[w_{k-1}, w_k]$ can be found. We conclude with the description of the general algorithm of finding the global optimum in Section 2.2.4.

2.2.1. Generating optimal schedules with fixed release rate

In order to describe the structure of the optimal schedule for a fixed value of w we formally state the algorithm from [20,24] which solves the problem with controllable processing times. We need to introduce the concepts of a slack of a job, a min-slack job, and a compressible job. A *slack* of job j is given by $\lambda_j = s_j - wr_j$, where s_j is the starting time of job j in the current schedule and wr_j is its actual release date. For a given job j_i we define a *min-slack* job u_i as the first job that follows j_i with the minimum slack λ_{u_i}. If the slack of a min-slack job is equal to zero, we call such a job *zero-slack* job. Job j is *compressible* if its actual processing time p_j in the current schedule is not crashed to its minimum value $\underline{p}_j$ and all subsequent jobs have non-zero slacks.

It can be seen that a job with $\alpha_j \geqslant 1$ will be left uncrashed in an optimal schedule, since crashing it, i.e., decreasing its processing time by x_j may reduce the makespan by at most x_j but increases the compression cost by $\alpha_j x_j$.

Algorithm CPT [20,24].

1. If necessary, renumber the jobs so that (1) holds and schedule them according to this numbering at their earliest starting times.
2. Find the critical job u_0 as the last job in the current schedule with an idle time before it (we assume that in any schedule there is always an idle time before the first job). Set $i = 1$. Define the set of zero-slack jobs $U = \{u_0\}$ and the set of crash jobs $J = \emptyset$.
3. Repeat the following steps while there exists a compressible job with $\alpha_j < 1$.
   (a) Find the first compressible job j_i which has minimum cost α_{j_i} among the compressible jobs from the set $\{u_{i-1}, \ldots, n\}$, compute the slacks λ_j for all jobs j that follow j_i and determine the corresponding min-slack job u_i.
   (b) Crash job j_i by $\min\{p_{j_i} - \underline{p}_{j_i}, \lambda_{u_i}\}$ and recalculate the slacks of all jobs of the set $\{u_i, \ldots, n\}$. Additionally, if the slack of job u_i becomes zero, set $i := i + 1$ and update $U := U \cup \{u_i\}$ and $J := J \cup \{j_i\}$.

The running time of Algorithm CPT does not exceed $O(n^2)$. A typical structure of a schedule created by the algorithm is shown in Fig. 2. For our purposes two following types of jobs in that schedule are of special importance:

- *zero-slack* jobs $u_0, u_1, \ldots, u_z$;
- *crash* jobs $j_1, \ldots, j_z$.

Note that in the schedule found by Algorithm CPT, for each i, $i \geqslant 1$, crash job j_i is the first job in the set $\{u_{i-1}, \ldots, n\}$ with the minimum compression cost α_{j_i}, $\alpha_{j_i} < 1$, which has not been fully crashed, i.e., $p_{j_i} > \underline{p}_{j_i}$, and u_i is the first job with zero slack, which follows j_i. See [10] for an animated presentation of this algorithm.

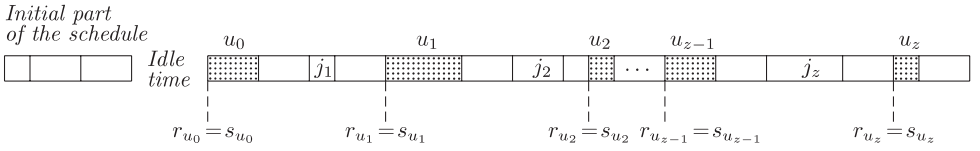

Fig. 2. The structure of a schedule found by Algorithm CPT.

2.2.2. Generating optimal schedules with controllable release rate and no release cost

In this subsection we temporarily disregard the cost of changing the release speed $c_r w^{-q}$, so that the objective function $\Phi(w, x_1, \ldots, x_n)$ becomes equal to $t(w, 0, \ldots, 0) - \sum_{j=1}^{n}(1-\alpha_j)x_j$.

Consider an arbitrary interval $[w_{k-1}, w_k]$, where w_{k-1} and w_k are two consecutive breakpoints of the function $t(w, 0, \ldots, 0)$ that are found by Meggido's algorithm. Let u_0 be the first critical job for all $w \in [w_{k-1}, w_k]$.

The quality of a schedule associated with a value of $w \in [w_{k-1}, w_k]$ and a collection of compressions $x_1, \ldots, x_n$ is measured by the function

$$\varphi_k(w, x_1, \ldots, x_n) = \left(w r_{u_0} + \sum_{j=u_0}^{n} \overline{p}_j \right) - \sum_{j=1}^{n}(1-\alpha_j)x_j,$$

where jobs $1, 2, \ldots, u_0$ are fully uncrashed ($x_1 = \cdots = x_{u_0} = 0$), while for each of the remaining jobs $u_0, u_0+1, \ldots, n$ the compression amount x_j does not exceed the maximum compression amount $\overline{p}_j - \underline{p}_j$ and guarantees that these jobs are processed as a block, without intermediate idle time.

Given a value of $w \in [w_{k-1}, w_k]$, define a schedule $S(w)$ that minimizes function $\varphi_k(w, x_1, \ldots, x_n)$, i.e., corresponds to the function

$$\varphi_k(w) = \min_{x_1,\ldots,x_n} \{\varphi_k(w, x_1, \ldots, x_n)\}, \tag{9}$$

which implies that for $w \in [w_{k-1}, w_k]$ the optimal compression values are functions of w. In what follows we derive analytical formulas for $x_1(w), \ldots, x_n(w)$. In fact, different formulas hold for different subintervals $[w_{k-1}, \xi_1]$, $[\xi_1, \xi_2], \ldots, [\xi_{l-1}, \xi_l], \ldots, [\xi_{d-1}, w_k]$ of interval $[w_{k-1}, w_k]$. The splitting of $[w_{k-1}, w_k]$ is done in such a way, that for any subinterval $[\xi_{l-1}, \xi_l] \subseteq [w_{k-1}, w_k]$, the optimal schedules have the same sets of crash jobs and the same sets of min-slack jobs. In order to find the subintervals $[\xi_{l-1}, \xi_l]$, we demonstrate how to generate all schedules $S(w)$ for w varying in the interval $[w_{k-1}, w_k]$. Recall that for any $w \in [w_{k-1}, w_k]$, job u_0 is critical in schedule $S(w)$, and there is idle time before that job for $w > w_{k-1}$; moreover, all jobs preceding job u_0 are uncrashed.

Let schedule $S(w)$ for $w = w_k$ be found by Algorithm CPT, and $J(w) = \{j_1, \ldots, j_z\}$ and $U(w) = \{u_0, u_1, \ldots, u_z\}$ be the corresponding sets of the crash jobs and the zero-slack jobs, respectively. Our algorithm starts with schedule $S(w)$ and proceeds by decreasing the value of w. In each iteration a transition from schedule $S(w)$ to schedule $S(w')$ is made, where $w' < w$.

Each time the algorithm modifies schedule $S(w)$ with compression amounts $x_j = x_j(w)$, $j = 1, \ldots, n$, into schedule $S(w')$ with compression amounts $x_j = x_j(w')$, $j = 1, \ldots, n$, so that

$$x_j(w') = x_j(w) + \Delta x_j. \tag{10}$$

The values of w' and Δx_j are selected in such a way that only the jobs of set $J(w)$ are crashed, at least one of them becoming fully crashed, while in schedule $S(w')$ each job of set $U(w)$ still has zero slack. Moreover, for all schedules $S(\tilde{w})$ for $\tilde{w} \in (w', w]$ the sets of crash jobs and of zero-slack jobs do not change, i.e., $J(\tilde{w}) = J(w)$ and $U(\tilde{w}) = U(w)$. The algorithm stores all relevant values of the release rate, which later will be shown to be the breakpoints $\xi_1, \xi_2, \ldots, \xi_{d-1}$ over the interval $[w_{k-1}, w_k]$.

Algorithm Min(φ_k)**.**

1. Set $w := w_k$, store this value and run Algorithm CPT to construct schedule $S(w)$ with the fixed release rate w and controllable processing times.

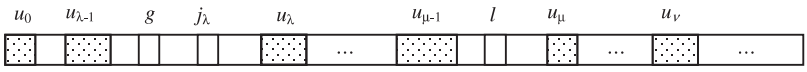

Fig. 3. Schedule $S(w)$.

2. For schedule $S(w)$, determine the sets $J(w)$ of crash jobs and $U(w)$ of zero-slack jobs.
3. If $J(w) = \emptyset$, then $x_j(w') = x_j(w)$ for any $w' \in [w_{k-1}, w]$. Stop.
4. Calculate

$$h_i = w - (p_{j_i} - \underline{p}_{j_i})/(r_{u_i} - r_{u_{i-1}}), \quad i = 1, \ldots, z, \tag{11}$$

where p_{j_i} is the actual processing time of job j_i in the current schedule, and

$$w' = \max\left\{w_{k-1}, \max_{i=1,\ldots,z} h_i\right\}. \tag{12}$$

5. For each $\tilde{w} \in [w', w]$, schedule $S(\tilde{w})$ is obtained from schedule $S(w)$ by decreasing w down to $\tilde{w}$ and by crashing jobs $j_i \in J(w)$ additionally by

$$\Delta x_{j_i} = (r_{u_i} - r_{u_{i-1}})(w - \tilde{w}). \tag{13}$$

6. If $w' = w_{k-1}$, then stop. Otherwise, set $w := w'$, store this value and go to Step 2.

It can be verified that the running time of Algorithm Min(φ_k) does not exceed $O(n^2)$.

Now we demonstrate that as the release rate w reduces, no job that has been earlier compressed will be decompressed.

Lemma 1. *If $S(w)$ and $S(\tilde{w})$ are schedules that are optimal for the values w and $\tilde{w}$, respectively, where $w_{k-1} \leqslant \tilde{w} < w \leqslant w_k$, then for any job of the set $\{u_0, \ldots, n\}$ its compression in $S(\tilde{w})$ is no smaller than that in $S(w)$.*

Proof. For schedule $S(w)$, let $J(w)$ and $U(w) = \{u_0, u_1, \ldots, u_z\}$ be the sets of crash jobs and of zero-slack jobs, respectively. We prove that a job of the set $\{u_0, \ldots, n\}$ that is (partly) compressed in $S(w)$ is not decompressed in $S(\tilde{w})$.

Suppose that our claim is not true and job $g \in \{u_{\lambda-1}, \ldots, u_\lambda - 1\}$ for some λ, $1 \leqslant \lambda \leqslant z$, is the first job that is decompressed in $S(\tilde{w})$, i.e., its processing time in $S(\tilde{w})$ is larger than that in $S(w)$ by some $\Delta y_g > 0$. Note that $\alpha_g < 1$ since job g is compressed in $S(w)$. Moreover,

$$\alpha_g \leqslant \alpha_{j_\lambda}; \tag{14}$$

otherwise, we could change the objective function value for schedule $S(w)$ by $(\alpha_{j_\lambda} - \alpha_g)\varepsilon < 0$ by decompressing g and compressing j_λ by the same amount ε. Here ε is chosen in such a way that the constraints on the largest possible and the smallest possible processing times of jobs g and j_λ are observed.

If in $S(\tilde{w})$ no job after g has zero slack, then $S(\tilde{w})$ cannot be optimal since we can compress g by some $\varepsilon \leqslant \Delta y_g$ such that the minimum slack for the jobs that follow g is still positive. This transformation reduces the makespan by ε and increases the compression cost by $\alpha_g \varepsilon < \varepsilon$.

Consider now the case that in $S(\tilde{w})$ some jobs processed after job g have zero slack. It can be verified that these jobs all belong to set $U(w)$. Let u_ν be the first of these jobs, $\nu \leqslant z$. Since in $S(\tilde{w})$ job g is compressed less than in $S(w)$, it follows that there exists a job l between the jobs $u_{\lambda-1}$ and u_ν that is compressed in $S(\tilde{w})$ more than in $S(w)$. Assume that job $l \in \{u_{\mu-1}, \ldots, u_\mu - 1\}$, $\lambda \leqslant \mu \leqslant \nu$, see Fig. 3. Then $S(\tilde{w})$ cannot be optimal. To see this, note that by the choice of the jobs of set $J(w)$ in Algorithm CPT we have that

$$\alpha_l \geqslant \alpha_{j_\mu} \geqslant \cdots \geqslant \alpha_{j_\lambda},$$

and due to (14) the inequality $\alpha_l \geqslant \alpha_g$ holds. We could decrease the objective function by $(\alpha_g - \alpha_l)\varepsilon < 0$ by compressing g and decompressing l by the same amount ε. Here ε is chosen in such a way that (i) positive slacks for all jobs between g and $u_\nu - 1$ are maintained, and (ii) the constraints on the smallest possible and the largest possible processing times of jobs g and l are observed. □

Let $\varrho(j)$ denote the sum of the current processing times of all jobs between u_0 and $j - 1$, inclusive.

Theorem 1. *Algorithm* $\mathrm{Min}(\varphi_k)$ *allows finding all optimal schedules for* w *running in the interval* $[w_{k-1}, w_k]$.

The proof of this theorem is given in Appendix.

Let $\xi_1, \ldots, \xi_{d-1}$ be the increasing sequence of all values w' generated and stored by Algorithm $\mathrm{Min}(\varphi_k)$; additionally, define $\xi_0 = w_{k-1}$ and $\xi_d = w_k$. It follows from the algorithm that for $w \in [\xi_{l-1}, \xi_l]$ the sets $J(w)$ and $U(w)$ remain the same; moreover, the optimal compressions x_j depend on w linearly due to formula (10). This implies that function $\varphi_k(w)$ is piece-wise linear over $[w_{k-1}, w_k]$, and $\xi_0, \ldots, \xi_d$ are its breakpoints.

Theorem 2. *Function* $\varphi_k(w)$ *is convex over* $[w_{k-1}, w_k]$.

Proof. To prove the convexity, we show that the slope of $\varphi_k(w)$ increases monotonously over $[w_{k-1}, w_k]$.

We compare the slopes of $\varphi_k(w)$ for any two consecutive segments $[\xi_{l-1}, \xi_l]$ and $[\xi_l, \xi_{l+1}]$ and prove that

$$\frac{\varphi_k(\xi_l) - \varphi_k(\xi_{l-1})}{\xi_l - \xi_{l-1}} \leqslant \frac{\varphi_k(\xi_{l+1}) - \varphi_k(\xi_l)}{\xi_{l+1} - \xi_l}. \tag{15}$$

Let $x_j(\xi_{l-1})$ and $x_j(\xi_l)$, $j = 1, \ldots, n$, be the optimal compressions for $w = \xi_{l-1}$ and $w = \xi_l$, respectively. We have that

$$\varphi_k(\xi_{l+1}) - \varphi_k(\xi_l) = \left(\xi_{l+1} r_{u_0} + \sum_{j=u_0}^{n} p_j - \sum_{j=1}^{n} (1-\alpha_j) \times x_j(\xi_{l+1})\right) - \left(\xi_l r_{u_0} + \sum_{j=u_0}^{n} p_j - \sum_{j=1}^{n} (1-\alpha_j) \times x_j(\xi_l)\right).$$

Let $J(\xi_{l+1}) = \{j_1, \ldots, j_z\}$ and $U(\xi_{l+1}) = \{u_0, u_1, \ldots, u_z\}$. Recall that only jobs $j_i \in J(\xi_{l+1})$ are subject to compression while w decreases from ξ_{l+1} to ξ_l. Due to (13), $x_{j_i}(\xi_l) - x_{j_i}(\xi_{l+1}) = (r_{u_i} - r_{u_{i-1}})(\xi_{l+1} - \xi_l)$ so that we derive

$$\begin{aligned}\varphi_k(\xi_{l+1}) - \varphi_k(\xi_l) &= r_{u_0}(\xi_{l+1} - \xi_l) + \sum_{i=1}^{z} (1-\alpha_{j_i})(r_{u_i} - r_{u_{i-1}})(\xi_{l+1} - \xi_l) \\ &= (\xi_{l+1} - \xi_l)\left(r_{u_z} - \sum_{i=1}^{z} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}})\right).\end{aligned}$$

The slope of function $\varphi_k(w)$ for $[\xi_l, \xi_{l+1}]$ is given by

$$H = r_{u_z} - \sum_{i=1}^{z} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}}). \tag{16}$$

In a similar way, we can find the slope H' of function $\varphi_k(w)$ for $[\xi_{l-1}, \xi_l]$; note that while H is determined for the sets $J(\xi_{l+1})$ and $U(\xi_{l+1})$, the value of H' is found with respect to the sets $J(\xi_l) = \{j'_1, \ldots, j'_y\}$ and $U(\xi_l) = \{u'_0 = u_0, u'_1, \ldots, u'_y\}$, where $y \leqslant z$. For simplicity, denote

$$J = J(\xi_{l+1}), \quad U = U(\xi_{l+1});$$
$$J' = J(\xi_l), \quad U' = U(\xi_l).$$

It follows from Algorithm $\mathrm{Min}(\varphi_k)$ that $U' \subseteq U$. This means that $u'_y = u_v$ for some v, $1 \leqslant v \leqslant z$.

To prove (15) we estimate the difference

$$H - H' = \left(r_{u_z} - \sum_{j_i \in J, u_i \in U} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}})\right) - \left(r_{u'_y} - \sum_{j'_i \in J', u'_i \in U'} \alpha_{j'_i}(r_{u'_i} - r_{u'_{i-1}})\right).$$

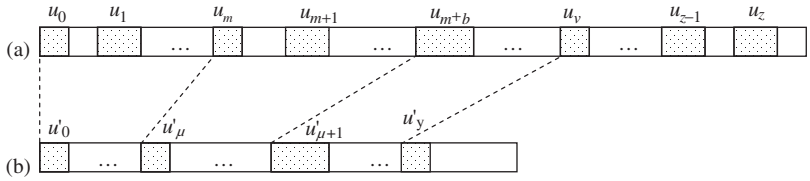

Fig. 4. Schedules (a) $S(\xi_l)$ and (b) $S(\xi_{l+1})$.

The two cases may occur.

Case 1: If $v = z$, i.e., $u'_y = u_z$, then we obtain

$$H - H' = \sum_{j'_i \in J', u'_i \in U'} \alpha_{j'_i}(r_{u'_i} - r_{u'_{i-1}}) - \sum_{j_i \in J, u_i \in U} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}}).$$

It follows from Algorithm Min(φ_k) that for each pair of jobs u'_μ and $u'_{\mu+1}$, there exist a pair of jobs u_m and u_{m+b}, such that $u_m = u'_\mu$, $u_{m+b} = u'_{\mu+1}$. See Fig. 4. This implies that for any μ, $0 \leqslant \mu \leqslant y - 1$, the equality

$$r_{u'_{\mu+1}} - r_{u'_\mu} = r_{u_{m+b}} - r_{u_m} = \sum_{i=m+1}^{m+b} (r_{u_i} - r_{u_{i-1}})$$

holds. By the definition of the crash jobs, we have that $\alpha_{j_m} \leqslant \alpha_{j_{m+1}} \leqslant \cdots \leqslant \alpha_{j_{m+b}} \leqslant \alpha_{j'_\mu}$, and hence

$$\alpha_{j'_\mu}(r_{u'_{\mu+1}} - r_{u'_\mu}) - \sum_{i=m+1}^{m+b} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}}) = \sum_{i=m+1}^{m+b} (\alpha_{j'_\mu} - \alpha_{j_i})(r_{u_i} - r_{u_{i-1}}) \geqslant 0. \tag{17}$$

The latter inequality yields $H - H' \geqslant 0$ and (15) is valid.

Case 2: If $v < z$, i.e., $u'_y = u_v < u_z$, then we obtain

$$\begin{aligned} H - H' &= \left(r_{u_z} - \sum_{i=1}^{v} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}}) - \sum_{i=v+1}^{z} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}}) \right) - \left(r_{u_v} - \sum_{j'_i \in J', u'_i \in U'} \alpha_{j'_i}(r_{u'_i} - r_{u'_{i-1}}) \right) \\ &= \left(r_{u_z} - \sum_{i=v+1}^{z} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}}) - r_{u_v} \right) + \left(\sum_{j'_i \in J', u'_i \in U'} \alpha_{j'_i}(r_{u'_i} - r_{u'_{i-1}}) - \sum_{i=1}^{v} \alpha_{j_i}(r_{u_i} - r_{u_{i-1}}) \right). \end{aligned}$$

Since $\alpha_{j_i} < 1$, the first term is positive. Besides, due to (17) the second term is non-negative. This implies that $H - H' > 0$ and (15) is valid. □

Remark 1. It follows from the proof of Theorem 2 that function $\varphi_k(w)$ over each interval $[\xi_l, \xi_{l+1}]$ is a linear function with a slope of H given by (16). Since all α_{j_i} are less than 1, the value of H is positive for each interval $[\xi_l, \xi_{l+1}]$, i.e., $\varphi_k(w)$ increases monotonously over $[w_{k-1}, w_k]$.

2.2.3. Optimal scheduling with release cost

We now turn back to the problem of minimizing the original function $\Phi(w, x_1, \ldots, x_n)$ of form (7) over an interval $[w_{k-1}, w_k]$, $1 \leqslant k \leqslant m+1$. Recall that while the rate w varies within $[w_{k-1}, w_k]$ the first critical job u_0 does not change.

For the last interval $[w_m, w_{m+1}]$,where w_{m+1} is equal to a sufficiently large number W, there is only one critical job u_0 and all jobs $u_0, u_0 + 1, \ldots, n$ have equal release dates. The jobs $u_0, u_0 + 1, \ldots, n$ are processed as a block,

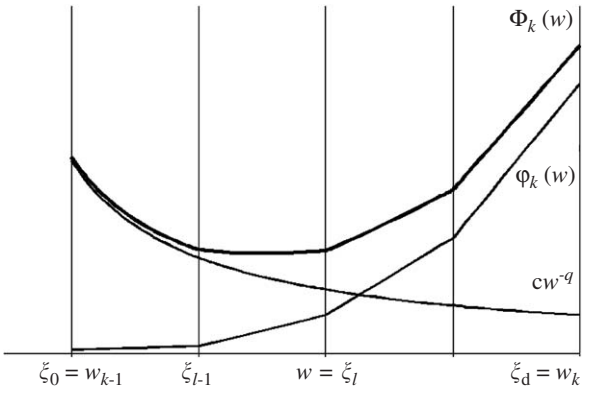

Fig. 5. Function $\Phi_k(w)$ as the sum of the convex functions $\varphi_k(w)$ and cw^{-q}.

without intermediate idle time. Those of them which have $\alpha_j < 1$ are fully crashed while the remaining jobs (including the first jobs $1, 2, \ldots, u_0 - 1$) are fully uncrashed.

For the fixed compression amounts $x_1, x_2, \ldots, x_n$, function $\Phi_m(w, x_1, x_2, \ldots, x_n)$ depends on a single parameter w and it is given by the formula:

$$\Phi_m(w) = \left(w r_{u_0} + \sum_{j=u_0}^{n} \overline{p}_j \right) + c_r w^{-q} - \sum_{j=u_0}^{n} (1 - \alpha_n) x_n$$

(see (8)). It follows that function $\Phi_m(w)$ can be minimized by solving the equation

$$\frac{\mathrm{d}(\Phi_m(w))}{\mathrm{d}w} = 0,$$

which yields

$$Q = \left(\frac{c_r q}{r_{u_0}} \right)^{1/(q+1)}$$

and the optimal value of w in $[w_m, w_{m+1}]$ is given by the formula:

$$w^* := \begin{cases} w_m & \text{if } Q \leqslant w_m, \\ Q & \text{otherwise.} \end{cases} \tag{18}$$

For each interval $[w_{k-1}, w_k]$ for $k = 1, \ldots, m$, the function $\Phi(w, x_1, \ldots, x_n)$ of form (7) is the sum of the function $\varphi_k(w, x_1, \ldots, x_n)$ studied in Section 2.2.2 and the released cost cw^{-q}. We have to determine both the release rate w and the processing time compressions $x_1, \ldots, x_n$. For each interval $[w_{k-1}, w_k]$ define $\Phi_k(w, x_1, \ldots, x_n) = \Phi(w, x_1, \ldots, x_n)$. Similar to Section 2.2.2, it can be seen that for each interval $[w_{k-1}, w_k]$ the compressions are in fact functions of w.

Given a value of $w \in [w_{k-1}, w_k]$, define a schedule $S(w)$ that minimizes function $\Phi_k(w, x_1, \ldots, x_n)$, i.e., corresponds to the function

$$\Phi_k(w) = \min_{x_1,\ldots,x_n} \{\Phi_k(w, x_1, \ldots, x_n)\},$$

which implies that for $w \in [w_{k-1}, w_k]$ the optimal compression values are functions of w. It follows from (9) that

$$\Phi_k(w) = \min_{x_1,\ldots,x_n} \{\Phi_k(w, x_1, \ldots, x_n)\} = \varphi_k(w) + c_r w^{-q}$$

and Theorem 2 implies that function $\Phi_k(w)$ is convex over $[w_{k-1}, w_k]$ as the sum of two convex functions, see Fig. 5.

Recall that function $\varphi_k(w)$ is piecewise-linear over $[w_{k-1}, w_k]$ and its breakpoints $\xi_0 = w_{k-1}, \xi_1, \ldots, \xi_d = w_k$ can be found by Algorithm Min(φ_k). Due to Remark 1, function $\Phi_k(w)$ over an interval $[\xi_{l-1}, \xi_l]$ for each $l = 1, \ldots, d$, can be written as

$$\Phi_k(w) = Hw + c_r w^{-q} + D,$$

where H is given by (16) and D does not depend on w, but may depend on l. The stationary point of $\Phi_k(w)$ can be found by solving the equation

$$\frac{\mathrm{d}(\Phi_k(w))}{\mathrm{d}w} = 0,$$

which yields

$$Q = \left(\frac{c_r q}{H}\right)^{1/(q+1)}. \tag{19}$$

For minimizing function $\Phi_k(w)$ we use the following method.

Algorithm Min(Φ_k)**.**

1. Use Algorithm Min(φ_k) for finding the increasing sequence of the breakpoints $\xi_0 = w_{k-1}, \xi_1, \ldots, \xi_d = w_k$ of function $\varphi_k(w)$ over the interval $[w_{k-1}, w_k]$.
2. Set $l := d$. Define $w^* := \xi_l$.
3. Set $w := \xi_l$. Determine the sets $J(w)$, $U(w)$. Find H by formula (16) and Q by formula (19). Define

$$w^*_{\mathrm{new}} := \begin{cases} \xi_{l-1} & \text{if } \xi_{l-1} \geqslant Q, \\ \xi_l & \text{if } \xi_l \leqslant Q, \\ Q & \text{if } \xi_{l-1} < Q < \xi_l \end{cases} \tag{20}$$

   and update $x_{j_i}(w^*_{\mathrm{new}}) = x_{j_i}(w) + \Delta x_{j_i}$ for all $j_i \in J(w)$, where

$$\Delta x_{j_i} = (r_{u_i} - r_{u_{i-1}})(w - w^*_{\mathrm{new}}),$$

   leaving $x_j(w^*_{\mathrm{new}}) = x_j(w)$ for all other jobs.
4. If $w^*_{\mathrm{new}} = Q$ then output w^*_{new}, the corresponding schedule $S(w^*_{\mathrm{new}})$ and stop. Otherwise, if $\Phi_k(w^*_{\mathrm{new}}) < \Phi_k(w^*)$ and $l \geqslant 2$ then set $w^* := w^*_{\mathrm{new}}$, $l := l - 1$ and go to Step 3, else output w^*, the corresponding schedule $S(w^*)$ and stop.

Algorithm Min(Φ_k) scans the segment $[w_{k-1}, w_k]$ by decreasing w starting from the right-hand endpoint. The value of Q in Step 3 of the algorithm is a stationary point of the function $\Phi_k(w)$ for $w \in [\xi_{l-1}, \xi_l]$ where ξ_{l-1} and ξ_l are two consecutive breakpoints of the function $\varphi_k(w)$, see Fig. 5. In any iteration, the current value of w^* delivers the minimum to the function $\Phi_k(w)$ over the intervals considered so far. Similarly, w^*_{new} minimizes $\Phi_k(w)$ over the current interval $[\xi_{l-1}, \xi_l]$. If $w^*_{\mathrm{new}} = Q$ then w^*_{new} corresponds to the global minimum of the function $\Phi_k(w)$ over $[w_{k-1}, w_k]$, since $\Phi_k(w)$ is convex. If $\Phi_k(w^*_{\mathrm{new}}) < \Phi_k(w^*)$ and $l = 1$, function $\Phi_k(w)$ monotonously increases over $[w_{k-1}, w_k]$, and the algorithm finds $w^* = w_{k-1}$. If $\Phi_k(w^*_{\mathrm{new}}) \geqslant \Phi_k(w^*)$, then the function does not decrease as w decreases below w^*; due to convexity of $\Phi_k(w)$ this implies that w^* will deliver the minimum to the function over $[w_{k-1}, w_k]$. The latter case includes the situation that the function is monotonously decreasing, so that the algorithm outputs $w^* = w_k$.

The running time of Algorithm Min(Φ_k) does not exceed $O(n^2)$. Step 1 alone requires $O(n^2)$ time. Steps 3 and 4 are repeated no more than n times. In each iteration, updating the sets $J(w)$ and $U(w)$ can be implemented in linear time. Finding H also takes $O(n)$ time. Note that for the empty sets $J(w)$ and $U(w)$ formula (16) reduces to $H = r_{u_0}$.

The results of this section can be summarized in the following statement.

Theorem 3. *Algorithm* Min(Φ_k) *determines the value of* w *that minimizes function* $\Phi_k(w)$ *over the interval* $[w_{k-1}, w_k]$ *in* $O(n^2)$ *time.*

2.2.4. General algorithm

We present now the complete algorithm for solving the single machine problem with controllable release speed and controllable processing times.

Algorithm CRS/CPT.

1. If necessary, renumber the jobs so that (1) holds.
2. Run the algorithm by Meggido [18] and find all breakpoints $w_0, w_1, \ldots, w_m$ of function $\Phi(w, 0, \ldots, 0)$.
3. Determine the optimum release rate $w^*(m+1)$ in the last interval $[w_m, w_{m+1}]$ using formula (18). Run Algorithm CPT to find $x_1^*(m+1), \ldots, x_1^*(m+1)$.
4. For each $k = m, m-1, \ldots, 2, 1$ run Algorithm Min(Φ_k) to find the optimal release rate $w^*(k)$ and compression amounts $x_1^*(k), \ldots, x_n^*(k)$. Compute $\Phi_k(w^*(k), x_1^*(k), \ldots, x_n^*(k))$.
5. Find the value w^* such that $w^* = w^*(h)$ for some h, $1 \leqslant h \leqslant m+1$, where

$$\Phi_h(w^*(h), x_1^*(h), \ldots, x_n^*(h)) = \min_{1 \leqslant k \leqslant m+1} \Phi_k(w^*(k), x_1^*(k), \ldots, x_n^*(k)).$$

6. Define $v_r^* = 1/w^*$.

The running time of the algorithm is $O(n^3)$.

3. Controllable release dates

In this section we consider the situation related to the supply chain in Fig. 1 that the supplier is willing to modify release dates for individual jobs. Recall that here job $j \in N$ is given a 'standard' release date $\bar{r}_j$ which can be reduced to the minimum value $\underline{r}_j$, where $\underline{r}_j \leqslant \bar{r}_j$. Reducing $\bar{r}_j$ to some actual value r_j, $\underline{r}_j \leqslant r_j \leqslant \bar{r}_j$, incurs additional cost $\beta_j y_j$, where $y_j = \bar{r}_j - r_j$ is the compression amount of the corresponding release date.

3.1. Controllable release dates and constant processing times

In this subsection we assume that the job processing times $p_1, \ldots, p_n$ remain unchanged, i.e., modifying the job release dates is the only type of coordination in the supply chain. Thus, our goal is to determine optimum release compression amounts $y_1, \ldots, y_n$ and an optimal permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ of jobs that minimize the function

$$F(\pi, y_1, \ldots, y_n) = t(\pi, y_1, \ldots, y_n) + \sum_{j=1}^{n} \beta_j y_j. \tag{21}$$

The first component of function $F(\pi, y_1, \ldots, y_n)$ represents the makespan, provided that the jobs are sequenced according to a permutation π and the release date of each job j is decreased by y_j, i.e.,

$$t(\pi, y_1, \ldots, y_n) = \max_{1 \leqslant u \leqslant n} \left\{ \bar{r}_{\pi(u)} - y_{\pi(u)} + \sum_{j=u}^{n} p_{\pi(j)} \right\},$$

while the second component is total compression cost.

Suppose that the value of the makespan of a schedule that minimizes function (21) is known to be equal to Y. Then the problem reduces to an auxiliary problem $1|\bar{r}_j - y_j, C_{\max} \leqslant Y| \sum \beta_j y_j$ of finding the compression values $y_1, \ldots, y_n$ and the permutation of jobs that minimize the expression $\sum_{j=1}^{n} \beta_j y_j$. The latter problem is equivalent to a single machine problem P_Y to minimize total weighted tardiness, usually denoted by $1\| \sum w_j T_j$, where the tardiness of job j that completes at time C_j with respect to a given due date d_j is defined as $T_j = \max\{C_j - d_j, 0\}$. To see this, it suffices to modify Problem P_Y by changing the direction of time and by setting the weights w_j equal to β_j and the job due dates equal to $d_j = Y - \bar{r}_j$.

Since problem $1\|\sum w_jT_j$ is NP-hard in the strong sense [15], we obtain the following statement (see also [20] for an alternative proof).

Theorem 4. *The single machine problem with controllable release dates to minimize function* (21) *is NP-hard in the strong sense, even if the makespan of the optimal schedule is known.*

Note that if all β_j are equal, then the original problem in a similar way can be associated with a single machine problem to minimize total (unweighted) tardiness, denoted by $1\|\sum T_j$. Since the latter problem is NP-hard in the ordinary sense (see [6]), we can conclude that the problem with controllable release dates and equal compression costs $\beta_j = \beta$ has the same complexity status. On the other hand, problem $1\|\sum T_j$ admits a pseudopolynomial dynamic programming algorithm by Lawler [15], and this allows us to use Lawler's algorithm as a subroutine to obtain a solution of the original problem with equal $\beta_j = \beta$ in pseudopolynomial time. The number of calls of Lawler's algorithm does not exceed $\max_{j\in N}\bar{r}_j + \sum_{j\in N} p_j$.

3.1.1. Controllable release dates and constant processing times: common bounds

In the remainder of this section we concentrate on the case that release dates have the same upper bound $\bar{r}$ and the same lower bound $\underline{r}$. We assume that $\bar{r}$, $\underline{r}$ and all processing times are integer. In terms of the supply chain, this assumption means that the supplier originally does not have any preferences regarding the order of releasing the jobs. By technological conditions, no job can be released earlier that time $\underline{r}$; on the other hand, it is possible to release all jobs by time $\bar{r}$. The manufacturer accepts that some of the jobs are released at time $\bar{r}$, and negotiates with the supplier a suitable order of those jobs which will be released earlier than $\bar{r}$.

Let the jobs that become available earlier than time $\bar{r}$ be called *early* jobs, while the other jobs are called *late*. Note that the late jobs have a common release date $\bar{r}$, while for the early jobs the release dates are reduced individually.

Without loss of generality, we may assume that $\beta_j < 1$ for all $j = 1, \ldots, n$. Indeed, the release date of a job j with $\beta_j \geqslant 1$ will remain equal to $\bar{r}$ in any optimal schedule.

We will distinguish between two versions of the problem: the *unrestricted* one for which the inequality

$$\bar{r} - \underline{r} \geqslant \sum_{j=1}^{n} p_j \tag{22}$$

holds and the *restricted* one for which (22) does not hold. Note that for the unrestricted version of the problem the manufacturer can schedule all jobs in the time interval $[\underline{r}, \bar{r}]$, while in the restricted case that is not possible. Thus, the unrestricted version is easier to solve; in particular the NP-hardness of the unrestricted version of the problem would imply the NP-hardness of the restricted counterpart. Separate consideration of the restricted and unrestricted versions of a scheduling problem is traditional for problems with a common due date, see, e.g., surveys [3,7].

As earlier in Section 3.1, we can associate the unrestricted version of the problem with common bounds on controllable release dates with a single machine problem to minimize total weighted tardiness with respect to a common due date, denoted by $1|d_j = d|\sum w_jT_j$. The latter problem is known to be NP-hard in the ordinary sense [25], and this implies the following statement.

Theorem 5. *The single machine problem with controllable release dates to minimize function* (21) *is NP-hard in the ordinary sense for both unrestricted and restricted cases even if the release dates have common bounds and the makespan of the optimal schedule is known.*

We now present pseudopolynomial algorithms for both versions of the problem with common bounds for controllable release dates.

We start with the unrestricted case. Consider an optimal schedule S^* associated with a permutation $\pi = \pi(1), \pi(2), \ldots, \pi(n)$ of jobs. In S^* either all jobs are late or there is a sequence or early jobs $\pi(1), \ldots, \pi(k)$ such that job $\pi(k)$ completes exactly at time $\bar{r}$. If this property does not hold, then by performing an appropriate shift (to the left or to the right depending on the sign of $k\beta_{\pi(k)} - 1$) we obtain a schedule with a smaller value of the objective function. Thus, the original release date $\bar{r}$ for job $\pi(k)$ is reduced by $y_{\pi(k)} = p_{\pi(k)}$, the release date for job $\pi(k-1)$

is reduced by $y_{\pi(k-1)} = p_{\pi(k)} + p_{\pi(k-1)}$, and so on, the release date for job $\pi(1)$ is reduced by $y_{\pi(1)} = \sum_{j=1}^{k} p_{\pi(j)}$. Hence, to minimize function (21) it suffices to minimize the function

$$f(\pi) = \min_{1 \leqslant k \leqslant n} \left\{ \bar{r} + \sum_{j=k+1}^{n} p_{\pi(j)} + \sum_{j=1}^{k} \beta_{\pi(j)} \sum_{i=j}^{k} p_{\pi(i)} \right\} \tag{23}$$

over all permutations of jobs.

If necessary, re-number the jobs in such a way that

$$\frac{p_1}{\beta_1} \geqslant \frac{p_2}{\beta_2} \geqslant \cdots \geqslant \frac{p_n}{\beta_n}, \tag{24}$$

which corresponds to the well-known Smith rule [21]. It follows from [21] that in an optimal schedule early jobs must be sequenced in the order of this numbering.

Our pseudopolynomial dynamic programming algorithm for the problem of minimizing function (23) is based on a backward recursion. The jobs are scanned in the order opposite to their numbering. Suppose that jobs $n, n-1, \ldots, k+1$ have been assigned, and $F_{k+1}(P)$ is the value of the objective function, provided that total processing time of the early jobs is P, i.e.,

$$P = \sum_{j \in E} p_j,$$

where E is the set of early jobs in the partial schedule. There are two options to schedule the next job k:

(i) Job k is scheduled late. Its exact position among other late jobs is immaterial. Scheduling job k does not affect the set of early jobs, while the value of the makespan increases by p_k, i.e.,

$$F_k(P) = F_{k+1}(P) + p_k.$$

(ii) Job k is scheduled early. In this case job k is inserted before the first early job, which does not affect the makespan. The release date of each previously scheduled early job does not change, the release date of job k is reduced by y_k equal to the total processing times of all early jobs, i.e.,

$$F_k(P) = F_{k+1}(P - p_k) + \beta_k P.$$

The initial conditions are given by

$$F_{n+1}(P) = \bar{r}$$

for all $P = 0, 1, \ldots, p(N) = \sum_{j \in N} p_j$. The main recursion is given by

$$F_k(P) = \min\{F_{k+1}(P) + p_k, F_{k+1}(P - p_k) + \beta_k P | k = n, n-1, \ldots, 1\}.$$

The algorithm stops having found

$$\min_{0 \leqslant P \leqslant p(N)} F_1(P).$$

The actual sequence of jobs corresponding to an optimal schedule can be found by backtracking. The running time of the algorithm is $O(np(N))$.

Thus, we have proved the following statement.

Theorem 6. *The unrestricted version of the single machine problem with controllable release dates, provided that all release dates have the same upper bound $\bar{r}$ and the same lower bound $\underline{r}$, is solvable in $O(np(N))$ time.*

We now pass to considering the restricted version of our problem. As above, assume that the jobs are numbered in accordance with (24).

Unlike the unrestricted case, there may exist a job h which *straddles* the release date $\bar{r}$, i.e., starts before and completes no earlier than time $\bar{r}$. The jobs that are processed before h are early and are sequenced in the order of their numbering.

Let in an optimal schedule S^* the jobs follow the sequence $\pi = \pi(1), \pi(2), \ldots, \pi(n)$. Suppose that $\pi(k+1) = h$ is the straddling job and the jobs $\pi(1), \ldots, \pi(k)$ are also early. If the original release date $\bar{r}$ for job $\pi(k+1)$ is reduced by $y_h = y_{\pi(k+1)}$, then the release date for job $\pi(k)$ is reduced by $y_{\pi(k)} = y_h + p_{\pi(k)}$, and so on, the release date for job $\pi(1)$ is reduced by $y_{\pi(1)} = y_h + \sum_{j=1}^{k} p_{\pi(j)}$. Hence, to minimize function (21) it suffices to minimize the function

$$f(\pi) = \min_{1 \leqslant k \leqslant n-1} \left\{ \bar{r} + (p_{\pi(k+1)} - y_{\pi(k+1)}) + \sum_{j=k+2}^{n} p_{\pi(j)} + \sum_{j=1}^{k} \beta_{\pi(j)} \left(\sum_{i=j}^{k} p_{\pi(i)} + y_{\pi(k+1)} \right) + \beta_{\pi(k+1)} y_{\pi(k+1)} \right\} \tag{25}$$

over all permutations of jobs. The compression amount y_h may take any integer value from 1 till p_h.

For any h from 1 to n, let $S(h, y_h)$ define a schedule for processing a single job h in such a way that this job straddles and is released at time $\bar{r} - y_h$.

For each combination of h and y_h, our dynamic programming algorithm starts from schedule $S(h, y_h)$. Let the sequence of the remaining $n-1$ jobs taken in the order of their numbering be denoted by $\varphi(1), \varphi(2), \ldots, \varphi(n-1)$. These jobs are scanned according to the sequence $\varphi(n-1), \varphi(n-2), \ldots, \varphi(1)$. Suppose that jobs $\varphi(n-1), \varphi(n-2), \ldots, \varphi(k+1)$ have been assigned, and $F_{k+1}(P, h, y_h)$ is the value of the objective function, provided that total processing time of the jobs that precede the straddling job h is P. Define the room available for scheduling additional early jobs as

$$z = (\bar{r} - \underline{r}) - P - y_h,$$

and call this value the *slack*.

There are two options to schedule the next job $\varphi(k)$:

(i) Job $\varphi(k)$ is scheduled late, i.e., after job h. This case is similar to the unrestricted version, so that

$$F_k(P, h, y_h) = F_{k+1}(P, h, y_h) + p_{\varphi(k)}.$$

(ii) Job $\varphi(k)$ is scheduled early, i.e., before the first early job in accordance with (24). This is only possible if $z \geqslant p_{\varphi(k)}$. The makespan does not change, the release date of job k is reduced by y_k equal to the total processing times of all early jobs plus y_h. The value of z is decreased by $p_{\varphi(k)}$.

$$F_k(P, h, y_h) = F_{k+1}(P - p_{\varphi(k)}, h, y_h) + \beta_{\varphi(k)} P + \beta_{\varphi(k)} y_h.$$

If both options are possible, the algorithm accepts the smaller of the found values $F_k(P, h, y_h)$.

For every pair h and y_h, where $1 \leqslant h \leqslant n$ and $1 \leqslant y_h \leqslant p_{\max} = \max\{p_j | j \in N\}$, the initial conditions are given by

$$F_n(P, h, y_h) = \beta_h y_h + \bar{r} + (p_h - y_h)$$

for all $P = 0, 1, \ldots, p(N) - p_h$.

The algorithm stops having found

$$\min\{F_1(P, h, y_h) | 0 \leqslant P \leqslant p(N) - p_h, 1 \leqslant h \leqslant n, 1 \leqslant y_h \leqslant p_h\}.$$

The actual sequence of jobs corresponding to an optimal schedule can be found by backtracking. The running time of the algorithm is $O(n^2 p(N) p_{\max})$.

Thus, we have proved the following statement.

Theorem 7. *The restricted version of the single machine problem with controllable release dates, provided that all release dates have the same upper bound $\bar{r}$ and the same lower bound $\underline{r}$, is solvable in $O(n^2 p(N) p_{\max})$ time.*

Finally, we show that the unrestricted version of the problem with common bounds on controllable release dates admits a fully polynomial approximation scheme (FPTAS). Recall that for a minimization problem a FPTAS is a series of approximation algorithms which for any positive ε generate a solution with the value of the objective function that is at most $(1+\varepsilon)$ times the optimal value, and with the running time that is polynomial with respect to both the length of the problem input and $1/\varepsilon$.

We formulate our scheduling problem as a special integer quadratic programming problem, known as *positive half-product* (PHP), which is a modification of the problem introduced in [2]. Problem PHP is a problem of minimization the function

$$H(z_1, z_2, \ldots, z_n) = \sum_{1 \leqslant i < j \leqslant n} a_i b_j z_i z_j + \sum_{j=1}^{n} h_j(1 - z_j) + \sum_{j=1}^{n} g_j z_j + D,$$

where $z_j \in \{0, 1\}$, $D \geqslant 0$, and all coefficients a_j, b_j, g_j and h_j are non-negative. A FPTAS for problem PHP is developed by Janiak et al. [14]. To see that our scheduling problem reduces to problem PHP, recall that the jobs are numbered in accordance with (24) and define the Boolean variable $z_j = 1$ if job j is scheduled early and $z_j = 0$, otherwise. The corresponding makespan can be written as $\bar{r} + \sum_{j=1}^{n} p_j(1 - z_j)$, while the release date compression amount y_i for an early job i is equal to $\sum_{i=j}^{n} p_j z_j$, so that the overall objective function becomes

$$\sum_{i=1}^{n} \beta_i z_i \left(\sum_{j=i}^{n} p_i z_j \right) + \bar{r} + \sum_{j=1}^{n} p_j(1 - z_j) = \sum_{1 \leqslant i < j \leqslant n} \beta_i p_j z_i z_j + \sum_{j=1}^{n} p_j(1 - z_j) + \sum_{j=1}^{n} \beta_j p_j z_j + \bar{r},$$

which is equivalent to function H with $a_j = \beta_j$, $b_j = p_j$, $h_j = p_j$, $g_j = \beta_j p_j$ and $D = \bar{r}$.

3.2. Controllable release dates and controllable processing speed: common bounds and equal compression costs

In this subsection we consider the situation that not only the supplier controls the job release dates, but also the manufacturer is ready to cooperate by modifying the machine processing speed. Due to the NP-hardness results established in Section 3.1, here we only consider the case that the release dates have a common upper bound $\bar{r}$ and a common lower bound $\underline{r}$, and additionally all release date compression costs are equal, i.e., $\beta_j = \beta$. Besides, we also distinguish between the unrestricted and restricted versions of the problem.

Assume that p_j is a given 'standard' value of the processing time of job j and we can increase the speed at which the jobs are processed on the machine. If the processing speed is equal to v_p then the processing time of job $j \in N$ equals $w_p p_j$, where $w_p = 1/v_p$ is the processing rate. Let w_p be a fixed processing rate. Speeding up the machine incurs the processing speed cost $c_p w_p^{-q}$, where c_p is a positive constant and q is a positive integer. We are looking for a schedule that minimizes the sum of the makespan, total release date compression cost and the processing speed cost.

We start with the unrestricted version first, for which condition (22) holds with respect to the standard processing times. Assume that the jobs are numbered according to the LPT rule, i.e.,

$$p_1 \geqslant p_2 \geqslant \cdots \geqslant p_n.$$

Using the results from [5], we can derive that for any fixed processing rate there exists an optimal schedule in which the jobs are processed in the LPT order. Moreover, there are $h = \lfloor 1/\beta \rfloor$ early jobs and the job in position h completes exactly at time $\bar{r}$.

For a schedule S associated with a fixed processing rate w_p we have that

$$C_{\max}(S) = \bar{r} + \sum_{j=h+1}^{n} w_p p_j,$$

while total release date compression cost is given by $\beta \sum_{j=1}^{h} \sum_{i=j}^{h} w_p p_i$. Hence, the objective function can be written as

$$F(w_p) = \bar{r} + \sum_{j=h+1}^{n} w_p p_j + \beta \sum_{j=1}^{h} \sum_{i=j}^{h} w_p p_i + c_p w_p^{-q}$$

and this function has to be minimized with respect to its only controllable parameter, w_p. Taking the derivative, we obtain

$$\frac{\mathrm{d}(F(w_p))}{\mathrm{d}w_p} = \sum_{j=h+1}^{n} p_j + \beta \sum_{j=1}^{h} \sum_{i=j}^{h} p_i - q c_p w_p^{-q-1} = 0,$$

so that the optimal processing rate w_p^* satisfies

$$w_p^* = \left(\frac{q c_p}{\sum_{j=h+1}^{n} p_j + \beta \sum_{j=1}^{h} \sum_{i=j}^{h} p_i} \right)^{1/(q+1)}. \tag{26}$$

The overall running time needed for finding w_p^* and the corresponding optimal schedule is $O(n \log n)$ due to the sorting of jobs in the LPT order.

For the restricted version, the jobs in an optimal schedule still follow the LPT order, but there may exist a job that straddles the release date $\bar{r}$. As in the restricted case, no optimal schedule may contain more than $h = \lfloor 1/\beta \rfloor$ early jobs.

With the standard processing times, find how many jobs taken in the LPT order can fit into the interval $[\underline{r}, \bar{r}]$ and denote this number by m.

For each job k, $m \leqslant k \leqslant h$, we determine the interval $[w'_p(k), w''_p(k)]$ such that while w_p varies within this interval only jobs $1, 2, \ldots, k$ are early with job k straddling. Note that

$$w''_p(k) = \frac{\bar{r} - \underline{r}}{\sum_{j=1}^{k} p_j} = w'_p(k-1).$$

For $w_p \in [w'_p(k), w''_p(k)]$ the objective function becomes

$$F(w_p) = \left(\underline{r} + \sum_{j=1}^{n} w_p p_j \right) + \beta \sum_{j=1}^{k} \left(\bar{r} - \sum_{i=1}^{j-1} w_p p_i \right) + c_p w_p^{-q}.$$

The function is convex and can be minimized as follows. Taking the derivative, we obtain

$$\frac{d(F(w_p))}{d w_p} = \sum_{j=1}^{n} p_j - \beta \sum_{j=1}^{k} \sum_{i=1}^{j-1} p_i - q c_p w_p^{-q-1} = 0.$$

The root of this equation is given by

$$Q = \left(\frac{q c_p}{\sum_{j=1}^{n} p_j - \beta \sum_{j=1}^{k} \sum_{i=1}^{j-1} p_i} \right)^{1/(q+1)}$$

and the optimal value of the processing rate $w_p^*(k)$ over the interval $[w'_p(k), w''_p(k)]$ satisfies

$$w_p^*(k) = \begin{cases} w'_p(k), & \text{if } w'_p(k) \geqslant Q \\ w''_p(k), & \text{if } w''_p(k) \leqslant Q \\ Q, & \text{if } w'_p(k) < Q < w''_p(k). \end{cases}$$

We find the values $w_p^*(k)$ for all k between m and h inclusive. Note that if the processing rate is further decreased, for $w_p \in (0, w'_p(h)]$ we obtain the situation similar to the unrestricted case. Therefore, we additionally compute w_p^* by (26) as in the unrestricted case and, if $w_p^*(h) > w_p^*$, redefine $w_p^*(h) := w_p^*$. An optimal processing speed corresponds to the found value $w_p^*(k)$ that delivers the minimum to the overall cost $F(w_p)$. The running time of this algorithm is $O(n \log n)$, since finding each of $O(n)$ values $w_p^*(k)$ takes constant time, and all relevant values of $F(w_p)$ can be computed in linear time.

Table 1
Summary of the results

Release dates	Processing times		
	Constant	Controllable processing speed	Controllable processing times
Constant	Section 1 $O(n \log n)$	Section 2.1 $O(n \log n)$	[20,24], Section 2.2 $O(n^2)$
Controllable release speed	Section 2.1 $O(n \log n)$	Section 2.1 $O(n \log n)$	Section 2.2 $O(n^3)$
Controllable release dates (general case)	Section 3.1 Unary NP Binary NP for $\beta_j = \beta$ Pseudopolynomial [15]	Section 3.1 Unary NP Binary NP for $\beta_j = \beta$	Section 3.1 Unary NP Binary NP for $\beta_j = \beta$
Controllable release dates (common bounds)	Section 3.1.1 Binary NP $O(np(N))$ unrestricted $O(n^2 p(N) p_{\max})$ restricted Section 3.2 $O(n \log n)$ for $\beta_j = \beta$	Section 3.1.1 Binary NP Section 3.2 $O(n \log n)$ for $\beta_j = \beta$	Section 3.1.1 Binary NP [5]: $O(n^3)$ for $\beta_j = \beta$ [5]: $O(n^2)$ for $\beta_j = \beta$, $\alpha_j = \alpha$

The results of this subsection can be summarized as follows.

Theorem 8. *Both restricted and unrestricted versions of the single machine problem with controllable release dates and controllable processing speed are solvable in* $O(n \log n)$ *time, provided that all release dates have the same upper bound* $\bar{r}$ *and the same lower bound* $\underline{r}$*, and the release date compression costs are equal.*

4. Conclusion

This paper provides a fairly complete complexity classification of single machine scheduling problems with both job-related and machine-related controllable processing and release parameters; see Table 1 for the summary. In particular, this paper is the first attempt to combine individually controlled processing time with an appropriate choice of the release speed, common for all jobs. We design a number of polynomial and pseudopolynomial algorithms which may serve for the justified choice of the values of the controllable parameters, thereby giving a powerful negotiation tool in many practical situations, including those arising in supply chain management.

Design and analysis of approximation algorithms will be an attractive direction for future research. For example, for problems studied in Section 3.1.1, it is interesting to find out whether they admit fully polynomial approximation schemes either by converting the corresponding dynamic programming algorithms or by adapting the existing scheme for a closely related integer quadratic programming problem HALF-PRODUCT (see [2]).

Acknowledgements

The authors are grateful to an anonymous referee for his/her comments that considerably improved the paper. The second author was partly supported by INTAS (Project 03-51-5501).

Appendix. Proof of Theorem 1

First, observe that u_0 remains critical for each $w > w_{k-1}$. Recall that w_{k-1} and w_k are obtained by Meggido's algorithm [18] applied to all uncrashed jobs. If for some value of w Algorithm Min(φ_k) finds schedule $S(w)$ in which

another job $u' > u_0$ is critical, then this schedule cannot be optimal, since some (partly) crashed job between jobs u_0 and u' could be decompressed thereby decreasing the compression cost without increasing the makespan.

In all schedules found by Algorithm Min(φ_k) all jobs preceding job u_0 remain uncrashed because their compression does not decrease the makespan of the schedule but increases the compression cost. Therefore, while making transition from schedule $S(w)$ to schedule $S(w')$ only some jobs of the set $\{u_0, \ldots, n\}$ could be subject to compression.

The algorithm starts with schedule $S(w_k)$ that is optimal since it is found by Algorithm CPT. Suppose that after some iterations schedule $S(w)$ for $w_{k-1} < w \leqslant w_k$ is found and that schedule is optimal. We demonstrate that schedule $S(w')$ found in the next iteration is also optimal.

For schedule $S(w)$, let $J(w)$ and $U(w)$ be the sets of the crash jobs and the zero-slack jobs, respectively, and z be the number of the crash jobs. If the value of w is decreased, the jobs of set $U(w)$ receive a positive slack, so that the jobs of set $J(w)$ can be further compressed. For each i, $1 \leqslant i \leqslant z$, Algorithm Min($\varphi_k$) determines the value h_i such that the slack of job u_i with respect to the new release date $h_i r_{u_i}$ remains zero, provided that job j_i is fully crashed.

Temporarily assume that there is no lower bound on possible values of the processing times. Take any $\tilde{w}$, $w_{k-1} \leqslant \tilde{w} < w$. Due to Lemma 1, the problem of finding an optimal schedule $S(\tilde{w})$ can be seen as the problem with controllable processing times and a fixed release rate $\tilde{w}$, provided that original processing times are equal to the current processing times in schedule $S(w)$. This problem can be solved by Algorithm CPT. The algorithm will first take a compressible job of set $\{u_0, \ldots, n\}$ with the minimum compression cost no greater than 1 (this job will be job j_1) and compress it till a zero-slack job appears (this job will be job u_1). In a similar way, job j_2 will be found and compressed and a zero-slack job u_2 will be formed, etc. Thus, in transition from $S(w)$ to $S(\tilde{w})$ only jobs $j_1, \ldots, j_z$ will be subject to compression. See [10] for an animated presentation of this discussion.

For each job $j_i \in J(w)$, the amount Δx_{j_i} of this compression satisfies (10). Indeed, for each i, $1 \leqslant i \leqslant z$, the jobs u_{i-1} and u_i have zero slacks in both schedules $S(w)$ and $S(\tilde{w})$. We obtain

$$wr_{u_i} = wr_{u_{i-1}} + [\varrho(u_i) - \varrho(u_{i-1})],$$
$$\tilde{w}r_{u_i} = \tilde{w}r_{u-1} + [\varrho(u_i) - \varrho(u_{i-1})] - \Delta x_{j_i},$$

so that (10) holds.

In order to satisfy the lower bound restrictions on the processing times, we choose w' as the smallest possible value of $\tilde{w} \geqslant w_{k-1}$, which guarantees that the compressions Δx_{j_i} are all feasible. It follows from (10) and $\Delta x_{j_i} \leqslant p_{j_i} - \underline{p}_{j_i}$ that for each i, $1 \leqslant i \leqslant z$, the inequalities

$$\tilde{w} \geqslant w - (p_{j_i} - \underline{p}_{j_i})/(r_{u_i} - r_{u_{i-1}})$$

hold. Therefore, due to (11), the smallest value of $\tilde{w}$ is determined by (12). The theorem is proved.

References

[1] A. Agnetis, N.G. Hall, D. Pacciarelli, Supply chain scheduling: sequence coordination, Discrete Appl. Math., this issue doi:10.1016/j.dam.2005.04.019.

[2] T. Badics, E. Boros, Minimization of half-products, Math. Oper. Res. 23 (1998) 649–660.

[3] B. Chen, C.N. Potts, G. Woeginger, A review of machine scheduling: complexity, algorithms and approximability, in: D.-Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Kluwer, Dordrecht, 1998, pp. 21–169.

[4] Z.-L. Chen, N.G. Hall, Supply chain scheduling: assembly systems, Working Paper, Fisher College of Business, The Ohio State University, 2001.

[5] T.C.E. Cheng, M.Y. Kovalyov, N.V. Shakhlevich, Scheduling with controllable release dates and processing times: Makespan minimization, European J. Oper. Res., to appear.

[6] J. Du, J.Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard, Math. Oper. Res. 15 (1990) 483–495.

[7] V.S. Gordon, J.-M. Proth, C. Chu, A survey of the state-of-the-art of common due date assignment and scheduling research, European J. Oper. Res. 139 (2002) 1–25.

[8] N.G. Hall, Supply chain scheduling, in: Book of Abstracts of the International Symposium on Combinatorial Optimization, Paris, 2002, pp. 10–11.

[9] N.G. Hall, C.N. Potts, Supply chain scheduling: batching and delivery, Oper. Res. 51 (2003) 566–584.

[10] ⟨http://staffweb.cms.gre.ac.uk/~sv02/papers/animations1.html⟩.

[11] H. Ishii, T. Masuda, T. Nishida, Two machine mixed shop scheduling problem with controllable machine speeds, Discrete Appl. Math. 17 (1987) 29–38.

[12] H. Ishii, T. Nishida, Two machine open shop scheduling problem with controllable machine speeds, J. Oper. Res. Soc. Japan 29 (1986) 123–131.

[13] A. Janiak, Single machine scheduling problem with a common deadline and resource dependent release dates, European J. Oper. Res. 53 (1991) 317–325.
[14] A. Janiak, M.Y. Kovalyov, W. Kubiak, F. Werner, Positive half-products and scheduling with controllable processing times, European J. Oper. Res. 165 (2005) 416–422.
[15] E.L. Lawler, A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness, Ann. Discrete Math. 1 (1977) 331–342.
[16] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Sequencing and scheduling: algorithms and complexity, in: S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin (Eds.), Handbooks in Operations Research and Management Science, vol. 4, Logistics of Production and Inventory, North-Holland, Amsterdam, 1993, pp. 455–522.
[17] C.L. Li, Scheduling with resource dependent release dates—a comparison of two different resource consumption functions, Naval Res. Logist. 41 (1994) 807–879.
[18] N. Meggido, Combinatorial optimization with rational objective functions, Math. Oper. Res. 4 (1979) 414–424.
[19] E. Nowicki, An approximation algorithm for a single machine scheduling problem with release times, delivery times and controllable processing times, European J. Oper. Res. 72 (1994) 74–81.
[20] E. Nowicki, S. Zdrzałka, A survey of results for sequencing problems with controllable processing times, Discrete Appl. Math. 26 (1990) 271–287.
[21] W.E. Smith, Various optimizers for single stage production, Naval Res. Logist. Quart. 3 (1956) 59–66.
[22] V.A. Strusevich, Two machine flow shop scheduling problem with no-wait in process: controllable machine speeds, Discrete Appl. Math. 59 (1995) 75–86.
[23] C.P.M. van Hoesel, A.P.M. Wagelmans, M. van Vliet, An $O(n \log n)$ algorithm for the two-machine flow shop problem with controllable machine speeds, INFORMS J. Comput. 8 (1996) 376–382.
[24] L.N. Van Wassenhove, K.R. Baker, A bicriterion approach to time/cost tradeoffs in sequencing, European J. Oper. Res. 11 (1982) 48–54.
[25] J. Yuan, The NP-hardness of the single machine common due date weighted tardiness problem, Systems Sci. Math. Sci. 5 (1992) 328–333.