SINGLE MACHINE SCHEDULING WITH CONTROLLABLE PROCESSING TIMES BY SUBMODULAR OPTIMIZATION

NATALIA V. SHAKHLEVICH
School of Computing, University of Leeds, Leeds LS2 9JT, U.K.

AKIYOSHI SHIOURA
Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan

and

VITALY A. STRUSEVICH
School of Computing and Mathematical Sciences, University of Greenwich, Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, U.K.

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

In scheduling with controllable processing times the actual processing time of each job is to be chosen from the interval between the smallest (compressed or fully crashed) value and the largest (decompressed or uncrashed) value. In the problems under consideration, the jobs are processed on a single machine and the quality of a schedule is measured by two functions: the maximum cost (that depends on job completion times) and the total compression cost. Our main model is bicriteria and is related to determining an optimal trade-off between these two objectives. Additionally, we consider a pair of associated single criterion problems, in which one of the objective functions is bounded while the other one is to be minimized. We reduce the bicriteria problem to a series of parametric linear programs defined over the intersection of a submodular polyhedron with a box. We demonstrate that the feasible region is represented by a so-called base polyhedron and the corresponding problem can be solved by the greedy algorithm that runs two orders of magnitude faster than known previously. For each of the associated single criterion problems, we develop algorithms that deliver the optimum faster than it can be deduced from a solution to the bicriteria problem.

Keywords: Single machine scheduling; Controllable processing times; Polymatroid; Base polyhedron

1. Introduction

In scheduling with controllable processing times, the actual durations of the jobs are not fixed in advance, but have to be chosen from a given interval. Assigning a small value to the actual processing time normally leads to an earlier completion of the job. However, the choice of a smaller value of the processing time is associated with some cost. The main issue in scheduling with controllable processing times is a trade-off between a good scheduling performance achieved by compressing the values of the processing times and the cost of that reduction. The surveys [12, 14] give good reviews of the research in this area.

Formally, the problems that we study in this paper can be described as follows. The jobs of the set $N = \{1, 2, \ldots, n\}$ are simultaneously available at time zero to

be processed on a single machine. For each job $j \in N$, its actual processing time p_j has to be chosen from an interval $[\underline{p}_j, \overline{p}_j]$, where the values $\underline{p}_j$ and $\overline{p}_j$ are given in advance. If job $j \in N$ is assigned the processing time p_j, we say that the job has been *compressed* or *crashed* by $h_j = \overline{p}_j - p_j$, and refer to h_j as the *compression amount* of job j.

A usual scheduling requirement says that the machine cannot process more than one job at a time. Preemption is not allowed. A schedule for processing the jobs of set N can be given by the split-values p_j and h_j that satisfy

$$\overline{p}_j = p_j + h_j \tag{1}$$

and

$$\underline{p}_j \leq p_j \leq \overline{p}_j. \tag{2}$$

for each job $j \in N$ and by a sequence π according to which the jobs are processed by the machine. The processing of the job sequenced in position $\pi(k)$, i.e., at the k-th position in permutation π, is completed at time

$$C_{\pi(k)} = C_{\pi(k-1)} + p_{\pi(k)},$$

where for completeness $C_{\pi(0)} = 0$.

Processing a job j incurs the following two costs: (i) the penalty for completing job $j \in N$ at time C_j given by the cost function $f_j(C_j)$, and (ii) the compression cost $\alpha_j h_j$. In this paper, we assume that each f_j is a non-decreasing piecewise linear function. Moreover, as in [8], we assume that each function is *explicitly represented*. This means that for each function $f_j(t)$ we are given the number of its break-points denoted by m_j; the sorted list of the break-points $\underline{p}_j = t_j^0 < t_j^1 < \cdots < t_j^{m_j} \leq \sum_{j \in N} \overline{p}_j$; the pieces of the function $Q_j^{(k)} t + R_j^{(k)}$ for $t \in (t_j^{k-1}, t_j^k]$, where $1 \leq k \leq m_j$ and $Q_j^{(k)} > 0$. We denote the total number of pieces of all functions f_j by L, i.e., $L = \sum_{j=1}^{n} m_j$.

The overall quality of a schedule is measured in the terms of the *maximum processing cost*

$$F = \max_{j \in N} f_j(C_j) \tag{3}$$

and *the total compression cost*

$$K = \sum_{j \in N} \alpha_j h_j. \tag{4}$$

Function F is of a very general nature. Here we only mention its two special cases widely studied in the scheduling literature: the makespan $F = \max_{j \in N} C_j$ if

$$f_j(C_j) = C_j$$

and the maximum tardiness if

$$f_j(C_j) = \max\{C_j - d_j, 0\} \tag{5}$$

defines the tardiness of job j with respect to a given due date d_j.

In the bicriteria problem, we want to determine a trade-off between the processing cost F of the form (3) and the compression cost K of the form (4). A solution to this bicriteria problem is obtained by the set of Pareto optimal points; see [17, Chapters 3 and 4] for definitions and a state-of-the-art survey of multicriteria scheduling. Here, we recall that a schedule S' is called *Pareto optimal* if there exists no schedule S'' such that $F(S'') \leq F(S')$ and $K(S'') \leq K(S')$, where at least one of these relations holds as a strict inequality. Having found the set of Pareto optimal solutions, usually in the form of a so-called *efficiency frontier* given by its break-points, the decision-maker may choose a schedule according to various combinations of the two criteria, e.g., their weighted sum; see [12, 17]. Extending standard notation for scheduling problems, we denote the bicriteria problem by $1|p_j = \bar{p}_j - h_j|(F, K)$. Here, the first field indicates that the jobs are processed on a single machine. We write '$p_j = \bar{p}_j - h_j$' in the middle field to stress that the processing times are controllable so that the given times $\bar{p}_j$ can be compressed. The last field specifies the functions to be minimized simultaneously.

Although the described bicriteria problem is the main target of this study, we also discuss associated problems to minimize one of the functions, while the value of the other function is bounded. In the problem with limited processing cost, the goal is to minimize the total compression cost $K = \sum \alpha_j h_j$, provided that the processing cost of each job does not exceed a given value U; we denote this problem by $1|p_j = \bar{p}_j - h_j,\ f_j(C_j) \leq U|\ \sum \alpha_j h_j$. In the problem with limited compression cost, the goal is to minimize the maximum processing cost F, provided that the total compression cost does not exceed a given value V; we denote this problem by $1|p_j = \bar{p}_j - h_j,\ \sum \alpha_j h_j \leq V|\ F$.

Notice that the optimal solution to the bicriteria problem given in the form of an efficiency frontier delivers a solution to either of the associated single criterion problems; thus, the purpose of study of these single criterion problems is to develop algorithms that run faster than the algorithm that solves the bicriteria problem.

To illustrate the problems under consideration, below we present several applications.

Make-or-Buy Decision-Making. Often a production manager realizes that either the existing production capabilities are insufficient to fulfill all orders internally in time or the cost of work-in-process of an order exceeds a desirable amount. Such an order can be partly subcontracted. Obviously, subcontracting incurs additional cost but that can be either compensated by quoting realistic deadlines for all jobs or balanced by a reduction in internal production expenses. The make-or-buy decisions should be taken to determine which part of each order is manufactured internally and which is subcontracted. Here the internal production facility is the machine. For each order $j \in N$, the value of $\overline{p}_j$ is interpreted as the processing requirement, provided that the order is manufactured internally in full, while $\underline{p}_j$ is a given mandatory limit on the internal production. Further, $p_j = \overline{p}_j - h_j$ is the chosen actual time for internal manufacturing, where h_j shows how much of the order is subcontracted and $\alpha_j h_j$ is the cost of this subcontracting. The cost function $f_j(C_j)$ is the work-in-process cost of the order. Finally, the function F of the form (3) represents the maximum cost of processing those orders and their parts that are accepted for internal manufacturing, while the function K expresses the total subcontracting cost.

Imprecise Computation. We illustrate this application by discussing problem $1|p_j = \bar{p}_j - h_j,\ f_j(C_j) \le U|\ \sum \alpha_j h_j$; see [11] for a review of research in this problem area. In image processing, to obtain a full quality picture $\bar{p}_j$ time units are required, and at least $\underline{p}_j$ time units are required for mandatory processing. The actual value $p_j = \overline{p}_j - h_j$ of the processing time is sufficient to obtain a picture of acceptable quality. In this case $\alpha_j h_j$ measures the loss of quality or an error of processing. Assuming that $f_j(C_j)$ is the tardiness defined by (5), the problem $1|p_j = \bar{p}_j - h_j,\ f_j(C_j) \le U|\ \sum \alpha_j h_j$ is that of finding a schedule that minimizes the total error, provided that the processing of each image is finished no more than U time units later than its due date; typically $U = 0$, so that the resulting schedule is deadline feasible.

Below we give a brief overview of the known results on single machine scheduling with controllable processing times. For problem $1|p_j = \bar{p}_j - h_j|(F, K)$ with arbitrary non-decreasing totally ordered cost functions f_j that satisfy $f_j(t) \le f_{j+1}(t)$ for all t and all $1 \le j \le n-1$, a procedure for finding the efficiency frontier on a grid of any given refinement is described by Van Wassenhove and Baker [20]. If the functions f_j are piecewise linear, the procedure requires $O(n^3)$ time, even if f_j is the tardiness defined by (5).

For problem $1|p_j = \bar{p}_j - h_j|(F, K)$ with general non-decreasing cost functions f_j, Tuzikov [18] develops an algorithm that finds an ε-approximation of the efficiency frontier.

Hoogeveen and Woeginger [8] solve problem $1|p_j = \bar{p}_j - h_j|(F, K)$ with arbitrary non-decreasing piecewise linear functions f_j in $O(n^4L^2)$ time where L is the total number of pieces of all functions f_j, $j \in N$. Notice that in the model considered in [8] the lower bounds on the processing times are all zero, i.e., $\underline{p}_j = 0$ for all $j \in N$. Observe also that Hoogeveen and Woeginger do not give any details of how to implement the pre-processing stage and do not analyze the running time of that stage. In this paper, we explain the details of the preprocessing stage and analyze its running time; see Section 2.

We are not aware of any prior work regarding a single criterion problem $1|p_j = \bar{p}_j - h_j, \sum \alpha_j h_j \le V|\ F$. As far as problem $1|p_j = \bar{p}_j - h_j,\ f_j(C_j) \le U|\ \sum \alpha_j h_j$ is concerned, it can be reduced to that of finding a schedule that is feasible with respect to some deadlines induced by the constraints $f_j(C_j) \le U$. This approach, originally due to [20], is outlined in Section 4.1. The resulting problem can be denoted by $1|p_j = \bar{p}_j - h_j, C_j \le d_j|\sum \alpha_j h_j$, where we write '$C_j \le d_j$' in the middle field to stress that each job j must be completed by its deadline d_j. In a symmetric and mathematically equivalent version of this problem, denoted by $1|p_j = \bar{p}_j - h_j, r_j, d_j = d|\sum \alpha_j h_j$, a job j becomes available at time r_j and all jobs must be completed by a common deadline d. In either problem, the processing time can be compressed to guarantee a deadline feasible schedule and the total compression cost is minimized. These two equivalent single criterion problems are the most studied scheduling problems with controllable processing times; see, e.g., [8, 9, 12, 15, 21]. Each of these problems is solvable in $O(n \log n)$ time, see the algorithms in [8, 9, 15]; however, only the latter paper provides all necessary implementation details.

It has been observed in [15, 16] that various single criterion and bicriteria problems with controllable processing times can be reduced to optimization problems over polymatroids and generalized polymatroids. This gives us an opportunity that arises very seldom in scheduling research: instead of designing a purpose-built

method to tackle an individual problem (and often this and only this problem) we can come up with a general framework of handling the whole range of scheduling problems of this type. Such an approach allows us to use powerful techniques of submodular optimization; see [5], [10], [13, Chapter 44] for books and surveys in this area. In this paper, we take further advantage from employing the polymatroidal approach. The main benefits include a clear justification of the greedy-like algorithms and a possibility to adapt known computationally efficient solution methods. As a result, we arrive at the solution procedures that are easier to justify and faster to run that those developed earlier. For example, all prior algorithms aimed at solving bicriteria problems with controllable processing times are based on searching for the candidate break-points of the efficiency frontier one by one; our algorithm presented in Sections 2 and 3 finds the break-points by solving a number of parametric linear programming problems.

The remainder of this paper is organized as follows. In Section 2, we discuss the bicriteria problem and reduce it to a series of parametric linear programs over regions described by nested constraints. In Section 3, we present a novel general method that reduces the obtained linear programs to optimization problems over so-called base polyhedra. We use this fact to design a fast polynomial-time algorithm for finding the trade-off curve for the original bicriteria problem. Section 4 is devoted to the pair of conjugate single criterion problems related to the bicriteria problem. In Subsection 4.1, we address the problem of minimizing the compression cost under a limited processing cost and show how this problem can be solved by the methods developed in [6, 7] for resource allocation problems. Subsection 4.2 studies the single criterion problem of minimizing the maximum processing cost subject to a constraint on the decompression cost. We give a fast solution procedure that is closely linked to the other algorithms of this paper. The conclusions are summarized in Section 5.

2. Bicriteria Problem: Mathematical Programming Formulation

In this section, we reformulate the bicriteria problem $1|p_j = \bar{p}_j - h_j|(F, K)$ in terms of a sequence of parametric linear programming (LP) problems with nested constraints. Recall that the objective functions F and K are defined by (3) and (4), respectively, and each function $f_j(C_j)$ is non-decreasing and piecewise linear with m_j break-points with the number of all break-points equal to L.

Following [8], we adopt a natural graphical interpretation of the problem. Imagine that the graphs of the functions $y = f_j(t)$ are drawn in the coordinate plane with t as the horizontal independent variable and y as the vertical dependent variable. Variable t can be thought of as a time variable related to the completion times C_j of jobs $j \in N$. By condition, all jobs cannot be completed earlier than time $\sum_{j=1}^{n} \underline{p}_j$ and later than time $\sum_{j=1}^{n} \bar{p}_j$. Determine the following sets of *special* points in the plane:

- Set S_1 consists of all break-points of all piecewise linear functions $f_j(t)$;
- Set S_2 consists of all intersection points of the graphs of two functions f_i and f_j for $i \in N, j \in N, i \neq j$;
- Set S_3 consists of all intersection points of the graphs of functions f_j with the vertical lines $t = \sum_{j=1}^{n} \underline{p}_j$ and $t = \sum_{j=1}^{n} \bar{p}_j$.

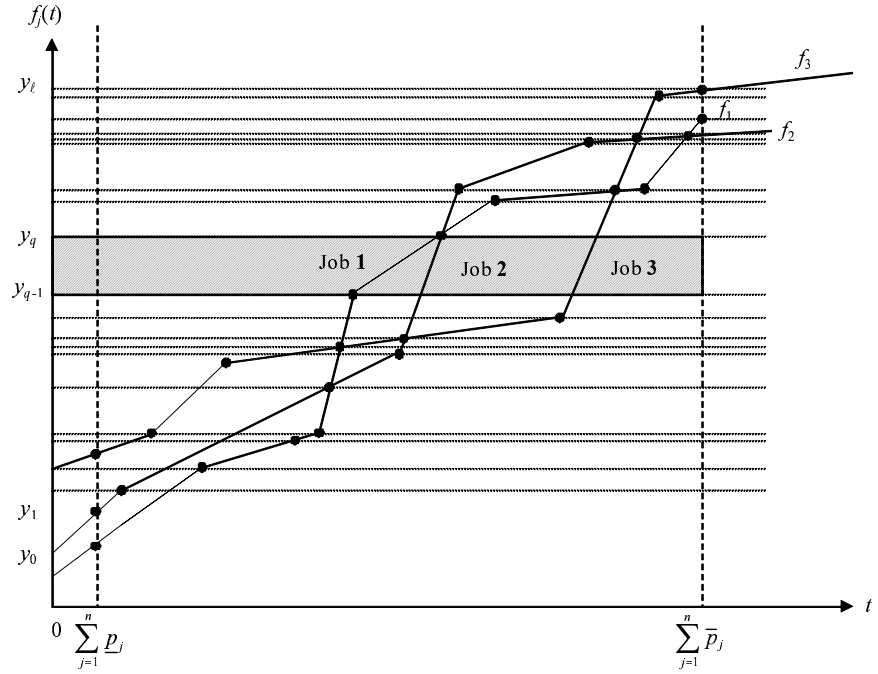

Fig. 1. An example with $n = 3$ jobs with proper numbering in stripe $[y_k, y_{k+1}]$

It is clear that $|S_2|$ does not exceed $O(L^2)$. The theorem below gives a tight upper bound on a possible number of points in set S_2.

Theorem 1 *Given n piecewise-linear functions f_j, each consisting of m_j linear pieces and defined over the interval $[0, T]$, the number of intersection points of the graphs of any two functions f_i and f_j for $i = 1, \ldots, n$, $j = 1, \ldots, n$, $i \neq j$, does not exceed $(n-1)L - n(n-1)/2$, where $L = \sum_{j=1}^{n} m_j$.*

Although Theorem 1 is of interest in its own right, we move its proof to the Appendix, in order not to affect the flow of logic of this section.

It is obvious that $|S_1| = L$ and $|S_3| = O(n)$, while Theorem 1 implies that $|S_2| = O(nL)$. Split the coordinate plane into horizontal stripes by drawing a horizontal line through each special point. Denote the total number of the obtained stripes by ℓ, where $\ell = O(n + L + nL) = O(nL)$. A stripe is defined by two lines $y = y_{q-1}$ and $y = y_q$, where

$$y_0 < y_1 < \cdots < y_\ell,$$

and we will refer to the stripe between $y = y_{q-1}$ and $y = y_q$ as $[y_{q-1}, y_q]$.

A stripe $[y_{q-1}, y_q]$ can be put into correspondence to a class of schedules for which (i) the value of the processing cost F lies within the interval $[y_{q-1}, y_q]$; (ii) each cost function $f_j(t)$ is represented by a linear expression; and (iii) the jobs are processed according to the same sequence obtained by scanning the segments within the stripe starting from the left-most segment. See Figure 1.

The process of finding all stripes $[y_{q-1}, y_q]$, $q = 1, \ldots, \ell$, will be called the pre-processing. Below we describe an efficient algorithm for finding all stripes.

Algorithm Preproc

Step 1. Find the sets S_1 and S_3. Determine the increasing sequence $\tilde{y}_0 < \tilde{y}_1 < \cdots < \tilde{y}_m$ of the ordinates of all points in $S_1 \cup S_3$.

Step 2. For each r from 1 to m do the following:

- **(a)** For each function f_j, $j \in N$, identify the linear piece that represents the function in the interval $[\tilde{y}_{r-1}, \tilde{y}_r]$.
- **(b)** Find intersections of the line segments of functions f_j, $j \in N$, in the interval $[\tilde{y}_{r-1}, \tilde{y}_r]$, either by the algorithm by Balaban [1] or by the algorithm by Bently and Ottmann [2].
- **(c)** Store the ordinates of the found intersection points in the increasing order between $\tilde{y}_{r-1}$ and $\tilde{y}_r$.

Step 3. Output the stripes $[y_{q-1}, y_q], q = 1, \ldots, \ell$, by renumbering the values $\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_m$ together with those found in Step 2.

Lemma 1 *Algorithm Preproc can be implemented in $O(nL \log n)$ time.*

Proof. Since each function f_j is explicitly represented, as described in Section 1, and is non-decreasing, it follows that for each function f_j, $j \in N$, its break-points are given as a list that is sorted in increasing order of their ordinates. Let $\sigma_1(j)$ be the increasing sequence of the ordinates of the break-points of function f_j, $j \in N$. Using a merge-sort algorithm we can merge all these sequences into a single increasing sequence σ_1. Since there are n sequences to be merged that contain in total L elements, the sequence σ_1 can be found in $O(nL)$ time.

An increasing sequence σ_3 of the ordinates of the points that belong to set S_3 can be found in $O(n \log n)$. To complete Step 1, we need to merge the sequences σ_1 and σ_3, which requires $O(L + n) = O(L)$ time. Thus, Step 1 can be implemented in $O(nL)$ time.

For $r = 1$, the linear pieces that represent the functions f_j in the interval $[\tilde{y}_{r-1}, \tilde{y}_r]$ are the first pieces of the functions. For any other r, a function f_j is either represented by the same piece as in the previous interval or by the next piece in its explicit representation. Thus, for each r, Step 2(a) can be implemented in $O(n)$ time.

For an r, $1 \leq r \leq m$, suppose that there exist k_r points in S_2 in the interval $[\tilde{y}_{r-1}, \tilde{y}_r]$. Notice that each function f_j is linear in each interval $[\tilde{y}_{r-1}, \tilde{y}_r]$, therefore in Step 2(b) either Balaban's algorithm [1] or the Bently-Ottmann algorithm [2] for finding intersections of line segments can be applied for identifying points of set S_2 with the ordinates from $[\tilde{y}_{r-1}, \tilde{y}_r]$.

Recall that Balaban's algorithm finds the intersecting pairs of ν line segments in $O(\nu \log \nu + \lambda)$ time, where λ is the number of intersecting pairs, while the Bently-Ottmann algorithm requires $O(\nu \log \nu + \lambda \log \nu)$. However, unlike the Bently-Ottmann algorithm, the algorithm by Balaban does not output the found intersection points in increasing order of their ordinates. This means that if for each r, $1 \leq r \leq m$, Balaban's algorithm is used in Step 2(b), then it takes $O(n \log n + k_r)$ time and we additionally need to sort the found points in Step 2(c), which requires extra $O(k_r \log k_r)$ time. Since $k_r \leq n^2$, we conclude that using Balaban's algorithm

we can run Step 2 in $O(n \log n + k_r \log n)$ time for each r, $1 \leq r \leq m$. If the Bently-Ottmann algorithm is used, the same running time is achieved.

Due to Theorem 1, the time complexity of Steps 2 and 3 can be estimated as

$$\begin{aligned} O\left(\sum_{r=1}^{m}(n \log n + k_r \log n)\right) &= O\left(mn \log n + \left(\sum_{r=1}^{m} k_r\right) \log n\right) \\ &= O(|S_1 \cup S_3| n \log n + |S_2| \log n) \\ &\leq O(Ln \log n + nL \log n) = O(nL \log n), \end{aligned}$$

as required. □

To find the efficiency frontier for the bicriteria problem $1|p_j = \bar{p}_j - h_j|(F, K)$ we need to find a set of Pareto-optimal points for each stripe.

Let $[\underline{y}, \overline{y}]$ be an arbitrary stripe. For each job $j \in N$ the cost function $f_j(t) = Q_j^{(k)} t + R_j^{(k)}$ for some k, $1 \leq k \leq m_j$, takes the values within the stripe. We may write $f_j(t) = Q_j t + R_j$ dropping the reference to the index k of the corresponding linear piece. For each job $j \in N$ and a small positive ε, determine the intersection points $(t_j, \overline{y})$ and $(t_j^{\varepsilon}, \overline{y} - \varepsilon)$ of the graph of the function $f_j(t) = Q_j t + R_j$ with the horizontal lines $y = \overline{y}$ and $y = \overline{y} - \varepsilon$, respectively. If required, renumber the jobs in such a way that $t_1 \leq t_2 \leq \cdots \leq t_n$, breaking ties by assigning a smaller number to the job with a smaller value t_j^{ε}. Call this numbering *proper* for this stripe; see the numbering of jobs in stripe $[\underline{y}, \overline{y}] = [y_{q-1}, y_q]$ shown in Figure 1, where the tie for jobs 1 and 2 is broken as described.

Suppose that we know the actual processing times p_j of all jobs in some Pareto-optimal schedule for a fixed $y \in [\underline{y}, \overline{y}]$. For some job $j \in N$, let t'_j be the value such that $y = f_j(t'_j) = Q_j t'_j + R_j$. Notice that the value of t'_j can be seen as the due date for job j, so that if job j completes at time $C_j \leq t'_j$ then the cost value $f_j(C_j)$ is no more than y. Under the proper numbering of jobs, we have that $t'_1 \leq t'_2 \leq \cdots \leq t'_n$. Using this interpretation, the proper numbering can be seen as obtained by sorting the jobs in accordance with a popular Earliest Due Date (EDD) rule. This means that the completion times of the jobs satisfy

$$C_j = \sum_{i=1}^{j} p_i, \ j = 1, \ldots, n.$$

We also deduce that the inequality

$$Q_j C_j + R_j \leq y$$

must hold for each job $j \in N$, which can be rewritten as

$$\sum_{i=1}^{j} p_i \leq \frac{y}{Q_j} - \frac{R_j}{Q_j}.$$

Thus, for finding a set of Pareto-optimal points for a stripe $[\underline{y}, \overline{y}]$ we need to solve the following parametric LP problem that we denote by Problem $LP(y)$:

$$LP(y): \quad \text{Maximize} \sum_{j\in N} \alpha_j p_j$$

$$\text{subject to} \quad \sum_{i=1}^{j} p_i \le a_j y + b_j, \quad 1 \le j \le n, \tag{6}$$

$$\underline{p}_j \le p_j \le \overline{p}_j, \quad 1 \le j \le n, \tag{7}$$

$$y \in [\underline{y}, \overline{y}], \tag{8}$$

where

$$a_j = \frac{1}{Q_j}, \; b_j = -\frac{R_j}{Q_j}, \; j \in N.$$

Notice that maximizing the quantity $\sum \alpha_j p_j$ is equivalent (up to an additive constant) to minimizing the total compression cost $\sum \alpha_j h_j$. For each stripe $[y_{q-1}, y_q]$, an optimal solution to Problem $LP(y)$ is a piecewise linear function of $y \in [y_{q-1}, y_q]$, so that all these functions found for all stripes together form the efficiency frontier for the original bicriteria problem $1|p_j = \bar{p}_j - h_j|(F, K)$.

3. Bicriteria Problem: Solution via Submodular Optimization

In this section, we demonstrate that Problem $LP(y)$ reduces to optimization of a linear function over a feasible region that is given by submodular constraints. As a result, we obtain an algorithm that solves this problem in $O(n^2)$ time, and therefore the original bicriteria problem is solvable in $O(n^3 L)$ time.

For completeness, we start this section with definitions related to submodular optimization. Unless stated otherwise, we follow a comprehensive monograph on this topic by Fujishige [5].

For a set $N = \{1, 2, \ldots, n\}$, let 2^N denote the set of all subsets of N. For two vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ and $\boldsymbol{z} = (z_1, z_2, \ldots, z_n) \in \mathbb{R}^n$, we write $\boldsymbol{x} \le \boldsymbol{z}$ to denote that $x_j \le z_j$ for each $j \in N$. A vector $\boldsymbol{x} \in X \subset \mathbb{R}^n$ is called *maximal* in X if there is no vector $\boldsymbol{z} \in X$ such that $\boldsymbol{x} \le \boldsymbol{z}$. For a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, define $x(\varnothing) = 0$ and $x(A) = \sum_{j\in A} x_j$ for a non-empty set $A \in 2^N$.

A collection $\mathcal{D}$ of subsets of N is called a *distributive lattice* if for any two sets in $\mathcal{D}$ their union and their intersection are both in $\mathcal{D}$, i.e., for $X \in \mathcal{D}$ and $Y \in \mathcal{D}$ we have that $X \cap Y \in \mathcal{D}$ and $X \cup Y \in \mathcal{D}$.

Definition 1 *A set-function $\psi : \mathcal{D} \to \mathbb{R}$ is called **submodular** if the inequality*

$$\psi(A \cup B) + \psi(A \cap B) \le \psi(A) + \psi(B) \tag{9}$$

holds for all sets $A, B \in \mathcal{D}$.

For a submodular function ψ defined on a distributive lattice $\mathcal{D} \subseteq 2^N$ such that $\varnothing, N \in \mathcal{D}$ and $\psi(\varnothing) = 0$, the pair $(\mathcal{D}, \psi)$ is called a *submodular system* on N, while ψ is referred to as the *rank function* of that system.

For a submodular system $(\mathcal{D}, \psi)$, define two polyhedra

$$P(\psi) = \{\boldsymbol{x} \in \mathbb{R}^n \mid x(A) \le \psi(A), A \in \mathcal{D}\}, \tag{10}$$

$$B(\psi) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in P(\psi), \; x(N) = \psi(N)\}, \tag{11}$$

called a *submodular polyhedron* and a *base polyhedron*, respectively, associated with the submodular system. Notice that $B(\psi)$ represents the set of all maximal vectors in $P(\psi)$. A submodular polyhedron associated with the pair $(2^N, \psi)$ is called a *polymatroid*, provided that the rank function ψ is monotone, i.e., ψ satisfies $\psi(A) \leq \psi(B)$ for $A \subseteq B$.

Consider the polyhedron defined by the inequalities (6) that partly define the region of Problem *LP*(y). Recall that the components of a vector $\boldsymbol{p}= (p_1, p_2, \ldots, p_n)$ follow the proper numbering for the stripe (8). Introduce the nested sets

$$N_0 = \varnothing,\ N_j = \{1, 2, \ldots, j\},\ j = 1, 2, \ldots, n.$$

It is obvious that the collection of sets $N_0, N_1, \ldots, N_n$ forms a distributive lattice $\mathcal{D}_{\text{nest}} \subseteq 2^N$. Moreover, for any set-function ψ defined on $\mathcal{D}_{\text{nest}}$ such that $\psi(\emptyset) = 0$, we deduce that the pair $(\mathcal{D}_{\text{nest}}, \psi)$ is a submodular system. Indeed, for any function ψ and any two sets $N_i, N_j \in \mathcal{D}_{\text{nest}}$, where $0 \leq i \leq j \leq n$, we have that $\psi(N_i) + \psi(N_j) = \psi(N_i \cap N_j) + \psi(N_i \cup N_j)$, i.e., ψ is submodular on $\mathcal{D}_{\text{nest}}$. If we define

$$\psi(N_j, y) = \begin{cases} 0, & j = 0, \\ a_j y + b_j, & 1 \leq j \leq n, \end{cases} \tag{12}$$

we see that the constraints (6) are equivalent to $p(N_j) \leq \psi(N_j, y)$, $0 \leq j \leq n$, and $P_{\text{nest}}(\psi) = \{\boldsymbol{p} \in \mathbb{R}^n \mid p(N_j) \leq \psi(N_j, y), j \in N\}$ is a submodular polyhedron. Thus, for a fixed $y \in [\underline{y}, \overline{y}]$, the system of constraints (6) and (7) defines a submodular polyhedron intersected with a box.

Various problems of scheduling with controllable processing times have been reduced to optimization of linear functions over polymatroids and so-called generalized polymatroids; see [15, 16]. The main advantage of this approach is that a linear function over a polymatroid can be maximized by the greedy algorithm. Such an algorithm considers the variables in non-increasing order of the coefficients of the objective function and gives each variable the largest possible value that does not violate feasibility of the current partial solution.

Below we demonstrate that problems similar to Problem *LP*(y) can be reduced to optimization over simpler structures, namely, over base polyhedra. In fact, we show that the problem of maximizing a linear function over an intersection of a submodular polyhedron and a box is equivalent to maximizing the same function over a base polyhedron. The benefit gained by such a reduction is that a solution to the problem of maximizing a linear function over a base polyhedron can be obtained essentially in closed form, as stated in the theorem below.

Theorem 2 (cf. [5]) *Consider the problem*

$$\begin{aligned} \text{Maximize} \quad & \sum_{j \in N} c_j x_j \\ \text{subject to} \quad & \boldsymbol{x} \in B(\psi), \end{aligned}$$

where $B(\psi)$ is a base polyhedron associated with a submodular function ψ. Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ be a permutation of indices such that

$$c_{\pi(1)} \geq c_{\pi(2)} \geq \cdots \geq c_{\pi(n)},$$

while $N_0^\pi = \varnothing$ *and* $N_j^\pi = (\pi(1), \pi(2), \ldots, \pi(j))$ *for* $j \in N$. *Then vector* $\boldsymbol{x}^* \in \mathbb{R}^n$ *given by*

$$x^*_{\pi(i)} = \psi(N_i^\pi) - \psi(N_{i-1}^\pi),\ i = 1, 2, \ldots, n$$

is an optimal solution.

For a submodular polyhedron $P(\psi)$ of the form (10) associated with a pair $(\mathcal{D}, \psi)$, define three polyhedra:

$$\begin{aligned} P(\psi)_l &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in P(\psi),\ \boldsymbol{x} \geq \boldsymbol{l}\}; \\ P(\psi)^u &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in P(\psi),\ \boldsymbol{x} \leq \boldsymbol{u}\}; \\ P(\psi)_l^u &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in P(\psi),\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}. \end{aligned}$$

The following two lemmas handle submodular polyhedra with either lower bounds or upper bounds on the decision variables.

Lemma 2 (cf. [5, Corollary 3.5, p. 49], [10, Lemma 3.5, p. 179]) *Polyhedron* $P(\psi)_l$ *is not empty if and only if* $\boldsymbol{l} \in P(\psi)$. *The set of maximal vectors of* $P(\psi)_l$ *is a base polyhedron* $B(\psi_l)$ *associated with* $(2^N, \psi_l)$, *where* $\psi_l(\varnothing) = 0$ *and*

$$\psi_l(A) = \min_{E \in \mathcal{D}, A \subseteq E} \{\psi(E) - l(E \backslash A)\}$$

for a non-empty set $A \in 2^N$.

The submodular system $(2^N, \psi_l)$ is called *the contraction of* $(\mathcal{D}, \psi)$ *by vector* $\boldsymbol{l}$.

Lemma 3 (cf. [5, Theorem 3.3, p. 48], [10, Lemma 3.4, p. 178]) *Polyhedron* $P(\psi)^u$ *is the same as a submodular polyhedron* $P(\psi^u)$ *associated with* $(2^N, \psi^u)$, *where* $\psi^u(\varnothing) = 0$ *and*

$$\psi^u(A) = \min_{E \in \mathcal{D}, E \subseteq A} \{\psi(E) + u(A \backslash E)\}$$

for a non-empty set $A \in 2^N$. *In particular, the set of maximal vectors of* $P(\psi)^u$ *forms a base polyhedron* $B(\psi^u)$ *associated with* $(2^N, \psi^u)$.

The submodular system $(2^N, \psi^u)$ is called *the restriction of* $(\mathcal{D}, \psi)$ *by vector* $\boldsymbol{u}$.

We now prove a similar statement regarding the polyhedron $P(\psi)_l^u$, which is the intersection of a submodular polyhedron with a box. In this case both transformations, restriction and contraction, should be applied to a submodular polyhedron.

Theorem 3 *Polyhedron* $P(\psi)_l^u$ *is not empty if and only if* $\boldsymbol{l} \in P(\psi)$ *and* $\boldsymbol{l} \leq \boldsymbol{u}$. *The set of maximal vectors of* $P(\psi)_l^u$ *is a base polyhedron* $B(\psi_l^u)$ *associated with* $(2^N, \psi_l^u)$, *where* $\psi_l^u(\varnothing) = 0$ *and*

$$\psi_l^u(A) = \min_{D \in \mathcal{D}} \{\psi(D) + u(A \backslash D) - l(D \backslash A)\} \tag{13}$$

for a non-empty set $A \in 2^N$.

Proof. By Lemma 3, the polyhedron $P(\psi)^u$ is a submodular polyhedron $P(\psi^u)$. Applying Lemma 2 to the submodular polyhedron $P(\psi^u)$ we obtain

$$\psi_l^u(A) = \min_{E \in 2^N, A \subseteq E} \{\psi^u(E) - l(E \backslash A)\}.$$

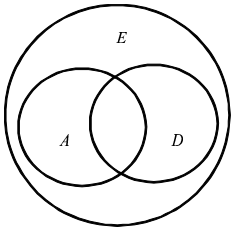

Fig. 2. The structure of sets A, D and E

In turn, writing out $\psi^u(E)$ in accordance with Lemma 3, we have that

$$\psi_l^u(A) = \min_{E\in 2^N, A\subseteq E} \left\{ \min_{D\in\mathcal{D}, D\subseteq E} \{\psi(D) + u(E\backslash D)\} - l(E\backslash A) \right\}.$$

Observe that

$$\begin{aligned} l(E\backslash A) &= l(D\backslash A) + l(E\backslash(A\cup D)); \\ u(E\backslash D) &= u(A\backslash D) + u(E\backslash(A\cup D)). \end{aligned}$$

See Figure 2 for illustration.

We further derive

$$\begin{aligned} \psi_l^u(A) &= \min_{E\in 2^N, A\subseteq E} \left\{ \min_{D\in\mathcal{D}, D\subseteq E} \{\psi(D) + u(A\backslash D) + u(E\backslash(A\cup D)) \right. \\ &\qquad \left. -l(D\backslash A) - l(E\backslash(A\cup D))\} \right\} \\ &= \min_{E\in 2^N, D\in\mathcal{D}, A\cup D\subseteq E} \{(\psi(D) + u(A\backslash D) - l(D\backslash A)) \\ &\qquad +u(E\backslash(A\cup D)) - l(E\backslash(A\cup D))\}. \quad (14) \end{aligned}$$

Since $u(E\backslash(A\cup D)) - l(E\backslash(A\cup D)) \geq 0$ and $A\cup D \subseteq E$, the minimum in (14) can be achieved by $E = A\cup D$, so that E can be removed, which yields (13). □

Notice that Theorem 3 can also be derived from [4, Proposition II.2.11], which addresses the truncation operation for generalized polymatroids.

Applying Theorem 3 to Problem $LP(y)$, we obtain the following statement.

Corollary 1 *Problem $LP(y)$ is equivalent to maximizing the objective function $\sum_{j\in N} \alpha_j p_j$ over the base polyhedron $B(\widetilde{\psi})$ for $y \in [\underline{y}, \overline{y}]$, such that $\widetilde{\psi}(\varnothing, y) = 0$ and for each non-empty set $A \subseteq N$*

$$\widetilde{\psi}(A, y) = \min_{0\leq j\leq n} \{\psi(N_j, y) + \overline{p}(A\backslash N_j) - \underline{p}(N_j\backslash A)\},$$

where $\psi(N_j, y)$ satisfies (12).

Corollary 1 enables us to show that Problem $LP(y)$ can be solved in $O(n^2)$ time. To solve Problem $LP(y)$ we may use Theorem 2 with $c_j = \alpha_j$ and $\psi(A) = \widetilde{\psi}(A, y)$. Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ be a permutation of indices such that

$$\alpha_{\pi(1)} \geq \alpha_{\pi(2)} \geq \cdots \geq \alpha_{\pi(n)}.$$

Since Problem $LP(y)$ is parametric, its optimal solution vector $\boldsymbol{p}^*(y) = (p_1^*(y), p_2^*(y), \ldots, p_n^*(y))$ depends on the parameter y. Due to Theorem 2 the solution is given by

$$p^*_{\pi(i)}(y) = \widetilde{\psi}(N_i^\pi, y) - \widetilde{\psi}(N_{i-1}^\pi, y),\ i = 1, 2, \ldots, n.$$

Recall that $\widetilde{\psi}(N_0^\pi, y) = 0$. To compute the values $\widetilde{\psi}(N_i^\pi, y)$ for all i, $1 \le i \le n$, determine a permutation σ of jobs such that

$$a_{\sigma(1)} \ge a_{\sigma(2)} \ge \cdots \ge a_{\sigma(n)}. \tag{15}$$

Let us fix some i, $1 \le i \le n$. Define $a_0 = b_0 = 0$, so that $\psi(N_j, y) = a_j y + b_j$ for all j, $0 \le j \le n$. Function

$$\begin{aligned} \widetilde{\psi}(N_i^\pi, y) &= \min_{0 \le j \le n} \{\psi(N_j, y) + \overline{p}(N_i^\pi \backslash N_j) - \underline{p}(N_j \backslash N_i^\pi)\} \\ &= \min_{0 \le j \le n} \{a_j y + b_j + \overline{p}(N_i^\pi \backslash N_j) - \underline{p}(N_j \backslash N_i^\pi)\} \end{aligned}$$

is a piecewise linear function of $y \in [\underline{y}, \overline{y}]$. Notice that all values $\overline{p}(N_i^\pi \backslash N_j) - \underline{p}(N_j \backslash N_i^\pi)$, $j = 0, 1, \ldots, n$, can be found in $O(n)$ time. Define

$$\beta_{i,j} = b_j + \overline{p}(N_i^\pi \backslash N_j) - \underline{p}(N_j \backslash N_i^\pi),\ j = 0, 1, \ldots, n,$$

so that

$$\widetilde{\psi}(N_i^\pi, y) = \min_{0 \le j \le n} \{a_j y + \beta_{i,j}\}.$$

Extend permutation (15) by additionally defining $a_{\sigma(n+1)} = a_0$ and $\beta_{i,\sigma(n+1)} = \beta_{i,0}$. The problem of computing the function

$$\widetilde{\psi}(N_i^\pi, y) = \min_{0 \le j \le n} \{a_j y + \beta_{i,j}\} = \min_{1 \le j \le n+1} \{a_{\sigma(j)} y + \beta_{i,\sigma(j)}\}$$

can be seen as the problem of finding the lower envelope of $n+1$ linear functions given in non-increasing order of their slopes $a_{\sigma(j)}$. Exactly this problem has been studied in [19] and has been shown to be solvable in $O(n)$ time.

Thus, for each set N_i^π, $0 \le i \le n$, the function $\widetilde{\psi}(N_i^\pi, y)$ can be found in linear time, i.e., for Problem $LP(y)$ finding the solution vector $\boldsymbol{p}^*(y) = (p_1^*(y), p_2^*(y), \ldots, p_n^*(y))$ takes $O(n^2)$ time.

The overall procedure for solving the original bicriteria problem can be stated as follows.

Algorithm Bicrit

Step 1. Find the stripes $[y_{q-1}, y_q]$, $q = 1, \ldots, \ell$, by running Algorithm Preproc.

Step 2. For $q = 1$ to ℓ, do the following:

- **(a)** Define $[\underline{y}, \overline{y}] = [y_{q-1}, y_q]$.
- **(b)** Formulate Problem $LP(y)$, i.e., find the proper numbering, the values a_j and b_j, $j \in N$, as described in Section 2 and determine permutation σ of the form (15).

(c) Solve Problem $LP(y)$.

Step 1 of Algorithm Bicrit takes $O(nL\log n)$ time, as stated in Lemma 1. For each q, $1 \leq q \leq \ell$, Step 2b of Algorithm Bicrit can be done in $O(n\log n)$ time, while Step 2c takes $O(n^2)$ time. As a result, in at most $O(nL(n\log n+n^2)) = O(n^3L)$ time all break-points of the efficiency frontier will be found and the following statement holds.

Theorem 4 *Problem* $1|p_j = \bar{p}_j - h_j|(F,K)$ *is solvable in* $O(n^3L)$ *time.*

Our algorithm compares favourably with an $O(n^4L^2)$-time algorithm for problem $1|p_j = \bar{p}_j - h_j|(F,K)$ presented in [8]. The improvement is achieved due to formulating a parametric linear program $LP(y)$ and applying the submodular optimization approach to finding its solution. For each stripe, this allows us to find the break-points of the efficiency frontier all at once, and not one by one as done previously. In fact, we deduce that for each stripe the number of break-points of the efficiency frontier is $O(n^2)$, against $O(n^3)$ as stated in [8]. Besides, our algorithm is accompanied with an accurate estimation of the number of stripes and an efficient algorithm for their finding.

In the subsequent sections we consider the single criterion problems associated with problem $1|p_j = \bar{p}_j - h_j|(F,K)$. In each of these problems, the value of one of the objective functions is bounded from above, while the other function is to be minimized.

4. Single Criterion Problems

In this section, we consider two conjugate single criterion problems related to the bicriteria problem $1|p_j = \bar{p}_j - h_j|(F,K)$. In each of this problems, the goal is to minimize one of the objective functions, either F or K, provided that the value of the other function does not exceed a given bound. We demonstrate that an optimal solution to each of these single criterion problems can be obtained faster than derived from the solution of the bicriteria problem.

4.1. Limited Processing Cost

Consider problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \leq U|\sum \alpha_j h_j$, which consists in minimizing the total compression cost, provided that the maximum processing cost $F = \max\{f_j(C_j)\}$ does not exceed a given upper bound U.

For problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \leq U|\sum \alpha_j h_j$, in which each function $f_j(C_j)$ is non-decreasing and piecewise linear with m_j break-points, the upper bound U induces the deadline d_j for each job $j \in N$ by which that job must be completed. A deadline d_j is determined by finding an appropriate piece of function f_j (which can be done in $O(\log m_j)$ time by binary search over m_j break-points of function f_j) and then by solving the linear equation

$$f_j(d_j) = U$$

for that piece. It follows that the deadlines for all jobs can be found in at most $O(\sum_{j\in N} \log m_j)$ time.

Renumber the jobs to satisfy the popular earliest due date (EDD) rule, so that

$$d_1 \leq d_2 \leq \cdots \leq d_n. \tag{16}$$

The original problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \leq U|\sum \alpha_j h_j$ reduces to problem $1|p_j = \overline{p}_j - h_j, C_j \leq d_j|\sum \alpha_j h_j$ with controllable processing times in which it is required to find a deadline feasible schedule with the smallest total compression cost.

It is well known that if a feasible schedule that respects the deadlines of all jobs exists, then it can be found by sequencing the jobs in accordance with the EDD numbering. Thus, in any feasible schedule the actual processing times $p_j = \bar{p}_j - h_j$, must satisfy the constraints

$$\sum_{i=1}^{j} p_i \leq d_j, \quad 1 \leq j \leq n, \tag{17}$$

$$\underline{p}_j \leq p_j \leq \overline{p}_j, \quad 1 \leq j \leq n. \tag{18}$$

Here, the inequalities (17) are nested constraints, while (18) can be seen as box constraints. Since minimizing the total compression cost $\sum \alpha_j h_j$ is equivalent to maximizing $\sum \alpha_j p_j$, it follows that problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \leq U|\sum \alpha_j h_j$ can be formulated as an LP problem of maximizing the function $\sum \alpha_j p_j$ subject to the constraints (17) and (18). We refer to this problem as Problem $LP1$.

This problem is a special case of the parametric Problem $LP(y)$, in which y is fixed to be U, $a_j = 0$ and $b_j = d_j$, $j \in N$. Therefore, the reasoning of Section 3 fully applies here and delivers an $O(n^2)$-time algorithm for Problem $LP1$. Thus, for problem $1|p_j = \overline{p}_j - h_j, C_j \leq d_j|\sum \alpha_j h_j$ the actual processing times can be found in closed form in at most $O(n^2)$ time. Notice that most of previously known algorithms for problem $1|p_j = \overline{p}_j - h_j, C_j \leq d_j|\sum \alpha_j h_j$ and its symmetric counterpart $1|r_j, p_j = \overline{p}_j - h_j, C_j \leq d|\sum \alpha_j h_j$ also require $O(n^2)$ time, but these algorithms essentially use scheduling reasoning and need a lengthy justification, see, e.g., [9, 12]. The algorithm from [8] reduces the problem to finding the maximum flow, but still does not deliver a solution in closed form. The algorithm from [15] is based on links to polymatroids and uses balanced 2-3-trees to guarantee the running time of $O(n \log n)$, but remains a pure scheduling algorithm.

Below we show that problem $1|p_j = \overline{p}_j - h_j, C_j \leq d_j|\sum \alpha_j h_j$ (or equivalently, Problem $LP1$) can be solved in $O(n \log n)$ time by the methods employed in [6, 7]. Among the variety of problems studied in these papers, for our purposes we only mention optimization problems over the region

$$\begin{aligned} &\sum_{i=1}^{j} z_i \leq v_j, \quad 1 \leq j \leq n, \\ &0 \leq z_j \leq \overline{z}_j, \quad 1 \leq j \leq n. \end{aligned} \tag{19}$$

It is fairly straightforward to see that Problem $LP1$ can be reformulated to fit this framework. Introduce the variables $z_j = p_j - \underline{p}_j$, $j \in N$. For each job j, the value of z_j represents the *decompression* amount by which the actual processing time of that job is extended compared to the minimum possible (mandatory) value $\underline{p}_j$.

With these new variables the feasible region for Problem $LP1$ can be written as

$$\sum_{i=1}^{j} z_i \le d_j - \sum_{i=1}^{j} \underline{p}_j, \quad 1 \le j \le n,$$
$$0 \le z_j \le \overline{p}_j - \underline{p}_j, \qquad 1 \le j \le n,$$

which is of the form (19). Problem $LP1$ becomes equivalent (up to an additive constant) to the problem of maximizing the linear function $\sum \alpha_j z_j$ over (19). As shown in [6], a linear function can be maximized over (19) by a greedy algorithm that considers the decision variables in nonincreasing order of α_j and gives each variable the largest possible value. To verify feasibility of a partial solution, [6] takes advantage of the nested structure of the region (19) and demonstrates that using an appropriate data structure and adopting the UNION-FIND algorithm all feasibility checks can be implemented in $O(n)$ time. Thus, it takes $O(n \log n)$ time to solve problem $1|p_j = \overline{p}_j - h_j, C_j \le d_j| \sum \alpha_j h_j$.

Since the original problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \le U| \sum \alpha_j h_j$ reduces to the corresponding problem $1|p_j = \overline{p}_j - h_j, C_j \le d_j| \sum \alpha_j h_j$ with controllable processing times in $O(\sum_{j \in N} \log m_j)$ time, we arrive at the following statement.

Theorem 5 *Problem* $1|p_j = \bar{p}_j - h_j,\ f_j(C_j) \le U| \sum \alpha_j h_j$ *to minimize the total compression cost subject to a bounded processing cost for each job is solvable in* $O(n \log n + \sum_{j \in N} \log m_j)$ *time.*

Notice that if each m_j is bounded by a constant, then the running time for solving problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \le U| \sum \alpha_j h_j$ reduces to $O(n \log n)$.

Observe that the technique developed in [6, 7] is very powerful and allows handling more general objective functions, e.g., maximization of separable concave functions or minimization of separable convex functions over (19). This leads to polynomial-time algorithms for scheduling problems with more general compression cost functions.

4.2. Limited Compression Cost

Consider problem $1|p_j = \bar{p}_j - h_j, \sum \alpha_j h_j \le V|F$, which consists in minimizing the maximum processing cost $F = \max\{f_j(C_j)\}$, provided that the total compression cost does not exceed a given upper bound V.

It is natural to interpret this problem graphically, similarly to Section 2; see Figure 1. Our algorithm consists of several stages and builds on the previous material of this paper. To make the algorithm more computationally efficient, the median finding procedure is employed, see [3, pp. 189–192] for details.

Assume that F^* is an optimal value of the objective function in problem $1|p_j = \bar{p}_j - h_j, \sum \alpha_j h_j \le V|F$. Let $\tilde{y}_0 < \tilde{y}_1 < \cdots < \tilde{y}_m$ be the sequence of the ordinates of the points in $S_1 \cup S_3$ found in Step 1 of Algorithm Preproc, see Section 2. Note that $m = O(L)$. In the first stage, we find an interval $[\tilde{y}_{r-1}, \tilde{y}_r]$ such that $F^* \in [\tilde{y}_{r-1}, \tilde{y}_r]$. This is done by binary search over the values $\{\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_m\}$. Start with $r = \lfloor m/2 \rfloor$, take $U = \tilde{y}_r$ by using a median finding algorithm, and solve a single criterion problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \le U| \sum \alpha_j h_j$ using the method described in Subsection 4.1. If the found value of the function $\sum \alpha_j h_j$ is greater than the upper bound V, we take a larger trial value $U = \tilde{y}_{r+\lfloor r/2 \rfloor}$. Otherwise, we

take a smaller trial value $U = \tilde{y}_{r-\lfloor r/2 \rfloor}$. The process is repeated until the required interval is found.

In the second stage, we find a stripe $[\underline{y}, \overline{y}]$ such that $F^* \in [\underline{y}, \overline{y}] \subseteq [\tilde{y}_{r-1}, \tilde{y}_r]$. For this, we first compute all the ordinates of the points in S_2 that belong to the stripe $[\tilde{y}_{r-1}, \tilde{y}_r]$, where S_2 is defined in Section 2. Since each function f_j is linear in the interval $[\tilde{y}_{r-1}, \tilde{y}_r]$, the required points in S_2 within the stripe $[\tilde{y}_{r-1}, \tilde{y}_r]$ are given as the intersection points of n linear pieces, and the number of such points is $O(n^2)$. To identify the required stripe $[\underline{y}, \overline{y}]$ we combine binary search with a median finding algorithm in a similar way as in the first stage.

The third and the final stage of the algorithm searches for the optimal value F^* in the determined stripe $[\underline{y}, \overline{y}]$. For each job j, let the piece that represents the processing cost function $f_j(t)$ in this stripe be $Q_j t + R_j$, and assume that the jobs are properly numbered, as introduced in Section 2, so that job j completes at time $\sum_{i=1}^{j} p_j$. Formulate Problem $LP(y)$ and solve it as described in Section 3. As a result, we will find an optimal solution vector $\boldsymbol{p}^*(y) = (p_1^*(y), p_2^*(y), \dots, p_n^*(y))$ and the optimal objective function $K(y) = \sum \alpha_j p_j^*(y)$. Function $K(y)$ is the sum of n piecewise linear functions, each consisting of $O(n)$ pieces. Thus, function $K(y)$ is piecewise linear in y and consists of $O(n^2)$ pieces. The optimal value of F^* is equal to such a $y^* \in [\underline{y}, \overline{y}]$ that $K(y^*) = V$.

Let us estimate the running time of the described procedure.

In the first stage, we need to use a median finding algorithm $O(\log m)$ times, which requires $O(m+m/2+m/4+\cdots) = O(m) = O(L)$ time. As follows from Theorem 5, for each trial value U, solving problem $1|p_j = \bar{p}_j - h_j, f_j(C_j) \le U| \sum \alpha_j h_j$ requires $O(\sum_{j \in N} \log m_j + n \log n)$ time, and we have $O(\log m) = O(\log L)$ trial problems to solve. Hence, the overall time complexity of that stage is

$$O\Big(L + \Big(\sum_{j \in N} \log m_j + n \log n\Big) \log L\Big).$$

In the second stage, we can compute the points in S_2 within the stripe $[\tilde{y}_{r-1}, \tilde{y}_r]$ in $O(n^2)$ time. The rest of the running time in the second stage can be analyzed in a similar way to that in the first stage. Since the number of points in S_2 within the stripe $[\tilde{y}_{r-1}, \tilde{y}_r]$ is $O(n^2)$, we need to solve $O(\log n)$ trial problems, and each of these problems can be solved in $O(\sum_{j \in N} \log m_j + n \log n)$ time. Hence, the time complexity of this stage is

$$O\Big(n^2 + \Big(\sum_{j \in N} \log m_j + n \log n\Big) \log n\Big).$$

Finally, in the third stage, Problem $LP(y)$ is solvable in $O(n^2)$ time, and the optimal value $F^* = y^*$ can be determined in $O(n^2)$ time by examining all pieces of function $K(y^*)$.

Summarizing, the overall time complexity of the three-stage procedure for solving problem $1|p_j = \bar{p}_j - h_j, \sum \alpha_j h_j \le V|F$ is

$$O\Big(L + n^2 + \Big(\sum_{j \in N} \log m_j + n \log n\Big) \log L\Big).$$

Theorem 6 *Problem* $1|p_j = \bar{p}_j - h_j, \sum \alpha_j h_j \leq V|F$ *to minimize the maximum processing cost subject to a bounded decompression cost is solvable in* $O(L + n^2 + (\sum_{j \in N} \log m_j + n \log n) \log L)$ *time.*

Notice that if each m_j is bounded by a constant, then $L = O(n)$ and $\sum_{j \in N} \log m_j = O(n)$, so that the running time for solving problem $1|p_j = \bar{p}_j - h_j, \sum \alpha_j h_j \leq V|F$ reduces to $O(n^2 + n \log^2 n)$.

5. Conclusion

This paper presents solution algorithms for the single machine scheduling problems with controllable processing times, which have applications to imprecise computation and to make-or-buy production planning. Our algorithms are less time consuming than those previously known.

It should be noted that the collection of solution methods developed in the area of scheduling with controllable processing times does not exhibit a general underlying idea that is common for several problems. For example, even though many algorithms in this area exploit some kind of greedy argument, usually in the literature the design and justification of an individual algorithm is very much problem dependent. On the other hand, as pointed out in [15, 16], many problems with controllable processing times can be reformulated in terms of optimization problems over regions described by submodular constraints, including polymatroids and generalized polymatroids. The main advantage of such a reformulation is that the greedy optimization algorithms over mentioned regions are immediately justified. The algorithm developed that way can be used as a subroutine in designing algorithms for other problems in this area; see, e.g., the paper by Wan et al. [22] which relies on one of the algorithms presented in [16].

This paper is another step towards a better understanding of the link between scheduling with controllable parameters and problems of submodular optimization. From that point of view, Theorem 3 of this paper makes an interesting contribution by providing the rank function for a base polyhedron related to a submodular polyhedron intersected with a box. Notice that box constraints with non-zero lower bounds are insufficiently studied in submodular optimization. Since box constraints occur in many applications, we expect that Theorem 3 will be found a useful tool. Algorithm Bicrit that is based on Theorem 3 demonstrates the power of our approach.

Acknowledgements

The second author is partially supported by the Humboldt Research Fellowship of the Alexander von Humboldt Foundation and by Grant-in-Aid of the Ministry of Education, Culture, Sports, Science and Technology of Japan. The authors are grateful to Professor Takeshi Tokuyama for his suggestions regarding the proof of Theorem 1.

Bibliography

1. I. J. Balaban, "An optimal algorithm for finding segments intersections," in *Proceedings of the 11th ACM Symposium on Computational Geometry* (1995), pp. 211–219.
2. J.L. Bently and T.A. Ottmann, "Algorithms for reporting and counting geometric

intersections," *IEEE Trans. Comput.* **C-28** (1979) 643–647.

3. T.H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 2nd edition (MIT Press, 2001).
4. A. Frank and E. Tardos, "Generalized polymatroids and submodular flows," *Math. Program.* **42** (1988) 489–563.
5. S. Fujishige, *Submodular Functions and Optimization*, *Annals of Discrete Mathematics* **58** (Elsevier, Amsterdam, 2005).
6. D.S. Hochbaum, "Lower and upper bounds for the allocation problem and other nonlinear optimization problems," *Math. Oper. Res.* **9** (1995) 390–408.
7. D.S. Hochbaum and S.-P. Hong, "About strongly polynomial time algorithms for quadratic optimization over submodular constraints," *Math. Program.* **69** (1995) 269–309.
8. H. Hoogeveen and G.J. Woeginger, "Some comments on sequencing with controllable processing times," *Computing* **68** (2001) 181–192.
9. A. Janiak and M.Y. Kovalyov, "Single machine scheduling subject to deadlines and resource dependent processing times," *Europ. J. Oper. Res.* **94** (1996) 284–291.
10. N. Katoh and T. Ibaraki, "Resource allocation problems," in *Handbook of Combinatorial Optimization*, Vol. 2, eds. D.-Z. Du and P.M. Pardalos (Kluwer, Dordrecht, 1998) pp. 159–260.
11. J.Y.-T. Leung, "Minimizing total weighted error for imprecise computation tasks," in *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, ed. J.Y.-T. Leung (Chapman & Hall/CRC, 2004) pp. 34-1 – 34-16.
12. E. Nowicki and S. Zdrzalka, "A survey of results for sequencing problems with controllable processing times," *Discrete Appl. Math.* **26** (1990) 271–287.
13. A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency* (Springer, Berlin, 2003).
14. D. Shabtay and G. Steiner, "A survey of scheduling with controllable processing times," *Discrete Appl. Math.* **155** (2007) 1643–1666.
15. N.V. Shakhlevich and V.A. Strusevich, "Preemptive scheduling problems with controllable processing times," *J. Scheduling* **8** (2005) 233–253.
16. N.V. Shakhlevich and V.A. Strusevich, "Preemptive scheduling on uniform parallel machines with controllable job processing times," *Algorithmica* **51** (2008) 451–473.
17. V. T'kindt and J.-C. Billaut, *Multicriteria Scheduling: Theory, Models and Algorithms* (Springer, Berlin, 2006).
18. A.V. Tuzikov, "A bicriterion problem of the scheduling theory taking the variation of servicing time into account," *USSR Comput. Math. Math. Physics* **24** (1984) 191–194.
19. S. van Hoesel, A. Wagelmans, and B. Moerman, "Using geometric techniques to improve dynamic programming algorithms for the economic lot-sizing problem and extensions," *Europ. J. Oper. Res.* **75** (1994) 312–331.
20. L.N. Van Wassenhove and K.R. Baker, "A bicriterion approach to time/cost tradeoffs in sequencing," *Europ. J. Oper. Res.* **11** (1982) 48–54.
21. R.G. Vickson, "Two single machine sequencing problems involving controllable job processing times," *AII Trans.* **12** (1980) 258–262.
22. G. Wan, J. Y.-T. Leung, and M.L. Pinedo, "Scheduling imprecise computation tasks on uniform processors," *Inform. Proc. Lett.* **104** (2007) 45–52.

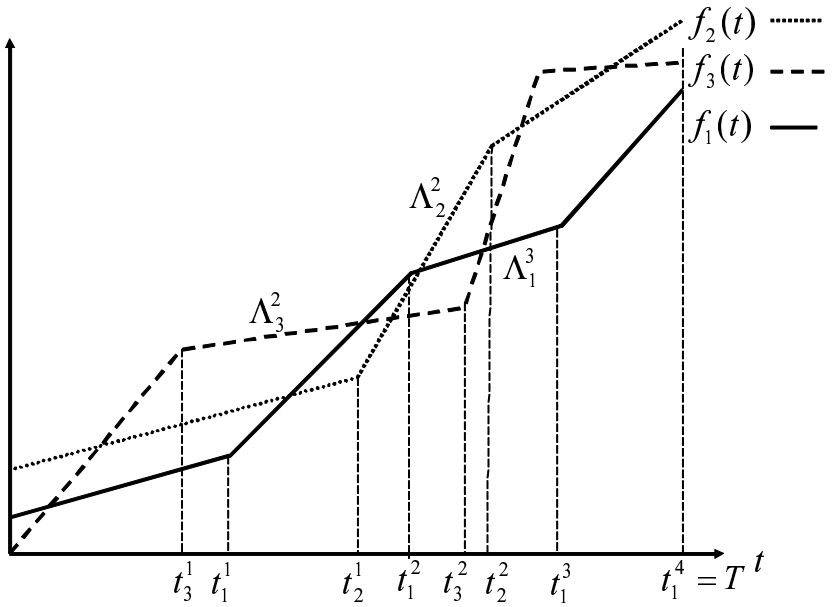

Fig. A.1. Example with three functions

Appendix A: Proof of Theorem 1

For each function $f_j(t)$, $1 \le j \le n$, we denote by $t_l^0 = 0, t_j^1, t_j^2, \ldots, t_j^{m_j} = T$ the increasing sequence of t-coordinates of the break-points of f_j. That is, each f_j is linear in the interval $[t_j^{k-1}, t_j^k]$ for $k = 1, 2, \ldots, m_j$. Let Λ_j^k denote the k-th linear piece of the function f_j which corresponds to the interval $[t_j^{k-1}, t_j^k]$.

Let us fix a linear piece Λ_j^k and determine which linear pieces $\Lambda_{j'}^{k'}$ of other functions $f_{j'}$ with $j' \neq j$ may intersect Λ_j^k in the interval $[t_j^{k-1}, t_j^k]$. We may restrict our attention to only those linear pieces $\Lambda_{j'}^{k'}$ for which $t_{j'}^{k'-1} \le t_j^{k-1}$. Under this assumption, the pieces Λ_j^k and $\Lambda_{j'}^{k'}$ may intersect if $t_{j'}^{k'} \ge t_j^{k-1}$. We call a pair $(\Lambda_j^k, \Lambda_{j'}^{k'})$ of linear pieces a *suitable* pair if the condition $t_{j'}^{k'-1} \le t_j^{k-1} < t_{j'}^{k'}$ is satisfied; the piece Λ_j^k is called the *primary* piece of the pair, while $\Lambda_{j'}^{k'}$ is called the *secondary* piece of the pair. We count suitable pairs $(\Lambda_j^k, \Lambda_{j'}^{k'})$ by fixing each linear piece as the primary piece Λ_j^k defined over $[t_j^{k-1}, t_j^k]$ and considering all secondary pieces $\Lambda_{j'}^{k'}$ defined over segments $[t_{j'}^{k'-1}, t_{j'}^{k'}]$ such that either $t_{j'}^{k'-1} = t_j^{k-1}$ or the left-end point $t_{j'}^{k'-1}$ is located to the left of t_j^{k-1}. Notice that a suitable pair does not necessarily correspond to an actual intersection point, but each intersection point can be associated with a suitable pair. The same primary piece Λ_j^k may intersect with a piece $\Lambda_{j'}^{k'}$, but the pair $(\Lambda_k^j, \Lambda_{k'}^{j'})$ is not suitable; in this case, however, the pair $(\Lambda_{k'}^{j'}, \Lambda_k^j,)$ will be counted as suitable. In what follows, we count the number Φ of all suitable pairs. It is clear that Φ is an upper bound on the total number of intersection points of all functions.

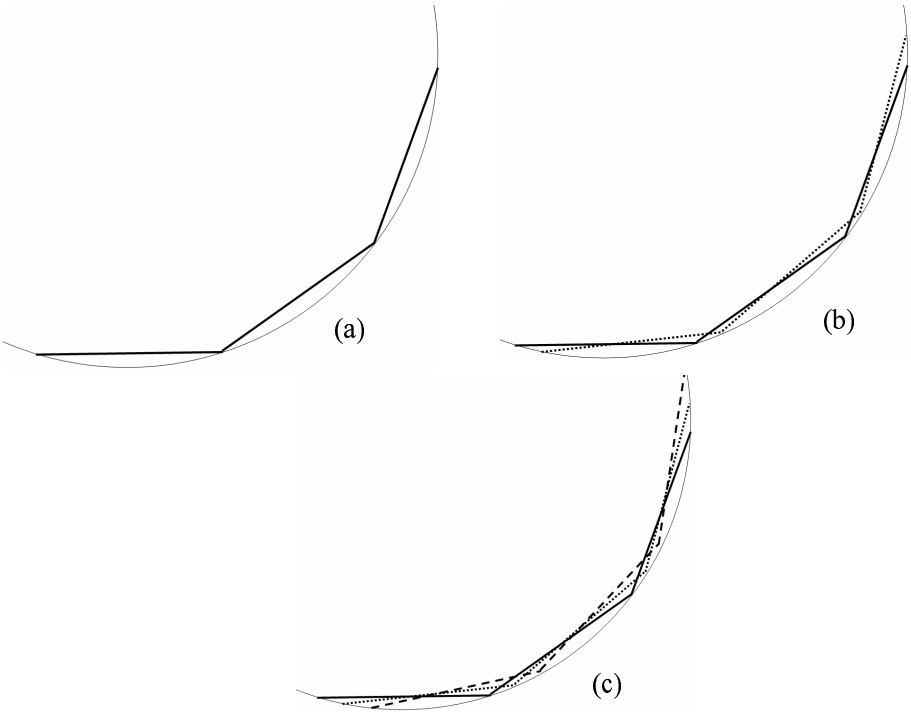

Fig. A.2. Generation of the tightness example: (a) one function, (b) two functions, (c) three functions

For a linear piece Λ_j^k, where $1 \le j \le n$ and $1 \le k \le m_j$, we denote

$$\mathcal{P}_j^k = \{(\Lambda_j^k, \Lambda_{j'}^{k'}) \mid j \ne j',\ t_{j'}^{k'-1} \le t_j^{k-1} < t_{j'}^{k'},\ \text{and } j' < j \text{ if } t_{j'}^{k'-1} = t_j^{k-1}\}.$$

If $t_{j'}^{k'-1} < t_j^{k-1} < t_{j'}^{k'}$ holds, then we have $(\Lambda_j^k, \Lambda_{j'}^{k'}) \in \mathcal{P}_j^k$; if $\eta_{k'-1}^{j'} = \eta_{k-1}^{j} < \eta_{k'}^{j'}$ holds, then we have either $(\Lambda_j^k, \Lambda_{j'}^{k'}) \in \mathcal{P}_k^j$ (provided that $j' < j$) or $(\Lambda_{j'}^{k'}, \Lambda_j^k,) \in \mathcal{P}_{j'}^{k'}$ (provided that $j' > j$). This implies that any pair $(\Lambda_k^j, \Lambda_{k'}^{j'})$ with $j \ne j'$ satisfying the condition $t_{j'}^{k'-1} \le t_j^{k-1} < t_{j'}^{k'}$ is contained in exactly one of the sets $\mathcal{P}_j^k$ or $\mathcal{P}_{j'}^{k'}$ and therefore we have

$$\Phi = \sum_{j=1}^{n} \sum_{k=1}^{m_j} |\mathcal{P}_j^k|.$$

Figure A.1 shows three piecewise linear functions f_1, f_2 and f_3, consisting of $4, 3$ and 4 linear pieces, respectively. For piece Λ_1^3 taken as the primary one, the set $\mathcal{P}_1^3$ consists of two suitable pairs $(\Lambda_1^3, \Lambda_2^2)$ and $(\Lambda_1^3, \Lambda_3^2)$, and only the former pair corresponds to an actual intersection point.

Notice that $|\mathcal{P}_j^k| \le n - 1$ since for each $j' \ne j$ there may exist at most one interval $[t_{j'}^{k'-1}, t_{j'}^{k'}]$ satisfying $t_{j'}^{k'-1} \le t_j^{k-1} < t_{j'}^{k'}$. Moreover, for $k = 1$ we have

$$\sum_{j=1}^{n} |\mathcal{P}_j^1| \le n(n-1)/2,$$

since the sets $\mathcal{P}_j^1$, $j = 1, \ldots, n$, are mutually disjoint subsets of pairs $(\Lambda_j^1, \Lambda_{j'}^1)$ with $j \neq j'$. Hence, we derive

$$\Phi \leq \sum_{j=1}^{n} |\mathcal{P}_j^1| + \sum_{j=1}^{n} \sum_{k=2}^{m_j} |\mathcal{P}_j^k| = \frac{n(n-1)}{2} + (n-1) \sum_{j=1}^{n} (m_j - 1) = (n-1)L - \frac{n(n-1)}{2},$$

which proves the required bound.

This bound is tight. It is possible to select n functions, each consisting of m linear pieces, so that for each function its first piece is intersected with exactly one piece of each of the other functions, while each piece, starting form the second, is intersected with exactly two pieces of each of the other functions. In this case, $\Phi = n(n-1)/2 + n(n-1)(m-1) = nm(n-1) - n(n-1)/2 = (n-1)L - n(n-1)/2$. This can be achieved by placing the break-points of the first function along a circle, followed by adding each of the next functions by an appropriate rotation of the original function. The idea of the generation of the functions for the tightness example is illustrated in Figure A.2.