Department of Industrial Engineering and Economics

Working Paper

No. 2017-3

A Review of Recent Approaches to Designing Fast Algorithms for Scheduling with Controllable Processing Times and Imprecise Computation

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

April, 2017

Tokyo Institute of Technology

2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, JAPAN
http://educ.titech.ac.jp/iee/

A Review of Recent Approaches to Designing Fast Algorithms for Scheduling with Controllable Processing Times and Imprecise Computation

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

Natalia V. Shakhlevich
School of Computing, University of Leeds, Leeds LS2 9JT, U.K.

Vitaly A. Strusevich
Department of Mathematical Sciences,
University of Greenwich, Old Royal Naval College,
Park Row, London SE10 9LS, U.K.

Abstract

This paper provides a review of recent results on scheduling with controllable processing times. The stress is on the methodological aspects that include parametric flow techniques and methods for solving mathematical programming problems with submodular constraints. We show that the use of these methodologies results into fast algorithms for solving problems on single machine or parallel machines, with either one or several objective functions. For a wide range of problems with controllable processing times we report algorithms with the running times which match those known for the corresponding problems with fixed processing times. As a by-product, we present the best possible algorithms for a number of problems on parallel machines that are traditionally studied within the body of research on scheduling with imprecise computation.

Keywords: scheduling with controllable processing times; scheduling with imprecise computation; flows in networks; optimization with submodular constraints

1 Introduction

Scheduling with Controllable Processing Times (*SCPT*) is an active area within scheduling research. It reflects the modern trend that, unlike the classical scheduling models, the processing times of the jobs are not given constants. One type of models that treat scheduling problems with changing times are those that allow dynamic changes of the processing times depending on the state of the processing machines, including various deterioration and/or learning effects, as well as machine maintenance. Another type of models, which is the topic of this review, gives the decision-maker the power of selecting the processing times from given intervals.

Finding a solution to an SCPT problem involves two decisions: (i) selecting actual processing times for all jobs, and (ii) allocating and sequencing the jobs on the machines in order to achieve a required level of quality. The first decision incurs a penalty that depends

on compression amounts of the jobs, i.e., on the reduction of a job's processing time from its given value. The second decision affects the system performance measured in terms of a scheduling objective depending on job completion times, e.g., the makespan.

The SCPT research has been active for more than 35 years. What is unusual is that during all these years there has been a parallel stream of research, termed *Scheduling with Imprecise Computation* (*SIC*). In the range of models studied within the SIC research the processing machines are seen as processors, the jobs are computational tasks, and these tasks are allowed to be processed partially, thereby generating errors of computation. No close examination is needed to observe that the SIC models are versions or, more precisely, particular meaningful interpretations of the SCPT models. Both the SCPT and the SIC studies address essentially the same range of problems, and often apply the same methods.

The word "parallel"used in the previous paragraph very well describes a surprising fact that until very recently the SCPT research and SIC research existed almost independently of each other, with almost no interaction or cross-referencing. This mutual neglect has taken place despite the fact that researchers of both groups are parts of the same wider scheduling community, with common main journals and conferences. For example, a rather comprehensive survey on the SCPT models by Shabtay and Steiner (2007) does not mention the results obtained by researchers who study the SIC models. Similarly, only a single paragraph in the survey on the SIC models by Leung (2004) admits a link between that area and SCPT, without exploiting that link.

The SCPT-SIC link is also overlooked within the stream of research on late work minimization, see the recent survey by Sterna (2011). As we show in Section 2, the SCPT and SIC models are exactly the same, while the preemptive late work model can be treated as a special case of either of these models.

Throughout this paper, we normally adopt the term SCPT as the main one; however, the imprecise computation scenarios will also be explicitly referred to, especially when we review and revise the results obtained within the SIC body of research.

Bringing together and establishing the true relations between the SCPT and the SIC models is an important, but secondary goal of this survey. Our main task is to review major changes that have taken place during the last decade, since the most recent reviews by Shabtay and Steiner (2007) and Leung (2004) were published.

From the historical perspective, it can be observed that most of earlier publications employed a range of rather straightforward approaches. These include simple reformulations of the SCPT problems in terms of finding either the maximum flow or the minimum cost flow in a special network. Many other papers supplied greedy-like procedures, normally accompanied by lengthy justification proofs, full of cumbersome details. Applicability of these methods was exhausted by the early 2000s, and new theoretical results on the SCPT models became rather rare.

Major components that have extended the toolkit of the SCPT techniques include the methods for solving parametric flow problems and methods for solving *submodular optimization* problems, i.e., mathematical programming problems with submodular constraints. Both of these components have been known for about 30 years, but the attempts to apply them in the SCPT context have been rather limited. Among noticeable examples is the paper by McCormick (1999) who developed a fast method for finding the maximum flow in networks with parametric capacities of some arcs and applied this method for solving quite general SCPT problems. With respect to the submodular optimization methodology, Nemhauser and Wolsey were among the first who noticed that the SCPT models could be handled by

methods of submodular optimization; see, e.g., Example 6.1 (Section 6 of Chapter III.3) of their book (Nemhauser and Wolsey (1988)).

Thus, the methods that we discuss in this paper, strictly speaking, are not new. We want to demonstrate that their systematic use, correct adaptation and appropriate further development lead to a range of efficient solution algorithms. This results into a general framework for handling the SCPT problems, which (i) replaces a collection of scattered purpose-built algorithms by providing faster and easier justifiable techniques; (ii) is able to solve problems which have not been addressed earlier; (iii) often supplies algorithms with the running times that cannot be improved, at least at the current stage of knowledge.

The paper is organized as follows. The SCPT model is formally introduced in Section 2, where we also review its applications to various problem areas, including the SIC and late work models. The main focus is on the problems of finding deadline-feasible preemptive schedules on either a single machine or on parallel machines.

Section 3 introduces the processing capacity function, a crucial concept for solving the SCPT models, as well as their counterparts with fixed processing times.

Section 4 presents various network flow techniques, which constitute the first of methodologies discussed in this paper. Among reviewed techniques are those for finding parametric maximum flow by Gallo et al. (1989) and their multiparametric extension by McCormick (1999). Since most of the SCPT applications of the network flow techniques deal with unbalanced bipartite network, we also review the speeding-up techniques by Ahuja et al. (1994).

Section 5 illustrates the use of Methodology 1 for solving feasibility scheduling problems with fixed processing times. In particular, we revise a perception widely accepted in the SIC community regarding the running time needed for finding a deadline-feasible schedule on identical parallel machines. Section 6 elaborates on Methodology 1 by applying it to the SCPT problems of minimizing total compression cost on parallel machines, where multiparametric network flow techniques by McCormick (1999) give the best results.

Section 7 overviews what we call Methodology 2: solving linear programming problems over a submodular polyhedron intersected with a box. Such a problem, that we call Problem (LP), is the main model for various SCPT problems that involve minimizing total compression cost. In particular, in Section 8 Problem (LP) and Methodology 2 are used to solve bicriteria problems on parallel machines to simultaneously minimize (in the Pareto sense) the maximum completion time and the total compression cost.

Methodology 3 presented in Section 9 can be seen as further development of Methodology 2. There we present a decomposition algorithm for solving Problem (LP) designed by Shioura et al. (2015, 2016a). In Section 10 we describe how to adapt Methodology 3 to solving a range of SCPT problems to minimize the total compression cost.

Sections 11 and 12 address the problems that involve minimizing the maximum compression cost. Such problems are traditionally considered within the SIC body of research. We develop new results that are based on application of Methodology 1, in particular on solving problems of lexicographic flow sharing which is done by adapting parametric flow techniques of Gallo et al. (1989). Resulting algorithms solve the problems on parallel machines to minimize the maximum cost as well as to minimize both the maximum cost and total cost (in the lexicographic sense). The running times of these algorithms are the best possible.

New results also appear in Section 13, where we study a quadratic cost function, either alone or in combination with another cost function, total or maximum. The algorithms of this section are natural adaptations of those from Sections 11 and 12 due to a link known in submodular optimization between the problems of minimizing a quadratic function and

finding a parametric flow.

Conclusions are summarized in Section 14.

2 Models and Applications

In this paper, we mainly address scheduling problems of the following type. The jobs of set $N = \{1, 2, \ldots, n\}$ have to be processed either on a single machine M_1 or on parallel machines $M_1, M_2, \ldots, M_m$, where $m \geq 2$.

In the classical scheduling settings, each job $j \in N$ is given its processing time $p(j)$. In the SCPT setting, the actual processing time $p(j)$ of job $j \in N$ is not given in advance but has to be chosen by a decision-maker from a given interval $[\underline{p}(j), \overline{p}(j)]$. Such a decision results in *compression* of the longest processing time $\overline{p}(j)$ down to $p(j)$, and the value $x(j) = \overline{p}(j) - p(j)$ is called the *compression amount* of job j. Compression may decrease the completion time of each job j but incurs additional cost.

Given m parallel machines, we distinguish between the *identical* machines and the *uniform* machines. In the former case, the machines have the same speed, so that for a job j with an actual processing time $p(j)$ the total length of the time intervals in which this job is processed in a feasible schedule is equal to $p(j)$. If the machines are uniform, then it is assumed that machine M_i has speed s_i, $1 \leq i \leq m$. If for job j the value $\overline{p}(j)$ is compressed to $p(j)$ and this job is assigned to machine M_i alone then the duration of such processing is $p(j)/s_i$. Throughout this paper, the uniform machines are numbered in non-increasing order of their speeds, i.e.,

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

Each job $j \in N$ is given a *release date* $r(j)$, before which it is not available, and a *deadline* $d(j)$, by which its processing must be completed. In the processing of any job, *preemption* is allowed, so that the processing can be interrupted on any machine at any time and resumed later, possibly on another machine (in the case of parallel machines). It is not allowed to process a job on more than one machine at a time, and a machine processes at most one job at a time.

Let $C(j)$ denote the completion time of job $j \in N$, provided that its processing time is equal to $p(j)$. A schedule is called *feasible* if no job j is processed outside the time interval $[r(j), d(j)]$. To solve a problem with fixed processing times means either to find a feasible schedule for the corresponding machine environment if it exists or to report that such a schedule does not exist. Adapting standard notation for scheduling problems by Lawler et al. (1993), we denote a generic feasibility problem with fixed processing times by $\alpha|r(j), C(j) \leq d(j), pmtn|-$. Here, in the first field α we write "1"for a single machine, "P"in the case of $m \geq 2$ identical machines and "Q"in the case of $m \geq 2$ uniform machines. In the middle field, the item "$r(j)$"implies that the jobs have individual release dates; this parameter is omitted if the release dates are equal. The condition "$C(j) \leq d(j)$"reflects the fact that in a feasible schedule the deadlines should be respected; we write "$C(j) \leq d$", if all jobs have a common deadline d. The abbreviation "*pmtn*"is used to point out that preemption is allowed.

Solving a typical problem from the SCPT range requires two decisions: (1) finding the compression amounts $x(j)$ for all jobs and (2) determining a deadline-feasible preemptive schedule with actual processing times $p(j) = \overline{p}(j) - x(j)$. The objective is to minimize a certain penalty function Φ that depends on compression amounts $x(j)$. For the range of problems traditionally considered in the SCPT literature, the most studied objective function

represents the *total compression cost* and we denote it by $\Phi_\Sigma = \sum_{j\in N} w_T(j)x(j)$, where $w_T(j)$ is the unit cost, i.e., the cost of compressing job $j \in N$ by one unit of time, and given by a non-negative real number. Problems of minimizing the *maximum compression cost* are mainly studied within the SIC body of research; we denote such an objective function by $\Phi_{\max} = \max\{x(j)/w_M(j)|j \in N\}$, where for a given positive weight $w_M(j)$ the fraction $1/w_M(j)$ represents the unit cost. Our choice of writing out function $\Phi_{\max}$ in terms of converting costs into weights will become clear in Section 11.

To refer to an SCPT problem, we use the generic notation $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j)$, $pmtn|\Phi$. Here, we write "$p(j) = \overline{p}(j) - x(j)$"to indicate that the processing times are controllable and $x(j)$ is the compression amount to be found. Besides, in the third field we indicate that Φ is a penalty function to be minimized. While the notation above is used to denote SCPT problems with a single criterion, it can be adjusted to refer to the multicriteria problems. We also look at the constrained problems, in which one type of the penalties, e.g., the total cost, is minimized in the class of the schedules with the minimum value of the other penalty function (such as minmax cost), or vice versa. Problems of the latter type are traditionally studied in the SIC literature; see Ho (2004).

We illustrate several scenarios of interpretation of the SCPT model $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j)$, $pmtn|\Phi$ in various application areas.

Resource-Dependent Times. Janiak and Kovalyov (1996) argue that the processing times are *resource-dependent*, so that the more units of a single additional resource is given to a job, the more it can be compressed. In their model, a job $j \in N$ has a 'normal' processing time $b(j)$ (no resource used), and its actual processing time becomes $p(j) = b(j) - a(j)u(j)$, provided that $u(j)$ units of the resource are allocated to the job, where $a(j)$ is interpreted as a compression rate. The amount of the resource to be allocated to a job is limited by $0 \leq u(j) \leq \tau(j)$, where $\tau(j)$ is a known job-dependent upper bound. The cost of using one unit of the resource for compressing job j is denoted by $v(j)$, and it is required to minimize the total cost of resource consumption. This interpretation of the resource-dependent times is essentially equivalent to that adopted in this paper, which can be seen by setting

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

Chen-McCormick Model. A very similar model for scheduling with controllable processing times is due to Chen (1994), later studied by McCormick (1999). In particular, McCormick (1999) gives algorithms for finding a preemptive schedule for parallel machines that is feasible with respect to arbitrary release dates and deadlines. The actual processing time of a job is determined by $p(j) = \max\{b(j) - a(j)\lambda(j), 0\}$ and the objective is to minimize the function $\sum_{j\in N} \lambda(j)$. This is also similar to our interpretation due to

$$\overline{p}(j) = b(j), \quad \underline{p}(j) = 0, \quad x(j) = a(j)\lambda(j), \quad w(j) = 1/a(j), \quad j \in N. \tag{2}$$

Make-or-Buy Decision Making. Manufacturing companies often do not fulfill the whole order internally but delegate a part of it to subcontractors. Then $p(j) = \overline{p}(j) - x(j)$ is understood as the chosen actual time for internal manufacturing of order j, where $x(j)$ shows how much of the order is subcontracted and $w(j)x(j)$ is the cost of this subcontracting. For example, in problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j)$, $pmtn|\Phi_\Sigma$ the goal is to minimize the total subcontracting cost and to find a deadline-feasible schedule for internally manufactured orders.

Imprecise Computation. The SCPT problems can be interpreted in terms of SIC as follows. The jobs are seen as computational tasks to be processed with preemption in

a computing system that consists either of one processor or of several parallel processors (machines). In computing systems that support imprecise computation, some computations (image processing programs, implementations of heuristic algorithms) can be run partially, producing less precise results. In our notation, a task with processing requirement $\overline{p}(j)$ can be split into a mandatory part which takes $\underline{p}(j)$ time, and an optional part that may take up to $\overline{p}(j)-\underline{p}(j)$ additional time units. To produce a result of reasonable quality, the mandatory part must be completed in full, while an optional part improves the accuracy of the solution. If instead of an ideal computation time $\overline{p}(j)$ a task is executed for $p(j)=\overline{p}(j)-x(j)$ time units, then computation is imprecise and $x(j)$ corresponds to the error of computation. In this settings, the objectives Φ_{Σ} and $\Phi_{\max}$ are understood as the total weighted error and the maximum weighted error, respectively. A popular research direction in SIC is related to the lexicographical optimization of the two criteria; see Ho (2004). If the maximum weighted error $\Phi_{\max}$ should be minimized first and then further optimization is performed in the obtained class of solutions to minimize the total weighted error Φ_{Σ}, then the relevant problem is generically denoted by $\alpha|r(j),p(j)=\overline{p}(j)-x(j),C(j)\leq d(j),\ pmtn|Lex(\Phi_{\max},\Phi_{\Sigma})$. In the counterpart with $Lex(\Phi_{\Sigma},\Phi_{\max})$, the goal is to find a schedule that minimizes maximum weighted error among all schedules with the smallest total weighted error.

Late Work Model. This model was introduced as the information loss model by Blazewicz (1984) in the context of information processing. The term "late work"was coined later on to reflect broader application areas. In a typical information processing scenario, processing any job $j\in N$ implies producing some output which is meaningful if it is obtained before its due date $d(j)$, while the part produced after $d(j)$ has no usage and therefore is lost. An alternative term for late work is the number of tardy job units as in Hochbaum and Shamir (1990). If job j with the processing time $p(j)$ is processed before its due date for $z(j)$ time units, then the late work of job j in such a schedule can be defined as $Y(j)=p(j)-z(j)$. The objective is to minimize the total late work $\sum_{j\in N}Y(j)$, or in a more general case the total weighted late work $\sum_{j\in N}w(j)Y(j)$. Clearly, in the preemptive version of the problem, all late parts can be placed at the end of the schedule or even removed from it. Thus, the preemptive late work model becomes a special case of the more general SCPT model if we interpret the given processing times $p(j)$ as upper bounds $\overline{p}(j)$ and define $\underline{p}(j)=0$, $j\in N$, so that the late work $Y(j)$ becomes compression $x(j)$. The objective function $\sum_{j\in N}w(j)Y(j)$ turns to total compression cost Φ_{Σ}. A comprehensive review of the late work studies is given in Sterna (2011). Interestingly, the link between late work scheduling and SIC is well recognized, with mutual cross-references in publications, but the link to SCPT is generally missing.

3 Processing Capacity Functions

Each SCPT problem can be seen as an extension of the corresponding problem with fixed processing times $p(j)=\overline{p}(j)$, $1\leq j\leq n$, where no job compression is allowed. Problems with fixed processing times are of interest in their own right, and algorithms for their solution are used within the algorithms for the corresponding SCPT problems for finding an optimal schedule.

In this section, we introduce an important notion of the *processing capacity function* that is widely used not only for the problems with fixed processing times, but also for the SCPT problems.

A set function is a function whose argument is a set. For a subset $X\subseteq N=\{1,2,\ldots,n\}$,

let $\mathbb{R}^X$ denote the set of all vectors $\mathbf{p}$ with real components $p(j)$, where $j \in X$. For a vector $\mathbf{p} \in \mathbb{R}^N$, define $p(X) = \sum_{j \in X} p(j)$ for every set $X \in 2^N$.

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

$$p(X) \leq \varphi(X) \tag{3}$$

holds for all sets $X \subseteq N$. In fact, the opposite statement is also true.

We illustrate this notion for several problems and review algorithms for their solution.

Let us start with a single machine problem $1|r(j),\ C(j) \leq d\,(j)\,,\ pmtn|-$. Divide the interval $[\min_{j \in N} r(j), \max_{j \in N} d(j)]$ into subintervals by using the release dates $r(j)$ and the deadlines $d(j)$ for $j \in N$ as breakpoints. Let $\tau_0, \tau_1, \ldots, \tau_\gamma$, where $1 \leq \gamma \leq 2n-1$, be the increasing sequence of distinct numbers in the list $(r(j), d(j) \mid j \in N)$. Introduce the intervals $I_h = [\tau_{h-1}, \tau_h],\ 1 \leq h \leq \gamma$, and define the set of all intervals $\mathcal{I} = \{I_h \mid 1 \leq h \leq \gamma\}$. Denote the length of interval I_h by $\Delta_h = \tau_h - \tau_{h-1}$. Interval I_k is *available* for processing job j if $r(j) \leq \tau_{k-1}$ and $d(j) \geq \tau_k$. For a job j, denote the set of the available intervals by $\Gamma(j)$, i.e.,

$$\Gamma(j) = \{I_h \in \mathcal{I} \mid I_h \subseteq [r(j), d(j)]\}. \tag{4}$$

For a set of jobs $X \subseteq N$, introduce the set function

$$\varphi_1(X) = \sum_{I_k \in \cup_{j \in X} \Gamma(j)} \Delta_k, \tag{5}$$

where the right-hand side represents the lengths of all time intervals available for processing the jobs of set X. Thus, for problem $1|r(j), C(j) \leq d(j), pmtn|-$ a feasible schedule exists if and only if (3) holds for all sets $X \subseteq N$ and $\varphi(X) = \varphi_1\,(X)$. Such a statement (in different terms) was first formulated by Gordon and Tanaev (1973) and Horn (1974). Checking the conditions (3) for problem $1|r(j),\ C(j) \leq d\,(j)\,,\ pmtn|-$ can be done in $O(n \log n)$ time by an algorithm that is due to Horn (1974). That algorithm, often called Algorithm EDF (Early Deadline First), at any time when either a job arrives or a job completes, assigns for processing the unfinished job with the smallest deadline. The running time reduces to $O(n)$, provided that a sorted sequence of distinct release dates and deadlines is available.

In the rest of this section, we turn to problems with a common deadline, i.e., $d\,(j) = d$, $j \in N$. Assume that if the jobs have different release dates, they are renumbered to satisfy

$$r(1) \leq r(2) \leq \ldots \leq r(n). \tag{6}$$

Notice that problem $\alpha|r(j),\ C(j) \leq d,\ pmtn|-$ is closely related to problem $\alpha|r(j), pmtn|C_{\max}$ of minimizing the maximum completion time $C_{\max} = \max\{C\,(j)\,|j \in N\}$, also known as the *makespan*. Indeed, the optimal value of $C_{\max}$ for an instance of problem $\alpha|r(j), pmtn|C_{\max}$ delivers the smallest value of d such that a feasible schedule exists in the corresponding instance of problem $\alpha|r(j),\ C(j) \leq d,\ pmtn|-$.

In the case of a single machine, problem $1|r(j), C(j) \leq d, pmtn|-$ with a common deadline d is solvable by Algorithm EDF. Since the algorithm still requires that the jobs are sorted in accordance with (6), it follows that problem $1|r(j), C(j) \leq d, pmtn|-$ is solvable in $O\,(n \log n)$ time.

Even in the case of parallel machines, the processing capacity function can also be easily derived. We illustrate this for problem $Q|C(j) \leq d, pmtn|-$ with zero release dates. Recall that the uniform machines are numbered in accordance with (1). We denote

$$S_0 = 0, \qquad S_k = s_1 + s_2 + \cdots + s_k, \quad 1 \leq k \leq m. \tag{7}$$

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

It is well known that for problem $Q|C(j) \leq d, pmtn|-$ a feasible preemptive schedule exists if and only if the following conditions hold, which we quote in accordance with Brucker (2007): d is large enough to guarantee a processing capacity that is sufficient for

- any job to be completed by time d if it is processed on the fastest machine M_1,
- for any u, $2 \leq u \leq m-1$, any subset of u jobs to be completed by d on the u fastest machines $M_1, M_2, \cdots, M_u$;
- all jobs to be completed by d on all m machines.

Given a set $X \subseteq N$ of jobs, define

$$m_X = \min\{m, |X|\}, \tag{8}$$

which specifies the largest possible number of machines for processing the jobs from X. Then the processing capacity functions for problems $\alpha|C(j) \leq d, pmtn|-$ can be written as

$$\varphi(X) = dS_{m_X}, \qquad \text{for } \alpha = Q; \tag{9}$$

$$\varphi(X) = dm_X, \qquad \text{for } \alpha = P. \tag{10}$$

Using this fact, problem $Q|C(j) \leq d, pmtn|-$ can be solved in $O(n + m\log m)$ time, which reduces to $O(n)$ time for the problems with identical machines; see Gonzales and Sahni (1978) and McNaughton (1959), respectively.

For the models with distinct release dates, given a set $X \subseteq N$ of jobs, define $r_i(X)$ to be the i-th smallest release date in set X, $1 \leq i \leq |X|$. The processing capacity functions for problems $\alpha|r(j), C(j) \leq d, pmtn|-$ can be written as

$$\varphi(X) = dS_{m_X} - \sum_{i=1}^{m_X} s_i r_i(X), \qquad \text{for } \alpha = Q; \tag{11}$$

$$\varphi(X) = dm_X - \sum_{i=1}^{m_X} r_i(X), \qquad \text{for } \alpha = P. \tag{12}$$

Formula (11) is shown in Martel (1982) and Shakhlevich and Strusevich (2008) in a different (but equivalent) form. Problem $Q|r(j), C(j) \leq d, pmtn|-$ can be solved in $O(nm + n\log n)$ time, which reduces to $O(n\log n)$ time for the problem on identical machines; see Sahni and Cho (1980) and Sahni (1979), respectively.

The running times of the relevant algorithms are summarized in Table 1. Additionally, that table also presents the results on parallel machine feasibility problems with distinct deadlines. Handling the problems of the latter type requires the use of algorithms for finding flows in networks. We classify these techniques as Methodology 1 and review in the following section. Their application for solving problems $\alpha|r(j), C(j) \leq d(j), pmtn|-$ with $\alpha \in \{P, Q\}$ is described in Section 5.

Problems	Results	
$1\|r(j), C(j) \leq d\|-$	$O(n \log n)$	Horn (1974)
$1\|r(j), C(j) \leq d(j)\|-$	$O(n \log n)$	Horn (1974)
$P\|C(j) \leq d, pmtn\|-$	$O(n)$	McNaughton (1959)
$P\|r(j), C(j) \leq d, pmtn\|-$	$O(n \log n)$	Sahni (1979)
$P\|r(j), C(j) \leq d(j), pmtn\|-$	$O(n^3)^*$	Horn (1974)
$Q\|C(j) \leq d, pmtn\|-$	$O(n + m \log m)$	Gonzales and Sahni (1978)
$Q\|r(j), C(j) \leq d, pmtn\|-$	$O(nm + n \log n)$	Sahni and Cho (1980)
$Q\|r(j), C(j) \leq d(j), pmtn\|-$	$O(mn^3)^*$	Federgruen and Groenevelt (1986)

* max-flow algorithm by Ahuja et al. (1994)

Table 1: Complexity Results for Problems with Fixed Processing Times

4 Methodology 1: Flows in Networks

Various network flow techniques form an essential part of the SCPT toolkit. In this section, we briefly review relevant techniques, including those that handle networks with parametric capacities. Further details on this topic can be found in the monograph by Ahuja et al. (1993).

Scheduling problems under consideration, with fixed and controllable processing times, can be reformulated in terms of various flow problems in networks of a particular structure. Introduce a generic network $G = (V, A)$, schematically shown in Figure 1. The results presented in this section normally hold for more general networks; however, for our purposes, we give an exposition of these results in relation to network G, as the most relevant to our review.

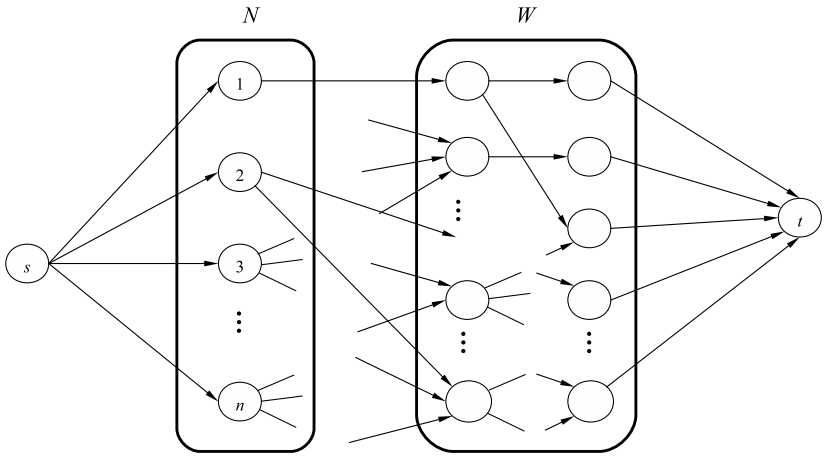

Figure 1: Network $G = (V, A)$

The set $V = \{s, t\} \cup N \cup W$ of nodes consists of the source s, the sink t and two subsets $N = \{1, 2, \ldots, n\}$ and W. The set A of arcs contains the arcs (s, j) for each node $j \in N$, but s is neither directly linked to the sink t nor to a node of set W. There are arcs from the nodes of set N to those of set W; arcs are also possible between the nodes of set W. The arcs entering the sink t only come from some nodes of set W.

The capacity of arc (v, v') is denoted by $\mu(v, v')$, which can be infinite for some arcs.

A flow f is a function $f : A \to \mathbb{R}$ that assigns real numbers to arcs. We say that a flow $f : A \to \mathbb{R}$ is *feasible* if it satisfies the *capacity constraint*

$$0 \le f(v, v') \le \mu(v, v'), \qquad (v, v') \in A, \tag{13}$$

and the *flow balance constraint*

$$\sum_{v' \in V,\ (v,v') \in A} f(v, v') = \sum_{v' \in V,\ (v',v) \in A} f(v', v), \qquad v \in V \setminus \{s, t\}. \tag{14}$$

In the *maximum flow problem,* it is required to find a feasible flow of the maximum value, where the *value* of a flow f is the total flow on the arcs that leave the source (or, equivalently, enter the sink):

$$\text{the value of flow } f = \sum_{v' \in N,\ (s,v') \in A} f(s, v') = \sum_{v \in W,\ (v,t) \in A} f(v, t).$$

In the case of network $G = (V, A)$, an algorithm due to Karzanov (1974) finds the maximum flow in $O(|V|^3)$ time, while one of the fastest strongly polynomial algorithms due to Goldberg and Tarjan (1988) takes $O(|V||A| \log(|V|^2/|A|))$ time; see rows 1 and 2 of Table 2.

A partition (S, T) of the node set V such that $s \in S$ and $t \in T$ is called an *s-t cut*. The capacity $\mu(S, T)$ of an *s-t* cut (S, T) is defined as the total capacity of the arcs that go from nodes of set S to nodes of set T, i.e.,

$$\mu(S, T) = \sum_{(v,v') \in A(S,T)} \mu(v, v'),$$

where $A(S, T) = \{(v, v') \in A \mid v \in S,\ v' \in T\}$. An *s-t* cut (S, T) is called a *minimum s-t cut* if its capacity $\mu(S, T)$ is the minimum among all *s-t* cuts in G. The maximum-flow minimum-cut theorem, the most well-known statement of network optimization, asserts that the value of the maximum flow is equal to the capacity of a minimum *s-t* cut.

In the *minimum-cost flow problem*, each arc $(v, v') \in A$ is associated with a cost $c(v, v')$ and it is required to find a feasible flow of a given value that has the smallest cost. In this paper, we will mainly be interested in the *minimum-cost maximum flow problem*, i.e., the problem of finding the maximum flow of the smallest cost. The problem can be solved by an algorithm by Orlin (1988), which is currently the fastest strongly polynomial algorithm; in the case of network G the algorithm requires $O(|A| \log |V|(|A| + |V| \log |V|))$ time; see row 3 of Table 2.

A range of network flow problems closely related to scheduling applications with variable processing times contains the problems of finding a *parametric maximum flow*. The work by Gallo et al. (1989) presents fast algorithms for solving the parametric maximum flow problem, provided that the capacities of all arcs are constant, except for the capacities of the arcs that leave the source (or enter the sink) which depend on a single parameter λ. More precisely, the capacity on an arc (s, j), $j \in N$, is given by $\mu_\lambda(s, j)$, which is a non-decreasing function of λ. There are several algorithms presented by Gallo et al. (1989) that find the maximum flow for all values of the parameter λ; for our purposes, we are interested in two of them, with the running times of $O(|V|^3)$ and $O(|V||A| \log(|V|^2/|A|))$, respectively; see rows 4 and 5 of Table 2. These algorithms are adaptations of the algorithms by Karzanov (1974)

#	**Flow Problem**	**General Network** G	**Bipartite Network** G, $\lvert N\rvert \leq \lvert W\rvert$
1	Max-Flow	$O(\lvert V\rvert^3)$ Karzanov (1974)	$O(\lvert N\rvert\lvert A\rvert + \lvert N\rvert^3)$ Ahuja et al. (1994)
2	Max-Flow	$O(\lvert V\rvert\lvert A\rvert \log(\lvert V\rvert^2/\lvert A\rvert))$ Goldberg and Tarjan (1988)	$O(\lvert N\rvert\lvert A\rvert \log(\lvert N\rvert^2/\lvert A\rvert + 2))$ Ahuja et al. (1994)
3	Min-Cost Max-Flow	$O(\lvert A\rvert \log \lvert V\rvert(\lvert A\rvert + \lvert V\rvert \log \lvert V\rvert))$ Orlin (1988)	
4	Parametric Max-Flow (single parameter, parametric capacities only on arcs leaving source/entering sink)	$O(\lvert V\rvert^3)$ Gallo et al. (1989)	$O(\lvert N\rvert^2\lvert W\rvert + \lvert N\rvert^3)$ Ahuja et al. (1994)
5		$O(\lvert V\rvert\lvert A\rvert \log(\lvert V\rvert^2/\lvert A\rvert))$ Gallo et al. (1989)	$O(\lvert N\rvert\lvert A\rvert \log(\lvert N\rvert^2/\lvert A\rvert + 2))$ Ahuja et al. (1994)
6	Parametric Max Flow (multiple parameters, parametric capacities only on arcs leaving source/entering sink)	$O(\lvert V\rvert^3)$ McCormick (1999)	$O(\lvert N\rvert^2\lvert W\rvert + \lvert N\rvert^3)$ Ahuja et al. (1994)
7		$O(\lvert V\rvert\lvert A\rvert \log(\lvert V\rvert^2/\lvert A\rvert))$ McCormick (1999)	$O(\lvert N\rvert\lvert A\rvert \log(\lvert N\rvert^2/\lvert A\rvert + 2))$ Ahuja et al. (1994)
8	Min-Cost Max Flow (non-zero costs only on arcs leaving source/entering sink)	$O(\lvert V\rvert^3)$ McCormick (1999) Hochbaum and Hong (1995)	$O(\lvert N\rvert^2\lvert W\rvert + \lvert N\rvert^3)$ Ahuja et al. (1994)
9		$O(\lvert V\rvert\lvert A\rvert \log(\lvert V\rvert^2/\lvert A\rvert))$ McCormick (1999) Hochbaum and Hong (1995)	$O(\lvert N\rvert\lvert A\rvert \log(\lvert N\rvert^2/\lvert A\rvert + 2))$ Ahuja et al. (1994)

Table 2: Running Times of Flow Algorithms Applied to Network $G = (V, A)$

and Goldberg and Tarjan (1988), respectively, and require the same running times as in the non-parametric case; see rows 1 and 2 of Table 2.

Among network flow problems considered by McCormick (1999) there is a parametric maximum flow problem, which can be stated with respect to our network G as follows. Suppose that on each arc (s, j), $j \in N$, with the source s the capacity is given as non-increasing linear function $b(j) - a(j)\lambda(j)$, where $b(j)$ and $a(j)$ are given constants, while $\lambda(j)$ is a non-negative parameter. It is required to find such values of $\lambda(j)$ that $\sum_{j \in N} \lambda(j)$ is minimum and there exists a flow saturating the arcs from s. The problem reduces to finding a maximum flow, provided the capacity on an arc (s, j) leaving the source depends on an individual parameter $\lambda(j)$, rather than on a single parameter λ, common for all these arcs, as in the models studied by Gallo et al. (1989). It is essentially proved in McCormick (1999) that for solving this multi-parameter problem the algorithms from Gallo et al. (1989) can be adapted without increasing their running times. For network G this means that the multi-parameter maximum flow can be found either in $O(|V|^3)$ or in $O(|V||A| \log(|V|^2/|A|))$ time; see rows 6 and 7 of Table 2.

Notice that Gallo et al. (1989) consider the problem of finding the maximum flow for all values of a single parameter λ and allow the capacity functions to be arbitrary monotone functions of λ. McCormick (1999) allows multiple parameters but considers only linear capacity functions and aims at finding the flow that corresponds to the minimum sum of the parameters, not the maximum flow for all values of the parameters.

McCormick (1999) also establishes the equivalence (with respect to the time complexity) between the problem of finding a maximum flow in a network with parametric capacities on the arcs leaving the source and the minimum-cost flow problem in a network with non-zero costs on some arcs entering the sink. In order to solve a more general version of the latter problem with a quadratic cost function, Hochbaum and Hong (1995) adapt the algorithms of Gallo et al. (1989) without increasing their running times; see rows 8 and 9 of Table 2. The results stated above also hold in a symmetric case, i.e., when the parametric capacities are applied to only the arcs that enter the sink and non-zero costs are assigned to the arcs that leave the source.

Notice that if there are no arcs between the nodes of W, then network G is bipartite. Moreover, in virtually all scheduling applications, network G is not balanced, i.e., $|N| \leq |W|$. It is demonstrated by Ahuja et al. (1994) that many network flow algorithms can be run faster on unbalanced bipartite networks, so that the running time depends not on the total number of nodes but rather on the number of nodes in the part of the lower cardinality. This is reflected in the last column of Table 2.

5 Fixed Processing Times. Parallel Machines. Distinct Release Dates and Deadlines

In this section, we discuss problems $\alpha|r(j),\ C(j) \leq d(j),\ pmtn|-$ with $\alpha \in \{P, Q\}$ of checking the existence of a feasible schedule, provided that the processing times are known and fixed. We illustrate how these problems reduce to the network flow problems, so that Methodology 1 can be used for their solution. In particular, we clarify that the fastest known correct algorithm for solving problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ requires $O\left(n^3\right)$ time, and not $O\left(n^2 \log^2 n\right)$, as is often assumed in the literature on the SIC models; see Leung (2004) and Ho (2004).

We start with the feasibility problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ on m identical par-

allel machines and with its special case $1|r(j),\ C(j) \leq d(j),\ pmtn|-$ on a single machine. Introduce network $G_P = (V, A)$ and define it as the following version of the generic network $G = (V, A)$ outlined in Figure 1. The node set V consists of the source s, the sink t, set N of job nodes and set $W = \mathcal{I} = \{I_1, I_2, \ldots, I_\gamma\}$ of the interval nodes. The set A of arcs is defined as $A = A^s \cup A^0 \cup A^t$, where

$$\begin{aligned} A^s &= \{(s, j) \mid j \in N\}, \\ A^0 &= \{(j, I_h) \mid j \in N,\ I_h \in \Gamma(j)\}, \\ A^t &= \{(I_h, t) \mid I_h \in \mathcal{I}\}. \end{aligned}$$

Thus, in G_P the source is connected to each job node, each interval node is connected to the sink, and each job node is connected to the nodes associated with the intervals during which the corresponding job can be processed; see Figure 2.

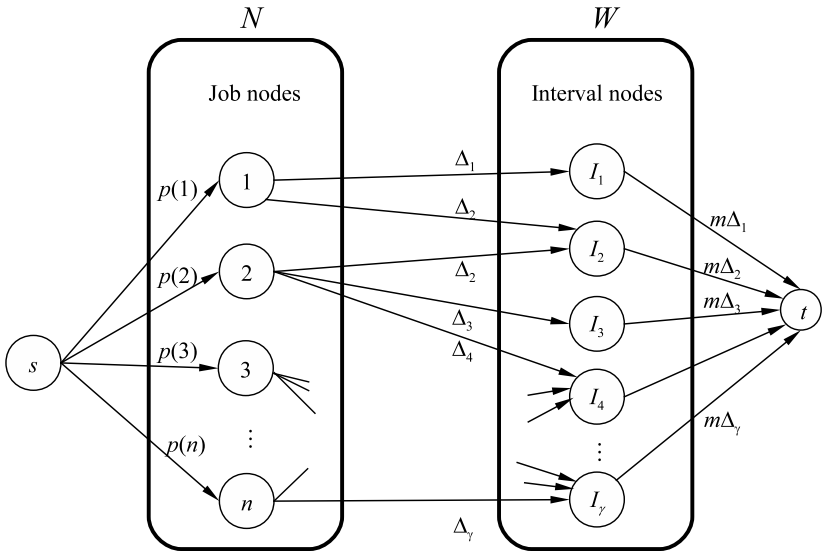

Figure 2: Network $G_P = (V, A)$

Given an instance of a feasibility problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ on m identical parallel machines (or of problem $1|r(j),\ C(j) \leq d(j),\ pmtn|-$ on a single machine), define the arc capacity function $\mu : A \to \mathbb{R}$ by

$$\begin{aligned} &\mu(s, j) = p(j), && (s, j) \in A^s, \\ &\mu(j, I_h) = \Delta_h, && (j, I_h) \in A^0, \\ &\mu(I_h, t) = m\Delta_h, && (I_h, t) \in A^t. \end{aligned}$$

Recall that solving a feasibility scheduling problem reduces to testing the inequality (3) for each set $X \subseteq N$ of jobs, where φ is a suitably defined processing capacity function. In the case under consideration, such a testing can be translated in terms of the network flow problem, as independently shown by Gordon and Tanaev (1973) and Horn (1974).

Lemma 1 (cf. Gordon and Tanaev (1973); Horn (1974)) *For positive real numbers $p(j)$, $j \in N$, there exists a feasible schedule for processing the jobs of set N on m parallel identical machines (or on a single machine if $m = 1$) such that job $j \in N$ has the actual processing time of $p(j)$ if and only if there exists a feasible flow $f : A \to \mathbb{R}_+$ in network G_P satisfying $f(s, j) = p(j)$ for all $j \in N$.*

Hence, problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ can be tested by solving the maximum flow problem in network G_P: if the value of the maximum flow is equal to $\sum_{j \in N} p(j)$, then problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ is feasible; otherwise, it is infeasible.

A feasible flow $f(j, I_h)$ on arc (j, I_h) defines for how long job j is processed in the time interval I_h. On a single machine, a feasible flow easily translates into a feasible schedule and vice versa, since there is a one-to-one correspondence between the flow incoming into an interval node I_h and durations of jobs processed within the corresponding time interval on a single machine. In the case of m identical parallel machines, the link between a feasible flow and a feasible schedule is less evident. To know the flow values $f(j, I_h)$ is insufficient to define a schedule. We need a linear time algorithm by McNaughton (1959) to find a feasible preemptive schedule for each interval I_h, and then the overall schedule can be found as a concatenation of these schedules.

Network G_P contains $O(n)$ nodes. For such a network, finding a maximum flow requires $O(n^3)$ time by Karzanov's algorithm; see row 2 of Table 2. The running time of $O(n^3)$ does not depend on the number of machines in the scheduling problem, and remains valid if the described flow approach is used for the single machine problem. However, the single machine feasibility problem $1|r(j),\ C(j) \leq d(j),\ pmtn|-$ can be solved much faster, in $O(n \log n)$ time by Algorithm EDF; see Section 3.

For a single machine, an algorithm that is based on the network flow reasoning but runs faster than in $O(n^3)$ time is developed by Chung et al. (1989) and Shih et al. (1989). The idea is to transform the original network G_P shown in Figure 2, replacing the set of the interval nodes by a balanced binary tree, in which the original interval nodes are the leaves at the lowest level. The tree is created recursively starting from the leaves, so that each pair of nodes of the same height that represent two adjacent intervals become children of a node of the higher level that represents the union of these intervals. The tree is completed with creating the root that is associated with the interval $[\tau_0, \tau_\gamma]$. The arc capacities are redistributed accordingly. Without going into technical details, which can be found in Chung et al. (1989), Shih et al. (1989), here we just illustrate this approach with a small size example.

Consider an instance of problem $1|r(j),\ C(j) \leq d(j),\ pmtn|-$ with three jobs and the set $\mathcal{I}$ of intervals consisting of four intervals $I_h = [\tau_{h-1}, \tau_h]$, $h \in \{1, 2, 3, 4\}$, such that the intervals $[\tau_0, \tau_4]$, $[\tau_1, \tau_3]$ and $[\tau_2, \tau_4]$ are available for processing job 1, job 2 and job 3, respectively. See Figure 3 for the corresponding network G_P, with the arcs capacities shown explicitly. The modified network G'_P, with the interval nodes organized as a binary tree, is shown in Figure 4.

For problem $1|r(j),\ C(j) \leq d(j),\ pmtn|-$ with γ intervals in set $\mathcal{I}$, there are $O(\gamma)$ nodes in the binary tree of the modified network G'_P. At most $2 \log \gamma$ arcs leave each job node. Thus, given that $\gamma = O(n)$, we deduce that in the network G'_P associated with problem $1|r(j),\ C(j) \leq d(j),\ pmtn|-$ there are $O(n)$ nodes and $O(n \log n)$ arcs. This network is not bipartite, but still is a version of the generic network G shown in Figure 1. We apply the algorithm by Goldberg and Tarjan (1988); see row 2 of Table 2. Since $|A| \geq n$, we deduce that a maximum flow in G'_P can be found in $O(n^2 \log^2 n)$ time.

Chung et al. (1989) and Shih et al. (1989) claim that this approach can be extended to parallel identical machines, but give no implementation details. That claim is known in the imprecise computation research community, and several authors, assuming that the claim is true, assert that problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$, and even its extension with controllable processing times $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ is solvable in $O(n^2 \log^2 n)$ time; see, e.g., surveys by Leung (2004) and Ho (2004).

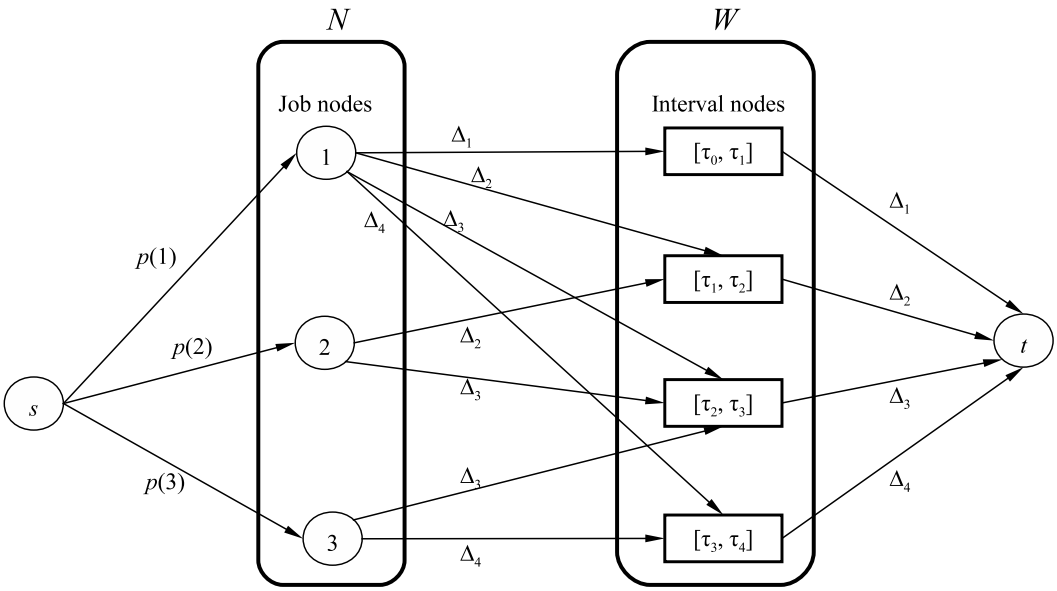

Figure 3: Network G_P for the three-job example with a single machine ($m = 1$)

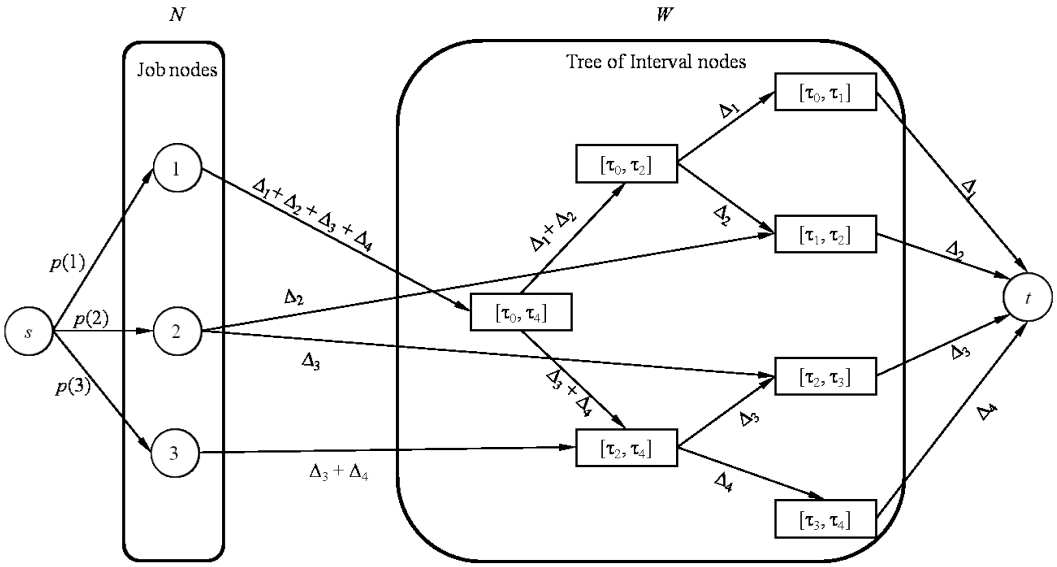

Figure 4: Network G'_P for the three-job example with a single machine ($m = 1$)

We, on the other hand, are confident that the claim that problem $P|r(j),\, C(j) \leq d\,(j)\,,$ $pmtn|-$ with $m \geq 2$ can be solved by finding a maximum flow in the modified network G'_P does not hold. To handle multiple machines, each interval that is contained in the binary tree of the interval nodes should be made available for all m machines. To achieve this, the capacity of each arc that leaves an interval node has to be multiplied by m (as is done in network G_P). But in this case a feasible flow does not necessarily translate into a feasible schedule. To illustrate this, for the example above assume that $m = 2$, $p(1) = 6$, $\Delta_1 = 3$, $\Delta_2 = 2$. Then the capacity of the arc that enters node $[\tau_0, \tau_2]$ should become equal to $m \times (\Delta_1 + \Delta_2) = 10$, while the capacity of the arc that enters node $[\tau_0, \tau_1]$ to $m \times \Delta_1 = 6$.

A feasible flow may be equal to 6 on each of these two arcs, but such a flow admits no scheduling interpretation, since it would imply that job 1 is processed during 6 time units in the interval $[\tau_0, \tau_1]$ of length 3, i.e., it is processed simultaneously on both machines.

A possible alternative attempt to reduce problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ to the maximum flow problem that is based on the binary tree representation of the interval nodes is to introduce m copies of the tree on interval nodes, one tree for each machine. However, a feasible flow again may lead to an infeasible schedule, since there is no mechanism to stop assigning one job to the same time interval on several machines.

The example given above shows that reducing the feasibility problem to the maximum flow problem in the network that uses a binary tree representation of the interval nodes works only for a single machine.

Remark 1 *The fastest correct algorithm for solving problem $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ requires $O(n^3)$ time. In the literature on imprecise computation, solving problems $P|r(j),\ C(j) \leq d(j),\ pmtn|-$ and $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ is often used as a subroutine for various problems on identical parallel machines, and the reported running times are derived under the assumption that each of these problems can be solved in $O(n^2 \log^2 n)$ time. In the subsequent sections, we will correct the estimates of earlier known algorithms that use that assumption, increasing the running time of the subroutine to $O(n^3)$ and making the reference to this remark.*

We now pass to the feasibility problem $Q|r(j),\ C(j) \leq d(j),\ pmtn|-$ on m uniform machines. For simplicity of exposition, assume that the machine speeds are pairwise distinct and the machines are numbered in the decreasing order of their speeds, i.e., $s_1 > s_2 > \cdots > s_m$. For completeness, define $s_{m+1} = 0$. Taking into consideration the speed of each machine, notice that in an interval I_h total processing that could be done on machine M_1 is $s_1\Delta_h$, on machine M_2 is $s_2\Delta_h$, and so on.

Federgruen and Groenevelt (1986) reduce the feasibility problem $Q|r(j),\ C(j) \leq d(j),\ pmtn|-$ to the maximum flow problem in a special network, which we call network G_Q; for illustration see Figure 5. This network is also a variant of the generic network G. In G_Q, the set of nodes contains the set N of job nodes, and the set W consists of machine-interval nodes (I_h, M_h). The set A of arcs is defined as $A = A^s \cup A^0 \cup A^t$, where

$$\begin{aligned} A^s &= \{(s,j) \mid j \in N\}, \\ A^0 &= \{(j,(I_h,M_i)) \mid j \in N,\ I_h \in \Gamma(j), 1 \leq i \leq m\}, \\ A^t &= \{((I_h,M_i),t) \mid I_h \in \mathcal{I}, 1 \leq i \leq m\}. \end{aligned}$$

The capacities on the arcs are as follows:

$$\begin{aligned} \mu(s,j) &= p(j), && (s,j) \in A^s, \\ \mu(j,(I_h,M_i)) &= \Delta_h(s_i - s_{i+1}), && (j,(I_h,M_i)) \in A^0, \\ \mu((I_h,M_i),t) &= i\Delta_h(s_i - s_{i+1}), && (I_h,M_i) \in A^t. \end{aligned}$$

Again, the feasibility scheduling problem can be reduced to the maximum flow problem: as shown by Federgruen and Groenevelt (1986) the statement of Lemma 1 holds for the case of uniform machines, with network G_P replaced by G_Q. For this problem we can apply Karzanov's algorithm adapted to an unbalanced bipartite network (row 1 of Table 2) to finding the maximum flow in the network G_Q. Since $|N| = n$ and $|A| = O(mn^2)$, such an algorithm will solve problem $Q|r(j),\ C(j) \leq d(j),\ pmtn|-$ in $O(mn^3)$ time.

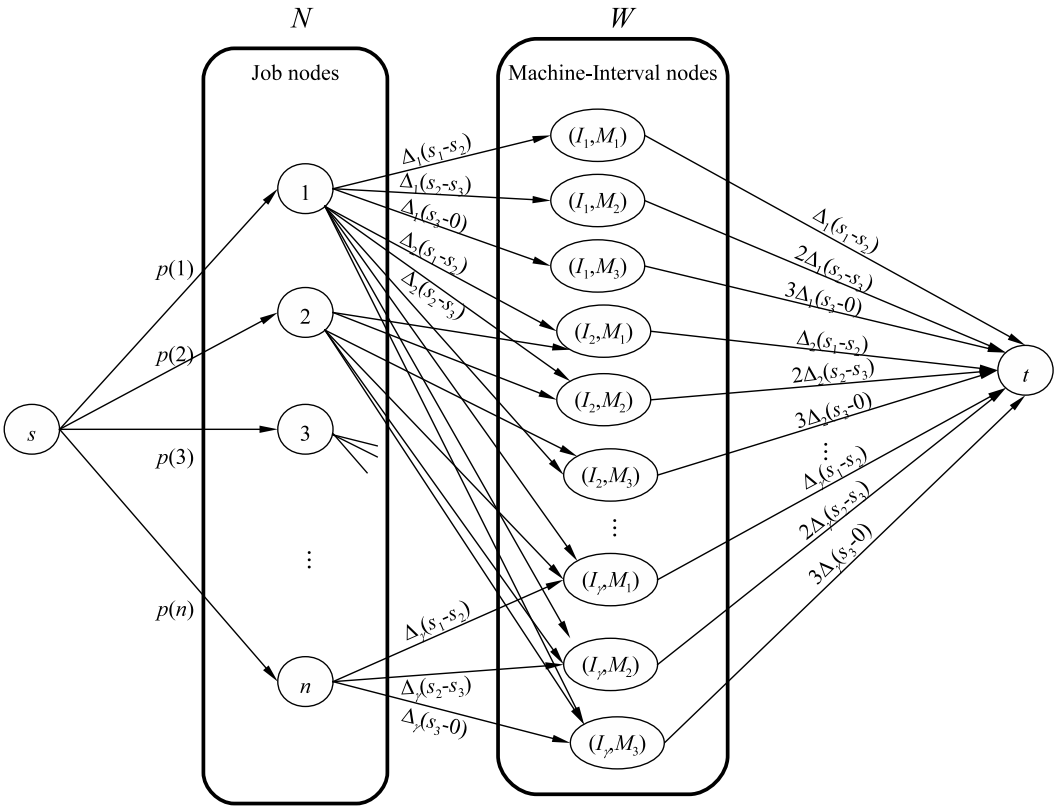

Figure 5: An illustration of network $G_Q = (V, A)$ for three uniform machines

See Table 1 for the summary of the results from Sections 3 and 5 for various versions of the feasibility problem $\alpha|r(j), C(j) \leq d(j), pmtn|-$ with fixed processing times.

The running times in Table 1 establish lower bounds on the running times of algorithms for solving problems with controllable processing times. One of the achievements reported in this paper is that almost all problems of the range $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi$ with controllable processing times and distinct release dates and deadlines are not harder computationally than their counterparts with fixed processing times. The tools needed for this purpose include parametric flow problems (Methodology 1) and/or techniques of submodular optimization that are reviewed further on.

6 Total Cost. Parallel Machines. Distinct Release Dates and Deadlines

In this section, we discuss the algorithms for solving problems $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d\,(j)\,,\ pmtn|\Phi_\Sigma$, where $\alpha \in \{P, Q\}$, to minimize the total cost $\Phi_\Sigma = \sum_{j \in N} w_T(j)x(j)$ on identical and uniform parallel machines. We assume that all weights $w_T(j)$ are non-negative. Notice that most of the previously known results on these problems are derived within the body of research of the SIC models. Below, we provide a critical review of these results by (i) clarifying the running time needed to solve problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d\,(j)\,,\ pmtn|\Phi_\Sigma$ based on Remark 1 of Section 5; (ii) demonstrating that the use of advanced techniques of Methodology 1, such as finding parametric flows, results into solution

Problem	Previously known	Methodology 1: Multiparametric flow
$P\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn\|\Phi_\Sigma$	$O(n^4 \log n)^*$ Blazewicz and Finke (1987) Chung et al. (1989) Shih et al. (1989, 1991) Leung (2004)	$O(n^3)$ McCormick (1999) Section 6
$Q\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn\|\Phi_\Sigma$	$O(m^2 n^4 \log mn)$ Blazewicz and Finke (1987) Leung (2004) $O(mn^4)$ Shakhlevich and Strusevich (2008)	$O(mn^3)$ McCormick (1999) Section 6

*After correcting a faulty claim that problem $P|r(j), C(j) \leq d(j), pmtn|-$ is solvable in $O(n^2 \log^2 n)$ time, see Remark 1 of Section 5

Table 3: Complexity of Problems with Different Deadlines

algorithms for problems $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d\,(j)\,,\ pmtn|\Phi_\Sigma$ which have the same running time as the best known for the counterparts of these problems with fixed processing times; see Table 1.

We start with the problems of minimizing the total compression or the *unweighted* compression cost $\Phi_{\mathrm{u}} = \sum_{j \in N} x(j)$, which is often considered in the SIC literature as a special case of the weighted error function; see Leung (2004). Clearly, minimizing $\sum_{j \in N} x(j)$ is equivalent to maximizing the sum of the actual processing times $\sum_{j \in N} p(j)$. As demonstrated in Section 5, the latter problem reduces to finding the maximum flow in either network G_P (if the machines are identical) or in G_Q (if the machines are uniform). The networks G_P and G_Q are of the same structure as described in Section 5, except each arc (s, j), $j \in N$, that leaves the source has an upper bound $\overline{p}(j)$ and a lower bound $\underline{p}(j)$ on its capacity. The resulting problems are computationally equivalent to problems $P|r(j),\ C(j) \leq d\,(j)\,,\ pmtn|-$ and $Q|r(j),\ C(j) \leq d\,(j)\,,\ pmtn|-$, and can be solved by Karzanov's algorithm adapted to an unbalanced bipartite network; see row 1 of Table 2. Thus, problems $\alpha|r(j), p(j) = \overline{p}(j) - x(j),\ C(j) \leq d\,(j)\,,\ pmtn|\Phi_{\mathrm{u}},\ \alpha \in \{P, Q\}$ to minimize the unweighted function $\sum_{j \in N} x(j)$, can be solved in $O(n^3)$ time and in $O(mn^3)$ time, respectively.

Further in this section, we show that minimizing the total *weighted* compression cost $\Phi_\Sigma = \sum_{j \in N} w_T(j) x(j)$ for problems $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d\,(j)\,,\ pmtn|\Phi_\Sigma,$ $\alpha \in \{P, Q\}$, is computationally no harder than their unweighted counterparts. Note that problems $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d\,(j)\,,\ pmtn|\Phi_\Sigma,\ \alpha \in \{P, Q\}$ are among the most popular problems studied within the body of research on SIC. The main solution approach has been based on the reduction of the problem to finding a minimum-cost maximum flow in a special network.

We start with illustrating this approach for problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d\,(j)\,,\ pmtn|\Phi_\Sigma$ with identical machines. The corresponding network, denoted by H_P, is shown in Figure 6. It can be described as an extension of network G_P, introduced in Section 5 for the feasibility problem with fixed processing times: the second set of nodes W is enlarged by adding nodes $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$. For every job $j \in N$, we introduce a so-called "compression"node X_j, with a single incoming arc (j, X_j) and a single outgoing arc (X_j, t),

both having capacity $\theta(j) = \overline{p}(j) - \underline{p}(j)$. For a feasible flow in H_P, the amount of flow passing via nodes X_j corresponds to the compression amounts $x(j)$ of jobs $j \in N$, while the flow via nodes in $\mathcal{I}$ specifies the actual schedule. The flow costs are zero except for arcs connecting the $\mathcal{X}$-nodes and t: the cost of one unit of flow via arc (X_j, t) is $w_T(j)$. For the corresponding minimum-cost maximum flow problem, the total cost is $\Phi_\Sigma = \sum_{j \in N} w_T(j)x(j)$, where $x(j)$ is the flow on arc (X_j, t). Notice that network H_P is slightly different from the one used in the literature on imprecise computation; see, e.g., Leung (2004, Figure 34.3). These differences are minor, and the numbers of arcs and nodes in both networks are of the same order, i.e., $|V| = O(n)$ and $|A| = O(n^2)$.

The minimum-cost maximum flow problem in network H_P can be solved by an algorithm by Orlin (1988) (row 3 of Table 2). Its direct application solves problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ in $O(n^4 \log n)$ time. Notice that the latter problem cannot be modelled as a modified network H_P with the interval nodes arranged in a balanced binary tree. Thus, in accordance with Remark 1 of Section 5, the estimate of $O(n^4 \log n)$ should replace the running time of $O(n^2 \log^3 n)$ reported in Leung (2004).

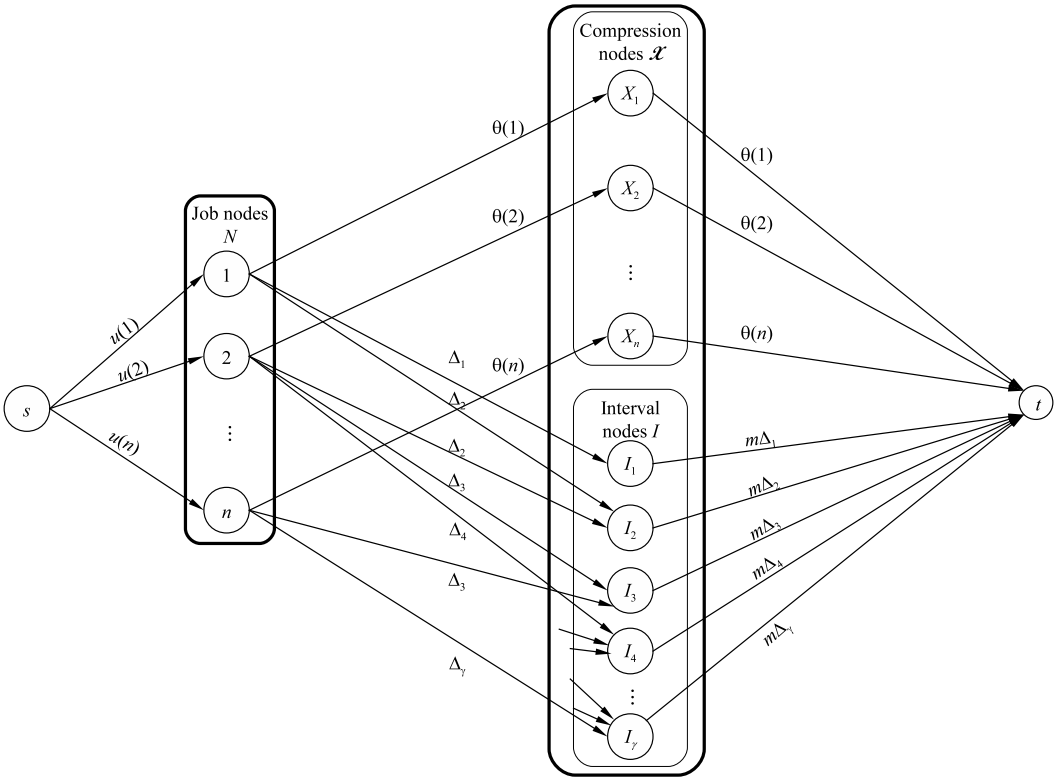

Figure 6: Network $H_P = (V, A)$ for problem $\Pi_\Sigma(P)$ on identical machines

Consider now problem $Q|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ with uniform machines. The corresponding network, denoted by H_Q, is shown in Figure 7. It is an extension of network G_Q from Section 5 obtained by adding the set of compression nodes $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ in the same way, as set $\mathcal{X}$ is added to G_P resulting in H_P.

Again, the introduced network H_Q is only slightly different from the one often used in the imprecise computation literature; see, e.g., Leung (2004, Figure 34.4), while the major characteristics, such as $|V| = O(mn)$ and $|A| = O(mn^2)$, are the same for both networks. The running time for solving problem $Q|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$

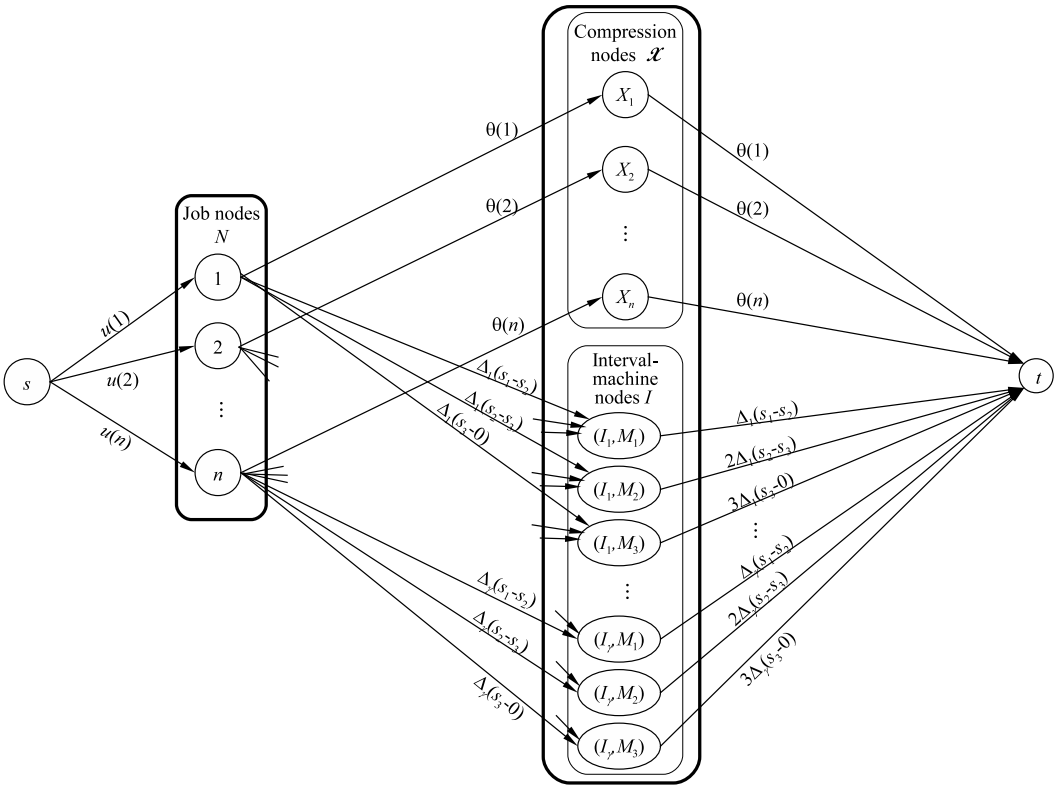

Figure 7: Network $H_Q = (V, A)$ for problem $\Pi_\Sigma(Q)$ on $m = 3$ uniform machines

reported in Leung (2004) is derived from applying Orlin's algorithm (row 3 of Table 2) to the network H_Q and is equal to $O(m^2n^4 \log mn)$. A faster algorithm for problem $Q|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$, $pmtn|\Phi_\Sigma$ is given by Shakhlevich and Strusevich (2008); it requires $O(mn^4)$ time.

As seen, the quoted best times known for problems $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$, $pmtn|\Phi_\Sigma$, $\alpha \in \{P, Q\}$, exceed those known for solving the corresponding feasibility problems $\alpha|r(j), C(j) \leq d\,(j)$, $pmtn|-$; see Table 1. Below we show that using an alternative approach, the times needed to solve problems $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$, $pmtn|\Phi_\Sigma$ can be reduced to match those for problems $\alpha|r(j)$, $C(j) \leq d\,(j)$, $pmtn|-$, as stated in Table 3.

We present the parametric maximum flow approach that results from adapting the methods developed by Chen (1994) and McCormick (1999). In those papers, a scheduling problem with controllable processing times is addressed, and the actual processing time of a job j is determined by $p(j) = \max\{b(j) - a(j)\lambda(j), 0\}$, where $b(j)$ and $a(j)$ are given constants while $\lambda(j)$ is a non-negative parameter, and the objective is to minimize $\sum_{j\in N} \lambda(j)$. This scheduling problem is equivalent to a special case of problem $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$, $pmtn|\Phi_\Sigma$ with zero lower bounds on processing times; see (2). Notice that the parametric flow algorithms by McCormick (1999) are developed for the flow problems with zero lower bounds on the arc capacities; in scheduling terms that means zero lower bounds on processing times, $\underline{p}(j) = 0$, $j \in N$. The algorithms can be extended to deal with non-zero lower bounds $\underline{p}(j)$, $j \in N$, by standard network flow techniques.

Following McCormick (1999), to solve problem $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$,

$pmtn|\Phi_\Sigma$, where $\alpha \in \{P, Q\}$, consider the problem of finding the maximum flow in network G_α, defined in Section 5, provided that the fixed capacity $\overline{p}(j)$ of each arc (s, j), $j \in N$, is replaced by a parametric capacity $\max\{\overline{p}(j) - x(j), 0\} = \max\{b(j) - a(j)\lambda(j), 0\}$. Suppose that $f(a), a \in A$, is the found maximum flow. Then for the arcs (s, j) entering the job-nodes the flow $f(s, j)$ determines $p(j)$, the actual processing time of job j. For network G_P, the flow on an arc (j, I_h) defines for how long job j is processed in the time interval I_h, while for network G_Q, the flow on an arc $(j, (I_h, M_i))$ defines for how long job j is processed in the time interval I_h on machine M_i.

Thus, problem $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ can be solved by McCormick's algorithms for solving the multi-parametric maximum flow problem; see rows 6 and 7 of Table 2. An important requirement, satisfied for both networks G_P and G_Q, that makes McCormick's techniques applicable, is the common tail s of the parametric arcs. Besides, since each network G_P and G_Q is bipartite, the running time of the algorithms can be sped-up by the techniques of Ahuja et al. (1994).

We apply the adapted McCormick's algorithm with the running time $O(|N|^2|W| + |N|^3)$; see row 6 of Table 2. In network G_P, we have that $|V| = |N| + |W| \leq 3n$; thus, the algorithm solves problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ in $O(n^3)$ time. In network G_Q, we have that $|W| \leq 2mn$, so that problem $Q|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ is solved in $O(mn^3)$ time.

Applying the other version of McCormick's algorithm of time complexity $O(|N||A| \log(|N|^2/|A| + 2))$ (row 7 of Table 2) proves to be less efficient. In the case of network G_P, we have that $n \leq |A| \leq 2n^2$, while for network G_Q the inequalities $mn \leq |A| \leq 2mn^2$ hold. Using the stated lower bounds on $|A|$, we deduce that $|N|^2/|A|$ is $O(n)$ for the identical machines, and is $O(n/m)$ for the uniform machines. Thus, the algorithm solves problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ in $O(n^3 \log n)$ time, while the time required to solve problem $Q|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ is no less than $O(mn^3 \log(n/m))$.

The main conclusion of this section is that the running times $O(n^3)$ and $O(mn^3)$ that are required to solve the problems of minimizing the total cost on parallel identical and uniform machines, respectively, coincide with those reported in Table 3. These values are much better that those previously known in the SIC literature.

7 Methodology 2: Optimization over Submodular Polyhedra

In this section, we briefly remind some basic definitions and approaches of submodular optimization. The key tool which provides efficient algorithms for many SCPT problems is the greedy algorithm that solves linear programming problems over submodular polyhedra.

We mainly follow a comprehensive monograph on submodular optimization by Fujishige (2005), see also Katoh and Ibaraki (1998) and Schrijver (2003).

For a positive integer n, let $N = \{1, 2, \ldots, n\}$ be a ground set, and let 2^N denote the family of all subsets of N. As in Section 3, for a subset $X \subseteq N$, let $\mathbb{R}^X$ denote the set of all vectors $\mathbf{p}$ with real components $p(j)$, where $j \in X$. For two vectors $\mathbf{p} = (p(1), p(2), \ldots, p(n)) \in \mathbb{R}^N$ and $\mathbf{q} = (q(1), q(2), \ldots, q(n)) \in \mathbb{R}^N$, we write $\mathbf{p} \leq \mathbf{q}$ if $p(j) \leq q(j)$ for each $j \in N$. For a vector $\mathbf{p} \in \mathbb{R}^N$, define $p(X) = \sum_{j \in X} p(j)$ for every set $X \in 2^N$.

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

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

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

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

$$\begin{aligned} P(\varphi) &= \{\mathbf{p} \in \mathbb{R}^N \mid p(X) \leq \varphi(X), \quad X \in 2^N\}; \\ B(\varphi) &= \{\mathbf{p} \in \mathbb{R}^N \mid \mathbf{p} \in P(\varphi), \quad p(N) = \varphi(N)\}, \end{aligned}$$

called the *submodular polyhedron* and the *base polyhedron*, respectively, associated with the submodular system.

The main problem of our interest is as follows:

$$\begin{array}{lll} \text{(LP)}: \ \max & \displaystyle\sum_{j \in N} w(j)p(j) & \\ \text{s.t.} & p(X) \leq \varphi(X), & X \in 2^N, \\ & \underline{p}(j) \leq p(j) \leq \overline{p}(j), & j \in N, \end{array} \tag{15}$$

where $\varphi : 2^N \to \mathbb{R}$ is a submodular function with $\varphi(\emptyset) = 0$, $\mathbf{p} \in \mathbb{R}^N$ is a vector of decision variables, $\mathbf{w} \in \mathbb{R}^N_+$ is a non-negative weight vector, and $\overline{\mathbf{p}}, \underline{\mathbf{p}} \in \mathbb{R}^N$ are vectors of upper and lower bounds on the components of vector $\mathbf{p}$, respectively. Further in this survey, we refer to (15) as *Problem (LP)*. This problem serves as a mathematical model for many SCPT problems, as demonstrated below.

Problem (LP) can be classified as a problem of maximizing a linear function over a submodular polyhedron intersected with a box. As shown in Shakhlevich et al. (2009), Problem (LP) can be reduced to optimization over a base polyhedron.

Theorem 2 (cf. Shakhlevich et al. (2009)) *If Problem (LP) has a feasible solution, then the set of its maximal feasible solutions is a base polyhedron $B(\tilde{\varphi})$ associated with the submodular system $(2^N, \tilde{\varphi})$, where the rank function $\tilde{\varphi} : 2^N \to \mathbb{R}$ is given by*

$$\tilde{\varphi}(X) = \min_{Y \in 2^N} \{\varphi(Y) + \overline{p}(X \setminus Y) - \underline{p}(Y \setminus X)\}. \tag{16}$$

Notice that in (16) computing the value $\tilde{\varphi}(X)$ for a given $X \in 2^N$ reduces to minimization of a submodular function. It is well known that an arbitrary submodular function can be minimized in polynomial time; see Schrijver (2000) and Iwata et al. (2001). However, the running time of these general algorithms is fairly large. In many special cases of Problem (LP), including its applications to the SCPT problems, the value $\tilde{\varphi}(X)$ can be computed more efficiently, as shown later.

Throughout this paper, we assume that Problem (LP) has a feasible solution, which is equivalent to the conditions $\underline{\mathbf{p}} \in P(\varphi)$ and $\underline{\mathbf{p}} \leq \overline{\mathbf{p}}$; see, e.g., Fujishige (2005). Theorem 2 implies that Problem (LP) reduces to the following problem:

$$\begin{array}{ll} \max & \displaystyle\sum_{j \in N} w(j)p(j) \\ \text{s.t.} & \mathbf{p} \in B(\tilde{\varphi}), \end{array} \tag{17}$$

where the rank function $\tilde{\varphi} : 2^N \to \mathbb{R}$ is given by (16).

An advantage of the reduction of Problem (LP) to a problem of the form (17) is that the solution vector can be obtained essentially in a closed form by a greedy algorithm. To determine an optimal vector $\mathbf{p}^*$, the algorithm starts with $\mathbf{p}^* = \underline{\mathbf{p}}$, considers the components of the current $\mathbf{p}^*$ in non-increasing order of their weights and gives the current component the largest possible increment that keeps the vector feasible.

Let $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be a permutation of elements in $N = \{1, 2, \ldots, n\}$ such that $w(\sigma(1)) \geq w(\sigma(2)) \geq \cdots \geq w(\sigma(n))$, and define $N_t(\sigma) = \{\sigma(1), \ldots, \sigma(t)\}$ for $t = 1, 2, \ldots, n$, where, for completeness, $N_0(\sigma) = \emptyset$.

Theorem 3 (cf. Fujishige (2005)) *Vector $\mathbf{p}^* \in \mathbb{R}^N$ given by*

$$p^*(\sigma(t)) = \tilde{\varphi}(N_t(\sigma)) - \tilde{\varphi}(N_{t-1}(\sigma)), \ t = 1, 2, \ldots, n,$$

is an optimal solution to problem (17) (and also to the problem (15)).

We now demonstrate that the SCPT problems to minimize the total weighted compression cost can be reformulated in terms of solving Problem (LP) of the form (15), where the rank function $\varphi(X)$ is a suitable processing capacity function defined in Section 3.

Take a generic problem $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$, for which $\varphi(X)$ is the corresponding processing capacity function. Notice that function $\varphi(X)$ is submodular; see, e.g., Shakhlevich and Strusevich (2005, 2008). Intuitively, submodularity of the processing capacity function can be naturally explained by using an equivalent definition of a submodular function, known as the law of diminishing returns: a set function φ is submodular if and only if the inequality

$$\varphi(X \cup \{j\}) - \varphi(X) \geq \varphi(Y \cup \{j\}) - \varphi(Y)$$

holds for all sets $X \subset Y \subseteq N$ and all $j \in N \setminus Y$. Since in our case $\varphi(X)$ is the total duration of all time intervals available for processing of jobs of set X, the value $\varphi(X \cup \{j\}) - \varphi(X)$ is the length of all intervals in which the job j can be processed, while none of the jobs of set X can. The inequality above holds due to $X \subset Y$.

Recall that in the problem $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$, it is required to find a feasible schedule with the job processing times $p(j) = \overline{p}(j) - x(j)$, $j \in N$, that minimizes the total compression cost $\Phi_\Sigma = \sum_{j \in N} w_T(j)\, x(j)$. As follows from Section 3 (see (3)), a feasible schedule exists if and only if the inequality $p(X) \leq \varphi(X)$ holds for each set $X \subseteq N$. Moreover, minimizing the total compression cost $\Phi_\Sigma = \sum_{j \in N} w_T(j)\, x(j)$ is equivalent to maximizing the total weighted processing time $W = \sum_{j \in N} w_T(j) p(j)$. Hence, problem $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ can be reformulated as Problem (LP) of the form (15).

Theorem 4 *In order to solve problem $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ it suffices to solve Problem (LP), where $\varphi(X)$ is the corresponding processing capacity function and $w(j) = w_T(j)$, $j \in N$.*

By Theorems 3 and 4, optimal processing times $p^*(j),\ j \in N$ of the problem $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_\Sigma$ are given as

$$p^*(\sigma(t)) = \tilde{\varphi}(N_t(\sigma)) - \tilde{\varphi}(N_{t-1}(\sigma)), \ t = 1, 2, \ldots, n,$$

and the optimal total weighted processing time is $W = \sum_{t=1}^{n} w_T(\sigma(t))p^*(\sigma(t))$. We can also obtain optimal compression amounts $x^*(j)$, $j \in N$ by $x^*(\sigma(t)) = \overline{p}(\sigma(t)) - p^*(\sigma(t))$, $t = 1, 2, \ldots, n$, and the optimal total compression cost Φ_Σ is given as

$$\Phi_\Sigma = \sum_{t=1}^{n} w_T(\sigma(t))(\overline{p}(\sigma(t)) - p^*(\sigma(t))) = \sum_{t=1}^{n} w_T(\sigma(t))\overline{p}(\sigma(t)) - \sum_{t=1}^{n} w_T(\sigma(t))p^*(\sigma(t))$$
$$= \sum_{t=1}^{n} w_T(\sigma(t))\overline{p}(\sigma(t)) - W. \tag{18}$$

As far as we are aware, the first observation of the link between the SCPT problems and Problem (LP) was made by Nemhauser and Wolsey (1988), who considered a single machine problem with no lower and upper bounds on the processing times. Since 2005, our team has performed a systematic exploration of that link and demonstrated how useful such a link is, bringing a powerful toolkit of submodular optimization into the study of SCPT.

If one reads earlier papers on SCPT, e.g., those reviewed in Nowicki and Zdrzałka (1990) and Shabtay and Steiner (2007), most of them have a common feature. An algorithm based on the greedy ideas is offered and its correctness is proved usually from the first principles using a problem-dependent scheduling argument. An immediate advantage of Theorem 4 is that most of the SCPT problems admit a solution by the greedy algorithm. Other advantages include a possibility of solving efficiently bicriteria problems without using a scheduling argument, as outlined below in Section 8, as well as single criterion problems, applying a novel decomposition algorithm developed for Problem (LP); see Section 10.

8 Controllable Processing Times. Bicriteria Problems

In this section, we review recent results on solving the SCPT problems in which it is required to simultaneously minimize two objective functions, e.g., F_1 and F_2. In problems of this type we need to find the set of Pareto-optimal schedules. Recall that a schedule S' is called *Pareto-optimal* if there exists no schedule S'' such that $F_1(S'') \leq F_1(S')$ and $F_2(S'') \leq F_2(S')$, where at least one of these inequalities is strict. We demonstrate that Methodology 2 provides the foundation to an approach to solve a range of bicriteria SCPT problems. In particular, for these problems Theorems 3 and 4 allow finding the efficiency frontier in a closed form.

Notice that for the problems of the range under consideration previously known algorithms are usually based on scheduling reasoning: typically, they enumerate the breakpoints of the efficiency frontier one by one, constructing the next breakpoint from the previous one by changing the underlying schedule by compressing/decompressing a job.

8.1 Parallel Machines

We start with problems $Q|p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, \Phi_\Sigma)$ and $\alpha|r(j), p(j) = \overline{p}(j) - x(j), pmtn|\,(C_{\max}, \Phi_\Sigma)$ with $\alpha \in \{P, Q\}$. In the third field of the above notation, we write $(C_{\max}, \Phi_\Sigma)$ to indicate that it is required to find the set of Pareto-optimal solutions with respect to two criteria, the makespan $C_{\max}$ and the total compression cost Φ_Σ. The material of this section is mainly based on Shioura et al. (2013).

Given an instance of a bicriteria problem of the indicated range, consider a schedule with a makespan $C_{\max} = d$ that minimizes the total compression cost $\Phi_\Sigma = \Phi_\Sigma(d)$. Note that $\Phi_\Sigma(d)$ depends on the makespan d and is a piecewise-linear function in d representing

the efficiency frontier. In a bicriteria problem, our task is to compute the piecewise-linear function $\Phi_\Sigma(d)$.

As discussed in Section 7, the value $\Phi_\Sigma(d)$ for a given d can be found by solving an appropriate problem of the form (15). Since in this case the rank functions $\varphi(X)$ and $\tilde{\varphi}(X)$ of the form (16) should be seen not only as functions of set X but also as functions of d, in this section we may write $\varphi(X,d)$ and $\tilde{\varphi}(X,d)$ whenever we want to stress that dependence on d. In particular, the value $\Phi_\Sigma(d)$ can be obtained by solving the problem

$$\begin{array}{lll} \max & \sum\limits_{j\in N} w_T(j)p(j) & \\ \text{s.t.} & p(X) \le \varphi(X,d), & X \in 2^N, \\ & \underline{p}(j) \le p(j) \le \overline{p}(j), & j \in N, \end{array} \tag{19}$$

where $\varphi(X,d)$ is a suitably chosen processing capacity function that guarantees that the jobs of set X can be completed by time d. This problem is a parametric version of Problem (LP), and the second line of its constraints describes a parametric submodular polyhedron. As in Section 7, let $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be a permutation of elements in $N = \{1, 2, \ldots, n\}$ such that $w_T(\sigma(1)) \ge w_T(\sigma(2)) \ge \cdots \ge w_T(\sigma(n))$, and define $N_t(\sigma) = \{\sigma(1), \ldots, \sigma(t)\}$, $1 \le t \le n$, where, for completeness, $N_0(\sigma) = \emptyset$. Then, an optimal solution $p^*(j,d)$, $j \in N$, to problem (19) is given as

$$p^*(\sigma(t), j) = \tilde{\varphi}(N_t(\sigma), d) - \tilde{\varphi}(N_{t-1}(\sigma), d),\ t = 1, 2, \ldots, n, \tag{20}$$

and the optimal value $W(d)$ of problem (19) is given by

$$W(d) = \sum_{t=1}^{n} w_T(\sigma(t))p^*(\sigma(t), d). \tag{21}$$

Hence, the total compression cost $\Phi_\Sigma(d)$ is represented as

$$\Phi_\Sigma(d) = \sum_{t=1}^{n} w_T(\sigma(t))\overline{p}(\sigma(t)) - W(d); \tag{22}$$

see equation (18). It should be noted that the term $\sum_{t=1}^{n} w_T(\sigma(t))\overline{p}(\sigma(t))$ in (22) is independent of the makespan d and is a constant for all d. Therefore, the function $\Phi_\Sigma(d)$ can be easily obtained from the piecewise-linear function $W(d)$ by a simple transformation (22), and it suffices to compute $W(d)$ instead of $\Phi_\Sigma(d)$.

Given a value of d, define a function

$$\psi_t(d) = \tilde{\varphi}(N_t(\sigma), d), \quad 1 \le t \le n. \tag{23}$$

By (20) and (21), the value $W(d)$ is represented as

$$\begin{aligned} W(d) &= \sum_{t=1}^{n} w_T(\sigma(t))(\psi_t(d) - \psi_{t-1}(d)) \\ &= \sum_{t=1}^{n-1} (w_T(\sigma(t)) - w_T(\sigma(t+1)))\psi_t(d) + w_T(\sigma(n))\psi_n(d). \end{aligned} \tag{24}$$

Thus, the piecewise-linear function $W(d)$ can be obtained by first computing the functions $\psi_t(d)$, $1 \le t \le n$, and then computing their weighted sum according to (24). It is shown

in (Shioura et al., 2013, Section 2) that once these functions $\psi_t(d)$, $1 \leq t \leq n$, are found, their weighted sum (24) can be computed in $O(nm \log n)$ time, provided that each function $\psi_t(d)$ has at most $O(m)$ breakpoints. Below we explain how to compute functions $\psi_t(d)$ for all $t = 1, 2, \ldots, n$.

It follows from (16) applied to $X = N_t(\sigma)$ that

$$\begin{aligned}\psi_t(d) &= \min_{Y \in 2^N} \left\{\varphi(Y,d) + \overline{p}(N_t(\sigma) \setminus Y) - \underline{p}(Y \setminus N_t(\sigma))\right\} \\ &= \overline{p}(N_t(\sigma)) + \min_{Y \in 2^N} \left\{\varphi(Y,d) - \overline{p}(N_t(\sigma) \cap Y) - \underline{p}(Y \setminus N_t(\sigma))\right\}. \end{aligned} \tag{25}$$

For the problem $Q|p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$, the value $\varphi(Y,d)$ is given by $\varphi(Y,d) = dS_{m_Y}$ with $m_Y = \min\{m, |Y|\}$; see (9). Using equation (25) and the fact that the value S_{m_Y} takes at most $m+1$ values, we can show that $\psi_t(d)$ is represented as a piecewise-linear (concave) function with $m+1$ pieces. For the other two scheduling problems under consideration, due to (11) and (12), we can still show that $\psi_t(d)$ is represented as a piecewise-linear (concave) function with $m+1$ pieces.

Computing functions $\psi_t(d)$, $1 \leq t \leq n$, for all relevant values of d is a problem dependent procedure. As shown in Shioura et al. (2013), such a procedure requires $O(n \log n + nm)$ time for problem $Q|p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$, and the overall time complexity for solving that problem is $O(nm \log m)$. For problems $\alpha|r(j), p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$ with non-zero release dates, computing functions $\psi_t(d)$, $1 \leq t \leq n$, takes $O\left(n^2 \log m\right)$ time and $O(n^2 m)$ time for $\alpha = P$ and $\alpha = Q$, respectively, and these values determine the running times needed for solving these problems.

We conclude this subsection by considering problem $P|p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$. In principle, it can be solved using the approach outlined above for a more general problem $P|r(j), p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$. However, even if all release dates are zero, we are not aware how to implement the approach faster than in $O(n^2)$ time. A more efficient approach presented in Shakhlevich and Strusevich (2005) uses the submodular optimization reasoning to justify the use of the greedy algorithm and is based on the following property of optimal solutions. For any fixed value d, there is a subset of decompressed jobs $N_t(\sigma)$ with the processing times $\min\{\overline{p}(j), d\}$, while each remaining job $j \in N \setminus N_t(\sigma)$ remains fully compressed, with the processing time $\underline{p}(j)$. Within the set $N \setminus N_t(\sigma)$, the preference for decompression is always given to the jobs with the largest weights $w_T(j)$. It is demonstrated in Shakhlevich and Strusevich (2005) that the solution with the smallest $C_{\max}$-value can be found in $O(n \log n)$ time. Starting from it, each next breakpoint of the efficiency frontier can be constructed in $O(\log n)$ time from the previous one. With the total number of breakpoints bounded by $2n+1$, the overall time complexity of that approach is $O(n \log n)$.

8.2 Single Machine

Below, we briefly review the results on single machine bicriteria SCPT problems.

Problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$ is a special case of the problem $P|p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$ considered in the last subsection, and therefore can be solved in $O(n \log n)$ time; see Shakhlevich and Strusevich (2005).

Problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn|(L_{\max}, \Phi_\Sigma)$, where job $j \in N$ has a due date $d(j)$ (not a deadline) and $L_{\max} = \max\{C(j) - d(j) \,|\, j \in N\}$ is the maximum lateness, is also studied in Shakhlevich and Strusevich (2005). Recall that in scheduling the difference

between the deadlines and the due dates is that the latter can be violated, which is usually associated with a penalty to be paid for a late completion of jobs. The submodular optimization reasoning is applied to justify and develop a version of the greedy algorithm, and the resulting algorithm requires $O\left(n^2\right)$ time.

Consider now problem $1|p(j) = \overline{p}(j) - x(j)|\,(F_{\max}, \Phi_\Sigma)$, where the first objective represents the schedule quality measured in the terms of the maximum processing cost $F_{\max} = \max_{j\in N} f_j(C\,(j))$. For job $j \in N$, function $f_j(C\,(j))$ is a non-decreasing piecewise-linear function that penalizes the completion of job j at time $C\,(j)$ and consists of l_j linear pieces.

This problem is among historically the first SCPT problems (see Van Wassenhove and Baker (1982)), and admits a natural interpretation in terms of the make-or-buy decision-making; see Section 2. Here the machine is seen as the internal production facility. The cost function $f_j(C_j)$ is the work-in-process cost of order j, so that $F_{\max}$ is the processing cost, i.e., represents the maximum cost of processing those orders and their parts that are accepted for internal manufacturing. The other objective function Φ_Σ expresses the total subcontracting cost.

The algorithm presented in Shakhlevich et al. (2009) combines the reformulations in terms of linear programming problems over parametric submodular polyhedra with computational geometry techniques. It starts with a pre-processing step, that requires $O(nL\log n)$ time with $L = \sum_{j\in N} l_j$, and is aimed at splitting the whole range of possible values of $f_j(C_j)$ into intervals $[y_{\ell-1}, y_\ell]$ such that within each interval the functions $f_j(t)$ do not intersect and do not change their linear shape. Such a splitting ensures that for every interval the job sequence is fixed, and the approach similar to that outlined in Section 8.1 is applicable to each of the $O(nL)$ intervals of the form $[y_{\ell-1}, y_\ell]$. Since the time complexity of finding $\Phi_\Sigma(d)$ in a single interval is $O(n^2)$ and the total number of the relevant intervals is $O(nL)$, the overall time complexity is $O(n^3L)$.

It is clear that the algorithm for solving problem $1|p(j) = \overline{p}(j) - x(j)|\,(F_{\max}, \Phi_\Sigma)$ delivers an optimal solution to a single criterion problem of minimizing one of the objectives $F_{\max}$ or Φ_Σ, provided that the other one is bounded. However, as shown in Shakhlevich et al. (2009), these single criterion problems can be solved faster by specialized algorithms. The problem of minimizing the total compression cost Φ_Σ subject to a bounded maximum processing cost $F_{\max}$ requires $O\,(n\log n + \lambda)$ time with $\lambda = \sum_{j\in N} \log l_j$. On the other hand, minimizing the maximum processing cost $F_{\max}$ subject to a bounded total compression cost Φ_Σ takes $O\left(L + n^2 + (\lambda + n\log n)\log L\right)$ time. If each l_j is bounded by a constant, the above estimates reduce to $O\,(n\log n)$ and $O\left(n^2 + n\log^2 n\right)$, respectively.

The summary of the results on the bicriteria problems and their comparison are presented in Table 4.

9 Methodology 3: Submodular Optimization via Decomposition Algorithm

Due to Theorems 3 and 4, Problem (LP) can be solved by a greedy algorithm in at most n iterations, each of which involves minimization of a submodular function. In this section, we present a recursive decomposition algorithm that solves Problem (LP) with a depth of recursion $O\,(\log n)$. The decomposition algorithm can be adapted to solving several SCPT problem to minimize the total compression cost. Our presentation of the decomposition algorithm is based on Shioura et al. (2015, 2016a).

Our algorithm is different from a well-known decomposition algorithm from Fujishige

Problem	Previously known	Methodology 2
$1\|r(j), p(j) = \overline{p}(j) - x(j),$ $pmtn\|(C_{\max}, \Phi_\Sigma)$	N/A	$O(n \log n)$ Shakhlevich and Strusevich (2005)
$1\|r(j), p(j) = \overline{p}(j) - x(j),$ $pmtn\|(L_{\max}, \Phi_\Sigma)$	N/A	$O(n^2)$ Shakhlevich and Strusevich (2005)
$1\|p(j) = \overline{p}(j) - x(j)\|$ $(\max_{j\in N} f_j(C(j)), \Phi_\Sigma)$	$O(n^4L^2)^*$ Hoogeveen and Woeginger (2001)	$O(n^3L)^*$ Shakhlevich et al. (2009)
$P\|p(j) = \overline{p}(j) - x(j),$ $pmtn\|(C_{\max}, \Phi_\Sigma)$	$O(n^2)$ Nowicki and Zdrzałka (1995)	$O(n \log n)$ Shakhlevich and Strusevich (2005)
$P\|r(j), p(j) = \overline{p}(j) - x(j),$ $pmtn\|(C_{\max}, \Phi_\Sigma)$	N/A	$O(n^2 \log m)$ Shioura et al. (2013)
$Q\|p(j) = \overline{p}(j) - x(j),$ $pmtn\|(C_{\max}, \Phi_\Sigma)$	$O(n \log n + nm^4)$ Shakhlevich and Strusevich (2008)	$O(nm \log m)$ Shioura et al. (2013)
$Q\|r(j), p(j) = \overline{p}(j) - x(j),$ $pmtn\|(C_{\max}, \Phi_\Sigma)$	N/A	$O(n^2 m)$ Shioura et al. (2013)

* L is the total number of pieces of of all functions f_j, $j \in N$

Table 4: Complexity of Bicriteria Problems

(1980, 2005) which minimizes a separable convex function over a base polyhedron. Even for a linear objective function, the depth of recursion of Fujishige's algorithm in the worst case is n. A detailed comparison of Fujishige's algorithm and our decomposition algorithm is provided in Shioura et al. (2015).

Given Problem (LP) of the form (15), a subset $\hat{N} \subseteq N$ is called a *heavy-element subset of N with respect to the weight vector* $\mathbf{w}$ if it satisfies the condition

$$\min_{j\in\hat{N}} w_T(j) \geq \max_{j\in N\setminus\hat{N}} w_T(j).$$

For completeness, we also regard the empty set as a heavy-element subset of N. For a given set $X \subseteq N$, in accordance with (16) define a set $Y_* \subseteq N$ such that the equality

$$\tilde{\varphi}(X) = \varphi(Y_*) + \overline{p}(X \setminus Y_*) - \underline{p}(Y_* \setminus X) \tag{26}$$

holds. In the remainder of this paper, we call Y_* an *instrumental* set for set X.

The statement below explains an important role that the instrumental set plays in solving Problem (LP).

Lemma 5 (cf. Shioura et al. (2015, 2016a)) *Let $\hat{N} \subseteq N$ be a heavy-element subset of N with respect to $\mathbf{w}$, and $Y_* \subseteq N$ be an instrumental set for set $\hat{N}$. Then there exists an optimal solution $\mathbf{p}^*$ of Problem (LP) such that*

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

In what follows, we use two fundamental operations on a submodular system $\left(2^N, \varphi\right)$, as defined in (Fujishige, 2005, Section 3.1). For a set $A \in 2^N$, define a set function $\varphi^A : 2^A \to \mathbb{R}$ by $\varphi^A(X) = \varphi(X),\ X \in 2^A$. Then, $(2^A, \varphi^A)$ is a submodular system on A and it is called a *restriction of* $(2^N, \varphi)$ *to* A. On the other hand, for a set $A \in 2^N$ define a set function $\varphi_A : 2^{N\setminus A} \to \mathbb{R}$ by $\varphi_A(X) = \varphi(X \cup A) - \varphi(A),\ X \in 2^{N\setminus A}$. Then, $(2^{N\setminus A}, \varphi_A)$ is a submodular system on $N \setminus A$ and it is called a *contraction of* $(2^N, \varphi)$ *by* A.

Theorem 6 (cf. Shioura et al. (2015, 2016a)) *Let $\hat{N} \subseteq N$ be a heavy-element subset of N with respect to $\mathbf{w}$, and Y_* be an instrumental set for set $\hat{N}$. Let $\mathbf{p_1} \in \mathbb{R}^{Y^*}$ and $\mathbf{p_2} \in \mathbb{R}^{N \setminus Y^*}$ be optimal solutions of the linear programs (LPR) and (LPC), respectively, given by*

$$\begin{array}{lll}
\text{(LPR)}: & \max & \displaystyle\sum_{j \in Y_*} w(j)p(j) \\
& \text{s.t.} & p(X) \le \varphi(X), \qquad X \in 2^{Y_*}, \\
& & \underline{p}(j) \le p(j) \le \overline{p}(j), \quad j \in Y_* \cap \hat{N}, \\
& & p(j) = \underline{p}(j), \qquad j \in Y_* \setminus \hat{N}, \\
\text{(LPC)}: & \max & \displaystyle\sum_{j \in N \setminus Y_*} w(j)p(j) \\
& \text{s.t.} & p(X) \le \varphi(X \cup Y_*) - \varphi(Y_*), \quad X \in 2^{N \setminus Y_*}, \\
& & \underline{p}(j) \le p(j) \le \overline{p}(j), \qquad j \in (N \setminus Y_*) \setminus \left(\hat{N} \setminus Y_*\right), \\
& & p(j) = \overline{p}(j), \qquad j \in \hat{N} \setminus Y_*.
\end{array}$$

Then, the vector $\mathbf{p}^* \in \mathbb{R}^N$ given by the direct sum $\mathbf{p}^* = \mathbf{p_1} \oplus \mathbf{p_2}$, where

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

is an optimal solution of Problem (LP).

Notice that Problem (LPR) is obtained from Problem (LP) as a result of restriction to Y_* and the values of components $p(j),\ j \in Y_* \setminus \hat{N}$, are fixed to their lower bounds in accordance with Property (c) of Lemma 5. Similarly, Problem (LPC) is obtained from Problem (LP) as a result of contraction by Y_* and the values of components $p(j),\ j \in \hat{N} \setminus Y_*$, are fixed to their upper bounds in accordance with Property (b) of Lemma 5.

Now we explain how the original problem (LP) can be decomposed recursively based on Theorem 6, until we obtain a collection of trivially solvable problems with no non-fixed variables. As described in Shioura et al. (2015, 2016a), in each stage of the recursive procedure, we need to solve a subproblem that can be written in the following generic form:

$$\begin{array}{lll}
\text{LP}(H, F, K, \mathbf{l}, \mathbf{u}): & \max & \displaystyle\sum_{j \in H} w_T(j)p(j) \\
& \text{s.t.} & p(X) \le \varphi_K^H(X) = \varphi(X \cup K) - \varphi(K), \quad X \in 2^H, \\
& & l(j) \le p(j) \le u(j), \qquad j \in H \setminus F, \\
& & p(j) = u(j) = l(j), \qquad j \in F,
\end{array} \tag{27}$$

where $H \subseteq N$ is the index set of components of vector $\mathbf{p}$; $\mathbf{l} = (l(j) \mid j \in H)$ and $\mathbf{u} = (u(j) \mid j \in H)$ are, respectively, the current vectors of the lower and upper bounds on variables $p(j), j \in H$; $F \subseteq H$ is the index set of fixed components, i.e., $l(j) = u(j)$ holds for each $j \in F$; $K \subseteq N \setminus H$ is the set that defines the rank function $\varphi_K^H : 2^H \to \mathbb{R}$ such that $\varphi_K^H(X) = \varphi(X \cup K) - \varphi(K)$, $X \in 2^H$.

Suppose that Problem $\text{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (27) contains at least one non-fixed variable, i.e., $|H \setminus F| > 0$. We define a function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ by

$$\widetilde{\varphi}_K^H(X) = \min_{Y \in 2^H} \{\varphi_K^H(Y) + u(X \setminus Y) - l(Y \setminus X)\}. \tag{28}$$

By Theorem 2, the set of maximal feasible solutions of Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ is given as a base polyhedron $B(\widetilde{\varphi}_K^H)$ associated with the rank function $\widetilde{\varphi}_K^H$. Therefore, if $|H \setminus F| = 1$ and $H \setminus F = \{j'\}$, then an optimal solution $\mathbf{p}^* \in \mathbb{R}^H$ is given by

$$p^*(j) = \begin{cases} \widetilde{\varphi}_K^H(\{j'\}) & (\text{if } j = j'), \\ u(j) & (\text{if } j \in F). \end{cases} \tag{29}$$

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

$$\widetilde{\varphi}_K^H(\hat{H}) = \varphi_K^H(Y_*) + u(\hat{H} \setminus Y_*) - l(Y_* \setminus \hat{H}). \tag{30}$$

Without going into implementation details, we follow Shioura et al. (2015, 2016a) and give a formal description of the recursive procedure. For the current problem LP$(H, F, K,$ $\mathbf{l}$, $\mathbf{u})$, we compute optimal solutions $\mathbf{p_1} \in \mathbb{R}^{Y_*}$ and $\mathbf{p_2} \in \mathbb{R}^{H \setminus Y_*}$ of the two subproblems by calling Procedures DECOMP$(Y_*, F_1, K, \mathbf{l_1}, \mathbf{u_1})$ and DECOMP$(H \setminus Y_*,\ F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$. By Theorem 6, the direct sum $\mathbf{p}^* = \mathbf{p}_1 \oplus \mathbf{p_2}$ is an optimal solution of Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$, which is the output of Procedure DECOMP$(H, F, K, \mathbf{l}, \mathbf{u})$.

Procedure Decomp$(H, F, K, \mathbf{l}, \mathbf{u})$

Step 1. If $|H \setminus F| = 0$, then output the vector $\mathbf{p}^* = \mathbf{u} \in \mathbb{R}^H$ and return.
If $|H \setminus F| = 1$ and $H \setminus F = \{j'\}$, then compute the value $\widetilde{\varphi}_K^H(\{j'\})$, output the vector $\mathbf{p}^*$ given by (29) and return.

Step 2. Select a heavy-element subset $\hat{H}$ of $H \setminus F$ with respect to $\mathbf{w}$, and determine an instrumental set $Y_* \subseteq H$ for set $\hat{H}$ satisfying (30).

Step 3. Define the vectors $\mathbf{l_1}, \mathbf{u_1} \in \mathbf{R}^{Y_*}$ and set F_1 by

$$l_1(j) = l(j),\ j \in Y_*, \qquad u_1(j) = \begin{cases} l(j), & j \in Y_* \setminus \hat{H}, \\ u(j), & j \in Y_* \cap \hat{H}, \end{cases} \quad ; \quad F_1 = Y_* \setminus \hat{H}.$$

Call Procedure DECOMP$(Y_*, F_1, K, \mathbf{l_1}, \mathbf{u_1})$ to obtain an optimal solution $\mathbf{p_1} \in \mathbb{R}^{Y_*}$ of Problem LP$(Y_*, F_1, K, \mathbf{l_1}, \mathbf{u_1})$.

Step 4. Define the vectors $\mathbf{l_2}, \mathbf{u_2} \in \mathbf{R}^{H \setminus Y_*}$ and set F_2 by

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

Call Procedure DECOMP$(H \setminus Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$ to obtain an optimal solution $\mathbf{p_2} \in \mathbb{R}^{H \setminus Y_*}$ of Problem LP$(H \setminus Y_*, F_2, K \cup Y_*, \mathbf{l_2}, \mathbf{u_2})$.

Step 5. Output the direct sum $\mathbf{p}^* = \mathbf{p_1} \oplus \mathbf{p_2} \in \mathbb{R}^H$ and return.

The original problem (LP) is solved by calling Procedure DECOMP$(N, \emptyset, \emptyset,\ \underline{\mathbf{p}}, \overline{\mathbf{p}})$. Its actual running time depends on the choice of a heavy-element subset $\hat{H}$ in Step 2 and on the time complexity of finding an instrumental set Y_*. As proved in Shioura et al. (2015),

if at each level of recursion a heavy-element set is chosen to contain roughly a half of the non-fixed variables, then the overall depth of recursion of Procedure DECOMP applied to Problem LP$(N, \emptyset, \emptyset, \underline{\mathbf{p}}, \overline{\mathbf{p}})$ is $O(\log n)$.

For a typical iteration of Procedure DECOMP applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ with $|H| = h$ and $|H \setminus F| = g$, let $T_{Y_*}(h)$ denote the running time for computing the value $\widetilde{\varphi}_K^H(\hat{H})$ for a given set $\hat{H} \subseteq H$ and finding an instrumental set Y_* in Step 2. In Steps 3 and 4, Procedure DECOMP splits Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ into two subproblems: one with h_1 variables among which $g_1 \leq \min\{h_1, \lceil g/2 \rceil\}$ variables are not fixed, and the other one with $h_2 = h - h_1$ variables, among which $g_2 \leq \min\{h_2, \lfloor g/2 \rfloor\}$ variables are not fixed. Let $T_{\text{Split}}(h)$ denote the time complexity for setting up the instances of these two subproblems. It is shown in Shioura et al. (2015, 2016a) that Problem (LP) can be solved by Procedure DECOMP in $O((T_{Y_*}(n) + T_{\text{Split}}(n)) \log n)$ time.

10 Total Cost. Decomposition Algorithm

In this section, we demonstrate the power of Methodology 3 by adapting the decomposition algorithm from Section 9 to solving the SCPT problems with the total compression cost objective. We split our consideration into two parts, depending on a particular way of finding an instrumental set Y_*.

10.1 Parallel Machines. Common Deadline

In this subsection, we assume that all jobs have a common deadline. We exclude from consideration problem $P|p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$, since this problem, the simplest of the range under consideration, admits a linear time algorithm. Indeed, as pointed out in Jansen and Mastrolilli (2004), the problem reduces to the continuous knapsack problem.

Thus, in this subsection, we focus on the problems $Q|p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$ and $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$ with $\alpha \in \{P, Q\}$. Of course, each of these problems can be solved by adapting an output of the corresponding algorithm for the relevant bicriteria problem; see Section 8. However, as shown below, each of these problems can be solved faster by applying the decomposition algorithm from Section 9. The material is this subsection is based on Shioura et al. (2015).

In accordance with Theorem 4, each of these three problems reduces to Problem (LP). The corresponding rank functions are given by (9) for problem $Q|p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$, by (12) for problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$ and by (11) for problem $Q|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$.

In line with the decomposition algorithm for Problem (LP), take an initial Problem LP$(N, \emptyset, \emptyset, \mathbf{l}, \mathbf{u})$, associated with one of the three scheduling problems above, where $l(j) = \underline{p}(j)$ and $u(j) = \overline{p}(j)$, $j \in N$. Assume that the following preprocessing is done in $O(n \log n)$ time before calling Procedure DECOMP$(N, \emptyset, \emptyset, \mathbf{l}, \mathbf{u})$: the jobs are numbered in non-decreasing order of their release dates in accordance with (6); the machines are numbered in non-increasing order of their speeds in accordance with (1), and the partial sums S_v are computed for all v, $0 \leq v \leq m$, by (7); the lists $(l(j) \mid j \in N)$ and $(u(j) \mid j \in N)$ are formed and their elements are sorted in non-decreasing order.

For each of the three problems under consideration, the rank functions are relatively simple, so that the instrumental set Y_* can be found directly, as a minimizer of a certain submodular function. In a typical iteration of Procedure DECOMP applied to Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (27) with the rank function $\varphi_K^H(Y) = \varphi(Y \cup K) - \varphi(K)$, it is

shown in Shioura et al. (2015) that for a given set $X \subseteq H$ the function $\widetilde{\varphi}_K^H : 2^H \to \mathbb{R}$ can be computed as

$$\widetilde{\varphi}_K^H(X) = u(X) - \varphi(K) + \min_{Y \in 2^H} \{\varphi(Y \cup K) - b(Y)\}, \tag{31}$$

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

$$b(j) = \begin{cases} u(j), & \text{if } j \in X, \\ l(j), & \text{if } j \in H \setminus X. \end{cases} \tag{32}$$

Notice that if the minimum in the right-hand side of (31) is achieved for $Y = Y_*$, then Y_* is an instrumental set for set X.

To illustrate this, consider, e.g., problem $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|\Phi_\Sigma$. For Problem LP$(H, F, K, \mathbf{l}, \mathbf{u})$ associated with that problem it follows from (9) and (31) that

$$\widetilde{\varphi}_K^H(X) = u(X) - dS_{\min\{m,k\}} + \min\{\Phi', \Phi''\}, \tag{33}$$

where $k = |K|$,

$$\Phi' = \begin{cases} \min\limits_{0 \leq v \leq \min\{h, m-k-1\}} \{dS_{v+k} - \sum\limits_{i=1}^{v} b_i\}, & \text{if } m > k, \\ +\infty, & \text{otherwise,} \end{cases}$$

with b_i being the i-th largest value in the list $(b(j) \mid j \in H)$, and

$$\Phi'' = \begin{cases} dS_m - b(H), & \text{if } h > m - k - 1, \\ +\infty, & \text{otherwise.} \end{cases}$$

In any case, in terms of the notions introduced in Section 9 we deduce that $T_{Y_*}(h) = T_{\text{Split}}(h) = O(h)$, so that the overall running time needed to solve problem $Q|p(j) = \overline{p}(j) - x(j), pmtn, C(j) \leq d|\Phi_\Sigma$ by the decomposition algorithm based on recursive applications of Procedure DECOMP is $O(n \log n)$. An alternative implementation of the same approach, also presented in Shioura et al. (2015), does not involve a full preprocessing and requires $O(n + m \log m \log n)$ time.

When Methodology 3 is applied to problems $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$ and $Q|r(j),\ p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$, the decomposition algorithm can be implemented in $O(n \log n \log m)$ time and in $O(mn \log n)$ time, respectively.

The summary of the results for the single criterion parallel machine problems with a common deadline is presented in Table 5.

10.2 Single Machine

Problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_\Sigma$ for many years has been an object of intensive study, mainly within the body of research on SIC. The history of studies on this problem is a race for developing an $O(n \log n)$-time algorithm, matching the best possible estimate of $O(n \log n)$ achieved for a simpler feasibility problem $1|r(j), C(j) \leq d(j), pmtn|-$, see Table 6.

The time complexity of problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_\Sigma$ is finally settled in Shioura et al. (2016a), where an $O(n \log n)$-time algorithm is produced using Methodology 3. The algorithm is based on the decomposition algorithm for Problem (LP)

Problem	Previously known	Methodology 3: Decomposition Algorithm
$P\|r(j), p(j) = \overline{p}(j) - x(j),$ $C(j) \le d, pmtn\|\Phi_\Sigma$	$O(n)$ Jansen and Mastrolilli (2004)	N/A
$P\|r(j), p(j) = \overline{p}(j) - x(j),$ $C(j) \le d, pmtn\|\Phi_\Sigma$	$O(n^4 \log n)^{*,\ \#}$ Blazewicz and Finke (1987) Chung et al. (1989) Shih et al. (1989, 1991) Leung (2004) $O(n^2 \log m)^{\dagger}$ Shioura et al. (2013)	$O(n \log n \log m)$ Shioura et al. (2015)
$Q\|p(j) = \overline{p}(j) - x(j),$ $C(j) \le d, pmtn\|\Phi$	$O(mn + n \log n)$ Nowicki and Zdrzałka (1995) Shakhlevich and Strusevich (2008) $O(mn \log m)^{\dagger}$ Shioura et al. (2013)	$O(\min\{n \log n,$ $n + m \log m \log n\})$ Shioura et al. (2015)
$Q\|r(j), p(j) = \overline{p}(j) - x(j),$ $C(j) \le d, pmtn\|\Phi$	$O(mn^4)^{*}$ Wan et al. (2007) Shakhlevich and Strusevich (2008) $O(mn^2)^{\dagger}$ Shioura et al. (2013)	$O(mn \log n)$ Shioura et al. (2015)

* Derived for the problem with arbitrary deadlines

After correcting a faulty claim that problem $P|r(j), C(j) \le d(j), pmtn|-$ is solvable in $O(n^2 \log^2 n)$ time, see Remark 1 of Section 5

† Methodology 2: Bicriteria problems via submodular optimization, Section 8

Table 5: Complexity of Problems with a Common Deadline

and uses an algorithm from Hochbaum and Shamir (1990) as a subroutine for solving auxiliary problems with the unweighted penalty function $\Phi_{\mathrm{u}} = \sum x(j)$.

The efficient implementation of the decomposition algorithm developed in Shioura et al. (2016a) is based on the following statement.

Theorem 7 (cf. (Fujishige, 2005, Corollary 3.4)) *For a submodular system $(2^H, \varphi)$ and a vector $\mathbf{b} \in \mathbb{R}^H$, the equality*

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

holds. In particular, if $\mathbf{b} \ge \mathbf{0}$ and $\varphi(X) \ge 0$ for all $X \subseteq N$ then the right-hand side is equal to $\max\{p(H) \mid \mathbf{p} \in P(\varphi),\ \mathbf{0} \le \mathbf{p} \le \mathbf{b}\}$.

Given Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (27), for a set $X \subseteq H$ define the vector

Problem	Previously known	Methodology 3 Decomposition Algorithm
$1\|r(j), p(j) = \overline{p}(j) - x(j),$ $C(j) \leq d(j), pmtn\|\Phi_\Sigma$	$O(n^2 \log^2 n)$ Chung et al. (1989), Shih et al. (1989) $O(n^2)$ for $\Phi_\Sigma = \sum w_T(j)x(j)$ $O(n \log n)$ for $\Phi_{\mathrm{u}} = \sum x(j)$ Hochbaum and Shamir (1990) $O(n \log n + \kappa n)^*$ Leung et al. (1994) $O(n \log^2 n)^\#$ Shih et al. (2000)	$O(n \log n)$ Shioura et al. (2016a)

* κ is the number of distinct weights $w_T(j)$

\# for integer input data

Table 6: Results for the Single Machine Problem

$\mathbf{b} \in \mathbb{R}^H$ by (32), and for a set $X \subseteq H$ represent $\tilde{\varphi}_K^H(X)$ in the form

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

Since $-l(H \setminus X)$ is a constant, in order to find an instrumental set Y_* that defines $\tilde{\varphi}_K^H(X)$ it suffices to find a minimizer for $\min_{Y \in 2^H} \{\varphi_K^H(Y) + b(H \setminus Y)\}$. By Theorem 7, the latter minimization problem is equivalent to the following auxiliary problem:

$$\begin{aligned} (\text{AuxLP}): \quad \max \quad & \sum_{j \in H} q(j) \\ \text{s.t.} \quad & q(Y) \leq \varphi_K^H(Y), \ Y \in 2^H; \\ & 0 \leq q(j) \leq b(j), \ j \in H. \end{aligned} \tag{34}$$

Let $\mathbf{q}_* \in \mathbb{R}^H$ be an optimal solution to Problem (AuxLP) with the values $b(j)$ defined with respect to a set $X \subseteq H$. It is proved in Shioura et al. (2016a) that a set Y_* is an instrumental set that defines $\tilde{\varphi}_K^H(X)$ if and only if

$$q_*(Y_*) = \varphi_K^H(Y_*); \qquad q(j) = b(j), \ j \in H \setminus Y_*.$$

Problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_\Sigma$ reduces to Problem (LP) with the rank function $\varphi = \varphi_1$ defined by (5). Consider a typical iteration of Procedure DECOMP applied to Problem $\mathrm{LP}(H, F, K, \mathbf{l}, \mathbf{u})$ of the form (27) related to the rank function $\varphi_K^H(Y) = \varphi(Y \cup K) - \varphi(K)$. For a set $X \subseteq H$ of jobs, a meaningful interpretation of $\varphi_K^H(X)$ is the total length of the time intervals originally available for processing the jobs of set $X \cup K$ after the intervals for processing the jobs of set K have been completely used up.

Select a heavy-element set $\hat{H}$ and define the values $b(j)$ by (32) applied to $X = \hat{H}$. Our goal is to find an instrumental set Y_* for set $\hat{H}$. As described above, for this purpose we may solve the auxiliary Problem (AuxLP).

Problem (AuxLP) can be seen as a version of a scheduling problem $1|r(j), q(j) = b(j) - x(j), C(j) \leq d(j), pmtn|\sum x(j)$, in which it is required to determine the actual processing times $q(j)$ of jobs of set H to maximize the total (unweighted) actual processing time, provided that $0 \leq q(j) \leq b(j)$ for each $j \in H$. It can be solved by an algorithm developed by Hochbaum and Shamir (1990), which uses the UNION-FIND technique and finds the actual processing times of all jobs and the corresponding optimal schedule in $O(h)$ time, provided that the jobs are renumbered in non-increasing order of their release dates. The algorithm is based on the latest-release-date-first rule. Informally, the jobs are taken one by one in the order of their numbering and each job $j \in H$ is placed into the current partial schedule to fill the available time intervals consecutively, from right to left, starting from the right-most available interval. The assignment of a job j is completed either if its actual processing time $q(j)$ reaches its upper bound $b(j)$ or if no available interval is left. Only a slight modification of the Hochbaum-Shamir algorithm is required to find not only the optimal values $q_*(j)$ of the processing times, but also an associated instrumental set. The running time of the modified algorithm is still $O(h)$.

In terms of the notions introduced in Section 9 we deduce that $T_{Y_*}(h) = T_{\text{Split}}(h) = O(h)$, so that the overall running time needed to solve problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_\Sigma$ by the decomposition algorithm based on recursive applications of Procedure DECOMP is $O(n \log n)$.

We conclude this section by reviewing the results for the special case of problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_\Sigma$ with a common due date d, which is probably one of the most studied SCPT problems; see, e.g., Vikson (1980), Nowicki and Zdrzałka (1990), Janiak and Kovalyov (1996), Hoogeveen and Woeginger (2001) and Shakhlevich and Strusevich (2005). Problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$ is known to be solvable in $O(n \log n)$ time. The algorithms by Janiak and Kovalyov (1996), Hoogeveen and Woeginger (2001) and Shakhlevich and Strusevich (2005) are justified by a schedule-based reasoning and the running time of $O(n \log n)$ is achieved by using special data structures, such as heaps or 2-3-trees. On the other hand, Methodology 2 delivers the same result for problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_\Sigma$ as a direct consequence of the fact that the bicriteria problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn|(C_{\max}, \Phi_\Sigma)$ is solvable in $O(n \log n)$ time; see Shakhlevich and Strusevich (2005) and Section 8.2.

11 Maximum Cost

In this section, we consider problems $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_{\max}$, $\alpha \in \{1, P, Q\}$, of minimizing the function $\Phi_{\max} = \max_{j \in N}\{x(j)/w_M(j)\}$, where $w_M(j)$ are positive weights. Problems of this type have been extensively studied in the SIC literature; see Ho (2004) and Wan et al. (2007) for reviews; see also Table 7. In the discussion in the forthcoming sections we may use the SIC terminology, i.e., to refer to jobs as tasks and to the compression costs (total or maximum) as errors.

Similarly to Section 6, below we provide a critical review of the earlier results by (i) clarifying the running time needed to solve problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_{\max}$ based on Remark 1 of Section 5; (ii) demonstrating that using advanced techniques of Methodology 1, such as solving the flow sharing problems by parametric flows methods, results into solution algorithms for problems $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_{\max}$ which have the same running times as the best algorithms known for the counterparts of these problems with fixed processing times; see Table 1.

Problem	Previously known	Methodology 1: Parametric flow
$1\|r(j), p(j) = \overline{p}(j) - x(j),$ $C(j) \leq d(j), pmtn\|\Phi_{\max}$	$O(n^2)$ Ho et al. (1994) Ho (2004)	N/A
$P\|r(j), p(j) = \overline{p}(j) - x(j),$ $C(j) \leq d(j), pmtn\|\Phi_{\max}$	$O(n^4)^*$ Ho et al. (1994) Ho (2004)	$O(n^3)$ Section 11
$Q\|p(j) = \overline{p}(j) - x(j),$ $C(j) \leq d(j), pmtn\|\Phi_{\max}$	$O(m^2n^4 \log mn)$ Wan et al. (2007)	$O(mn^3)$ Section 11

*After correcting a faulty claim that problem $P|r(j), C(j) \leq d(j), pmtn|-$ is solvable in $O(n^2 \log^2 n)$ time, see Remark 1 of Section 5

Table 7: Complexity of Problems with Maximum Cost

Note that for identical parallel machines, the best algorithm known within SIC implements an idea of an appropriate redistribution of the minimum total compression amount $\sum_{j \in N} x(j)$; see Ho et al. (1994) and Leung et al. (1994). The algorithm is iterative, and each of its n steps requires finding the minimum total cost for a modified system of tasks. It is claimed that such a step can be done in $O(n^2 \log^2 n)$ time, by solving an appropriate problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\sum x(j)$ with the unweighted total cost function. However, as follows from our consideration in Section 6, solving such a problem requires $O(n^3)$ time, even for the unweighted case. Thus, in accordance with Remark 1 of Section 5, we deduce that the previously known approaches are able to solve problem $P|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_{\max}$ only in $O(n^4)$ time, not in $O(n^3 \log^2 n)$ time, as claimed in Ho et al. (1994) and Ho (2004). For problem $1|r(j),\ p(j) = \overline{p}(j) - x(j),$ $C(j) \leq d(j),\ pmtn|\Phi_{\max}$ the algorithm in Ho (2004) requires $O\left(n^2\right)$ time and remains the fastest.

In the case of uniform machines, the best known algorithm for problem $Q|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_{\max}$ is due to Wan et al. (2007). The algorithm requires $O(mn^4)$ time and is based on the algorithm for problem $Q|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \leq d(j),\ pmtn|\Phi_{\Sigma}$ by Shakhlevich and Strusevich (2008).

We now present more efficient algorithms for parallel machines that are based on Methodology 1, in particular on solving the *flow sharing* problems in networks H_P and H_Q, which are solved by parametric maximum flow algorithms, as described in Gallo et al. (1989).

In the flow sharing problems, we consider a network that is structurally similar to network H_P or H_Q, introduced in Section 6, where each arc $a = (v, t) \in A^t$ entering the sink t has a positive weight $w(a)$, and it is required to find a maximum flow f that guarantees certain properties of the ratios $f(a)/w(a)$. In particular, for our purposes we are interested in three versions of the flow sharing problems:

- *minimax sharing*: find a maximum flow $f(a)$, $a \in A^t$, that minimizes the largest ratio $f(a)/w(a)$;
- *lexicographic sharing*: find a maximum flow $f(a)$, $a \in A^t$, such that the sequence of the ratios $f(a)/w(a)$, $a \in A^t$, arranged in the *non-decreasing* order

$$\frac{f(a_1)}{w(a_1)} \leq \frac{f(a_2)}{w(a_2)} \leq \cdots \leq \frac{f(a_g)}{w(a_g)}$$

is lexicographically *maximum* (here $g = |A^t|$);

- *co-lexicographic sharing*: find a maximum flow $f(a)$, $a \in A^t$, such that the sequence of the ratios $f(a)/w(a)$, $a \in A^t$, arranged in the *non-increasing* order

$$\frac{f(a_1)}{w(a_1)} \geq \frac{f(a_2)}{w(a_2)} \geq \cdots \geq \frac{f(a_g)}{w(a_g)}$$

is lexicographically *minimum*.

Clearly, an optimal solution to the co-lexicographic sharing problem delivers an optimal solution to the minimax sharing problem, although the converse does not necessarily hold. It is known (cf. (Fujishige, 2005, Section 9.1)) that the lexicographic and the co-lexicographic sharing problems are equivalent; more precisely, a maximum flow is an optimal solution to the lexicographic sharing problem if and only if it is an optimal solution to the co-lexicographic sharing problem. Hence, in the remainder of this paper, we will use the term "the lexicographic sharing problem" to refer to the co-lexicographic sharing problem.

For $\alpha \in \{P, Q\}$, take a network H_α and for each arc $a_j = (X_j, t)$, $j \in N$, define the weight $w(a_j) = w_M(j)$; for all other arcs $a \in A^t$ entering t prescribe infinitely large weights $w(a)$. Then a solution to problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_{\max}$ can be derived from a solution to either the minimax sharing problem or the lexicographic sharing problem for the network H_α introduced above.

As demonstrated by Gallo et al. (1989), both problems can be reduced to finding a parametric maximum flow. In the case of network H_α, define the capacity of each arc $a_j = (X_j, t)$, $j \in N$, to be equal to $w(a_j)\lambda$, where λ is a non-negative parameter. Below we remind how that reduction works for the lexicographic sharing problem, since (i) solving this problem suffices for solving the associated problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_{\max}$, and (ii) as shown later in Section 13, a solution to the lexicographic sharing problem also helps solving problems $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_{\text{quad}}$ with a quadratic cost function.

Suppose that $f(a)$, $a \in A$, is a maximum flow in network H_α with the modified capacities of the arcs entering the sink t. Let $\kappa(\lambda)$ represent the capacity of a minimum cut, as a function of λ. It follows from Gallo et al. (1989) that $\kappa(\lambda)$ is a piecewise-linear function of λ and has n breakpoints, one for each arc $a_j = (X_j, t)$, $j \in N$. Moreover, Gallo et al. (1989) present an algorithm that finds all these breakpoints. For an arc a_j, let λ_j be the breakpoint at which node X_j moves from the source side of a minimum cut to the sink side. Change the capacity of each arc $a_j = (X_j, t)$, $j \in N$, to $w(a_j)\lambda_j$, and find a maximum flow f^* in the resulting network. It is proved by Gallo et al. (1989), that flow f^* solves the lexicographic sharing problem.

Once the flow f^* is found, it determines an optimal solution to problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_{\max}$. For $j \in N$, the flow $f^*(s, j)$ on the arc (s, j) entering the job-node j determines the value $p(j)$, the actual processing time of job j, while the flow $f^*(X_j, t)$ on the arc (X_j, t) entering the sink determines $x(j)$, the compression amount of job j. Similarly to Section 6, for network H_P the flow on an arc (j, I_h) defines for how long job j is processed in the time interval I_h, while for network H_Q the flow on an arc $(j, (I_h, M_i))$ defines for how long job j is processed in the time interval I_h on machine M_i.

Since network H_α, $\alpha \in \{P, Q\}$, is bipartite, the techniques by Ahuja et al. (1994) can be used to speed up the algorithm by Gallo et al. (1989); see row 4 of Table 2. Thus, problem $P|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_{\max}$ can be solved in $O(n^3)$ time, and

problem $Q|r(j),\ p(j)=\overline{p}(j)-x(j),\ C(j)\leq d\,(j)\,,\ pmtn|\Phi_{\max}$ in $O(mn^3)$ time, as stated in Table 7.

12 Lexicographic Minimization of Total Cost and Maximum Cost Criteria

In this section, we consider SCPT problems of lexicographical minimization. Each job $j\in N$ is associated with two weights: $w_T(j)$ that defines the total compression cost $\Phi_\Sigma=\sum w_T(j)x(j)$ and $w_M(j)$ that determines the maximum compression cost $\Phi_{\max}=\max\{x(j)/w_M(j)\}$. Problems of this range are known in the SIC literature as the 'doubly weighted' problems. Under the SIC interpretation Φ_Σ is called the total error, while $\Phi_{\max}$ is called the maximum error. A review of the previously known results on the doubly weighted SIC problems is contained in Ho (2004), where the focus is on the lexicographically ordered objective functions. One of these objectives (the primary function Φ_1) is minimized and then the minimum of the other objective (the secondary function Φ_2) is sought among the schedules that are optimal with respect to the primary function. To denote the problems of lexicographic minimization, in the three-field scheduling notation we write $Lex\,(\Phi_1,\Phi_2)$.

As in Sections 6 and 11, below we present improved algorithms for identical parallel machines and uniform parallel machines obtained by advanced techniques of Methodology 1. In particular, we show that the doubly-weighted problems with the lexicographically ordered objectives $Lex(\Phi_{\max},\Phi_\Sigma)$ and $Lex(\Phi_\Sigma,\Phi_{\max})$ can be solved in a similar way and within the same running time as the singly-weighted problems $\alpha|r(j),\ p(j)=\overline{p}(j)-x(j),\ C(j)\leq d\,(j)\,,\ pmtn|\Phi$, where $\alpha\in\{P,Q\}$ and $\Phi\in\{\Phi_\Sigma,\Phi_{\max}\}$, see Table 8. For completeness, the table also contains the previously known results on the relevant single machine problems. It remains to be seen whether the running times for these single machine problems can be reduced. The approaches to solving parallel machine problems with the objectives $Lex(\Phi_{\max},\Phi_\Sigma)$ and $Lex(\Phi_\Sigma,\Phi_{\max})$ are presented in Sections 12.1 and 12.2, respectively.

12.1 (Total Cost, Maximum Cost) Lexicographic Minimization

Consider problems $\alpha|r(j),p(j)=\overline{p}(j)-x(j),C(j)\leq d(j)|Lex(\Phi_{\max},\Phi_\Sigma)$ for $\alpha\in\{P,Q\}$, where the goal is to minimize the total cost Φ_Σ in the class of schedules with the smallest maximum cost $\Phi_{\max}$. The previously known algorithms are based on the following idea: find an optimal schedule for problem $\alpha|r(j),\ p(j)=\overline{p}(j)-x(j),\ C(j)\leq d\,(j)\,,\ pmtn|\Phi_{\max}$, redefine the durations of the mandatory and optional parts for each task and output a schedule that delivers the minimum total cost for the modified task system. For identical parallel machines, the best previously known algorithm is due to Ho et al. (1994) and Ho (2004). Solving problem $P|r(j),\ p(j)=\overline{p}(j)-x(j),\ C(j)\leq d\,(j)\,,\ pmtn|\Phi_{\max}$ is the most time-consuming part of the algorithm, which should be estimated as $O(n^4)$ (again, due to Remark 1, the running time of $O(n^3\log^2 n)$ claimed in Ho et al. (1994) and Ho (2004) is incorrect). For a single machine, the algorithm requires $O\left(n^2\right)$ time. For uniform parallel machines, the described approach can be implemented in $O(mn^4)$ time; see Wan et al. (2007).

Below, we outline a straightforward approach to solving problems $\alpha|r(j),p(j)=\overline{p}(j)-x(j),C(j)\leq d(j)|Lex(\Phi_{\max},\Phi_\Sigma)$, where $\alpha\in\{P,Q\}$, which leads to faster algorithms. Let $\xi_{\min}$ be the minimum value of $\Phi_{\max}$ (with respect to weights $w_M(j)$, $j\in N$) obtained by solving problem $\alpha|r(j),\ p(j)=\overline{p}(j)-x(j),\ C(j)\leq d\,(j)\,,\ pmtn|\Phi_{\max}$. It is clear that the problem $\alpha|r(j),\ p(j)=\overline{p}(j)-x(j),\ C(j)\leq d\,(j)\,,\ pmtn|Lex(\Phi_{\max},\Phi_\Sigma)$ is nothing else but

Problem	Previously known	Methodology 1
$1\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j),$ $pmtn\|Lex(\Phi_{\max},\Phi_{\Sigma})$	$O(n^2)$ Ho et al. (1994) Ho (2004)	N/A
$1\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j),$ $pmtn\|Lex(\Phi_{\Sigma},\Phi_{\max})$	$O(n^3)$ Ho and Leung (2004) Ho (2004)	N/A
$P\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j),$ $pmtn\|Lex(\Phi_{\max},\Phi_{\Sigma})$	$O(n^4)^*$ Ho et al. (1994) Ho (2004)	$O(n^3)$ Section 12.1
$P\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j),$ $pmtn\|Lex(\Phi_{\Sigma},\Phi_{\max})$	$O(n^5)^*$ Ho and Leung (2004) Ho (2004)	$O(n^3)$ Section 12.2
$Q\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j),$ $pmtn\|Lex(\Phi_{\max},\Phi_{\Sigma})$	$O(mn^4)$ Wan et al. (2007)	$O(mn^3)$ Section 12.1
$Q\|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j),$ $pmtn\|Lex(\Phi_{\Sigma},\Phi_{\max})$	$O(mn^5)$ Wan et al. (2007)	$O(mn^3)$ Section 12.2

* after correcting a faulty claim that problem $P|r(j), C(j) \leq d(j), pmtn|-$ is solvable in $O(n^2 \log^2 n)$time, see Remark 1 of Section 5.

Table 8: Complexity of Problems with Ordered Criteria

problem $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$, $pmtn|\Phi_{\Sigma}$ with additional upper bounds on the x-values:

$$x(j) \leq \xi_{\min} w_M(j), \qquad j \in N.$$

As discussed in Section 6, the resulting problem, and therefore the original problem $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$, $pmtn|Lex(\Phi_{\max}, \Phi_{\Sigma})$ can be solved in $O(n^3)$ time for $\alpha = P$ and in $O(mn^3)$ time for $\alpha = Q$.

12.2 (Maximum Cost, Total Cost) Lexicographic Minimization

We now consider problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j)|Lex(\Phi_{\Sigma}, \Phi_{\max})$ for $\alpha \in \{P, Q\}$, where the goal is to minimize the maximum cost $\Phi_{\max}$ in the class of schedules with the smallest total error cost Φ_{Σ}. The previously known algorithms are based on the following approach. The algorithm consists of k iterations, where k is the number of distinct weights $w_T(j), j \in N$. In an iteration j, a modified task system is treated, in which the optional parts $\underline{p}(j)$ are set to zero, except for those tasks whose w_T-weight is the j-th largest. For such a system a schedule that delivers the minimum total cost is found, and the durations of mandatory parts are appropriately adjusted to be used in the next iteration.

For identical parallel machines, an algorithm that implements this idea is due to Ho and Leung (2004). The running time of such an algorithm should be estimated as $O(kn^4)$ (again the claimed running time of $O(kn^3 \log^2 n)$ time is incorrect, see Remark 1). In the worst case, all w_T-weights are distinct, and hence the algorithm solves problem $P|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d\,(j)$, $pmtn|Lex(\Phi_{\Sigma}, \Phi_{\max})$ in $O(n^5)$ time. For a single machine, the algorithm requires $O\left(n^3\right)$ time. For uniform machines, Wan et al. (2007) give an implementation of this approach in $O(kmn^4)$ time, which in the worst case of $k = O(n)$ leads to $O(mn^5)$. An alternative approach to solving problem $Q|r(j)$, $p(j) = \overline{p}(j) - x(j)$,

$C(j) \leq d(j)$, $pmtn|Lex(\Phi_\Sigma, \Phi_{\max})$ requires $O(kcmn^3)$, where c is not a strongly polynomial parameter that depends on the problem's input; see Wan et al. (2007). Below, we describe an approach that leads to faster algorithms for solving problems $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d(j)$, $pmtn|Lex(\Phi_\Sigma, \Phi_{\max})$ for $\alpha \in \{P, Q\}$.

Recall that problem $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d(j)$, $pmtn|\Phi_\Sigma$ can be solved as a minimum-cost maximum flow problem for the underlying network $H_\alpha = (V, A)$, in which an upper bound on the capacity of an arc $a \in A$ is denoted by $\mu(a)$, and an arc (X_j, t) has the cost $w_T(j)$, $j \in N$, while the weights of all other arcs are zero; see Section 6. Hence, feasible schedules for problem $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d(j)$, $pmtn|Lex(\Phi_\Sigma, \Phi_{\max})$ correspond to minimum-cost maximum flows in H_α, and our scheduling problem reduces to the problem of finding a minimum-cost maximum flow f that minimizes the maximum cost

$$\Phi_{\max} = \max \left\{ \frac{f(X_j, t)}{w_M(j)} \,\middle|\, j \in N,\ f \text{ is a min-cost max-flow in } H_\alpha \right\}.$$

Notice that compression amounts $x(j)$ correspond to $f(X_j, t)$, $j \in N$.

To perform the search over the minimum-cost maximum flows f in H_α, we use their characterization in terms of node potentials and reduced costs; see, e.g., Ahuja et al. (1993, Theorem 9.4). For node potentials $\pi(v)$, $v \in V$, the *reduced cost* of an arc $a = (v', v'') \in A$ is defined as

$$c^\pi(a) = w_T(a) - \pi(v') + \pi(v'').$$

We denote by f_Σ a minimum-cost maximum flow in network H_α, which is fixed in the following discussion. The lemma formulated below can be seen as the complementary slackness theorem for linear programming problems applied to the minimum-cost maximum flow problem.

Lemma 8 *There exist node potentials $\pi(v)$, $v \in V$, such that a maximum flow f in H_α is a minimum-cost maximum flow if and only if it satisfies the following conditions:*

$$\left.\begin{array}{ll} \text{if } c^\pi(a) > 0, & \text{then } f(a) = 0 \ (= f_\Sigma(a)), \\ \text{if } 0 < f(a) < \mu(a), & \text{then } c^\pi(a) = 0, \\ \text{if } c^\pi(a) < 0, & \text{then } f(a) = \mu(a) \ (= f_\Sigma(a)). \end{array}\right\} \tag{35}$$

Moreover, such node potentials can be computed by solving a specially defined shortest path problem in a residual network associated with f_Σ.

Let us call the arcs a with $c^\pi(a) \neq 0$ *fixed arcs*. If for some $j \in N$, only one arc of the pair (j, X_j) and (X_j, t) is fixed and the other is not, then we treat the other arc also as fixed. This can be done since $f(j, X_j) = f(X_j, t)$ holds for all feasible flows f.

The discussion above implies that problem $\alpha|r(j)$, $p(j) = \overline{p}(j) - x(j)$, $C(j) \leq d(j)$, $pmtn|Lex(\Phi_\Sigma, \Phi_{\max})$ reduces to the minimax sharing (or lexicographic sharing) problem (with weights $w_M(j)$) in the network H_α with an additional condition that

$$f(a) = f_\Sigma(a) \qquad \text{for all fixed arcs } a \in A. \tag{36}$$

We will find an optimal flow f^* of this problem by using the algorithm by Gallo et al. (1989) as in Section 5, adjusting it to handle the additional condition (36). Namely, we find f^* as the sum of two flows f' and f'', such that for each arc (X_j, t) at most one value $f'(X_j, t)$ or $f''(X_j, t)$ is positive.

Flow f' is a feasible flow in network H_α that is responsible for keeping the flow on fixed arcs a to $f_\Sigma(a)$. It satisfies the following conditions:

(i) for any fixed arc a, the equality $f'(a) = f_\Sigma(a)$ holds;

(ii) for any non-fixed arc (X_j, t) the equality $f'(X_j, t) = 0$ holds.

Flow f'' delivers the optimal flow on non-fixed arcs (X_j, t). It satisfies the conditions:

(iii) for any fixed arc a, the equality $f''(a) = 0$ holds;

(iv) flow $f''(X_j, t)$ for every non-fixed arc (X_j, t) is part of an optimal flow f^*, i.e., $f''(X_j, t) = f^*(X_j, t)$.

Flow f' can be found as follows. Recall that for each $j \in N$, the arcs of each pair (j, X_j) and (X_j, t) are either both fixed or both non-fixed. For each pair of non-fixed arcs (j, X_j) and (X_j, t), define $f'(j, X_j) = f'(X_j, t) = 0$ and $f'(s, j) = \overline{p}(j) - f_\Sigma(X_j, t)$. For the remaining arcs a of network H_α, define $f'(a) = f_\Sigma(a)$. It is easy to verify that f' satisfies the properties (i) and (ii) above.

In order to find flow f'', we need to solve the lexicographic sharing problem in a residual network. Let $R_\alpha(f')$ be the residual network associated with the flow f'. Since the amount of flow on fixed arcs in H_α must be fixed, we delete all (forward and reverse) arcs in $R_\alpha(f')$ that are associated with fixed arcs in H_α. In the obtained network, which we still denote $R_\alpha(f')$, for each non-fixed arc (X_j, t) the forward arc has capacity $\theta(j)$, while the reverse arc does not exist since $f'(X_j, t) = 0$. Flow f'' can be determined by finding a lexicographically optimal flow in network $R_\alpha(f')$. Notice that the algorithm by Gallo et al. (1989) is applicable, since every arc has zero lower bound for the amount of flow. Hence, conditions (iii) and (iv) are satisfied, and the flow $f^* = f' + f''$ gives an optimal flow of the lexicographic sharing problem in the network H_α with the additional condition (36).

To summarize, the algorithm for solving problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d(j), pmtn|Lex(\Phi_\Sigma, \Phi_{\max})$ performs the following steps.

Algorithm Total-Max

Step 1. Find a minimum-cost maximum flow f_Σ in network H_α.

Step 2. Compute the node potentials and the reduced costs that satisfy the conditions (35), and determine fixed and non-fixed arcs in H_α.

Step 3. Find flow f'.

Step 4. Create the residual network $R_\alpha(f')$ and find flow f'' by solving the lexicographic sharing problem in that network.

Step 5. Set $f^* = f' + f''$. Output the vector x given by $x(j) = f^*(X_j, t)$, $j \in N$, as an optimal solution to $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d(j), pmtn|Lex(\Phi_\Sigma, \Phi_{\max})$.

We now analyze the time complexity of the Algorithm Total-Max. Step 1 requires solving the minimum-cost maximum flow problem and can be done in $O(n^3)$ time for $\alpha = P$ and in $O(mn^3)$ time for $\alpha = Q$ by adapting McCormick's algorithm, as demonstrated in Section 6. In Step 2, we create and process the residual network associated with flow f_Σ. It is known that for the minimum-cost maximum flow problem, the node potentials and the associated reduced costs that satisfy (35) can be found by solving the shortest path problem in the residual network (see, e.g., Ahuja et al. (1993, Section 9.3)), which requires $O(|V| \cdot |A|)$ time,

so that Step 2 can be implemented in $O(n^3)$ time for H_P and in $O(mn^3)$ time for H_Q. Step 3 can be done easily in $O(|A|)$ time. Step 4 solves the lexicographic sharing problem and hence requires $O(n^3)$ and $O(mn^3)$ time for $\alpha = P$ and $\alpha = Q$, respectively. Step 5 can be done in $O(|A|)$ time in a straightforward way. Thus, we conclude that problem $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|Lex(\Phi_\Sigma, \Phi_{\max})$ can be solved in $O(n^3)$ time for $\alpha = P$ and in $O(mn^3)$ time for $\alpha = Q$.

13 Quadratic Costs

In this section, we turn to the SCPT problems with quadratic cost functions. Notice that the cost functions of this type have not been earlier studied in the context of SCPT and SIC, although the smallest weighted sum of squares is a very natural measure, often used in mathematics and statistics. We demonstrate that again advanced techniques of Methodology 1 allow us to handle the whole range of relevant problems.

Consider problems $\alpha|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_{\text{quad}}$ with $\alpha \in \{P, Q\}$, in which the objective function is the weighted sum of squares of compression amounts

$$\Phi_{\text{quad}} = \sum_{j \in N} w'_T(j) x(j)^2, \tag{37}$$

where $w'_T(j)$, $j \in N$, are positive weights. For problems of these range, each job $j \in N$ may be associated with up to three weights: $w_T(j)$ used for computing total error Φ_Σ, $w_M(j)$ used for computing the maximum error $\Phi_{\max}$, and $w'_T(j)$ that is involved in (37). Below, we demonstrate that for all versions of the problem involving Φ_{quad}, even in combination with Φ_Σ and $\Phi_{\max}$, the running times of the algorithms remain the same as for the problems with fixed processing times.

Optimizing quadratic functions is a popular topic of research within submodular optimization; see, e.g., Fujishige (1980) and Hochbaum and Hong (1995). It appears that the SCPT problems defined as the network flow problems in networks H_P and H_Q belong to the class of optimization problems over submodular polyhedra. Indeed, for a network H_α, $\alpha \in \{P, Q\}$, let V_t be the set of nodes connected to the sink t. Assume that the value of a maximum flow in H_α is equal to $\sum_{j \in N} \overline{p}(j)$, i.e., there exists a feasible flow f such that $f(s, j) = \overline{p}(j)$ for all $j \in N$. For $\alpha \in \{P, Q\}$, consider the polyhedron

$$B_\alpha = \{\mathbf{y} \in \mathbb{R}^{V_t} \mid \text{there exists a maximum flow } f \text{ in } H_\alpha \text{ such that } y(v) = f(v, t) \text{ for } v \in V_t\}. \tag{38}$$

It is known (see, e.g., Lemma 4.1 from Megiddo (1974), Hochbaum and Hong (1995), and Section 2.2 from Fujishige (2005), where flow problems in a similar network are considered) that B_α is the base polyhedron with the rank function $\varphi_\alpha : 2^{V_t} \to \mathbb{R}$ given by

$$\varphi_\alpha(X) = \max \left\{ \sum_{v \in X} f(v, t) \mid f \text{ is a feasible flow in } H_\alpha \right\}, \quad X \subseteq V_t.$$

The problems considered in Sections 6 and 11 can be respectively reformulated as the following optimization problems $\Pi_\Sigma(\alpha)$ and $\Pi_{\max}(\alpha)$ over the corresponding base polyhedra:

$$\Pi_\Sigma(\alpha): \quad \text{minimize} \quad \sum_{j \in N} w_T(j) x(j) \quad \text{subject to} \quad \mathbf{x} \in B_\alpha, \tag{39}$$

$$\Pi_{\max}(\alpha): \quad \text{minimize} \quad \max_{j\in N} \frac{x(j)}{w_M(j)} \quad \text{subject to} \quad \mathbf{x} \in B_\alpha, \tag{40}$$

while the problem under consideration can be formulated as

$$\Pi_{\text{quad}}(\alpha): \quad \text{minimize} \quad \sum_{j\in N} w'_T(j)x(j)^2 \quad \text{subject to} \quad \mathbf{x} \in B_\alpha,$$

where $x(j) = x(X_j)$ for $j \in N$. Adapting a general result from Fujishige (1980) and from Section 9.2 of Fujishige (2005) on the equivalence of quadratic and lexicographic optimization with submodular constraints, we deduce the following statement.

Lemma 9 *For network H_α, let $f^*(a)$, $a \in A$, be a maximum flow that is lexicographically optimal with respect to the ratios $f^*(X_j,t)/w'_T(j)$, $j \in N$. Then, the flow f^* minimizes the quadratic objective function $\sum_{j\in N} w'_T(j)f^*(X_j,t)^2$ among all maximum flows in H_α, and the values $x(j) = f^*(X_j,t)$, $j \in N$, define an optimal solution to problem $\Pi_{\text{quad}}(\alpha)$.*

Using the algorithm for solving the lexicographic sharing problem discussed in Section 11, we deduce that problem $\Pi_{\text{quad}}(\alpha)$ can be solved in $O(n^3)$ time for $\alpha = P$ and in $O(mn^3)$ time for $\alpha = Q$.

We now pass to considering various constrained versions of scheduling problems of imprecise computation that involve quadratic cost. Let parameters $\eta_{\min}$ and $\xi_{\min}$ denote the minimum value of the total cost and the minimum value of the maximum cost, respectively; in other words, $\eta_{\min}$ is the optimal value of the objective function for problem (39), while $\xi_{\min}$ is that for problem (40).

We start with the constrained problem $\Pi_{Lex(\max,\text{quad})}(\alpha)$, in which the optimal value of Φ_{quad} is to be found among the solutions with the smallest maximum cost $\xi_{\min}$. The corresponding problem is of the form:

$$\begin{aligned} \Pi_{Lex(\max,\text{quad})}(\alpha): \quad & \text{minimize} \quad \sum_{j\in N} w'_T(j)x(j)^2 \\ & \text{subject to} \quad \mathbf{x} \in B_\alpha, \qquad \frac{x(j)}{w_M(j)} \le \xi_{\min},\ j \in N. \end{aligned}$$

This problem can be treated in a similar way as the problems with $Lex(\Phi_{\max},\Phi_\Sigma)$ objective considered in Section 12.1. That is, problem $\Pi_{Lex(\max,\text{quad})}(\alpha)$ reduces to problem $\Pi_{\text{quad}}(\alpha)$ with the additional upper bounds $x(j) \le \xi_{\min} w_M(j)$ on variables $x(j)$.

Consider the constrained problem $\Pi_{Lex(\Sigma,\text{quad})}(\alpha)$, in which the optimal value of function Φ_{quad} is to be found among the solutions with the minimum total weighted cost $\eta_{\min}$. This can be expressed as

$$\begin{aligned} \Pi_{Lex(\Sigma,\text{quad})}(\alpha): \quad & \text{minimize} \quad \sum_{j\in N} w'_T(j)x(j)^2 \\ & \text{subject to} \quad \mathbf{x} \in B_\alpha, \qquad \sum_{j\in N} w_T(j)x(j) = \eta_{\min}. \end{aligned}$$

Problem $\alpha|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \le d(j), pmtn|\ Lex(\Phi_\Sigma,\Phi_{\max})$ presented in Section 12.2 has the same feasible region as problem $\Pi_{Lex(\Sigma,\text{quad})}(\alpha)$. This implies that the first three steps of Algorithm Total-Max, which manipulate the feasible region, are applicable to problem $\Pi_{Lex(\Sigma,\text{quad})}(\alpha)$. As a result of these steps, problem $\Pi_{Lex(\Sigma,\text{quad})}(\alpha)$ is reduced to

problem $\Pi_{\text{quad}}(\alpha)$, similarly to the reduction of problem $\Pi_{Lex(\Sigma,\max)}(\alpha)$ to problem $\Pi_{\max}(\alpha)$ presented in Section 12.2. An optimal solution to problem $\Pi_{\text{quad}}(\alpha)$ is delivered by solving the corresponding problem of lexicographic flow sharing.

Finally, consider the constrained problems $\Pi_{Lex(\text{quad},\Sigma)}(\alpha)$ and $\Pi_{Lex(\text{quad},\max)}(\alpha)$, in which Φ_Σ and $\Phi_{\max}$ are respectively minimized subject to the minimum value of Φ_{quad}. The problems $\Pi_{Lex(\text{quad},\Sigma)}(\alpha)$ and $\Pi_{Lex(\text{quad},\max)}(\alpha)$ have the same feasible region

$$\mathbf{x} \in B_\alpha, \qquad \sum_{j\in N} w'_T(j)x(j)^2 = \zeta_{\min},$$

where $\zeta_{\min}$ is the minimum value of Φ_{quad}.

In order to find $\zeta_{\min}$, we solve problem $\Pi_{\text{quad}}(\alpha)$, which is done by solving the lexicographic sharing problem, as shown in Lemma 9. Notice that an optimal solution to the lexicographic sharing problem is unique (see Fujishige (1980, Theorem 3.1)). This implies that for the original problems $\Pi_{Lex(\text{quad},\Sigma)}(\alpha)$ and $\Pi_{Lex(\text{quad},\max)}(\alpha)$, their feasible regions consist of a single solution. This leaves no freedom for optimizing Φ_Σ and $\Phi_{\max}$. Thus, the solution to problem $\Pi_{Lex(\text{quad},\Sigma)}(\alpha)$ and problem $\Pi_{Lex(\text{quad},\max)}(\alpha)$ does not depend on the weights $w_T(j)$ and $w_M(j)$ of the objective functions Φ_Σ and $\Phi_{\max}$ and remains the same for any of these two functions.

Hochbaum and Hong (1995) deal with a quadratic function with a linear term:

$$\Phi_{\text{Quad}} = \sum_{j\in N} w'_T(j)x(j)^2 + \sum_{j\in N} w_T(j)x(j), \tag{41}$$

in addition to a quadratic function Φ_{quad} without a linear term. It is demonstrated in Hochbaum and Hong (1995) that the techniques by Gallo et al. (1989) cannot be applied directly to solve the problem of minimizing Φ_{Quad} with a linear term. They also show how to adjust the parametric flow algorithms by Gallo et al. (1989) to make them handle this problem without increasing their running times. Hence, we deduce that all results mentioned in this section remain valid if a quadratic function without a linear term is extended to the one with a linear term (informally, if the subscript "quad" is replaced by "Quad" in the notation of the problem). Thus, we can conclude that for all SCPT problems which involve a quadratic objective function, with or without a linear term, time complexities $O(n^3)$ and $O(mn^3)$ hold for the parallel machine models with $\alpha = P$ and $\alpha = Q$, respectively.

It is clear that all results discussed in this section for the problems on parallel machines carry on for a single machine counterpart. It remains to be seen whether for the single machine problems that involve minimization of a quadratic cost function it is possible to develop an algorithm of the running time lower than $O\left(n^3\right)$. Methodology 1 cannot be used for this purpose. Indeed, if such an algorithm existed it would be based on the ideas different from finding the parametric flow, since for the latter problem the fastest known algorithm requires $O\left(n^3\right)$ time. The use of Methodology 2 seems to be more promising, as we demonstrate below for the problem of minimizing function (41), provided that the jobs have a common deadline d.

Using compressions $x(j), j \in N$, problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \le d, pmtn|\Phi_{\text{Quad}}$ can be written as a quadratic programming problem with submodular constraints:

$$\begin{array}{lll} \min & \displaystyle\sum_{j\in N} w'_T(j)x(j)^2 + \sum_{j\in N} w_T(j)x(j) & \\ \text{s.t.} & \overline{p}(X) - x\left(X\right) \le d - \min\limits_{h\in X} r(h), & X \in 2^N, \\ & 0 \le x(j) \le \overline{p}(j) - \underline{p}(j), & j \in N. \end{array}$$

If the jobs are renumbered in accordance with (6) then the problem above simplifies to

$$\begin{array}{lll} \min & \sum\limits_{j\in N} w'_T(j)x(j)^2 + \sum\limits_{j\in N} w_T(j)x(j) & \\ \text{s.t.} & \sum\limits_{j=k}^{n} (\overline{p}(j) - x\,(j)) \le d - r(k), & k = 1,\ldots,n; \\ & 0 \le x(j) \le \overline{p}(j) - \underline{p}(j), & j \in N. \end{array}$$

Now we change the decision variables to the actual processing times $p\,(j) = \overline{p}(j) - x\,(j)$, $j \in N$. The objective function becomes

$$\begin{aligned} & \sum_{j\in N} w'_T(j)(\overline{p}(j) - p(j))^2 + \sum_{j\in N} w_T(j)\,(\overline{p}(j) - p\,(j)) \\ & \quad = \sum_{j\in N} w'_T(j)p\,(j)^2 - \sum_{j\in N} \left(2\overline{p}(j)w'_T(j) + w_T(j)\right) p(j) + L, \end{aligned}$$

where

$$L = \sum_{j\in N} w'_T(j)\overline{p}(j)^2 + \sum_{j\in N} w_T(j)\overline{p}(j).$$

If the constant L is removed, problem $1|r(j),\ p(j) = \overline{p}(j) - x(j),\ C(j) \le d,\ pmtn|\Phi_{\text{Quad}}$ reduces to

$$\begin{array}{lll} \min & \sum\limits_{j\in N} w'_T(j)p\,(j)^2 - \sum\limits_{j\in N} \left(2\overline{p}(j)w'_T(j) + w_T(j)\right) p(j) & \\ \text{s.t.} & \sum\limits_{j=k}^{n} p\,(j) \le d - r\,(k)\,, & k = 1,\ldots,n; \\ & \underline{p}(j) \le p(j) \le \overline{p}(j), & j \in N. \end{array}$$

This problem can be classified as the resource allocation problem with a separable quadratic function under nested constraints. Such a problem can be solved in $O\,(n \log n)$ time, as proved in Hochbaum and Hong (1995).

14 Conclusions

To conclude, we summarize the main points addressed in the survey.

1. The term "scheduling with controllable processing times" and associated terminology are sufficiently abstract and general to provide a unified framework for all associated models in which the actual processing time is to be selected from a given interval. The models of scheduling with imprecise computation and of late work minimization (with preemption) should be seen as meaningful interpretations of the general SCPT models, driven by particular applications. We hope that a correct positioning of these and other specialized models within the body of research on SCPT will help avoiding potential rediscoveries and duplications. Besides, such a positioning will allow merging the corresponding toolkits to attack joint research challenges.

2. Processing capacity set functions φ introduced in Section 3 are crucial for solving problems with fixed and controllable data. Their submodularity links SCPT to optimization with submodular constraints.

3. Methodology 1 based on the network flow methods is useful for the most general models with multiple parallel machines (models P and Q) and arbitrary $r(j)$ and $d(j)$, $j \in N$.

   (a) For fixed data, standard max-flow techniques deliver a solution as presented in Section 5.

   (b) Multiparametric network flow methods by McCormick (1999) are useful in handling SCPT problems that involve minimizing the total compression cost Φ_{Σ}; see Section 6.

   (c) Single-parameter flow algorithms by Gallo et al. (1989) form the basis for solving SCPT problems that involve minimizing the maximum compression cost $\Phi_{\max}$ and/or the quadratic cost; see Sections 11-13.

   For the networks that have a structure relevant to this study, the running times of the parametric flow algorithms match those developed for solving the problems on networks with fixed arc capacities, see Table 2. This is the main reason why the whole range of the SCPT problems on parallel machines with different release dates and deadlines require the same running times as their feasibility counterparts with fixed data: $O\left(n^3\right)$ in the case of identical machines and $O\left(mn^3\right)$ in the case of uniform machines. The range of these problems include not only problems with a single objective (total, maximum, or quadratic cost), but also all problems with two lexicographically ordered criteria. The reported running times should be seen as the best possible and can only be improved if faster algorithms are found for solving the feasibility problems with fixed data.

4. In Section 5, we demonstrate that checking the existence of a feasible schedule on identical parallel machines requires $O\left(n^3\right)$ time, and not $O\left(n^2 \log^2 n\right)$ as has been claimed in the SIC literature. This leads to repairing of the running times of algorithms previously known for solving the corresponding SCPT problems; see Remark 1 and the references to that remark throughout.

5. Methodology 2 makes use of the important property of processing capacity functions φ, their submodularity. It provides a rather direct way to finding Pareto-optimal solutions to bicriteria problems with Φ_{Σ} being one of the objectives; see Section 8. The feasible region of the relevant problems is a parametric submodular polyhedron intersected with a box, and this allows deducing an analytical description of the efficiency frontier. The power of this analytical approach can be seen from the fact that among the problems, that are handled that way, there are those on parallel machines with distinct release dates, which were previously open. The analytical approach appears to be more efficient and less tedious than the traditional one, based on generating breakpoints of the frontier one after another by tracing changes in the structure of schedules subject to compression/decompression of some jobs.

   On the other hand, Section 8 gives examples of bicriteria problems for which fast algorithms are derived based on the traditional approach, since those problems are relatively simple and the corresponding efficiency frontier has a rather small number of breakpoints. But even then Methodology 2 provides a natural justification of actions taken by these algorithms.

6. Methodology 3 is a further development of Methodology 2; see Section 9. The decomposition algorithm is applicable to solving linear programming problems for which the

feasible region is the intersection of an arbitrary submodular polyhedron and a box. In Section 10, we discuss its adaptation for solving SCPT problems to minimize the total compression cost Φ_{Σ}. Those problems can be solved by a straightforward application of the greedy algorithm (Methodology 2), but the application of the decomposition algorithm delivers solutions faster. In particular, for problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d(j), pmtn|\Phi_{\Sigma}$ the best possible running time of $O(n \log n)$ is achieved, which has been a long-standing goal.

7. Sections 11 and 12 contain new results on the SCPT problems which involve minimizing the maximum cost $\Phi_{\max}$. The problems of this type form the main direction in the SIC research. We demonstrate that an appropriate use of Methodology 1, i.e., an application of the single-parameter flow techniques, results in a collection of algorithms with the best possible running times.

8. Section 13 addresses the problems that involve minimizing a quadratic cost function. Such SCPT problems have not been studied before. It appears to be fairly easy to extend the methods developed in Sections 11 and 12 to achieve the best possible algorithms for the parallel machine problems of that range.

Now we state several open questions that might motivate further studies in the area of scheduling with controllable processing times.

1. For the SCPT problems with arbitrary release dates and deadlines that involve minimizing the maximum or quadratic cost functions on a single machine there is still a complexity gap between the running times of the best known algorithms and $O(n \log n)$, i.e., the time needed to solve the feasibility problem with fixed data. Such a gap has been removed in the case of parallel machines due to the use of parametric flow techniques. The flow approach will not give algorithms faster than $O\left(n^3\right)$ time and is therefore not applicable for getting improved algorithms for a single machine. We hope that the existing gaps could be eventually closed, as has happened to all problems of minimizing the total compression cost Φ_{Σ}.

2. For the SCPT problems with a common deadline that involve minimizing the maximum or quadratic cost functions there is a need for developing algorithms with the running times better than those available in the case of arbitrary deadlines. In Section 13, we present an $O(n \log n)$-time algorithm for problem $1|r(j), p(j) = \overline{p}(j) - x(j), C(j) \leq d, pmtn|\Phi_{\text{Quad}}$ and expect that algorithms with a similar performance can be developed for the remaining problems.

3. There are no results on finding Pareto-optimal solutions to the SCPT problems in which one of the objectives is either the maximum cost or quadratic cost.

4. There is a lack of results on finding Pareto-optimal solutions to the SCPT problems with arbitrary due dates in which one of the objective is the maximum lateness $L_{\max}$. In Section 8.2, we mention the solved problem $1|r(j), p(j) = \overline{p}(j) - x(j), pmtn|\left(L_{\max}, \Phi_{\Sigma}\right)$ and expect that similar bicriteria problems will be addressed.

5. In our recent paper (Shioura et al. (2016b)), we demonstrate how the flow and submodular optimization techniques can be applied to the off-line problems of speed scaling. These problems reduce to minimizing convex separable functions under submodular

constraints. The algorithms that we develop outperform those previously known in the area and also are able to tackle problems with more general objectives than studied before. We hope that a systematic methodological study, similar to that done for the SCPT problem, can also be performed in the area of speed scaling.

6. We are interested in finding other areas, even not related to scheduling, in which the described methodologies can be useful. In particular, we would like to find out practical situations that give rise to Problem (LP) so that the decomposition algorithm can be applied.

Acknowledgement

This research was supported by the EPSRC funded project EP/J019755/1 "Submodular Optimisation Techniques for Scheduling with Controllable Parameters". The first author was partially supported by JSPS KAKENHI Grant Numbers 15K00030, 15H00848.

References

R.K. Ahuja, T.L. Magnanti, J.B. Orlin (1993) Network Flows: Theory, Algorithms and Applications. Prentice Hall, NJ, USA.

R.K. Ahuja, J.B. Orlin, C. Stein, R.E. Tarjan (1994) Improved algorithms for bipartite network flow. SIAM J Comput. 23, 906–933.

J. Blazewicz (1984) Scheduling preemptive tasks on parallel processors with information loss. Technique et Science Informatiques 3, 415–420.

J. Blazewicz, G. Finke (1987) Minimizing mean weighted execution time loss on identical and uniform processors. Inform. Process. Lett. 24, 259–263.

P. Brucker (2007) Scheduling Algorithms. 5th edition, Springer, Berlin.

Y.L. Chen (1994) Scheduling jobs to minimize total cost. Eur. J. Oper. Res. 74, 111–119.

J.Y. Chung, W.-K. Shih, J.W.S. Liu, D.W. Gillies (1989) Scheduling imprecise computations to minimize total error. Microproc. Microprogram. 27, 767–774.

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

S. Fujishige (1980) Lexicographically optimal base of a polymatroid with respect to a weight vector. Mathematics of Operations Research 5, 186–196.

S. Fujishige (2005) Submodular Functions and Optimization. 2nd Edition, Ann. Discr. Math. 58, Elsevier.

G. Gallo, M.D. Grigoriadis, R.E. Tarjan (1989) A fast parametric maximum flow algorithm and applications. SIAM J Comput. 18, 30–55.

A.V. Goldberg, R.E. Tarjan (1988) A new approach to the maximum flow problem. J. ACM 35, 921–940.

T.F. Gonzales, S. Sahni (1978) Preemptive scheduling of uniform processor systems. J. ACM 25, 92–101.

V.S. Gordon, V.S. Tanaev (1973) Deadlines in single-stage deterministic scheduling. Optimization of Systems for Collecting, Transfer and Processing of Analogous and Discrete Data in Local Information Computing Systems. Materials of the 1st Joint Soviet-Bulgarian seminar (Institute of Engineering Cybernetics of Academy of Sciences of BSSR – Institute of Engineering Cybernetics of Bulgarian Academy of Sciences, Minsk), 53–58 (in Russian).

K.I.-J. Ho (2004) Dual criteria optimization problems for imprecise computation tasks. In: J.Y.-T. Leung, ed. Handbook of Scheduling: Algorithms, Models and Performance Analysis (Chapman & Hall/CRC), 35-1 – 35-26.

K.I.-J. Ho, J.Y.-T. Leung, W.-D. Wei (1994) Minimizing maximum weighted error for imprecise computation tasks. J. Algorithms 16, 431–452.

K.I.-J. Ho, J.Y.-T. Leung (2004) A dual criteria preemptive scheduling problem for minimax error of imprecise computation tasks. Int. J. Found. Comput. Sci. 15, 717–731.

D.S. Hochbaum, S.P. Hong (1995) About strongly polynomial time algorithms for quadratic optimization over submodular constraints. Math. Program. 69, 269–309.

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

H. Hoogeveen and G.J. Woeginger (2001) Some comments on sequencing with controllable processing times, Computing 68, 181–192.

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

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

A. Janiak, M.Y. Kovalyov (1996) Single machine scheduling with deadlines and resource dependent processing times. Eur. J. Oper. Res. 94, 284–291.

K. Jansen, M. Mastrolilli (2004) Approximation schemes for parallel machine scheduling problems with controllable processing times. Comput. Oper. Res. 31, 1565–1581.

A.V. Karzanov (1974) Determining the maximal flow in a network by the method of preflows. Soviet Math Dokl 15, 434–437.

N. Katoh, T. Ibaraki (1998) Resource allocation problems. In Du, D.-Z., Pardalos, P.M. (eds) Handbook of Combinatorial Optimization, Kluwer, Dordrecht, Vol. 2, pp. 159–260.

E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (1993) Sequencing and scheduling: algorithms and complexity. In Graves, S.C., Rinnooy Kan, A.H.G. Zipkin, P.H. (eds.) Handbooks in Operations Research and Management Science, Vol. 4, Logistics of Production and Inventory, Elsevier, North–Holland, Amsterdam, pp. 445–522.

J.Y.-T. Leung (2004) Minimizing total weighted error for imprecise computation tasks. In Leung, J.Y.-T. (ed) Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapman & Hall/CRC, pp. 34-1 – 34-16.

J.Y.-T. Leung, V.K.M. Yu, W.-D. Wei (1994) Minimizing the weighted number of tardy task units. Discr. Appl. Math. 51, 307–316.

C. Martel (1982) Preemptive scheduling with release times, deadlines, and due dates, J. ACM 29, 812–829.

S.T. McCormick (1999) Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. Oper. Res. 47, 744–756.

R. McNaughton (1959) Scheduling with deadlines and loss functions. Manage. Sci. 12, 1–12.

N. Megiddo (1974) Optimal flows in networks with multiple sources and sinks. Math. Progr. 7, 97–107.

G.L. Nemhauser, L.A. Wolsey (1988) Integer and Combinatorial Optimization. Wiley.

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

E. Nowicki, S. Zdrzałka (1995) A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times. Discr. Appl. Math. 63, 237–256.

J.B. Orlin (1988) A faster strongly polynomial minimum cost flow algorithm. Proceedings of the 20th ACM Symposium on Theory of Computing, 377-387.

S. Sahni (1979) Preemptive scheduling with due dates. Oper. Res. 27, 925–934.

S. Sahni, Y. Cho (1980) Scheduling independent tasks with due times on a uniform processor system. J. ACM 27, 550–563.

A. Schrijver (2000) A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theory B 80, 346–355.

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

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

N.V. Shakhlevich, V.A. Strusevich (2005) Pre-emptive scheduling problems with controllable processing times. J. Sched. 8, 233–253.

N.V. Shakhlevich, V.A. Strusevich (2008) Preemptive scheduling on uniform parallel machines with controllable job processing times. Algorithmica 51, 451–473.

N.V. Shakhlevich, A. Shioura, V.A. Strusevich (2009) Single machine scheduling with controllable processing times by submodular optimization. Int. J. Found. Comput. Sci. 20, 247–269.

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

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

W.-K. Shih, C.-R. Lee, C.H. (2000) A fast algorithm for scheduling imprecise computations with timing constraints to minimize weighted error. Proc. 21th IEEE Real-Time Systems Symposium (RTSS2000), 305–310.

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

A. Shioura, N.V. Shakhlevich, V.A. Strusevich (2015) Decomposition algorithms for submodular optimization with applications to parallel machine scheduling with controllable processing times. Math. Progr. A 153: 495–534.

A. Shioura, N.V. Shakhlevich, V.A. Strusevich (2016a) Application of submodular optimization to single machine scheduling with controllable processing times subject to release dates and deadlines. INFORMS J. Comp. 28, 148–161.

A. Shioura, N.V. Shakhlevich, V.A. Strusevich (2016b) Energy saving computational models with speed scaling via submodular optimization. Proc. 3rd International Conference on Green Computing, Technology and Innovation (ICGCTI2015), Serdang, Malaysia, 7-18.

M. Sterna (2011) A survey of scheduling problems with late work criteria. Omega 29, 120–129.

V. T'kindt, J.-C. Billaut (2006) Multicriteria Scheduling: Theory, Models and Algorithms, Springer, Berlin.

L.N. Van Wassenhove, K.R. Baker (1982) A bicriterion approach to time/cost tradeoffs in sequencing, Eur. J. Oper. Res. 11, 48–54.

R.G. Vickson (1980) Two single machine sequencing problems involving controllable job processing times, AII Trans. 12, 258–262.

G. Wan, J.Y.-T. Leung, M.L. Pinedo (2007) Scheduling imprecise computation tasks on uniform processors. Inform. Process. Lett. 104, 45–52.