DOI: 10.15439/2023F4164
ISSN 2300-5963 ACSIS, Vol. 35

Scheduling Jobs to Minimize a Convex Function of Resource Usage

Evelin Szögi
0009-0008-5818-374X
ELKH SZTAKI
Kende str. 13-17, Budapest, 1111, Hungary
and
Department of Operations Research
Loránd Eötvös University
Email: szogi.evelin@sztaki.hu

Tamás Kis
0000-0002-2759-1264
ELKH SZTAKI
Kende str. 13-17, Budapest, 1111, Hungary
Email: kis.tamas@sztaki.hu

***Abstract*—In this paper we describe polynomial time algorithms for minimizing a separable convex function of the resource usage over time of a set of jobs with individual release dates and deadlines, and admitting a common processing time.**

I. Introduction

In THIS paper we study variants of the following scheduling problem. There are n jobs $\mathcal{J} = \{J_1,\ J_2,\ \ldots,\ J_n\}$, and a common resource required by a subset of the jobs. Each job J_i has a release date r_i, a deadline d_i, and requires $\mu_i \in \{0, 1\}$ unit of the common resource. All jobs have the same processing time p. A schedule $\mathcal{S}$ specifies a starting time S_i for each job J_i, and it is *feasible*, if $r_i \leq S_i \leq d_i - p$ holds for each job J_i.

The goal is to find a feasible schedule $\mathcal{S}$, which minimizes a convex function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ 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., $\ell^{\mathcal{S}}(t) = |\{i \mid S_i \leq t \leq S_i + p, \mu_i = 1\}|$.

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., [9], [10], [6]. 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 $\ell^{\mathcal{S}}(t_0) \geq \ell^{\mathcal{S}}(t_1) + 2$ along with a subset of jobs, which can be rescheduled such that the load of t_0 decreases by one, while the load t_1 increases by one, and the load of all other time slots do not change. The convexity of f ensures that the objective function value of the new schedule is smaller than that of $\mathcal{S}$. Yet another variant is when the jobs have arbitrary integer processing times, but the preemption is allowed, i.e., the processing of any job can be interrupted and resumed later. This variant is studied in [8]. The authors have shown that the above two properties of optimal solutions are preserved. Drótos and Kis [7] study the following resource leveling problem. There is a set of m machines, a set of renewable resources, and a set of n jobs associated with release times, deadlines and resource requirements, each pre-assigned to one of the machines. The jobs have to be sequenced on the machines, while minimizing the sum of the convex functions of the loads of the resources over time. They show that if the starting times of all tasks on all but one machines are fixed, the problem is NP-hard. However, if the ordering of tasks on the remaining machine is also given, then a polynomial time algorithm exists. They also give a heuristic as well as exact branch-and-bound algorithm for solving the problem.

The above scheduling problem can be extended to parallel machine problems, where the jobs have to be assigned to machines, and the jobs assigned to the same machine have to be sequenced. The latter problem was introduced by Blazewicz [4], 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 time. 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 two variants of the scheduling

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

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 common processing time p is arbitrary positive integer.

We will show by way of an example that in unlike the load balancing problem with $p = 1$, for general p, there is no universally optimal solution (Section III). 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 IV).
Problem P_2. Only a subset of the jobs require one unit of the common resource, and $p = 1$. In addition, there are m machines (resources of unit capacity), and each job has to be assigned to one of the machines. The jobs assigned to the same machine must be processed in non-overlapping time slots.

We describe a network-flow based method with convex cost functions on the arcs in Section V. As a by-product, our method can also answer the decision problem of [4].

II. Related work

We have already summarized the most relevant results on load balancing, resource leveling, 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 [5] investigate the problem where equal-length jobs have to be scheduled on m identical parallel machines. For each job, a release time 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 [12], 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. Further results can be found in [2], [3], and for a survey, see [13].

III. Properties of optimal solutions

In the introduction we emphasized that the load balancing problem with common 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 III.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 $x \leq 6$, and $f_2(x) = x - 6$ for $x \geq 6$. 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 $p > 1$.

IV. A combinatorial approach for problem P_1

We first give a linear programming formulation of problem P_1 in Section IV-A. Then, 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 (Sections IV-B), and how to determine an optimal solution for our scheduling problem from an optimal circulation in Section IV-C. Finally, based on these results, we propose a new combinatorial algorithm for a parallel machine scheduling problem (Section IV-D).

A. Initial problem formulation and solution method

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

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

The following lemma states an important property about the structure of an optimal schedule, and it generalizes Lemma 3 of [2].

Lemma 1. *There is 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 with jobs $\mathcal{J}$ and processing time p. Let $\mathcal{S}^*$ be an optimal schedule such that the number of jobs which are not scheduled in some time slot in $\mathcal{I}$ is minimal. Let H be the subset of all those jobs that are not scheduled in some time slot 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 $\ell \in \{0, \ldots, n\}$, δ_ℓ 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 H by ϵ time earlier, or later, at least one of the jobs in

H is scheduled in a time slot $I \in \mathcal{I}$. Let $\mathcal{S}_1$ be the resulting schedule. Such a shift induces a vector $\mu \in \mathbb{R}^{n+1}$ such that $\delta + \mu$ represents the load of the resource in schedule $\mathcal{S}_1$.

The cost of the schedules $\mathcal{S}^*$ and $\mathcal{S}_1$ are related by

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

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

Since $\sum_{\ell=0}^{n} \ell \cdot \mu_\ell = 0$ must hold, $\delta - \mu$ also represents the resource usage of some feasible schedule $\mathcal{S}_2$, namely, the one obtained by shifting all jobs in H in the opposite direction by ϵ. The cost of $\mathcal{S}^*$ and $\mathcal{S}_2$ are related by

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

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 job in H is scheduled in a time slot in $\mathcal{I}$, which contradicts the choice of $\mathcal{S}^*$. □

Let $\mathcal{C} = \{c_0, c_1, \ldots, c_H\}$ 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 the time slots in $N(X)$ are consecutive. 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 that require the resource in K_h, that is, $t_h = \sum_{I_k \supset K_h} x_k$.

Consider the following mathematical program:

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

$$(IP): \quad \begin{aligned} \text{s.t.} \sum_{I_k \in N(X)} x_k - b_X &= 0, \quad \text{for all } X \in \mathcal{X} && (1)\\ b_X &\geq |X|, \quad \text{for all } X \in \mathcal{X} && (2)\\ b_{\mathcal{J}} &= n && (3)\\ \sum_{k: I_k \supseteq K_h} x_k - t_h &= 0, \quad \text{for each interval } K_h && (4)\\ x_k &\geq 0, \quad \text{for all } k = 1, \ldots, L && (5)\\ x_k &\in \mathbb{Z}, \quad \text{for all } k = 1, \ldots, L. && (6) \end{aligned}$$

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 $v_k^{(1)}, \ldots, v_k^{(x_k)}$ corresponding to I_k. For each job J_i and $I_k \subset [r_i, d_i]$, E contains the edge $(v_{J_i}, v_k^{(l)})$ for $l = 1, \ldots, x_k$. 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, constraints in (1) and (2) 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 $N(V_{X_1}), N(V_{X_2}), \ldots, N(V_{X_r})$ 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 $l = 1, \ldots, r$. Since (x, b, t) is a feasible solution to (IP), $|N(V_{X_i})| \geq |V_{X_i}|$ holds for all $i = 1, 2, \ldots r$ and $|N(V_X)| \geq |V_X|$ follows. □

Constraint (3) ensures $|V_{\mathcal{J}}| = n$, therefore, we have:

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

Proof. It follows from Lemma 2 and from the well-known theorem of Hall (see e.g. [1]). □

The following result shows how to map feasible solutions of (IP) to feasible schedules of the same cost and vice versa.

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 the same cost satisfying (1) - (6).*

Proof. To prove the first part of the lemma, suppose (x, b, t) satisfies (1) - (6). 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 represent the load of interval K_h in the solution by eq. (4), 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, $h = 1, \ldots, H$, 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 (1) - (6) and it has the same cost as $\mathcal{S}$. □

The following statement shows how to map optimal solutions (x, b, t) to perfect matchings in $G_{(x,b,t)}$, and vice versa.

Proposition 1. *If (x, b, t) is an optimal solution to (IP), then any perfect matching in $G_{(x,b,t)}$ corresponds to an optimal*

Algorithm 1 Calculation of optimal schedule

```
Require: $n \geq 0$, $p \geq 1$, $\{r_i, d_i\}$ for $i = 1, \ldots, n$, 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 $M$;
```

schedule. Conversely, any optimal schedule $\mathcal{S}$ induces an optimal solution (x, b, t) to (IP).

Proof. Follows easily from Lemma 3 and Lemma 4. □

By Proposition 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 [1], 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 (1)-(6) is polynomial in the size of the input.*

Proof. To prove that the size of system (1)-(6) is polynomial in the size of the input, it is enough to show that the number of constraints in (2) and (4) is polynomial in the input size. Observe that $L = |\mathcal{I}|$ is $\mathcal{O}(n^2)$. All $X \in \mathcal{X}$ can be obtained the following way. One can construct $\mathcal{O}(L^2)$ possible connected $N(X)$ sets of time slots by determining the earliest and latest time slot. Then it remains to check whether there is an $X \subseteq \mathcal{X}$ such that $N(X)$ is exactly the set of time slots feasible for at least one jobs in X. Therefore $|\mathcal{X}|$ is polynomial in L. There are H intervals K_h and the endpoints of each of them coincide with endpoints of time slots in $\mathcal{I}$, therefore $H = \mathcal{O}(L)$ and the number of constraints in (4) 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 $z \in \mathbb{Z}_{\geq 0}$. 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

$$(IPWL): \qquad \text{minimize} \sum_h |K_h| \tilde{f}(t_h) \quad \text{s.t. } (1) - (6).$$

In fact, the matrix of (1) - (6) is totally unimodular, see Lemma 6, which permits to get rid of the integrality condition (6) by a result of Meyer. Meyer [14] has shown that an optimization problem with a separable piecewise linear convex cost function with integer break points, and a totally unimodular constraint matrix always admits and integer optimal solution. This means that in (IPWL) we can drop constraint (6), while preserving integer optimal solutions:

$$(PWL): \qquad \text{minimize} \sum_h |K_h| \tilde{f}(t_h) \quad \text{s.t. } (1) - (5).$$

Karzanov and McCormick [11] 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 have shown 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 $c_e^{\dashv}(r) \leq s \leq c_e^{\vdash}(r)$.

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

Karzanov and McCormick proposed two different algorithms for solving such problems: the Minimum Mean Canceling Method (MMCM) and the Cancel and Tighten algorithm. Both methods are iterative, and in general case, these algorithms involve solving linear programs 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.

B. 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 $0, 1, \ldots, n$, and the slopes between these breakpoints are $s_l = |K_h|(\tilde{f}(l) - \tilde{f}(l-1))$ for $l = 1, \ldots, n$, noting that $s_1 \leq s_2 \leq \cdots \leq s_n$, since $\tilde{f}$ is convex (if $s_l = s_{l+1}$ then

b_l is not a real breakpoint, but it is not a problem). Then we have

$$w_h(z) = \begin{cases} |K_h|\tilde{f}(0) + s_1 z & \text{if } 0 \le z \le 1 \\ |K_h|\tilde{f}(1) + s_2(z-1) & \text{if } 1 \le z \le 2 \\ \dots \\ |K_h|\tilde{f}(n-1) + s_n(z-n+1) & \text{if } n-1 \le z. \end{cases}$$

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. Let K be a sufficiently large positive number. 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

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

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 M^T is a 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 (1) and (4). Therefore if y is a column of M^T, 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 with a spanning tree T and show that each non-tree edge corresponds to a column of M.

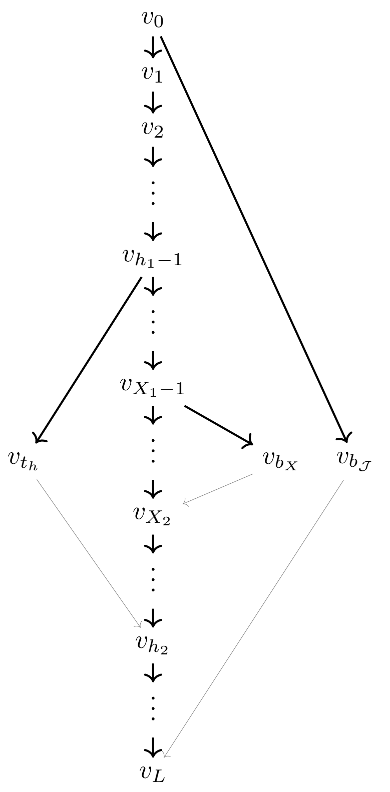
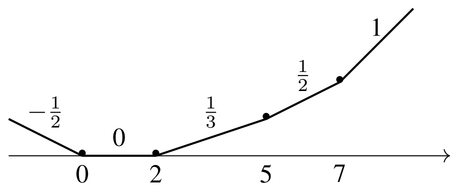
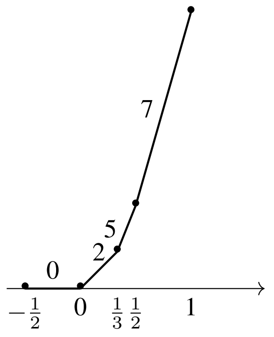


Fig. 1: Network D. Thin arcs correspond to non-tree edges, thick arcs are the edges of T.

Let $P = v_0 \to v_1 \to \dots \to v_L$ 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, $D = P$ and $T = P$. Then we add new nodes and edges to D and T the following way: we take all constraints from (1) and (4) 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 (1) and suppose equation $\sum_{I_k\in N(X)} x_k - b_X = 0$ is one of them. We add a new node 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 $v_{X_1-1} \to v_{b_X}$ and $v_{b_X} \to v_{X_2}$. We extend T with the first new edge $v_{X_1-1} \to v_{b_X}$, and we say the tree edge $v_{X_1-1} \to v_{b_X}$ corresponds to variable b_X. The other new edge $v_{b_X} \to v_{X_2}$ becomes a non-tree edge. We proceed similarly with equations in (4) of the form $\sum_{k:I_k\supseteq K_h} x_k - t_h = 0$: we connect the first and the last time slot in $I_k : I_k \supseteq K_h$ with a 2-length path containing two new edges and extend T with the first new edge in the same way as before. Fig. 1 illustrates the network constructed this way. It is not difficult to check that a column in M^T corresponding to a constraint from (1) or (4) is represented by a non-tree edge in the network. $\square$



(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. 2a.

Fig. 2: A piecewise linear convex function and its dual.

For simplicity, tree edges in D corresponding to variables x_k, b_X and t_h are denoted by e_{x_k}, e_{b_X} and e_{t_h}. In the next part, we 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, let z_X denote the corresponding dual variable if the edge derives from a constraint in (1), and let z_h be the dual variable if it derives from a constraint in (4). For a non-tree edge e, let C_e denote the tree edges in e's fundamental cycle.

In order to obtain the dual of $(PWL2)$, we have to determine the duals of the functions w_k, w_X and w_h, respectively. It is known (see e.g. [11]) 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. 2 for an illustration.

The dual functions of w_X, w_k and w_h are denoted by w_X^*, w_k^* and w_h^*, respectively, and these functions have the following forms.

If $X \neq \mathcal{J}$, we have

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

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

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

For w_k^*, we have

$$w_k^*(z) = 0, \qquad z \geq 0,$$

and finally w_h^* can be written as

$$w_h^*(z) = \begin{cases} 0, & \text{if } z \leq s_1, \\ z - s_1, & \text{if } s_1 \leq z \leq s_2, \\ -s_1 + s_2 + 2(z - s_2) & \text{if } s_2 \leq z \leq s_3, \\ \dots & \\ -\sum_{i=1}^{r-1} s_i + (r-1)s_r + & \\ \quad r(z - s_r), & \text{if } s_r \leq z \leq s_{r+1}, \\ \dots & \\ -\sum_{i=1}^{n-1} s_i + (n-1)s_n + & \\ \quad n(z - s_n), & \text{if } z \geq s_n. \end{cases}$$

Using w_X^* and w_h^*, the dual of $(PWL2)$ can be concisely expressed as follows:

$$\begin{aligned} & \text{minimize} \sum_{X \in \mathcal{X}} w_X^*(z_X) + \sum_{h=1}^{H} w_h^*(-z_h) \\ & \text{s.t.} \\ & \qquad \sum_{e_{b_X}: e_{x_k} \in C_{e_{b_X}}} z_X + \sum_{e_{t_h}: e_{x_k} \in C_{e_{t_h}}} z_h + \lambda_k = 0, \\ (DP) & \qquad \lambda_k \geq 0, \text{ for all } k = 1, \dots, L \\ & \qquad 0 \leq z_X \leq K, \quad \text{for all } X \in \mathcal{X}, \ X \neq \mathcal{J} \\ & \qquad -K \leq z_{\mathcal{J}} \leq K. \end{aligned}$$

Notice that the lower bound for λ_k coincides with the left endpoint of the domain of w_k^*, and 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 $w_{\mathcal{J}}^*$.

Since the matrix of problem $(PWL2)$ is the transpose of a network matrix, (DP) is a network circulation problem, the only problem with it is that in the objective function we have $w_h^*(-z_h)$, i.e., the negative of z_h is substituted in the convex function w_h^*. However, it is easy to overcome this issue by reversing the arcs e_{t_h}. So, our final network has the same set of nodes and arcs as D, except that the arcs e_{t_h} are directed oppositely. For an edge e_{x_k}, the lower bound for the flow value λ_k is 0 and there is no upper bound, while the cost is 0. For an 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 an 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 in D, and the cost function is 0.

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

The minimum cost circulation problem defined above can be solved by the Cancel and Tighten method described in [11]. Remember, 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 $C = \mathcal{O}(n)$ holds in our case. It can be assumed that the length of the scheduling horizon $d_{\max} - r_{\min}$ is at most $2np$. Therefore, the number of different p-length time slots is $\mathcal{O}(n^2)$, and the number of constraints in (1) and (4) is $\mathcal{O}(n^4)$. Hence, $|V(G)| = \mathcal{O}(n^4)$ and $|E(G)| = \mathcal{O}(n^4)$ in our case.

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

By Proposition 2, an optimal solution to (DP) can be obtained by solving a minimum cost network circulation problem with convex cost functions on the arcs, using an algorithm of [11]. 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 [11].

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 $c_e^{\dashv}(x_e^*) \leq h_e \leq c_e^{\vdash}(x_e^*)$ for all $e \in E$, then h_e, $e \in E$ is an optimal solution to the dual problem, where $c_e^{\dashv}$ and $c_e^{\vdash}$ are the corresponding left and right derivatives, respectively.*

If an optimal solution x_e^*, $e \in E$ is given, then by Lemma 7, an optimal co-circulation is easy to obtain. 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 $c_e^{\dashv}(x_e^*) \leq h_e \leq c_e^{\vdash}(x_e^*)$, $e \in E$ is equivalent to finding node potentials π satisfying $\pi(v_e) - \pi(u_e) \leq c_e^{\vdash}(x_e^*)$ and $\pi(u_e) - \pi(v_e) \leq -c_e^{\dashv}(x_e^*)$ for all $e \in E$. Such node potentials π can be found by a shortest path algorithm in the directed graph we get by adding edges $e \in E$ to the graph in reverse direction as well. The edge lengths are $c_e^{\vdash}(x_e^*)$ for edges with original orientation and $-c_e^{\dashv}(x_e^*)$ for edges with reverse orientation. Since $c_e^{\vdash}(x_e^*)$ and $c_e^{\dashv}(x_e^*)$ are integral values in our case, the optimal co-circulation found by a shortest path algorithm is integral as well.

D. 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 previously introduced techniques to a problem investigated by Brucker and Kravchenko [5]. There are n jobs with common 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 at any time. 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.

To begin with, we formulate the problem similarly to (IP). Recall the definitions of the set of time slots $\mathcal{I}$, and set of intervals $\{K_h\}_{h=1,\ldots,H}$. 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).

$$(IP'):\quad \begin{aligned}
&\text{minimize} \sum_{k=1}^{L} C(I_k)x_k && (7)\\
&\text{s.t.} \sum_{I_k \in N(X)} x_k - b_X = 0, \quad \text{for all } X \in \mathcal{X} && (8)\\
&b_X \geq |X|, \quad \text{for all } X \in \mathcal{X} && (9)\\
&b_{\mathcal{J}} = n && (10)\\
&\sum_{k: I_k \supseteq K_h} x_k - t_h = 0, \quad \text{for each interval } K_h && (11)\\
&t_h \leq m, \quad \text{for all } h = 1, \ldots, H && (12)\\
&x_k \geq 0, \quad \text{for all } k = 1, \ldots, L && (13)\\
&x_k \in \mathbb{Z}, \quad \text{for all } k = 1, \ldots, L. && (14)
\end{aligned}$$

The objective function (7) expresses the total completion time of the jobs. The rest of the constraints are analogous to that of (IP). For a feasible solution (x, b, t), one can construct a bipartite graph $G_{(x,b,t)} = (V_{\mathcal{I}} \cup V_{\mathcal{J}}, E)$ in the same way as in Section IV-A. Analogously to Lemma 2, one can show that $G_{(x,b,t)}$ admits a perfect matching and Proposition 1 holds. Therefore, the problem can be solved efficiently if one can solve (IP') efficiently. From Lemma 5, it follows that the size of (IP') is polynomial in the size of the input, and similarly to Lemma 6, one can show that the transpose of the matrix of the system is a network matrix. Since the cost functions on the arcs are linear functions of the flows, the dual is a circulation problem with liner costs on the arcs, and a simple minimum cost flow computation finds an optimal solution.

V. 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 is as balanced as possible. 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 show that this problem can be solved by a single minimum cost flow computation in a network with convex costs on the arcs in Section V-A. Then, we apply our formulation to the decision problem of [4] in Section V-B.

A. 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 $\mathcal{I}' = \{I_1, \ldots, I_{L'}\}$ be the set of all different unit-length time slots in the set

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

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

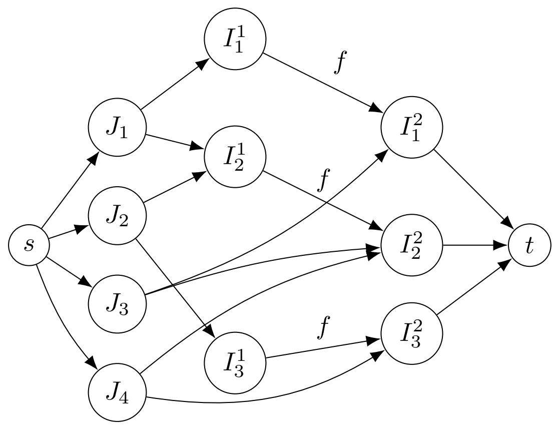


Fig. 3: Network D' for Problem P_2, where jobs J_1 and J_2 require one resource unit and jobs J_3 and J_4 require 0. The cost is measured by the convex function f on three edges, the remaining edges have zero cost.

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 this 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. 3 for an illustration.

Proposition 3. *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 in a similar way as in [8], and the problem can be solved by any minimum cost network flow algorithm.

B. 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 Blazewicz [4] as described in Section I. 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 time and deadline if and only if there is feasible flow of zero cost in the previously constructed network D'. Notice that there is no need 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_k^1 to I_k^2 to c. Then we have the following result.

Proposition 4. *The scheduling problem of [4] with a bounded capacity resource admits a feasible solution if and only of the network D'' admits a feasible flow of value n.*

VI. 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 test runs was to assess how sensitive is the method to two problem parameters: the number of the jobs n, and the ratio of the common 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 common 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 $r_i \geq 0$, $d_i = \lceil r_i + p/q \rceil$, and $d_i \leq 100$. We used the same piecewise linear convex function f in all cases, where $f(0) = 0$, f has breakpoints at $1, 2, \dots, n$, 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 I. We provide averages (rounded to nearest integers) over 3 problem instances for each combination of the parameters n and q. In each case, we provide the average number of nodes and edges of the network D, the average number of iterations, the average CPU time, and also the average optimum values. All values are rounded to the nearest integers. As we can see, the CPU time strongly correlates with the number of graph edges. On the other hand, the number of iterations lightly increases with the number of the jobs, but for the same number of jobs, it is smaller for $q = 0.4$ than for $q = 0.1$. In fact, this is what we expected, since problem instances with a larger ratio q permit less freedom to choose the starting times of the jobs.

VII. Conclusion

In this paper we gave polynomial algorithms to two load balancing problem. A possible direction for a future research is to investigate a problem slightly more general than Problem P_2. While the common processing time in P_2 is one time unit, it is an interesting question what can be said if the common processing time is greater than one time unit. The complexity of this problem is still open. One possible next step would be to derive an approximation algorithm for the problem.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows.* Prentice-Hall, Inc., New Jersey, 1993.

[2] Philippe Baptiste. Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, 103(1):21–32, 2000.

[3] Philippe Baptiste, Peter Brucker, Sigrid Knust, and Vadim G. Timkovsky. Ten notes on equal-processing-time scheduling. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2:111–127, 2004.

[4] Jacek Błażewicz. Deadline scheduling of tasks with ready times and resource constraints. *Information Processing Letters*, 8(2):60–63, 1979.

[5] Peter Brucker and Svetlana Kravchenko. Scheduling jobs with equal processing times and time windows on identical parallel machines. *Journal of Scheduling*, 11:229–237, 08 2008.

TABLE I: 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 (millisec)	36	58	52	74	59	135

[6] Mihai Burcea, Wing-Kai Hon, Hsiang-Hsuan Liu, Prudence W. Wong, and David K. Yau. Scheduling for electricity cost in a smart grid. *Journal of Scheduling*, 19(6):687–699, 2016.

[7] Márton Drótos and Tamás Kis. Resource leveling in a machine environment. *European Journal of Operational Research*, 212(1):12–21, 2011.

[8] Péter Györgyi, Tamás Kis, and Evelin Szögi. A polynomial time algorithm for solving the preemptive grid-scheduling problem. *unpublished*, 2023.

[9] Bruce Hajek. Performance of global load balancing by local adjustment. *IEEE Transactions on Information Theory*, 36(6):1398–1414, 1990.

[10] Nicholas JA Harvey, Richard E Ladner, László Lovász, and Tami Tamir. Semi-matchings for bipartite graphs and load balancing. *Journal of Algorithms*, 59(1):53–78, 2006.

[11] Alexander V. Karzanov and S. Thomas McCormick. Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM Journal on Computing*, 26(4):1245–1275, 1997.

[12] Svetlana Kravchenko and Frank Werner. Minimizing the number of machines for scheduling jobs with equal processing times. *European Journal of Operational Research*, 199:595–600, 12 2009.

[13] Svetlana Kravchenko and Frank Werner. Parallel machine problems with equal processing times: a survey. *Journal of Scheduling*, 14:435–444, 10 2011.

[14] R. R. Meyer. A class of nonlinear integer programs solvable by a single linear program. *SIAM J. Control Optim.*, 15(6):935–946, 1977.