Computers & Operations Research 169 (2024) 106748

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Scheduling jobs to minimize a convex function of resource usage[☆]

Tamás Kis[*], Evelin Szögi

HUN-REN Institute for Computer Science and Control, Kende utca 13-17, Budapest, 1111, Hungary
Department of Mathematics, Loránd Eötvös University, Pázmány Péter sétány 1/C, Budapest, 1117, Hungary

ARTICLE INFO

Keywords:
Scheduling
Load balancing
Convex optimization
Network flows

ABSTRACT

Given a finite set of jobs and a common resource. Each job has the same processing time p, alongside an individual release date and deadline, and utilizes either zero or one unit from the resource. A schedule specifies a star time for each job, and it determines the resource usage over time. The objective is to minimize a separable convex function of the resource usage. Prior to our work, the existing body of research only tackled the scenario where $p = 1$. We explore three variations of this fundamental problem, accompanied by applications drawn from existing literature. In the first variant, all jobs require one unit of the resource each. In the second and third variants, there are m parallel machines, and at most m jobs may be processed concurrently at any given moment. Furthermore, in the second variant, each job has a unit processing time, and may require either 0 or 1 unit of the resource. In the third case, there are v distinct resource types each linked with a convex function, and each job requires precisely one of these resources types. The jobs have a uniform processing time p and possess agreeable release dates and deadlines. For each of these cases, we introduce novel polynomial-time algorithms designed to determine optimal solutions.

1. Introduction

This paper delves into variants of the scheduling problem outlined as follows: A set of n jobs $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$ is considered, alongside a shared resource required by some subset of these jobs. Each job J_i is characterized by a release date r_i, a deadline d_i, and a resource requirement $\mu_i \in \{0, 1\}$. All jobs share a uniform processing time p. A schedule $\mathcal{S}$ specifies a starting time S_i for each job J_i, and is deemed *feasible*, if the condition $r_i \le S_i \le d_i - p$ is met for each job J_i.

The goal is to find a feasible schedule $\mathcal{S}$, which minimizes a convex function $f : \mathbb{R}_{\ge 0} \to \mathbb{R}$ of the load of the resource throughout the scheduling horizon, i.e.,

$$\min_{\mathcal{S}} \int_{r_{\min}}^{d_{\max}} f(\ell^{\mathcal{S}}(t))dt,$$

where $r_{\min} = \min_i r_i$ is the earliest release date and $d_{\max} = \max_i d_i$ is the last deadline, and $\ell^{\mathcal{S}}(t)$ is the load of the resource at time point t in schedule $\mathcal{S}$, i.e., $\ell^{\mathcal{S}}(t) = |\{i \mid S_i \le t \le S_i + p,\ \mu_i = 1\}|$.

A variant of this problem, where $p = 1$, $\mu_i = 1$ for all $J_i \in \mathcal{J}$, and for each job J_i a subset of time slots $D_i \subset \mathbb{Z}_+$ is given, rather than an interval $[r_i, d_i]$, is known as the *load balancing problem* and it has been extensively studied by several authors, see e.g., Hajek (1990), Harvey et al. (2006) and Burcea et al. (2016). As it is established in all these papers, this special case has some very nice properties: (i) there always exists a universally optimal solution $\mathcal{U}$, which is optimal for any convex function f of the load $\ell^{\mathcal{U}}$, and (ii) if a schedule $\mathcal{S}$ is *not* optimal, then there exists a pair of time slots $t_0, t_1 \in \mathbb{Z}$, such that $\ell^{\mathcal{S}}(t_0) \ge \ell^{\mathcal{S}}(t_1)+2$ alongside a subset of jobs which can be rescheduled. This rescheduling results in a decrease of one in the load of time slot t_0, an increase of one in the load of time slot t_1, while the workload of the other time slots remains unchanged. The convexity of f ensures that the objective function value of the new schedule is smaller than that of $\mathcal{S}$. While it might seem logical to adapt the methodologies utilized in the $p = 1$ scenario to instances where $p > 1$, such a direct transfer is not possible. To begin with, the two properties observed in the $p = 1$ case do not hold when $p > 1$ (see Section 3), indicating a fundamentally distinct problem structure. The algorithms devised for the $p = 1$ case heavily rely on concepts such as the legal graph and legal paths, which indicate potential job reassignments improving the current schedule. However, when $p > 1$, it seems hopeless to establish analogous versions of legal graphs and legal paths. Hence, there is a need for new ideas and methodologies to efficiently solve the problem when $p > 1$.

The above scheduling problem can be extended to parallel machine problems, where each job has to be assigned to a machine, and the jobs assigned to the same machine have to be sequenced. The latter problem

☆ This research has been supported by the TKP2021-NKTA-01 NRDIO, Hungary grant on "Research on cooperative production and logistics systems to support a competitive and sustainable economy".

* Corresponding author at: HUN-REN Institute for Computer Science and Control, Kende utca 13-17, Budapest, 1111, Hungary.
E-mail addresses: kis.tamas@sztaki.hu (T. Kis), szogi.evelin@sztaki.hu (E. Szögi).

https://doi.org/10.1016/j.cor.2024.106748
Received 9 November 2023; Received in revised form 9 June 2024; Accepted 19 June 2024
Available online 24 June 2024

was introduced by Błażewicz (1979), where all jobs have processing time $p = 1$, and require 0 or 1 unit of a common resource of capacity c. In a feasible solution the jobs are scheduled between their release dates and deadlines, and at most m jobs are processed concurrently, where m is the number of the parallel machines. Moreover, at most c jobs of resource requirement 1 are processed in parallel at any given moment. Blazewicz described a proprietary polynomial time algorithm for deciding whether a feasible schedule exists. This problem can be reformulated as a scheduling problem with a separable convex cost function. We define a piecewise-linear convex function f as follows: $f(x) = 0$ for $x \leq c$ and $f(x) = x - c$ for $x \geq c$. It is easy to see that there is a feasible schedule in which at most c jobs with resource requirement 1 are scheduled concurrently if and only if there exists a feasible schedule of cost 0 w.r.t function f.

In this paper, we deal with three variants of the scheduling problem with non-preemptive jobs, all of processing time p:

Problem P_1. All jobs require one unit of the common resource, i.e., $\mu_i = 1$ for each job J_i, and the joint processing time p is an arbitrary positive integer.

We will show by way of an example that unlike in the load balancing problem with $p = 1$, for general p, there is no universally optimal solution (Section 3). Furthermore, improving a non-optimal schedule may be far more complicated than in the case with unit length jobs. We will reduce the problem to a minimum cost circulation problem with convex cost functions on the arcs in an appropriately defined network, which permits the application of efficient combinatorial methods for finding optimal solutions (Section 4).

Problem P_2. The jobs have unit processing times, integer release dates and deadlines, and require 0 or 1 unit of a common resource. In addition, there are m parallel machines, which can process at most m jobs concurrently, and each job must be assigned to precisely one machine.

We describe a network-flow based method with convex cost functions on the arcs in Section 5. As a by-product, our method can also answer the decision problem of Błażewicz (1979).

Problem P_3. There is a constant number of v resource types $R_1, \ldots, R_v$, and each R_k is associated with a convex cost function f_k, $k = 1, \ldots, v$. There are m parallel machines, which can process at most m jobs concurrently. Each job J_i has the same processing time p, it requires one unit from exactly one resource type $\mu_i \in \{1, \ldots, v\}$, and has to be assigned to precisely one machine. The goal is to find a schedule $\mathcal{S}$ respecting the above constraints and minimizing the objective function

$$\min_{\mathcal{S}} \sum_{k=1}^{v} \int_{r_{\min}}^{d_{\max}} f_k(\ell_k^{\mathcal{S}}(t))dt,$$

where $\ell_k^{\mathcal{S}}(t) = |\{i \mid S_i \leq t \leq S_i + p,\ \mu_i = k\}|$.

In Section 6, we give dynamic programming based algorithms to solve P_3 in two special cases: the release dates and deadlines of the jobs are agreeable, or the set of jobs can be partitioned into a constant number of subsets with this property.

Finally, we mention that this work is an extended version of Szögi and Kis (2023).

2. Related work

We have already summarized the most relevant results on load balancing and deadline scheduling of jobs on parallel machines using a bounded capacity resource in the introduction. In the following we focus on parallel machine scheduling problems with equal job processing times.

Brucker and Kravchenko (2008) investigate the problem where equal-length jobs have to be scheduled on m identical parallel machines. For each job, a release date and a deadline is given. They present a polynomial time algorithm that finds a feasible schedule and minimizes the weighted sum of the completion times. Their method is based on an integer programming formulation of the problem, solving the linear relaxation and rounding the solution appropriately. A similar problem is considered by Kravchenko and Werner (2009), but the time interval between the earliest release date and the latest deadline is divided into several smaller intervals, and for each of them, the number of available machines is given. They present a linear programming approach to find a feasible schedule that minimizes the maximum number of machines used by the jobs. In Simons (1983), Simons considers a problem where n unit-time jobs are given with rational release dates and deadlines, and they have to be scheduled on m machines. A polynomial time algorithm is devised that finds a feasible schedule, if one exists, which minimizes both the makespan and the sum of job completion times. Since the integrality of the release dates and deadlines is not assumed, the same algorithm works if the jobs have the same but not necessarily unit processing time. Simons and Sipser (1984) consider a scheduling problem, where n unit-length jobs have to be scheduled on m machines. Each job has several feasible intervals with integral endpoints in which it can be processed. They give a polynomial time algorithm that finds a feasible schedule, if one exists, and in this case, a feasible schedule that minimizes the makespan or the sum of the completion times can be obtained as well. While the case with unit processing time is manageable, they prove NP-hardness for a slightly different variant of the problem. If the length of each job is 2, the problem is NP-hard even if each job has only two feasible intervals. Baptiste (2000) investigates the problem, where n equal-length jobs are given, each of them associated with a release date and deadline. The jobs have to be scheduled on m identical parallel machines, where m is a constant. Baptiste gives a method using dynamic programming that finds a feasible solution minimizing $\sum_{i=1}^{n} f_i(C_i)$, where the functions f_i are non-decreasing and for any $1 \leq i, j \leq n$, $f_i - f_j$ is monotonous. Baptiste et al. (2004) consider several scheduling problems of open complexity status, where the jobs share the same processing time. They devise new polynomial time algorithms to many of these problems, some of them derived from well-known scheduling algorithms and some of them involving new techniques. We refer the reader to the survey (Kravchenko and Werner, 2011) for further results.

Drótos and Kis (2011) study the following resource leveling problem. Given m machines, a finite set of renewable resources, and n jobs, each of which pre-allocated to one of the machines. The jobs have individual processing times, release dates, deadlines and resource requirements. One has to determine the starting times of the jobs such that the jobs processed on the same machine do not overlap, and the sum of some convex functions of the loads of the resources is minimized. The authors prove that if the starting times of all jobs on all but one machines are fixed, the problem is NP-hard. However, if the ordering of jobs on the remaining machine is also given, then a polynomial time algorithm exists. They also give a heuristic method as well as an exact branch-and-bound algorithm for solving the problem.

Many authors investigate online load balancing problems. In e.g. Shaikhet et al. (2013), Liu et al. (2020) and Chau et al. (2021), online energy management problems are studied. Shaikhet et al. (2013) propose an on-line scheduling algorithm for demands with stochastic energy demands and stochastic malleability constraints and prove that the presented algorithm is asymptotically optimal and runs in linear time. Liu et al. (2020) consider the online, non-preemptive problem involving jobs with arbitrary processing times and resource requirements. The convex function to be minimized is $f(x) = x^{\alpha}$, where $\alpha > 1$ is a constant. The authors devise an online algorithm based on online dynamic speed scaling methods. Chau et al. (2021) study a problem in demand response management in a smart grid. A sequence of jobs arrives online to be scheduled within their time windows on unrelated parallel machines. These machines could represent different resources in a smart grid or distinct electrical sub-networks. Each machine is linked with an energy-power function, and the goal is to determine a non-migrative schedule that minimizes the total energy consumption. When the energy-power functions adhere to the form $f(x) = x^{\alpha}$, the authors provide an α^{α}-competitive algorithm. They also propose a 2^α-approximation algorithm for the offline setting, which involves unit-time jobs, constant power requests and identical machines with identical energy-power functions.

In Stewart et al. (2023), the authors investigate the problem of scheduling precedence related tasks on parallel processors with variable speeds. If two precedence related tasks are scheduled on distinct processors, a positive communication time is necessary between the completion of the predecessor task and the start of the successor task. The authors consider a bi-objective version of the problem aiming to minimize both the makespan and the energy consumption in a Pareto-efficient manner. The survey of Xie et al. (2022) discusses three kind of scheduling methods involving energy: energy-efficient parallel scheduling, energy-aware parallel scheduling and energy-conscious parallel scheduling algorithms. The authors divide the algorithms into five groups: heuristic, meta-heuristic, integer programming, machine learning and game-theory algorithms. They also discuss in detail the computing system architectures and power and energy models.

3. Properties of optimal solutions

In the introduction we emphasized that the load balancing problem with joint job processing time $p = 1$ admits a universally optimal solution, i.e., one which is optimal for any convex cost function. The following example shows that for $p > 1$ this is not the case by providing two different convex functions with two different unique optimal solutions.

Example 1. There are 9 jobs, where J_1 and J_2 have release dates $r_1 = r_2 = 0$ and deadlines $d_1 = d_2 = 5$, and J_3 through J_8, have release dates $r_3 = \cdots = r_8 = 5$ and deadlines $d_3 = \cdots = d_8 = 10$. Furthermore, job J_9 has release date $r_9 = 0$ and deadline $d_9 = 14$. The processing time of all the jobs is $p = 5$. Notice that in a feasible schedule, the place of jobs $J_1, \ldots, J_8$ is fixed: J_1 and J_2 are processed from 0 to 5, and $J_3, \ldots, J_8$ are processed from 5 to 10. The feasible schedules differ only in the starting time of J_9. Firstly, we want to minimize the function $f_1(x) = x^2$ of the load. It is not difficult to check that in the optimal solution, J_9 is processed in the interval $[9, 14]$. Let S_1 denote this solution, and the cost of S_1 is $5 \cdot 2^2 + 4 \cdot 6^2 + 7^2 + 4 \cdot 1^2 = 217$. We get a feasible, but not optimal solution by processing J_9 in $[0, 5]$. Let S_2 denote the solution we get in this way. The cost of S_2 with respect to f_1 is $5 \cdot 3^2 + 5 \cdot 6^2 = 225 > 217$, and indeed, S_2 is not an optimal solution w.r.t. f_1. Now consider the convex function $f_2(x) = 0$ for $x \le 6$, and $f_2(x) = x - 6$ for $x \ge 6$. It is easy to show that S_2 is the only optimal solution w.r.t. f_2.

The example above suggests that the approaches for $p = 1$ may not be straightforwardly generalized for $p > 1$.

4. A combinatorial approach for problem P_1

We first give a convex programming formulation for problem P_1 in Section 4.1. Then, we sketch a method for solving the convex program in Section 4.2. For efficient implementation, we show that the problem can be equivalently described as a minimum-cost circulation problem in a network with piecewise-linear convex cost functions on the arcs (Section 4.3), and then how to determine an optimal solution for our scheduling problem from an optimal circulation (Section 4.4). We also implemented our method and conducted some computational experiments on randomly generated test instances (Section 4.5). Based on the above results, we propose a new combinatorial algorithm for a parallel machine scheduling problem with one additional resource (Section 4.6).

4.1. Initial problem formulation

Firstly, we introduce additional notation. Let $\mathcal{I} = \{I_1, I_2, \ldots, I_L\}$ be the set of all different time slots of length p in

$$\bigcup_{i=1}^{n} \Big(\{[\, r_i + kp, r_i + kp + p] \mid k \in \mathbb{Z},\ r_{\min} \le r_i + kp \le d_{\max} - p\} \\ \cup \{[\, d_i + kp, d_i + kp + p] \mid k \in \mathbb{Z},\ r_{\min} \le d_i + kp \le d_{\max} - p\} \Big). \tag{1}$$

We assume that the left endpoint of I_k is smaller than that of I_{k+1} for all $k = 1, \ldots, L-1$. The following lemma states an important property about the structure of an optimal schedule, and it generalizes Lemma 3 of Baptiste (2000).

Lemma 1. *There always exists an optimal schedule, where each job is processed in one of the intervals in $\mathcal{I}$.*

Proof. Suppose the statement of the lemma does not hold for a problem instance. Let $\mathcal{S}^*$ be an optimal schedule such that the number of jobs which are not scheduled in some time slots in $\mathcal{I}$ is minimal. Let $\mathcal{J}'$ be the subset of all those jobs that are not scheduled in some time slots in $\mathcal{I}$ by $\mathcal{S}^*$.

Let $\delta \in \mathbb{R}^{n+1}$ be a vector representing the load of the resource in $\mathcal{S}^*$, that is, for $\ell \in \{0, \ldots, n\}$, $\delta(\ell)$ equals the total size of time intervals in which the load of the resource is ℓ. Clearly, $\sum_{\ell=0}^{n} \ell \cdot \delta(\ell) = n \cdot p$, and the cost of $\mathcal{S}^*$ is $\sum_{\ell=0}^{n} f(\ell)\delta(\ell)$.

Let $\epsilon > 0$ be the smallest value such that starting all the jobs in $\mathcal{J}'$ by ϵ time earlier, or later, at least one of the jobs in $\mathcal{J}'$ is scheduled in a time slot $I \in \mathcal{I}$. Let $\mathcal{S}_1$ be the resulting schedule. Such a shift induces a vector $\gamma \in \mathbb{R}^{n+1}$ such that $\delta + \gamma$ represents the load of the resource in schedule $\mathcal{S}_1$.

The costs of the schedules $\mathcal{S}^*$ and $\mathcal{S}_1$ are related in the following way:

$$cost(\mathcal{S}_1) = cost(\mathcal{S}^*) + \Delta,$$

where $\Delta = \sum_{\ell=0}^{n} f(\ell)\gamma(\ell)$.

Since $\sum_{\ell=0}^{n} \ell \cdot \gamma(\ell) = 0$ must hold, $\delta - \gamma$ also represents the resource usage of some feasible schedule $\mathcal{S}_2$, namely, the one obtained by shifting all jobs in $\mathcal{J}'$ in the opposite direction by ϵ. The costs of $\mathcal{S}^*$ and $\mathcal{S}_2$ are related in the following way:

$$cost(\mathcal{S}_2) = cost(\mathcal{S}^*) - \Delta.$$

Unless $\Delta = 0$, this implies that $\mathcal{S}^*$ is not optimal. Hence, $\Delta = 0$, and $\mathcal{S}_1$ is an optimal schedule in which some jobs in $\mathcal{J}'$ are scheduled in a time slot in $\mathcal{I}$, which contradicts the choice of $\mathcal{S}^*$. □

Let $C = \{c_0, c_1, \ldots, c_H\}$ be the ordered set of all different left and right endpoints of $I_1, I_2, \ldots, I_L$ such that $c_0 = r_{\min}$ and $c_H = d_{\max}$. Let K_h denote the interval $[\,c_{h-1}, c_h]$ for $h = 1, \ldots, H$.

For any subset X of the jobs, let $N(X)$ consist of all those time slots $I_k \in \mathcal{I}$, which are feasible for at least one of the jobs in X, that is, $N(X) = \{I_k \mid I_k \subseteq [r_i, d_i] \text{ for some } J_i \in X\}$. We say that $N(X)$ is *connected*, if there exist $1 \le a \le b \le L$ such that $N(X) = \{I_a, \ldots, I_b\}$, see Fig. 1 for an illustration.

Let $\mathcal{X}$ denote those subsets X of $\mathcal{J}$ such that $N(X)$ is connected.

In our formulation we have the following three types of variables:

- x_k: the number of jobs processed in time slot I_k;
- b_X: the number of jobs scheduled in the time slots $N(X)$;
- t_h: the number of jobs processed in interval K_h, that is, $t_h = \sum_{I_k \supset K_h} x_k$.

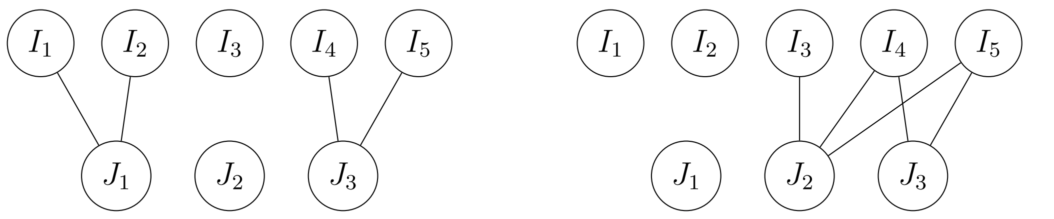
Consider the following mathematical program:

$$\text{minimize} \sum_{h=1}^{H} |K_h| f(t_h) \tag{2a}$$

$$\text{s.t.} \sum_{I_k \in N(X)} x_k - b_X = 0, \quad \text{for all } X \in \mathcal{X} \tag{2b}$$

$$b_X \ge |X|, \quad \text{for all } X \in \mathcal{X} \tag{2c}$$

$$(IP): \quad b_{\mathcal{J}} = n \tag{2d}$$



(a) Suppose $X = \{J_1,\ J_2\}$, then $N(X) = \{I_1,\ I_2,\ I_4,\ I_5\}$. $N(X)$ is not connected.

(b) Suppose $X = \{J_2,\ J_3\}$, then $N(X) = \{I_3,\ I_4,\ I_5\}$. $N(X)$ is connected.

Fig. 1. Suppose we have jobs J_1, J_2 and J_3. Time slots I_1 and I_2 are feasible for J_1, time slots I_3, I_4 and I_5 are feasible for J_2 and time slots I_4 and I_5 are feasible for J_3. Then for $X = \{J_1,\ J_3\}$, $N(X)$ is not connected and for $X = \{J_2,\ J_3\}$, $N(X)$ is connected.

$$\sum_{k:I_k \supseteq K_h} x_k - t_h = 0, \quad \text{for each interval } K_h \tag{2e}$$

$$x_k \geq 0, \quad \text{for all } k = 1,\dots,L \tag{2f}$$

$$x_k \in \mathbb{Z}, \quad \text{for all } k = 1,\dots,L. \tag{2g}$$

Note that $|K_h| = c_h - c_{h-1}$, while $|X|$ denotes the cardinality of X. In order to show that (IP) is a proper formulation for problem P_1, we define the bipartite graph $G_{(x,b,t)} = (V_{\mathcal{I}} \cup V_{\mathcal{J}}, E)$ for a feasible solution (x,b,t) of (IP). For each job $J \in \mathcal{J}$, $V_{\mathcal{J}}$ contains a unique node v_J, and for each time slot $I_k \in \mathcal{I}$, $V_{\mathcal{I}}$ contains x_k nodes $v_k^{(1)},\dots,v_k^{(x_k)}$ corresponding to I_k. For each job J_i and $I_k \subset [r_i, d_i]$, E contains the edge $(v_{J_i}, v_k^{(\ell)})$ for $\ell = 1,\dots,x_k$. Since $\sum_{k=1}^L x_k = n$ by (2b) and (2d), $|V_{\mathcal{J}}| = |V_{\mathcal{I}}| = n$.

Recall that a matching in a bipartite graph is a subset of node-disjoint edges. A matching is *perfect* if it covers all the nodes. We will prove the following result:

Theorem 1. *If (x,b,t) is an optimal solution to (IP), then $G_{(x,b,t)}$ admits a perfect matching and any perfect matching in $G_{(x,b,t)}$ corresponds to an optimal schedule. Conversely, any optimal schedule $\mathcal{S}$ induces an optimal solution (x,b,t) to (IP).*

For any $X \subseteq \mathcal{J}$, let V_X denote the set of nodes $\{v_J \mid J \in X\}$, and $N(V_X)$ the set of time slot nodes adjacent to any node in V_X in $G_{(x,b,t)}$. Then, the constraints in (2b) and (2c) ensure that for any subset of jobs $X \in \mathcal{X}$, $N(V_X) \geq |V_X|$. The following result shows that this condition holds for all nonempty subsets of the nodes, not only for those in $\mathcal{X}$.

Lemma 2. *Let (x,b,t) be a feasible solution to (IP). Then, for every nonempty subset X of the jobs $\mathcal{J}$, $|N(V_X)| \geq |V_X|$.*

Proof. Let $X \subseteq \mathcal{J}$ be arbitrary subset of the jobs and let V_X denote the corresponding subset of nodes in $V_{\mathcal{J}}$. Let $N(V_X) \subseteq V_{\mathcal{I}}$ denote the nodes that are adjacent to at least one node in V_X. Then $N(V_X)$ can be partitioned into $N(V_{X_1}), N(V_{X_2}),\dots,N(V_{X_r})$ such that $X = X_1 \cup X_2 \cdots \cup X_r$, the X_l are disjoint and $N(X_l)$ is connected for each $l = 1,\dots,r$. Since (x,b,t) is a feasible solution to (IP), $|N(V_{X_i})| \geq |V_{X_i}|$ holds for all $i = 1,2,\dots r$, and $|N(V_X)| \geq |V_X|$ follows. □

The following result is well-known.

Theorem 2 (*Hall's Condition for Bipartite Graphs*)**.** *Given a bipartite graph with bipartition A, B, there is a matching of size $|A|$ if and only if every subset $S \subseteq A$ is connected to at least $|S|$ vertices in B.*

Lemma 3. *Let (x,b,t) be a feasible solution to (IP). Then $G_{(x,b,t)}$ admits a perfect matching.*

Proof. The statement follows from Lemma 2 and from Hall's condition. □

Now we will map feasible solutions of (IP) to feasible schedules of the same cost and vice versa.

Algorithm 1 Calculation of optimal schedule

```
Require: $n \geq 1$, $p \geq 1$, $\{r_i, d_i\}$ for $i = 1,\dots,n$, function $f$.
Ensure: Optimal schedule $\mathcal{S}$.
1: Solve $(IP)$, and let $(x,b,t)$ be an optimal solution.
2: Define the graph $G_{(x,b,t)}$, and find a perfect matching $M$ in it.
3: Construct schedule $\mathcal{S}$ by assigning the jobs to the time slots as
   specified by $M$.
```

Lemma 4. *For any feasible solution (x,b,t) of (IP), there is a feasible schedule, where x_k jobs are processed in time slot I_k and the load of the interval K_h is t_h. Conversely, from every feasible schedule, where the jobs are scheduled in the time slots of $\mathcal{I}$, one can obtain a solution (x,b,t) of identical cost for (IP) satisfying (2b)–(2g).*

Proof. To prove the first part of the lemma, suppose (x,b,t) satisfies (2b)–(2g). Then by Lemma 3, $G_{(x,b,t)}$ admits a perfect matching M. If $(v_J, v_I) \in M$, schedule job J in time slot I. Then for any $I_k \in \mathcal{I}$, the number of jobs processed in I_k is the number of nodes in $V_{\mathcal{I}}$ representing I_k which is exactly x_k. t_h represents the load of interval K_h in the solution by Eq. (2e), therefore, the load of K_h is t_h in the schedule.

To show the second part, let $\mathcal{S}$ denote a feasible schedule. For all time slots $I_k \in \mathcal{I}$, let x_k denote the number of jobs processed in I_k, and for each interval K_h, $h = 1,\dots,H$, let t_h denote the number of jobs that are executed during K_h. For an arbitrary $X \in \mathcal{X}$, let b_X denote the total number of jobs that are processed in the time slots of $N(X)$. Then (x,b,t) satisfies (2b)–(2g) and it has the same cost as $\mathcal{S}$. □

As a corollary to Lemma 4 we get Theorem 1.

4.2. Solution method

By Theorem 1, we can solve the scheduling problem by Algorithm 1. Since finding a perfect matching in a bipartite graph can be done in polynomial time (Ahuja et al., 1993), it remains to solve (IP) efficiently. In the remainder of this section, we sketch our approach for solving (IP) in polynomial time by a combinatorial method based on network flows.

Firstly, we observe that the size of (IP) can be polinomially bounded in the size of the input.

Lemma 5. *The size of the linear system (2b)–(2g) is polynomial in the size of the input.*

Proof. To prove that the size of system (2b)–(2g) is polynomial in the size of the input, it is enough to show that the number of constraints in (2c) and (2e) is polynomial in the input size. Observe that $L = |\mathcal{I}| = \mathcal{O}(n^2)$. All $X \in \mathcal{X}$ can be constructed the following way. For each pair of intervals (I_k, I_ℓ) with $k \leq \ell$ consider the set X of those jobs J_i such that r_i is not before the left endpoint of I_k and d_i is not after the right endpoint of I_ℓ. Then $X \in \mathcal{X}$ if and only if $N(X) = \{I_j \mid k \le j \le \ell\}$. It follows that the size of $\mathcal{X}$ is $\mathcal{O}(L^2)$. There are H intervals K_h and the endpoints of each of them coincide with endpoints of time slots in $\mathcal{I}$, hence, $H = \mathcal{O}(L)$ and the number of constraints in (2e) is polynomial in the input size. □

Since we are only interested in the values of f at integer loads only, we can replace f with a piecewise-linear convex function $\tilde f$ with integer break points, where $\tilde f(z) = f(z)$ for all $z \in \mathbb{Z}_{\ge 0}$. Furthermore, one can assume that the first break point of $\tilde f$ is at 0 and the last one is at most n, since there are n jobs. It is not difficult to see that for such an $\tilde f$, the optimal value of (IP) coincides with the optimal value of

$$(IPWL): \qquad \text{minimize} \sum_h |K_h| \tilde f(t_h)$$

s.t. (2b) – (2g).

In fact, the matrix of (2b)–(2f) is totally unimodular, see Lemma 6 and Corollary 1, which permits to get rid of the integrality condition (2g) by a result of Meyer. Meyer (1977) showed that an optimization problem with a separable piecewise-linear convex cost function with integer break points, and a totally unimodular constraint matrix always admits an integer optimal solution. This means that in (IPWL), constraint (2g) can be omitted, while preserving integer optimal solutions:

$$(PWL): \qquad \text{minimize} \sum_h |K_h| \tilde f(t_h)$$

s.t. (2b) – (2f).

Karzanov and McCormick (1997) describe efficient polynomial time algorithms for minimizing a separable convex cost function over the linear space $Mx = 0$, provided M is a totally unimodular matrix. More specifically, for every coordinate x_e of x, there is a convex function $w_e : \mathbb{R} \to \mathbb{R}$. If E is the set of coordinates of x, the problem is to find x such that $Mx = 0$, while $\sum_{e \in E} w_e(x_e)$ is minimized. It is assumed that $\{x : Mx = 0\}$ contains a non-zero point and the minimization problem has a finite optimal solution. The authors show how to solve the problem efficiently for different classes of convex functions, assuming there is an oracle that solves the following problems:

1. given a point $r \in \mathbb{R}$, return $c_e^{\vdash}(r)$ and $c_e^{\dashv}(r)$, where $c_e^{\vdash}(r)$ and $c_e^{\dashv}(r)$ are the right and left derivatives of w_e at r. The convexity of w_e implies the existence of the right and left derivatives;
2. given a slope $s \in \mathbb{R}$, return a point r with $c_e^{\dashv}(r) \le s \le c_e^{\vdash}(r)$.

In the case of piecewise-linear convex functions, such an oracle is easy to implement.

Karzanov and McCormick (1997) propose two different algorithms for solving such problems: the Minimum Mean Canceling Method (MMCM) and the Cancel and Tighten method. Both methods are iterative, and in the most general case, both algorithms solve a linear program in each iteration. Although the number of iterations is polynomial in the size of the input, the running time is not as impressive due to solving linear programs. However, when $\{x : Mx = 0\}$ is the space of circulations in a graph G (that is, M is the node-edge incidence matrix of G), then there is no need to solve linear programs, and each iteration of the Cancel and Tighten algorithm takes $\mathcal{O}(|E(G)| \log |V(G)|)$ time, and the total number of iterations is $\mathcal{O}(|V(G)| \log(|V(G)|C))$, where C denotes the absolutely largest finite slope. The impressive running time motivates the question whether our problem can be reformulated as a minimum cost circulation problem in a network with piecewise-linear convex cost functions on the arcs.

4.3. Reformulation as a circulation problem in a network

First of all, observe that in the objective function of (PWL), we have convex functions only for variables t_h, and the system has lower or upper bound constraints for variables x_k and b_X. The function corresponding to t_h is $|K_h|\tilde f(\cdot)$. The breakpoints of $\tilde f(\cdot)$ are $0, 1, \dots, n$, and the slopes between these breakpoints are $s_\ell = \tilde f(\ell) - \tilde f(\ell - 1)$ for $\ell = 1, \dots, n$, noting that $s_1 \le s_2 \le \dots \le s_n$, since $\tilde f$ is convex (if $s_\ell = s_{\ell+1}$ then b_ℓ is not a real breakpoint, but it is not a problem). Define $s_{h,\ell}$ as $s_{h,\ell} = |K_h|(\tilde f(\ell) - \tilde f(\ell - 1))$ for $h = 1, \dots, H$, $\ell = 1, \dots, n$. Then we have

$$w_h(z) = \begin{cases} |K_h|\tilde f(0) + s_{h,1} z & \text{if } 0 \le z \le 1 \\ |K_h|\tilde f(1) + s_{h,2}(z-1) & \text{if } 1 \le z \le 2 \\ \dots & \\ |K_h|\tilde f(n-1) + s_{h,n}(z-n+1) & \text{if } n-1 \le z. \end{cases}$$

It is not difficult to show that we get an equivalent problem by introducing convex cost functions w_k and w_X for the variables x_k and b_X and moving the lower and upper bound constraints to w_k and w_X the following way. We define the symbol $K = +\infty$ with the following meaning. For any $x > 0$, $K \cdot x = K$, $K \cdot x = -K$ for any $x < 0$, and $K \cdot 0 = 0$. Moreover, $K + x = K$ for any real number x. In the computations, we will only compare a real number x to K, and we only need that $-K < x < K$.

Since we want a variable x_k to be non-negative, w_k has a breakpoint at 0 and the slope before 0 is $-K$ and the slope after 0 is 0, that is

$$w_k(z) = \begin{cases} -Kz & \text{if } z \le 0 \\ 0 & \text{if } z \ge 0. \end{cases}$$

Similarly, if $X \ne \mathcal{J}$, w_X has a breakpoint at $|X|$ and the slope before $|X|$ is $-K$ and after $|X|$ is 0, i.e.,

$$w_X(z) = \begin{cases} -K(z - |X|) & \text{if } z \le |X| \\ 0 & \text{if } z \ge |X|. \end{cases}$$

If $X = \mathcal{J}$, w_X has a breakpoint at n and the slope before n is $-K$ and after n it is K, that is

$$w_{\mathcal{J}}(z) = \begin{cases} -K(z-n) & \text{if } z \le n \\ K(z-n) & \text{if } z \ge n. \end{cases}$$

Therefore, (PWL) can be reformulated as

$$\begin{aligned} & \text{minimize} \sum_{k=1}^{L} w_k(x_k) + \sum_{X \in \mathcal{X}} w_X(b_X) + \sum_{h=1}^{H} w_h(t_h) \\ & \text{s.t.} \\ (PWL2): \quad & \sum_{I_k \in N(X)} x_k - b_X = 0 \quad \text{for all } X \in \mathcal{X}; \\ & \sum_{k : I_k \supseteq K_h} x_k - t_h = 0 \quad \text{for all intervals } K_h. \end{aligned}$$

The following lemma plays a crucial role in our method.

Lemma 6. *Let M denote the matrix of the following system:*

$$\sum_{I_k \in N(X)} x_k - b_X = 0 \quad \textit{for all } X \in \mathcal{X};$$
$$\sum_{k : I_k \supseteq K_h} x_k - t_h = 0 \quad \textit{for all intervals } K_h.$$

Then there exists a network $D = (V, E)$ and a tree $T = (V, E')$ such that $M^\top$ is the corresponding network matrix.

Proof. Observe that M can be written as $M = (A, -I)$, where $A \in \mathbb{R}^{a \times L}$ is an interval matrix and $-I \in \mathbb{R}^{a \times a}$ is a negative identity matrix, where $a = |\mathcal{X}| + H$ is the number of constraints in (2b) and (2e). Therefore, if y is a column of $M^\top$, then y has some 1 entries in consecutive positions in the first L coordinates, and in the remaining a coordinates, y has a unique -1 entry. All other coordinates of y are 0. We construct a network $D = (V, E)$ and a spanning tree $T = (V, E')$ with $E' \subset E$ and show that each edge in $E \setminus E'$ corresponds to a column of $M^\top$.

Let $P = v_0 \to v_1 \to \cdots \to v_L$ denote a directed path of length L, where the kth edge $v_{k-1} \to v_k$ represents the kth p-length time slot I_k, and we say it corresponds to variable x_k. At the beginning,

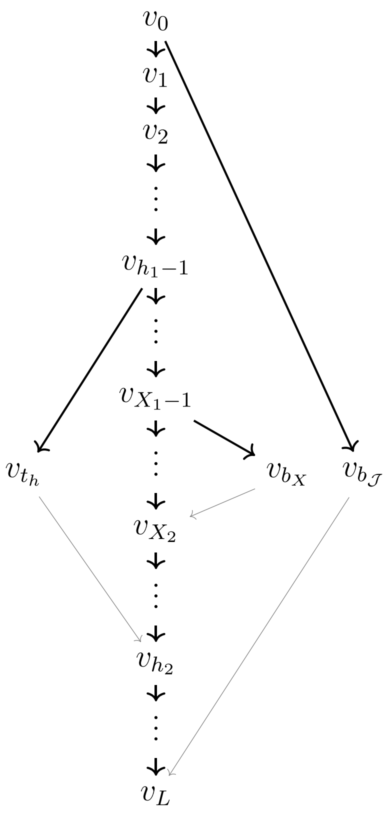


Fig. 2. Network D. Thin arcs correspond to non-tree edges, thick arcs are the edges of T.

$V = \{v_0, v_1, \dots, v_L\}$ and $E = E' = \{(v_{i-1}, v_i), i = 1, \dots, L\}$. Then we add new nodes and edges to $D = (V, E)$ and $T = (V, E')$ the following way: we take all constraints from (2b) and (2e) one by one, and for each of them, we connect nodes representing the first and the last time slot in the constraint with a new path of length 2. More precisely, consider constraints in (2b) and suppose equation $\sum_{I_k \in N(X)} x_k - b_X = 0$ is one of them. We add a new node to V denoted by v_{b_X}. Let X_1 denote the index of the first time slot and X_2 the index of the last time slot in $N(X)$. We add edges (v_{X_1-1}, v_{b_X}) and (v_{b_X}, v_{X_2}) to E. We extend the set of tree edges E' with the first new edge (v_{X_1-1}, v_{b_X}), and we say that the tree edge (v_{X_1-1}, v_{b_X}) corresponds to the variable b_X. The other new edge (v_{b_X}, v_{X_2}) becomes a non-tree edge. We proceed similarly with the equations in (2e) of the form $\sum_{I_k \supseteq K_h} x_k - t_h = 0$. That is, we connect the first and the last time slot in $\{I_k \mid I_k \supseteq K_h\}$ with a 2-length path containing two new edges and extend tree E' with the first new edge in the same way as before. Fig. 2 illustrates the network constructed this way. It is not difficult to check that a column in $M^\top$ corresponding to a constraint from (2b) or (2e) is represented by a non-tree edge in the network. $\square$

Corollary 1. *The matrix of* (2b)–(2f) *is totally unimodular.*

Proof. It is known that a matrix A is totally unimodular if and only if the matrix $[A^\top, -A^\top, I, -I]^\top$ is totally unimodular (see e.g., Schrijver's book (Schrijver, 1998), page 268). In our case the matrix A consists of the left-hand-side of the Eqs. (2b) and (2e), which by Lemma 6 is totally unimodular. Since the matrix of (2b)–(2f) is a submatrix of $[A^\top, -A^\top, I, -I]^\top$, it is totally unimodular. $\square$

The tree edges of D corresponding to variables x_k, b_X and t_h are denoted by e_{x_k}, e_{b_X} and e_{t_h}, respectively. A non-tree edge derived from a constraint of type (2b) is denoted by e_X and a non-tree edge derived from a constraint of type (2e) is denoted by e_h. We will dualize the problem and solve the dual problem instead of the original primal formulation. To this end, we need dual variables for non-tree edges represented by the rows of M. For a non-tree edge e_X, let z_X denote the corresponding dual variable and for a non-tree edge e_h, the corresponding dual variable is z_h. For non-tree edges e_X and e_h, let $C(e_X)$ and $C(e_h)$ denote the tree edges in the corresponding fundamental cycles,[1] respectively.

To express the dual problem of $(PWL2)$ as a network circulation problem, we introduce the following piecewise-linear functions. Recall the definition of the values $s_{h,\ell}$ at the beginning of this section. For $h = 1, \dots, H$, we have

$$w_h^*(z) = \begin{cases} 0, & \text{if } z \le s_{h,1}, \\ z - s_{h,1}, & \text{if } s_{h,1} \le z \le s_{h,2}, \\ -s_{h,1} + s_{h,2} + 2(z - s_{h,2}) & \text{if } s_{h,2} \le z \le s_{h,3}, \\ \dots & \\ -\sum_{i=1}^{r-1} s_{h,i} + (r-1)s_{h,r} + r(z - s_{h,r}), & \text{if } s_{h,r} \le z \le s_{h,r+1}, \\ \dots & \\ -\sum_{i=1}^{n-1} s_{h,i} + (n-1)s_{h,n} + n(z - s_{h,n}), & \text{if } z \ge s_{h,n}. \end{cases}$$

If $X \neq \mathcal{J}$, let w_X^* be

$$w_X^*(z) = -|X|z, \qquad 0 \le z \le K,$$

and if $X = \mathcal{J}$,

$$w_X^*(z) = -nz, \qquad -K \le z \le K,$$

where K is a sufficiently large positive number.

Proposition 1. *The dual of* $(PWL2)$ *is the convex program*

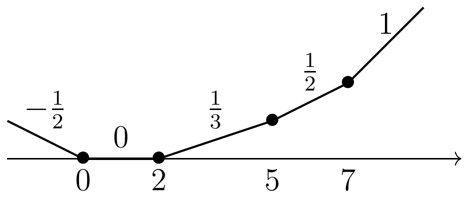
$$(D - PWL2) \quad \begin{aligned} \textit{minimize} \quad & \sum_{X \in \mathcal{X}} w_X^*(z_X) + \sum_{h=1}^{H} w_h^*(-z_h) \\ \textit{s.t.} \quad & \sum_{e_X : e_{x_k} \in C(e_X)} z_X + \sum_{e_h : e_{x_k} \in C(e_h)} z_h + \lambda_k = 0, \\ & \lambda_k \ge 0, \ \textit{for all } k = 1, \dots, L \\ & 0 \le z_X \le K, \quad \textit{for all } X \in \mathcal{X}, \ X \neq \mathcal{J} \\ & -K \le z_{\mathcal{J}} \le K. \end{aligned}$$

For the proof, see Appendix.

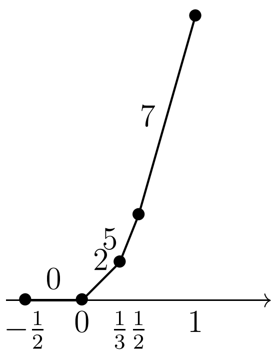
Remark 2. It is known (see e.g. Karzanov and McCormick (1997)) that the dual $\tilde{f}^*(\cdot)$ of a piecewise-linear convex function $\tilde{f}(\cdot)$ is obtained by exchanging the slopes and breakpoints of $\tilde{f}$, that is, the slopes of $\tilde{f}$ will be the breakpoints of $\tilde{f}^*$ and the breakpoints of $\tilde{f}$ will be the slopes $\tilde{f}^*$, see Fig. 3 for an illustration. Note that our piecewise-linear functions have the same relationship: w_h^* and w_h are dual functions, as are w_X^* and w_X.

Since the matrix of problem $(PWL2)$ is the transpose of a network matrix, $(D - PWL2)$ is a network circulation problem. However, a notable issue arises in its objective function, where we encounter $w_h^*(-z_h)$, indicating the substitution of the negative of z_h in the convex function w_h^*. However, it is easy to overcome this issue by reversing the tree edges e_{t_h}. Therefore, our final network has the same set of nodes and arcs as D, except that the arcs e_{t_h} are in opposite direction. Next, we define lower and upper bounds for the edges, together with edge

[1] Suppose we are given a network $D = (V, E)$ together with one of its spanning tree $T = (V, E')$. By taking an edge $e \in E \setminus E'$ and adding it to E', the resulting graph contains a unique cycle $C(e)$, which is called the fundamental cycle of e corresponding to the spanning tree T.



(a) A piecewise-linear convex function with breakpoints at 0, 2, 5, 7 and slopes $-\frac{1}{2}$, 0, $\frac{1}{3}$, $\frac{1}{2}$ and 1.



(b) The dual piecewise-linear convex function. The breakpoints are the slopes and the slopes are the breakpoints of the function in Fig. 3a.

Fig. 3. A piecewise-linear convex function and its dual.

costs. For a tree edge e_{x_k}, let the lower bound for the flow value λ_k be 0 and there is no upper bound, while the cost is 0. For a tree edge e_{b_X}, let the cost function be the piecewise-linear w^*_X and if $X \neq \mathcal{J}$, the lower bound is 0 and the upper bound is K, and when $X = \mathcal{J}$, the lower bound is $-K$ and the upper bound is K. For a tree edge e_{t_h}, we have no lower or upper bounds and the cost function is the piecewise-linear w^*_h. There are no lower or upper bounds for the flows on non-tree edges e_X and e_h in D, and the cost function is 0. Then Proposition 2 follows immediately.

Proposition 2. *The minimum cost circulation problem defined above is equivalent to the problem $(D-PWL2)$. The cost of a minimum cost circulation is equal to the optimal value of $(D-PWL2)$.*

The minimum cost circulation problem defined above can be solved by the Cancel and Tighten method of Karzanov and McCormick (1997). Remember that the number of iterations is $\mathcal{O}(|V(G)|\log(|V(G)|C))$, and one iteration takes $\mathcal{O}(|E(G)|\log|V(G)|)$ time. Observe that in our case $C = n$, by the definition of w^*_h and w^*_X, noting that $X \subseteq \mathcal{J}$ and $n = |\mathcal{J}|$. It can be assumed that $\mathcal{I}$ contains at most $4n$ different p-length time slot for each job. Hence, the number of constraints in (2b) and (2e) is $\mathcal{O}(n^4)$. Hence, $|V(G)| = \mathcal{O}(n^4)$ and $|E(G)| = \mathcal{O}(n^4)$ in our case. Proposition 3 states the running time of the Cancel and Tighten method applied to our network.

Proposition 3. *Using the Cancel and Tighten algorithm presented in Karzanov and McCormick (1997), $(D-PWL2)$ can be solved in $\mathcal{O}(n^8(\log n)^2)$ time.*

4.4. Solution of the primal problem $(PWL2)$

By Proposition 2, an optimal solution to $(D-PWL2)$ can be obtained by solving a minimum cost network circulation problem with convex cost functions on the arcs, using an algorithm of Karzanov and McCormick (1997). It remains to show how to determine the primal optimal solution for $(PWL2)$. Since the dual space of circulations is the space of co-circulations, the optimal primal solution is represented by an appropriate co-circulation. One can read out the following lemma from Karzanov and McCormick (1997).

Lemma 7. *Let x^*_e, $e \in E$ denote an optimal solution to the minimum cost circulation problem. If h_e, $e \in E$ is a co-circulation satisfying $c^{\dashv}_e(x^*_e) \le h_e \le c^{\vdash}_e(x^*_e)$ for all $e \in E$, then h_e, $e \in E$ is an optimal solution to the dual problem, where $c^{\dashv}_e$ and $c^{\vdash}_e$ are the corresponding left and right derivatives, respectively.*

Given an optimal solution x^* of the minimum cost circulation problem. Using Lemma 7, we can compute an optimal co-circulation as follows. For $e \in E$, let u_e and v_e denote the starting and ending node of e, respectively. Then finding a co-circulation satisfying $c^{\dashv}_e(x^*_e) \le h_e \le c^{\vdash}_e(x^*_e)$, $e \in E$ is equivalent to finding node potentials π satisfying $\pi(v_e) - \pi(u_e) \le c^{\vdash}_e(x^*_e)$ and $\pi(u_e) - \pi(v_e) \le -c^{\dashv}_e(x^*_e)$ for all $e \in E$. Such node potentials π can be found by a shortest path algorithm in the directed graph we obtain by adding each edge $e \in E$ to the graph in reverse orientation as well. The edge lengths are $c^{\vdash}_e(x^*_e)$ for the original graph edges and $-c^{\dashv}_e(x^*_e)$ for the edges with reverse orientation. Since $c^{\vdash}_e(x^*_e)$ and $c^{\dashv}_e(x^*_e)$ are integer values in our case, the optimal co-circulation found by a shortest path algorithm is integer as well.

4.5. Preliminary computational results

We have implemented Algorithm 1 including the Cancel and Tighten method in C++ for solving Problem P_1 on randomly generated problem instances. The goal of the computational tests was to evaluate the impact of two problem parameters on the running time of the method. The two parameters were the number of the jobs n, and the ratio between the joint processing time and the size of the time windows of the jobs, i.e, $q = p/(d_i - r_i)$. We generated three problem instances for each combination $(n, q) \in \{20, 50, 100\} \times \{0.1, 0.4\}$. The joint job processing time was $p = 8$. The time horizon spanned 100 time units, and for each job J_i the release date and deadline satisfied the constraint $r_i \ge 0$, $d_i = \lceil r_i + p/q \rceil$, and $d_i \le 100$. We used the same piecewise-linear convex function f in all cases, where $f(0) = 0$, f has breakpoints at $1, 2, \ldots, n$, and the slope after breakpoint i is i.

The code was compiled with Visual Studio 2019, and the tests were run on a notebook computer with Intel Core I7 processor and Windows 11. We summarize the computational results in Table 1. We provide averages (rounded to the nearest integers) over 3 problem instances for each combination of the parameters n and q. In each case, we supply the average number of nodes and edges of the network D, the average number of iterations of the algorithm, the average CPU time, and also the average optimum values. All values are rounded to the nearest integer. As can be seen, the CPU time is strongly correlated with the number of edges of the graph. On the other hand, the number of iterations slightly increases with the number of jobs, but it is smaller for the same number of jobs for $q = 0.4$ than for $q = 0.1$. In fact, this is what we expected, since problem instances with a higher q allow less freedom in choosing when to start the jobs.

4.6. Application to a parallel machine scheduling problem to minimize the total completion time of the jobs

In this subsection we show how to apply the techniques presented earlier to a problem investigated by Brucker and Kravchenko (2008).

Table 1
Preliminary computational results for randomly generated inputs. n is the number of the jobs and q is the ratio of the job length and the size of the time windows of the jobs.

	$n = 20$ $q = 0.1$	$n = 20$ $q = 0.4$	$n = 50$ $q = 0.1$	$n = 50$ $q = 0.4$	$n = 100$ $q = 0.1$	$n = 100$ $q = 0.4$
Avg. number of graph nodes	87	86	92	89	92	92
Avg. number of graph edges	289	374	360	924	379	1785
Avg. number of iterations	1275	1102	1511	1293	1633	1493
Avg. optimal objective value	223	229	1028	1071	3685	3804
Avg. CPU time (ms)	36	58	52	74	59	135

Given n jobs with the same processing time p, each of them having a release date and deadline. In addition, there are m identical parallel machines. In a feasible schedule each job is processed between its release date and deadline on one of the machines, and at most m jobs are processed concurrently. The goal is to find a feasible schedule, if one exists, that minimizes $\sum C_i$, where C_i denotes the completion time of J_i.

We formulate the problem similarly to (IP). Recall the definitions of the set of time slots $\mathcal{I}$, and set of intervals $\{K_h\}_{h=1,\dots,H}$. Let $c(I_k)$ denote the right endpoint of $I_k \in \mathcal{I}$. Variables x_k, b_X and t_h denote the same quantities as in (IP).

$$(IP'):\quad \begin{aligned}
&\text{minimize} \sum_{k=1}^{L} c(I_k)x_k && \text{(3a)}\\
&\text{s.t.} \sum_{I_k\in N(X)} x_k - b_X = 0, \quad \text{for all } X \in \mathcal{X} && \text{(3b)}\\
&b_X \ge |X|, \quad \text{for all } X \in \mathcal{X} && \text{(3c)}\\
&b_{\mathcal{J}} = n && \text{(3d)}\\
&\sum_{k:I_k\supseteq K_h} x_k - t_h = 0, \quad \text{for each interval } K_h && \text{(3e)}\\
&t_h \le m, \quad \text{for all } h = 1,\dots,H && \text{(3f)}\\
&x_k \ge 0, \quad \text{for all } k = 1,\dots,L && \text{(3g)}\\
&x_k \in \mathbb{Z}, \quad \text{for all } k = 1,\dots,L. && \text{(3h)}
\end{aligned}$$

The objective function (3a) is a linear function expressing the total completion time of the jobs. The rest of the constraints are analogous to that of (IP). The sole difference is the upper bound (3f) on the variables t_h, potentially causing (IP') to become infeasible. When (IP') is feasible, for any feasible solution (x, b, t), we can construct a bipartite graph $G_{(x,b,t)} = (V_{\mathcal{I}} \cup V_{\mathcal{J}}, E)$ following the method outlined in Section 4.1. Analogously to Lemma 2, it can be shown that $G_{(x,b,t)}$ possesses a perfect matching, and Theorem 1 holds. Consequently, efficiently solving the problem hinges on efficiently tackling (IP'). From Lemma 5, it follows that the size of (IP') is polynomial in the size of the input, and akin to Lemma 6, we can show that the transpose of the matrix of the system is a network matrix. Similarly to Corollary 1, we can prove that the matrix of (3) is totally unimodular, hence, the LP relaxation is either infeasible, or has an integer optimal solution. Since the cost functions on the arcs are linear functions, the dual is a circulation problem with linear arc costs. Hence, a minimum cost flow computation suffices to find an optimal solution, or prove that no feasible solution exists.

5. A solution to problem P_2

This section is devoted to problem P_2. We aim to schedule the jobs on m parallel machines in a way that the load of the resource is evenly balanced. We deal only with the case, where the processing time of the all jobs is $p = 1$, but the resource requirement of the jobs can be 0 or 1. We prove that this problem can be solved by a single minimum cost flow computation in a network with convex costs on the arcs in Section 5.1. Then, we apply our formulation to the decision problem of Błażewicz (1979) in Section 5.2.

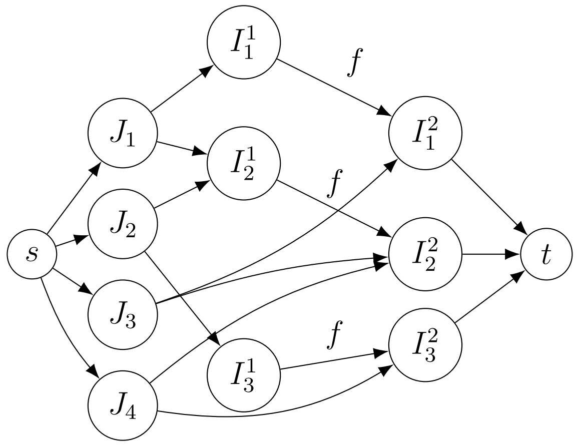


Fig. 4. Network D' for Problem P_2, where jobs J_1 and J_2 require one resource unit and jobs J_3 and J_4 require 0. The cost is measured by the convex function f on three edges, the remaining edges have zero cost.

5.1. A network flow formulation to solve P_2

In order to describe a network flow representation of the scheduling problem, we define the time slots for the jobs. Let $\mathcal{I}' = \{I_1, \dots, I_{L'}\}$ be the set of all different unit-length time slots in the set

$$\bigcup_{i=1}^{n} \Big\{ [r_i + k, r_i + k + 1] \mid \forall\, k \in \mathbb{Z} \\ \text{s.t. } 0 \le k \le \min\{n-1, d_i - r_i - 1\} \Big\}.$$

Given that r_i and d_i are assumed to be integers in problem P_2, for any two distinct indices $k \neq k'$, the intersection of I_k and $I_{k'}$ is either empty or comprises a single integer number. Note that for each job it suffices to consider only the first n unit-length time slots, since there are n jobs. We define the network D' as follows. For every job there is a job node in D'. For simplicity, the job nodes are denoted by $J_1, \dots, J_n$. For each $I_k \in \mathcal{I}'$, there are two time slot nodes in D' denoted by I_k^1 and I_k^2. Furthermore, there is a source node s and a sink node t. From s, there is an arc to every job node of capacity 1. If J_i requires 1 unit of the resource, that is, $\mu_i = 1$, then there are arcs from J_i to all the nodes I_k^1 such that I_k is feasible for J_i. When $\mu_i = 0$, there are arcs from J_i to all the nodes I_k^2, such that I_k is feasible for the job. All these arcs have infinite capacity. For all $I_k \in \mathcal{I}$, there is an arc from I_k^1 to I_k^2 of infinite capacity, and the cost function on the arc is f. The cost function on all other arcs is 0. Moreover, there is an arc from I_k^2 to t of capacity m. See Fig. 4 for an illustration.

Proposition 4. *The optimal feasible schedules are in one-to-one correspondence with the integral minimum cost feasible flows, where the total flow leaving s is n.*

Despite of D' having convex costs on some edges, a similar network having only linear costs can be constructed, and the problem can be solved by any minimum cost network flow algorithm.

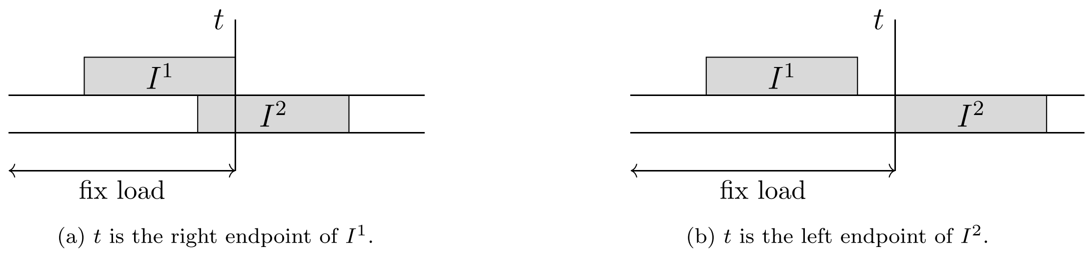

t
I^1
I^2
fix load
(a) t is the right endpoint of I^1.
t
I^1
I^2
fix load
(b) t is the left endpoint of I^2.

Fig. 5. The load is fixed until time point t, since any further jobs are scheduled later than t.

5.2. *Application to a scheduling problem with a resource of bounded capacity*

By choosing the convex function f properly, one can decide the feasibility problem considered by Błażewicz (1979) as described in Section 1. If c denotes the resource capacity, then let f denote the following piecewise-linear function: $f(x) = 0$ if $x \leq c$ and $f(x) = x - c$ if $x \geq c$, where c is the capacity of the resource. Then there exists a feasible schedule using m machines, where each job is processed in a time slot between its release date and deadline if and only if there is feasible flow of zero cost in the previously constructed network D'. Notice that it is not necessary to solve a flow problem with arc costs. Let the network D'' be obtained from D' by removing all arc costs and setting the capacity of the arcs from I^1_k to I^2_k to c. Then we have the following result.

Proposition 5. *The scheduling problem of Błażewicz (1979) with a bounded capacity resource admits a feasible solution if and only of the network D'' admits a feasible $s-t$ flow of value n.*

Given that finding a feasible $s-t$ flow with a value of n in a network can be reduced to a maximum-flow problem in the same network, and the latter problem can be solved in polynomial time (Ahuja et al., 1993), the scheduling problem of Błażewicz (1979) can be solved by a single maximum-flow computation in polynomial time.

6. Solution to $P3$ in special cases

In this section, we devise dynamic programming based algorithms to determine optimal solutions to $P3$ when there are only a constant number of machines, and the jobs satisfy some additional conditions. To simplify the presentation, we assume that there are 2 resource types and 2 machines, but our approach can be extended to any constant number of resource types and machines. We denote by $\mathcal{J}_1$ the subset of all those jobs that require resource type R_1 and by $\mathcal{J}_2$ the subset of all those jobs that require resource type R_2. We say that the jobs J_i and J_j have *agreeable deadlines* if $r_i \leq r_j$ implies $d_i \leq d_j$.

First, we show how to find an optimal schedule in polynomial time with dynamic programming if the jobs in $\mathcal{J}_1$ have agreeable deadlines and the jobs in $\mathcal{J}_2$ have agreeable deadlines. It is easy to see that under this condition there is an optimal schedule, where the jobs in $\mathcal{J}_1$ are scheduled in non-decreasing release date order, and the same holds for the jobs in $\mathcal{J}_2$. This observation plays a key role in our dynamic program. First we describe the algorithm for the case with only two machines and two resource types, then we show how to generalize the algorithm for a constant number of machines and resource types.

Second, we weaken the agreeable deadlines assumption. Suppose the job set $\mathcal{J}$ can be covered by ρ disjoint job chains $C_1, \ldots, C_\rho$, where ρ is a constant number, chain $C_i = \{J^i_1, J^i_2, \ldots, J^i_{|C_i|}\}$ is a set of jobs with $r^i_1 \leq r^i_2 \leq \cdots \leq r^i_{|C_i|}$ and $d^i_1 \leq d^i_2 \leq \cdots \leq d^i_{|C_i|}$, all jobs in C_i require the same resource, $C_i \cap C_j = \emptyset$ if $i \neq j$, and $\bigcup_{i=1}^{\rho} C_i = \mathcal{J}$. Observe that if J and J' are two jobs in the same chain and J precedes J' in the order, then there is an optimal schedule where J is scheduled no later than J'. We will show that if the jobs can be covered by a constant number of chains, an optimal solution can be computed in polynomial time, see Section 6.2. The proposed dynamic program is an extension of the one devised for the case with agreeable deadlines.

Finally, we generalize Lemma 1, which is needed to ensure a polynomial time complexity of our methods. Recall the definition (1) of the set of time slots $\mathcal{I}$ in Section 4.

Lemma 8. *There exists an optimal schedule for the problem P3 in which all jobs are scheduled in some time slot of $\mathcal{I}$.*

Proof. (sketch) Firstly, we introduce a new resource R_0 along with a function f_0 such that $f_0(\ell) = 0$ for $\ell \leq m$, and $f_0(\ell) = +\infty$ for $\ell > m$. Moreover, each job J_i requires R_0 in addition to R_{μ_i}. Then, we apply the proof technique of Lemma 1 with the extension that for each resource type R_k there is a distinct resource load function δ_k, $k \in \{0, \ldots, \nu\}$. With these extensions, the proof follows the same steps as that of Lemma 1. □

6.1. *Jobs with agreeable deadlines*

In this subsection we give a polynomial time algorithm that determines an optimal schedule in case of problem instances where jobs in $\mathcal{J}_1$ and jobs in $\mathcal{J}_2$ have agreeable deadlines, respectively. As it was mentioned before, in this case there is an optimal schedule, where the jobs in $\mathcal{J}_1$ and the jobs in $\mathcal{J}_2$ are scheduled in non-decreasing order of their release dates, respectively. Therefore, we will schedule the jobs in $\mathcal{J}_1$ as well as in $\mathcal{J}_2$ in non-decreasing release date order.

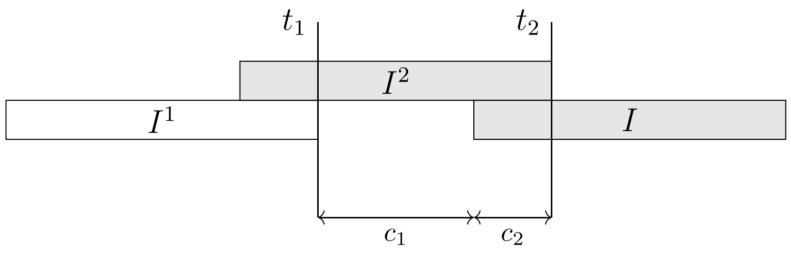
We build a schedule step-by-step. In each step, we assign the next job from $\mathcal{J}_1$ or $\mathcal{J}_2$ to the end of the schedule, i.e. to a feasible time slot that does not start earlier than the time slot of the job scheduled last. Let t be the starting time of the last scheduled job. Then the total cost up to time t can be computed, and will not change by scheduling the remaining jobs after time point t.

Our dynamic program builds a directed acyclic graph, where the nodes represent partial schedules with the same attributes. Node $S(J^1, I^1, J^2, I^2, i_1, i_2)$ corresponds to the set of all partial schedules, where:

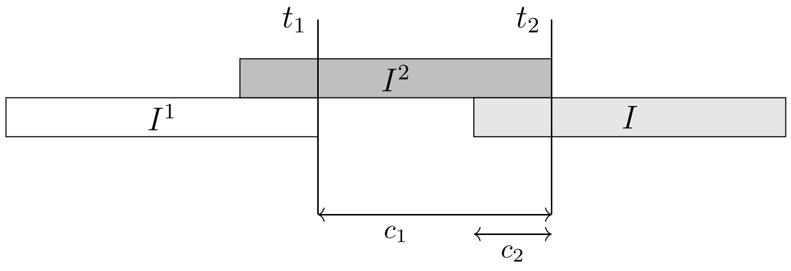
- I^1 and I^2 are the last occupied time slots on the two machines. We assume I^1 is not later than I^2;
- J^1 is the job in I^1, and J^2 is the job in I^2 (I^1 is feasible for J^1 and I^2 is feasible for J^2);
- The first i_1 jobs from $\mathcal{J}_1$ and the first i_2 jobs from $\mathcal{J}_2$ are scheduled.

We introduce a dummy job $J_\emptyset$ which does not require any resource and an "earliest" time slot I^0, where the right endpoint of I^0 is the left endpoint of the first real time slot I_1. This way, the state $S_0 = S(I^0, J_\emptyset, I^0, J_\emptyset, 0, 0)$ represents the empty schedule, and if one machine is empty, we set $I^1 = I^0$ and $J^1 = J_\emptyset$.

For partial schedules represented by $S(J^1, I^1, J^2, I^2, i_1, i_2)$, we can determine the latest time point $t(J^1, I^1, J^2, I^2, i_1, i_2)$ such that the load of the schedules until the time point $t(J^1, I^1, J^2, I^2, i_1, i_2)$ does not change anyhow we schedule the remaining jobs in time slots later than I^1 and I^2. Since I^1 is not later than I^2, this time point can



(a) When $J^2 \in \mathcal{J}_1$ and $J \in \mathcal{J}_1$, the cost increment can be computed as $c_1 + c_2$, where $c_1 = f_1(1) \cdot |[t_1,t_2] \cap (I^2 - I)|$ and $c_2 = f_1(2) \cdot |[t_1,t_2] \cap I^2 \cap I|$.



(b) When $J^2 \in \mathcal{J}_2$ and $J \in \mathcal{J}_1$, the cost increment can be computed as $c_1 + c_2$, where $c_1 = f_2(1) \cdot |[t_1,t_2] \cap I^2|$ and $c_2 = f_1(1) \cdot |[t_1,t_2] \cap I|$.

Fig. 6. Fix cost increment between t_1 and t_2 if $J \in \mathcal{J}_1$.

be determined as $t(J^1, I^1, J^2, I^2, i_1, i_2) = \max\{$left endpoint of I^2, right endpoint of $I^1\}$, see Fig. 5.

For a node $S(J^1, I^1, J^2, I^2, i_1, i_2)$, we construct the outgoing edges the following way. First, take the next job J from $\mathcal{J}_1$ or $\mathcal{J}_2$, and take all of its feasible time slots I, where I does not start earlier than $t(J^1, I^1, J^2, I^2, i_1, i_2)$. We schedule J in I, and the resulting state is $S(J^2, I^2, J, I, i_1+1, i_2)$, if $J \in \mathcal{J}_1$ and it is $S(J^2, I^2, J, I, i_1, i_2+1)$, if $J \in \mathcal{J}_2$. It remains to determine the value of the arc from $S(J^1, I^1, J^2, I^2, i_1, i_2)$ to $S(J^2, I^2, J, I, i_1+1, i_2)$ or $S(J^2, I^2, J, I, i_1, i_2+1)$. It can be computed as the cost of the load between time points $t_1 = t(J^1, I^1, J^2, I^2, i_1, i_2)$ and $t_2 = t(J^2, I^2, J, I, i_1+1, i_2)$ or $t_2 = t(J^2, I^2, J, I, i_1, i_2+1)$, since this is the fix cost increment when assigning a new job J to time slot I. When J is taken from $\mathcal{J}_1$, the cost increment can be computed as

$$f_1(2) \cdot |[t_1,t_2] \cap I^2 \cap I| + f_1(1) \cdot |[t_1,t_2] \cap (I^2 - I)|, \text{ if } J^2 \in \mathcal{J}_1,$$
$$f_1(1) \cdot |[t_1,t_2] \cap I| + f_2(1) \cdot |[t_1,t_2] \cap I^2|, \text{ if } J^2 \in \mathcal{J}_2,$$

and if $J \in \mathcal{J}_2$, it is

$$f_1(1) \cdot |[t_1,t_2] \cap I^2| + f_2(1) \cdot |[t_1,t_2] \cap I|, \text{ if } J^2 \in \mathcal{J}_1,$$
$$f_2(2) \cdot |[t_1,t_2] \cap I^2 \cap I| + f_2(1) \cdot |[t_1,t_2] \cap (I^2 - I)|, \text{ if } J^2 \in \mathcal{J}_2.$$

See Figs. 6 and 7 for an illustration.

If $S(J^1, I^1, J^2, I^2, i_1, i_2)$ represents partial schedules where all jobs are scheduled, i.e. $i_1 = |\mathcal{J}_1|$ and $i_2 = |\mathcal{J}_2|$, we connect it to a terminal state S_T. The value of the arc is the cost of any schedule represented by $S(J^1, I^1, J^2, I^2, i_1, i_2)$ between time point $t(J^1, I^1, J^2, I^2, i_1, i_2)$ and the right endpoint of I^2. It is an easy observation that this cost is equal for all schedules represented by $S(J^1, I^1, J^2, I^2, i_1, i_2)$. Let t_2 denote the right endpoint of the time slot I^2 and let $t_1 = t(J^1, I^1, J^2, I^2, i_1, i_2)$. Then the arc value is the following:

$$f_1(1) \cdot |[t_1,t_2] \cap I^2|, \text{ if } J^2 \in \mathcal{J}_1, \text{ and}$$
$$f_2(1) \cdot |[t_1,t_2] \cap I^2|, \text{ if } J^2 \in \mathcal{J}_2.$$

For an illustration, see Fig. 8. If $S(J^1, I^1, J^2, I^2, i_1, i_2)$ represents partial schedules where only one job is scheduled, i.e. $i_1 + i_2 = 1$, then there is an arc from S_0 to $S(J^1, I^1, J^2, I^2, i_1, i_2)$ with zero cost. Lemma 9 follows from the construction of the digraph.

Lemma 9. *Every directed path from S_0 to S_T corresponds to a feasible schedule, where all jobs are scheduled, jobs in $\mathcal{J}_1$ and $\mathcal{J}_2$ are scheduled in order of their release dates, and the cost of the schedule coincides with the length of the path. For all feasible schedules S that contain all jobs and schedule jobs in $\mathcal{J}_1$ and $\mathcal{J}_2$ in order of their release dates, there is a directed path from S_0 to S_T. The path represents a schedule equivalent to S, and the cost of the schedule and the length of the path are the same.*

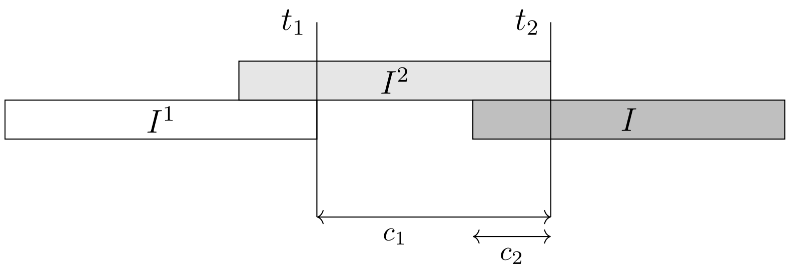
Next, we show that the size of the constructed digraph is polynomial in the input size.

Lemma 10. *The number of different $S(J^1, I^1, J^2, I^2, i_1, i_2)$ nodes is $\mathcal{O}(n^6)$.*

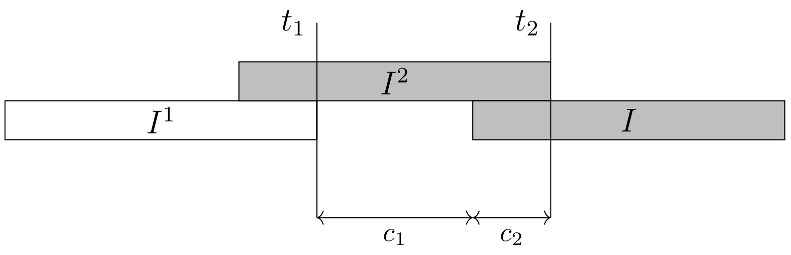
Proof. The number of different I^1 values can be upper bounded by the number of different time slots, which is $\mathcal{O}(n^2)$. The same holds for the number of different I^2 values. The number of different (i_1, i_2) pairs is $\mathcal{O}(n^2)$. If i_1 and i_2 is fixed, then there are four different possibilities for J^1 and J^2: J^1 is the $(i_1 - 1)$th and J^2 is the i_1th job in $\mathcal{J}_1$, J^1 is the $(i_2 - 1)$th and J^2 is the i_2th job in $\mathcal{J}_2$, J^1 is the i_1th job in $\mathcal{J}_1$ and J^2 is the i_2th job in $\mathcal{J}_2$, or J^2 is the i_1th job in $\mathcal{J}_1$ and J^1 is the i_2th job in $\mathcal{J}_2$. Therefore the number of different $S(J^1, I^1, J^2, I^2, i_1, i_2)$ nodes is $\mathcal{O}(n^6)$. □

Observe that a similar dynamic program can be applied when we have more than two, but constant number of machines. If the number of machines is m, we have to record the last job on each machine together with its time slot, J^i and I^i, $i = 1, \ldots, m$. Again, we take jobs one by one from $\mathcal{J}_1$ or $\mathcal{J}_2$ in release date order and schedule them in a feasible time slot which is not earlier than t, where $t = \max\{$left endpoint of I^m, right endpoint of $I^1\}$. The construction of the directed graph is similar as in the two machines case. The computation of the arc costs is similar, one only has to consider the number of jobs processed together for a time interval and using the same resource. Thus, the size of the constructed directed graph is polynomial in n.

The problem can be further generalized by allowing more resources, or allowing jobs using one unit from more resources during their execution. If jobs from the same group, that is, jobs that require the same subset of resources have agreeable deadlines, then there is an optimal schedule where jobs from the same group are scheduled in

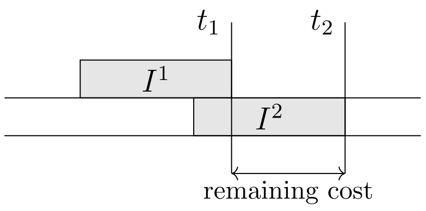


(a) When $J^2 \in \mathcal{J}_1$ and $J \in \mathcal{J}_2$, the cost increment can be computed as $c_1 + c_2$, where $c_1 = f_1(1) \cdot |[t_1, t_2] \cap I^2|$ and $c_2 = f_2(1) \cdot |[t_1, t_2] \cap I|$.

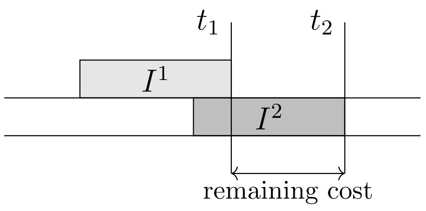


(b) When $J^2 \in \mathcal{J}_2$ and $J \in \mathcal{J}_2$, the cost increment can be computed as $c_1 + c_2$, where $c_1 = f_2(1) \cdot |[t_1, t_2] \cap (I^2 - I)|$ and $c_2 = f_2(2) \cdot |[t_1, t_2] \cap I^2 \cap I|$.

Fig. 7. Fix cost increment between t_1 and t_2 if $J \in \mathcal{J}_2$.



(a) The remaining cost can be computed as $f_1(1) \cdot |[t_1, t_2] \cap I^2|$, if $J^2 \in \mathcal{J}_1$.



(b) The remaining cost can be computed as $f_2(1) \cdot |[t_1, t_2] \cap I^2|$, if $J^2 \in \mathcal{J}_2$.

Fig. 8. The computation of the remaining cost, when all jobs are scheduled.

release date order. Therefore, by maintaining the last job and last occupied time slot on each machine alongside the number of scheduled jobs from each group, a directed graph can be constructed and an optimal schedule can be found by a shortest path computation from the source node S_0 to the terminal node S_T. If the number of machines and the number of different job groups are constants, the size of the graph is bounded by a polynomial in n.

6.2. Constant number of chains

In this subsection, we assume that the set of jobs can be covered by disjoint chains $C_1, \dots C_\rho$, where ρ is a constant number. Similarly to the agreeable deadlines case, we schedule jobs one by one in time slots that are not earlier than the occupied ones, and the newly scheduled job is the next job from one of the chains. More precisely, a state of the dynamic program denoted by $S(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho)$ represents the set of all partial schedules, where:

- I^1 and I^2 are the last occupied time slots on the machines, assuming I^1 is not later than I^2;
- J^1 is the job in I^1, and J^2 is the job in I^2 (I^1 is feasible for J^1 and I^2 is feasible for J^2);
- The first i_1 jobs are scheduled from C_1, the first i_2 jobs are scheduled from C_2 and so on.

Similarly to Section 6.1, we construct a directed graph with edge costs, where the nodes correspond to partial schedules $S(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho)$. Then a shortest path from S_0 to S_T represents an optimal schedule, where S_0 denotes the empty state where no jobs are scheduled, and S_T denotes the final state.

Again, for partial schedules represented by $S(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho)$, we can determine the latest time point $t(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho)$ until which the load does not change by scheduling new jobs in time slots later than I^1 and I^2. This time point can be determined as $t(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho) = \max\{$left endpoint of I^2, right endpoint of $I^1\}$.

Determining the outgoing edges from a state $S(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho)$ goes analogously as in the agreeable deadlines case. We can obtain a new state by choosing one of the chains where there is at least one unscheduled job, let C_i denote such a chain. Then by taking the first unscheduled job J according to the job order in C_i, together with a feasible time slot I that is no earlier than I^2, the resulting state is $S(J^2, I^2, J, I, i_1, \dots, i_i + 1, \dots, i_\rho)$. It remains to calculate the fix cost increment introduced by scheduling J in I. Determining time points t_1 and t_2 and calculating the cost increment between them can be done by the exact same formulas as in Section 6.1.

If $S(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho)$ represents schedules where all jobs are scheduled, that is $i_1 = |C_1|$, $i_2 = |C_2|$, …, $i_\rho = |C_\rho|$, then there is a directed edge from $S(J^1, I^1, J^2, I^2, i_1, i_2, \dots, i_\rho)$ to S_T. Again, the cost of the edge can be determined the same way as in Section 6.1. Then Lemma 11 is the analogous version of Lemma 9 and follows immediately from the construction.

Lemma 11. *Every directed path from S_0 to S_T corresponds to a feasible schedule, where all jobs are scheduled, jobs in $C_1, C_2, \ldots, C_\rho$ are scheduled in order of their release dates, and the cost of the schedule coincides with the length of the path. For all feasible schedules $\mathcal{S}$ that contain all jobs and schedule jobs in $C_1, C_2, \ldots, C_\rho$ in order of their release dates, there is a directed path from S_0 to S_T. The path represents a schedule equivalent to $\mathcal{S}$, and the cost of the schedule and the length of the path are the same.*

Lemma 12 shows the size of the constructed directed graph is polynomial in the input size, therefore, the dynamic program can be solved in polynomial time.

Lemma 12. *The number of different $S(J^1, I^1, J^2, I^2, i_1, i_2, \ldots, i_\rho)$ nodes is $\mathcal{O}(\rho^2 n^{4+\rho})$.*

Proof. The number of different I^1 or I^2 values can be upper bounded by the number of different time slots, which is $\mathcal{O}(n^2)$, therefore, the number of different I^1, I^2 pairs is $\mathcal{O}(n^4)$. The number of different $(i_1, i_2, \ldots, i_\rho)$ tuples is $\mathcal{O}(n^\rho)$. If $i_1, i_2, \ldots, i_\rho$ are fixed, the number of different possibilities for J^1 and J^2 is ρ, if both of them are in the same chain, and it is $\rho(\rho-1)$, if they are in different chains. Therefore the number of different $S(J^1, I^1, J^2, I^2, i_1, i_2, \ldots, i_\rho)$ nodes is $\mathcal{O}(\rho^2 n^{4+\rho})$. □

Similarly to Section 6.1, the dynamic program can be extended for the case where the number of machines is a constant $m \geq 2$, and it can be further generalized by allowing more resources or jobs using one unit from more than one distinct resources. If the number of distinct job groups is a constant and each group can be partitioned into a constant number of chains, the problem can be solved by a polynomial time dynamic program as well.

7. Conclusions

In this paper we have considered three variants P_1, P_2 and P_3 of scheduling jobs while minimizing a convex function of the resource usage of the jobs. We devised polynomial time algorithms to problems P_1 and P_2: P_1 can be reduced to a minimum cost circulation problem with convex cost functions on the edges and it can be solved by the algorithm of Karzanov and McCormick (1997), while P_2 can be modeled by a minimum cost network flow problem. We also provided polynomial time algorithms based on dynamic programming for some special cases of P_3.

A possible direction for future research is to determine the computational complexity of problem P_3 in the general case.

CRediT authorship contribution statement

Tamás Kis: Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Formal analysis, Conceptualization. **Evelin Szögi:** Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Appendix. Proof of Proposition 1

We derive the dual of $(PWL2)$ by first rewriting it as a linear program. Then by using LP duality, we obtain the dual linear program. After that, we show that the dual linear program can be rewritten as a problem with linear constraints and with a piecewise-linear convex cost function. Finally, we show that the matrix of the problem is the network matrix corresponding to network D defined in Section 4.3.

We can model the piecewise-linear functions w_X, w_k and w_h by new variables and linear constraints the following way. For w_X, we introduce two variables y_X^1 and y_X^2 and the constraints

$$b_X = |X| - y_X^1 + y_X^2,$$
$$y_X^1 \geq 0,\ y_X^2 \geq 0.$$

In the objective function, the term $w_X(b_X)$ is replaced by Ky_X^1 if $X \neq \mathcal{J}$, and it is $Ky_X^1 + Ky_X^2$ if $X = \mathcal{J}$, where $K = +\infty$ was defined in 4.3. In this way we ensure that in an optimal integral solution b_X is at least $|X|$, if $X \neq \mathcal{J}$, and b_X is exactly $|\mathcal{J}| = n$, if $X = \mathcal{J}$.

For the w_h functions, we use variables $y_h^1, \ldots, y_h^n$, and

$$t_h = y_h^1 + y_h^2 + \cdots + y_h^n,$$
$$y_h^1 \geq 0,\ y_h^2 \geq 0,\ \ldots, y_h^n \geq 0$$
$$y_h^1 \leq 1,\ y_h^2 \leq 1,\ \ldots, y_h^n \leq 1.$$

In the objective, the term $w_h(t_h)$ is replaced by $y_h^1 s_{h,1} + y_h^2 s_{h,2} + \cdots + y_h^n s_{h,n}$.

We simply remove the $w_k(x_k)$ terms from the objective, and add $x_k \geq 0$ to the problem as a constraint. Then the resulting linear program looks as follows:

$$\begin{aligned}
&\text{minimize} \sum_{X \in \mathcal{X}, X \neq \mathcal{J}} Ky_X^1 + (Ky_\mathcal{J}^1 + Ky_\mathcal{J}^2) + \sum_{h=1}^{H} (y_h^1 s_{h,1} + y_h^2 s_{h,2} + \cdots + y_h^n s_{h,n}) \\
&\qquad \text{s.t.} \\
&\qquad \sum_{I_k \in N(X)} x_k - b_X = 0, \\
&\qquad b_X = |X| - y_X^1 + y_X^2, \\
(LP):\ &\qquad y_X^1 \geq 0,\ y_X^2 \geq 0 \quad \text{for all } X \in \mathcal{X}; \\
&\qquad \sum_{k: I_k \supseteq K_h} x_k - t_h = 0, \\
&\qquad t_h = y_h^1 + y_h^2 + \cdots + y_h^n, \\
&\qquad 0 \leq y_h^1 \leq 1, \\
&\qquad \quad \vdots \\
&\qquad 0 \leq y_h^n \leq 1 \quad \text{for all intervals } K_h; \\
&\qquad x_k \geq 0 \quad \text{for all } 1 \leq k \leq L.
\end{aligned}$$

Note that if y_X^1 is positive for any $X \in \mathcal{X}$, then the objective function value is K by the arithmetic defined in Section 4.3.

Lemma 13. *There is a one-to-one correspondence between the optimal integer solutions of $(PWL2)$ and that of (LP).*

Proof. Let (x^*, b^*, t^*) be an optimal integer solution to $(PWL2)$. By the choice of K, (x^*, b^*, t^*) satisfies constraints

$$b_X^* \geq |X|, \quad \text{for all } X \in \mathcal{X}, X \neq \mathcal{J},$$
$$b_\mathcal{J}^* = n,$$
$$x_k^* \geq 0, \quad \text{for all } k = 1, \ldots, L,$$

since otherwise the objective value of (x^*, b^*, t^*) is strictly greater than the objective value of any integer solutions satisfying the constraints by the definition of symbol K. Therefore, (x^*, b^*, t^*, y^*) with $y^* = 0$ is a solution to (LP) with the same objective value.

Let (x^*, b^*, t^*, y^*) be an optimal integer solution to (LP). Then (x^*, b^*, t^*) satisfies the constraints above and $y^* = 0$, otherwise the objective value of (x^*, b^*, t^*, y^*) is at least K, and the objective value of any other integer solution with $y^* = 0$ is strictly less by the choice of K. Therefore (x^*, b^*, t^*) is a solution to $(PWL2)$ with the same objective value.

Therefore, the optimal integer solutions of $(PWL2)$ and that of (LP) are in one-to-one correspondence. □

Next, we determine the dual problem of (LP). Recall the definitions of $e_{x_k}, e_{b_X}, e_{t_h}, e_X, e_h, C(e_X)$, and $C(e_h)$ from Section 4.3. The dual linear program of (LP) is the following:

$$\begin{aligned}
\text{maximize } & \sum_{X\in\mathcal{X}} \lambda_X |X| - \sum_{h=1}^{H}(\lambda_h^1 + \lambda_h^2 + \cdots + \lambda_h^n)\\
\text{s.t.}\\
& \sum_{e_X : e_{x_k}\in C(e_X)} z_X + \sum_{e_h : e_{x_k}\in C(e_h)} z_h + \lambda_k = 0,\\
& \lambda_k \ge 0 \text{ for all } 1\le k\le L,\\
& \sum_{e_X : e_{b_X}\in C(e_X)} (-z_X) + \lambda_X = 0,\\
& \lambda_X \le K \text{ for all variables } X\in\mathcal{X};\\
& -\lambda_X \le 0 \text{ for all variables } X\in\mathcal{X}, X\neq\mathcal{J} \text{ and}\\
& -\lambda_{\mathcal{J}} \le K;\\
& \sum_{e_h : e_{t_h}\in C(e_h)} (-z_h) + \lambda_h = 0,\\
& -\lambda_h - \lambda_h^1 \le s_{h,1},\\
& -\lambda_h - \lambda_h^2 \le s_{h,2},\\
& \ldots\\
& -\lambda_h - \lambda_h^n \le s_{h,n},\\
& \lambda_h^1, \ldots, \lambda_h^n \ge 0 \text{ for all } 1\le h\le H.
\end{aligned}$$

It will be convenient to rewrite this as a minimization problem:

$$(D-LP)\qquad \begin{aligned}
\text{minimize } & -\sum_{X\in\mathcal{X}} \lambda_X |X| + \sum_{h=1}^{H}(\lambda_h^1 + \lambda_h^2 + \cdots + \lambda_h^n)\\
\text{s.t.}\\
& \sum_{e_X : e_{x_k}\in C(e_X)} z_X + \sum_{e_h : e_{x_k}\in C(e_h)} z_h + \lambda_k = 0,\\
& \lambda_k \ge 0 \text{ for all } 1\le k\le L,\\
& \sum_{e_X : e_{b_X}\in C(e_X)} (-z_X) + \lambda_X = 0,\\
& \lambda_X \le K \text{ for all variables } X\in\mathcal{X};\\
& -\lambda_X \le 0 \text{ for all variables } X\in\mathcal{X}, X\neq\mathcal{J} \text{ and}\\
& -\lambda_{\mathcal{J}} \le K;\\
& \sum_{e_h : e_{t_h}\in C(e_h)} (-z_h) + \lambda_h = 0,\\
& -\lambda_h - \lambda_h^1 \le s_{h,1},\\
& -\lambda_h - \lambda_h^2 \le s_{h,2},\\
& \ldots\\
& -\lambda_h - \lambda_h^n \le s_{h,n},\\
& \lambda_h^1, \ldots, \lambda_h^n \ge 0 \text{ for all } 1\le h\le H.
\end{aligned}$$

Now we determine the optimal values of $\lambda_h^1, \lambda_h^2, \ldots, \lambda_h^n$, based on the value of λ_h, and show how to decrease the number of variables in $(D-LP)$ by introducing piecewise-linear convex functions.

Consider variables $\lambda_h^1, \lambda_h^2, \ldots, \lambda_h^n$. Each of them appears in two inequalities, that is, for λ_h^i, we have

$$\begin{aligned}
-\lambda_h - \lambda_h^i &\le s_{h,i},\\
\lambda_h^i &\ge 0.
\end{aligned}$$

Equivalently, we have

$$\begin{aligned}
\lambda_h^i &\ge -\lambda_h - s_{h,i},\\
\lambda_h^i &\ge 0.
\end{aligned}$$

If $-\lambda_h < s_{h,1}$, then it can be assumed that $\lambda_h^1 = \lambda_h^2 = \cdots = \lambda_h^n$ in an optimal solution to $(D-LP)$. In this case, the contribution to the objective function is $\lambda_h^1 + \lambda_h^2 + \cdots + \lambda_h^n = 0$. Now define $s_{h,n+1}$ as $s_{h,n+1} = \infty$ and suppose that for an index $2 \le i \le n+1$, we have

$$s_{h,i-1} \le -\lambda_h \le s_{h,i},$$

then in an optimal solution

$$\begin{aligned}
\lambda_h^1 &= -\lambda_h - s_{h,1},\\
\lambda_h^2 &= -\lambda_h - s_{h,2},\\
&\vdots\\
\lambda_h^{i-1} &= -\lambda_h - s_{h,i-1},\\
\lambda_h^i &= \lambda_h^{i+1} = \cdots = \lambda_h^n = 0.
\end{aligned}$$

The contribution to the objective function is

$$\begin{aligned}
&\lambda_h^1 + \lambda_h^2 + \cdots + \lambda_h^n = (-\lambda_h - s_{h,1}) + (-\lambda_h - s_{h,2}) + \cdots + (-\lambda_h - s_{h,i-1})\\
&= \sum_{j=1}^{i-2} j(s_{h,j+1} - s_{h,j}) + (i-1)(-\lambda_h - s_{h,i-1}).
\end{aligned}$$

Remember the definition of w_h^* and w_X^* from Section 4.3. If we omit the variables $\lambda_h^1, \lambda_h^2, \ldots, \lambda_h^n$ together with inequalities in which one of them appears, and in the objective we replace $\sum_{h=1}^{H}(\lambda_h^1 s_{h,1} + \lambda_h^2 s_{h,2} + \cdots + \lambda_h^n s_{h,n})$ by $\sum_{h=1}^{H} w_h^*(-\lambda_h)$, using the observations above, the optimal value of λ_h and its contribution to the objective is the same in the two formulations.

We continue with variables λ_X. If $-\sum_{X\in\mathcal{X}} \lambda_X|X|$ is replaced by $\sum_{X\in\mathcal{X}} w_X^*(\lambda_X)$ in the objective function, the problem does not change.

Summarizing the observations above, the problem can be concisely expressed as

$$(D-PWL2)\qquad \begin{aligned}
\text{minimize } & \sum_{X\in\mathcal{X}} w_X^*(z_X) + \sum_{h=1}^{H} w_h^*(-z_h)\\
\text{s.t.}\\
& \sum_{e_X : e_{x_k}\in C(e_X)} z_X + \sum_{e_h : e_{x_k}\in C(e_h)} z_h + \lambda_k = 0,\\
& \lambda_k \ge 0, \text{ for all } k = 1,\ldots,L\\
& 0 \le z_X \le K, \quad \text{for all } X\in\mathcal{X},\ X\neq\mathcal{J}\\
& -K \le z_{\mathcal{J}} \le K.
\end{aligned}$$

Notice that the lower and upper bounds for z_X and $z_{\mathcal{J}}$ coincide with the left and right endpoints of the domain of w_X^* and $w_{\mathcal{J}}^*$. Using the observations above, Lemma 14 follows immediately.

Lemma 14. *The optimal solutions to $(D-LP)$ are in one-to-one correspondence with the optimal solutions to $(D-PWL2)$.*

The above arguments prove that the convex program $(D-PWL2)$ is indeed the dual of $(PWL2)$.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network Flows. Prentice-Hall, Inc., New Jersey.

Baptiste, P., 2000. Scheduling equal-length jobs on identical parallel machines. Discrete Appl. Math. 103 (1), 21–32. http://dx.doi.org/10.1016/S0166-218X(99)00238-3.

Baptiste, P., Brucker, P., Knust, S., Timkovsky, V.G., 2004. Ten notes on equal-processing-time scheduling. Q. J. Belgian French Italian Oper. Res. Soc. 2, 111–127. http://dx.doi.org/10.1007/s10288-003-0024-4.

Błażewicz, J., 1979. Deadline scheduling of tasks with ready times and resource constraints. Inform. Process. Lett. 8 (2), 60–63. http://dx.doi.org/10.1016/0020-0190(79)90143-1.

Brucker, P., Kravchenko, S., 2008. Scheduling jobs with equal processing times and time windows on identical parallel machines. J. Sched. 11, 229–237. http://dx.doi.org/10.1007/s10951-008-0063-y.

Burcea, M., Hon, W.-K., Liu, H.-H., Wong, P.W., Yau, D.K., 2016. Scheduling for electricity cost in a smart grid. J. Sched. 19 (6), 687–699. http://dx.doi.org/10.1007/s10951-015-0447-8.

Chau, V., Feng, S., Thăng, N.K., 2021. Competitive algorithms for demand response management in a smart grid. J. Sched. 1–8. http://dx.doi.org/10.1007/s10951-021-00690-x.

Drótos, M., Kis, T., 2011. Resource leveling in a machine environment. European J. Oper. Res. 212 (1), 12–21. http://dx.doi.org/10.1016/j.ejor.2011.01.043.

Hajek, B., 1990. Performance of global load balancing by local adjustment. IEEE Trans. Inform. Theory 36 (6), 1398–1414. http://dx.doi.org/10.1109/18.59935.

Harvey, N.J., Ladner, R.E., Lovász, L., Tamir, T., 2006. Semi-matchings for bipartite graphs and load balancing. J. Algorithms 59 (1), 53–78. http://dx.doi.org/10.1016/j.jalgor.2005.01.003.

Karzanov, A.V., McCormick, S.T., 1997. Polynomial methods for separable convex optimization in unimodular linear spaces with applications. SIAM J. Comput. 26 (4), 1245–1275. http://dx.doi.org/10.1137/S0097539794263695.

Kravchenko, S., Werner, F., 2009. Minimizing the number of machines for scheduling jobs with equal processing times. European J. Oper. Res. 199, 595–600. http://dx.doi.org/10.1016/j.ejor.2008.10.008.

Kravchenko, S., Werner, F., 2011. Parallel machine problems with equal processing times: a survey. J. Sched. 14, 435–444. http://dx.doi.org/10.1007/s10951-011-0231-3.

Liu, F.-H., Liu, H.-H., Wong, P.W., 2020. Non-preemptive scheduling in a smart grid model and its implications on machine minimization. Algorithmica 82 (12), 3415–3457. http://dx.doi.org/10.1007/s00453-020-00733-3.

Meyer, R.R., 1977. A class of nonlinear integer programs solvable by a single linear program. SIAM J. Control Optim. 15 (6), 935–946. http://dx.doi.org/10.1137/0315059.

Schrijver, A., 1998. Theory of Linear and Integer Programming. John Wiley & Sons.

Shaikhet, G., Karbasioun, M.M., Kranakis, E., Lambadaris, I., 2013. Asymptotic convex optimization for packing random malleable demands in smart grid. In: 2013 IEEE International Conference on Communications. ICC, IEEE, pp. 2555–2560. http://dx.doi.org/10.1109/ICC.2013.6654919.

Simons, B., 1983. Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. SIAM J. Comput. 12 (2), 294–299. http://dx.doi.org/10.1137/0212018.

Simons, B., Sipser, M., 1984. On scheduling unit-length jobs with multiple release time/deadline intervals. Oper. Res. 32 (1), 80–88. http://dx.doi.org/10.1287/opre.32.1.80.

Stewart, R., Raith, A., Sinnen, O., 2023. Optimising makespan and energy consumption in task scheduling for parallel systems. Comput. Oper. Res. 154, 106212. http://dx.doi.org/10.1016/j.cor.2023.106212.

Szögi, E., Kis, T., 2023. Scheduling jobs to minimize a convex function of resource usage. In: Ganzha, M., Maciaszek, L., Paprzycki, M., Slezak, D. (Eds.), Proceedings of the 18th Conference on Computer Science and Intelligence Systems. In: Annals of Computer Science and Information Systems, Vol. 35, IEEE, pp. 791–799. http://dx.doi.org/10.15439/2023B4164.

Xie, G., Xiao, X., Peng, H., Li, R., Li, K., 2022. A survey of low-energy parallel scheduling algorithms. IEEE Trans. Sustain. Comput. 7 (1), 27–46. http://dx.doi.org/10.1109/TSUSC.2021.3057983.