Operations Research Letters 51 (2023) 99–104

Contents lists available at ScienceDirect

Operations Research Letters

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

SPT optimality (mostly) via linear programming

Woo-Hyung Cho *, David Shmoys, Shane Henderson

Cornell University, United States of America

ARTICLE INFO

Article history:
Received 4 July 2022
Received in revised form 4 December 2022
Accepted 11 December 2022
Available online 16 December 2022

Keywords:
Scheduling
Linear programming
Gantt charts

ABSTRACT

One of the oldest results in scheduling theory is that the Shortest Processing Time (SPT) rule finds an optimal solution to the problem of scheduling jobs on identical parallel machines to minimize average job completion times. We present a new proof of correctness of SPT based on linear programming (LP). Our proof relies on a generalization of a single-machine result that yields an equivalence between two scheduling problems. We first identify and solve an appropriate variant of our problem, then map its solutions to solutions for our original problem to establish SPT optimality. Geometric insights used therein may find further uses; we demonstrate two applications of the same principle in generalized settings.

1. Introduction

Consider the problem of scheduling jobs on identical parallel machines to minimize average job completion times. Each job j is available at time 0 and requires $p_j > 0$ units in uninterrupted processing time. Each machine can only process one job at a time. Letting C_j denote the completion time of job j, this problem is $P||\sum C_j$ in the notation of Graham et al. [6]. One of the oldest and most widely known results in scheduling theory is that this problem is solvable in polynomial time [3]. An optimal schedule can be constructed by the Shortest Processing Time (SPT) rule that begins processing a job not yet processed with the shortest processing time whenever a machine is idle.

We present a new proof of correctness of SPT via linear programming (LP). We use an LP formulation previously introduced by Balas [1] and further developed by Wolsey [16], Queyranne [11], Queyranne and Wang [13], Schulz [14] and Hall, Schulz, Shmoys and Wein [7]. Earlier proofs of correctness of the SPT rule rely on *coefficient matching* (see Brucker [2] and Lawler [8,9], for example), but to the best of our knowledge, this is the first LP-based proof.

Our proof of correctness of SPT uses a second scheduling problem that involves job weights $w_j > 0$ for each job j. The general problem $P||\sum w_j C_j$ is NP-hard [8]. One reaction to this NP-hardness result is that an LP-based proof for $P||\sum C_j$ (and the associated structural results then implied by LP duality) should not be possible. However, some special cases of the weighted problem $P||\sum w_j C_j$ are known to be polynomial-time solvable, and in fact, one such special case has an equivalence with our main problem $P||\sum C_j$. This equivalent weighted problem comes with strong structural properties that we are able to exploit using LP techniques. The resulting LP solution is then transformed into an optimal solution for $P||\sum C_j$, which gives an alternate LP-based proof of correctness of SPT.

Identifying an appropriate weighted variant is a critical step in our proof. Our methods generalize a single-machine result that has been observed by many in the Scheduling field, and are based on geometric insights from two-dimensional Gantt charts. Gantt charts have already proven useful for tackling various scheduling problems [14,7,5], so we expect our methods to also find further uses. To demonstrate this, we apply the same principles in more generalized settings in the last section of this paper.

The remainder of this paper is organized as follows. In Section 2, we discuss the geometric insights from two-dimensional Gantt charts that reveal a related scheduling problem. Linear programming methods are used to solve this problem and establish SPT optimality in Section 3. Section 4 extends the idea in uniform and unrelated parallel machine settings.

2. Insights from two-dimensional Gantt charts

Gantt charts are useful for visualizing schedules over time, especially for a single machine. Traditional Gantt charts are unidimensional in time. In a nonpreemptive schedule, a machine may block off p_j units in uninterrupted processing time for job j. If job j begins processing at time t, then its completion time $C_j = t + p_j$. See Fig. 1.

* Corresponding author.
E-mail addresses: wc563@cornell.edu (W.-H. Cho), david.shmoys@cornell.edu (D. Shmoys), sgh9@cornell.edu (S. Henderson).

https://doi.org/10.1016/j.orl.2022.12.007
0167-6377/© 2022 Elsevier B.V. All rights reserved.

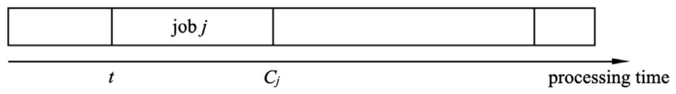

Fig. 1. A one-dimensional Gantt chart.

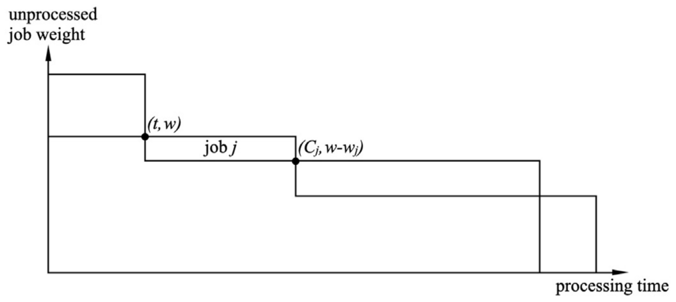

Fig. 2. A two-dimensional Gantt chart.

We introduce job weights in two-dimensional Gantt charts. With time on the horizontal axis and the total remaining unprocessed job weight on the vertical axis, job j with weight w_j is represented as a rectangular block of width p_j and height w_j. When job j begins processing with coordinate (t, w) on its upper-left corner, it completes with coordinate $(C_j, w - w_j)$ on its lower-right corner as illustrated in Fig. 2. Letting $\mathcal{N}$ denote the set of all jobs in a schedule, $\sum_{j\in\mathcal{N}} w_jC_j$ is the area under the curve in a two-dimensional Gantt chart. Solving the single-machine problem $1||\sum w_jC_j$ is therefore equivalent to finding a sequence of jobs that minimizes this area under the curve in a two-dimensional Gantt chart.

Two-dimensional Gantt charts have been widely explored for various single-machine problems, as shown in Hall et al. [7], Schulz [14], and Goemans and Williamson [5], for example. In comparison, their applications in parallel-machine settings are relatively limited (a notable exception is the paper by Eastman, Even and Isaacs [4] that introduced the concept of two-dimensional Gantt charts in 1964). Parallel-machine schedules are difficult to interpret graphically when multiple jobs of varying widths are being processed at the same time, each on a different machine.

Two observations are used to transform a two-dimensional Gantt chart for $P||\sum C_j$ for better interpretability. This procedure will also reveal a second scheduling problem that has an equivalence relation with $P||\sum C_j$. As will become clear shortly, the equal-weighted nature of our objective is a key feature in this process.

The first observation is that any feasible parallel-machine schedule with m machines may be decomposed into m feasible single-machine schedules without altering the objective value. Of course, the converse is also true when the set of jobs $\mathcal{N}$ is partitioned into m disjoint sets of jobs.

The next insight is due to the fact that reflecting any feasible, bounded two-dimensional Gantt chart over the identity line preserves the objective value. This fact has been observed by many researchers over the years, becoming a kind of folklore in the Scheduling field. More precisely, suppose there is a feasible schedule for $1||\sum w_jC_j$ where jobs are described by the set of parameters $\{(p_j, w_j) : j \in \mathcal{N}\}$. Then, we can construct an instance that shares the same area under the curve by scheduling in *reverse order* the set of jobs described by $\{(\hat{p}_j, \hat{w}_j) : j \in \mathcal{N}\}$, where $\hat{p}_j = w_j$ and $\hat{w}_j = p_j$ for each j. We shall call this the *weight–processing time flip*. See Fig. 3 for an illustration. Another way of arriving at this equivalence is by observing that the area under the curve in a two-dimensional Gantt chart can also be obtained by having every scheduled job j pay p_j for the delay it causes itself and all jobs scheduled thereafter.

Now consider any feasible schedule for $P||\sum C_j$ with job inputs $\{(p_j, w_j) : j \in \mathcal{N}\}$. In the absence of weights, let $w_j = p$ where p is some constant. We decompose this schedule into m single-machine schedules, then flip the weights and processing times of each job according to the weight–processing time flip. Doing so creates a set of jobs with *equal processing times* and general weights described by $\{(\hat{p}_j, \hat{w}_j) : j \in \mathcal{N}\}$, where $\hat{p}_j = w_j = p$ and $\hat{w}_j = p_j$, and reverses the order in which jobs are processed on each machine. Putting these m newly created single-machine schedules together, we obtain a feasible parallel-machine schedule for the problem $P|p_j = p|\sum w_jC_j$: an equal-processing-time variant of the weighted problem. The schedules for $P||\sum C_j$ and $P|p_j = p|\sum w_jC_j$ share the same objective value.

This bijection between an input to $P||\sum C_j$ and what we shall call a *flipped input* to $P|p_j = p|\sum w_jC_j$ implies that solving one solves the other. Between the two, $P|p_j = p|\sum w_jC_j$ is a much more attractive problem to solve given its equal processing time structure. Since all jobs are available at time 0 and require p units in processing time, job completion times are always at integer multiples of p in an optimal schedule. This problem denoted $P|p_j = p|\sum w_jC_j$ is technically equivalent to $P|p_j = 1|\sum w_jC_j$ where all jobs have unit processing time requirements. This equivalence is lost, however, once we relax the assumption that all jobs are available at time 0. To highlight this distinction, we retain the notation $p_j = p$ throughout this paper.

Given our observations, a two-dimensional Gantt chart for $P|p_j = p|\sum w_jC_j$ reads like that of a single-machine problem in which each job is a collection of at most m jobs of equal width p. Sequencing jobs in order of nonincreasing w_j/p_j is optimal for the single-machine problem $1||\sum w_jC_j$ by Smith's Weighted Shortest Processing Time (WSPT) rule [15]. By extension, sorting jobs in order of nonincreasing w_j and sequencing collections of m jobs in sorted order is optimal for $P|p_j = p|\sum w_jC_j$. We give a formal proof of this in the following section.

3. SPT optimality

Consider the following linear program for $P|p_j = p|\sum w_jC_j$, which refines the frameworks of Wolsey [16] and Queyranne [11].

$$\min \sum_{j\in\mathcal{N}} w_jC_j \tag{1}$$

$$\text{s.t.} \sum_{j\in\mathcal{S}} C_j \geq f(\mathcal{S}) \quad \text{for all } \mathcal{S} \subseteq \mathcal{N}, \tag{2}$$

where

$$f(\mathcal{S}) = \frac{p}{2}\left(\left\lceil \frac{|\mathcal{S}|}{m} \right\rceil^2 \cdot (|\mathcal{S}| \bmod m) + \left\lfloor \frac{|\mathcal{S}|}{m} \right\rfloor^2 \cdot (m - |\mathcal{S}| \bmod m) + |\mathcal{S}|\right).$$

The derivation for the functional form of $f(\mathcal{S})$ builds on earlier works that describe the convex hull of feasible completion time vectors. Balas [1], Wolsey [16], Queyranne and Wang [13], Queyranne [11], and Queyranne and Schulz [12] have extensively studied scheduling polyhedra for single machines. Of particular interest are the valid inequalities

$$\sum_{j\in\mathcal{S}} p_jC_j \geq \frac{1}{2}\left(\sum_{j\in\mathcal{S}} p_j\right)^2 + \frac{1}{2}\sum_{j\in\mathcal{S}} p_j^2 \quad \text{for all } \mathcal{S} \subseteq \mathcal{N} \tag{3}$$

that capture all permutations of completion times as shown by Wolsey [16] and Queyranne [11]. Valid inequalities have also

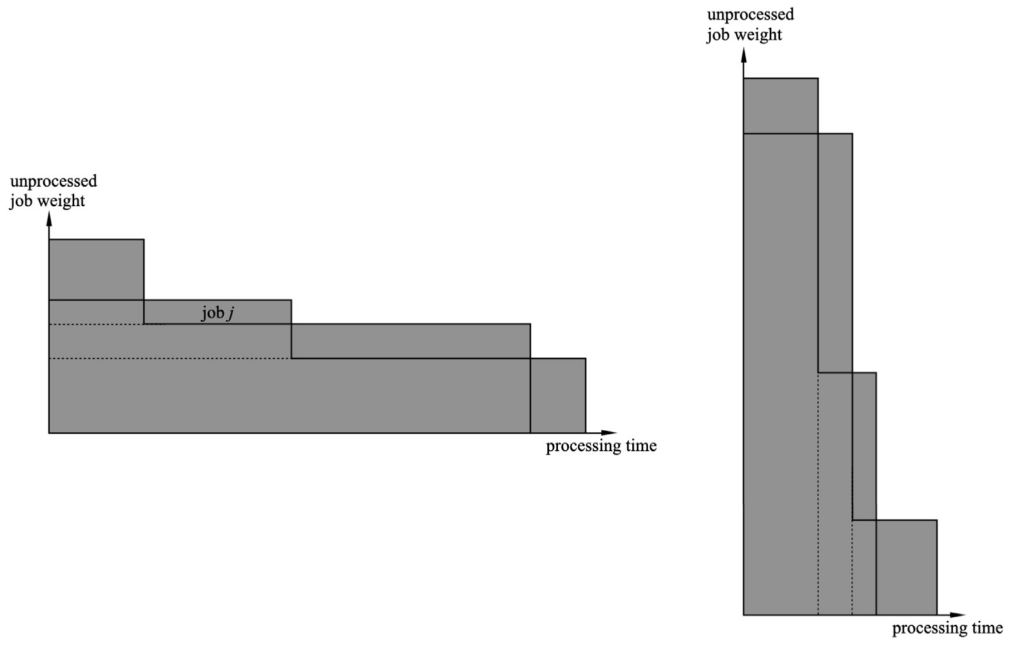

Fig. 3. An illustration of the weight–processing time flip.

been derived for parallel machines and subsequently tightened by Schulz [14] and Hall et al. [7]. We expand on Schulz's 1996 derivation [14] to obtain tighter inequalities given equal processing times. For every $\mathcal{S} \subseteq \mathcal{N}$, consider any partition $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_m$:

$$\begin{aligned}\sum_{j\in\mathcal{S}} p_jC_j &= \sum_{i=1}^{m}\sum_{j\in\mathcal{S}_i} p_jC_j \\ &\geq \sum_{i=1}^{m}\left\{\frac{1}{2}\left(\sum_{j\in\mathcal{S}_i} p_j\right)^2 + \frac{1}{2}\sum_{j\in\mathcal{S}_i} p_j^2\right\} \qquad \text{by (3)} \\ &= \frac{1}{2}\left\{\sum_{i=1}^{m}\left(\sum_{j\in\mathcal{S}_i} p_j\right)^2 + \sum_{j\in\mathcal{S}} p_j^2\right\}.\end{aligned}$$

Given $p_j = p$, it follows that

$$\sum_{j\in\mathcal{S}} C_j \geq \frac{p}{2}\left(\sum_{i=1}^{m}|\mathcal{S}_i|^2 + |\mathcal{S}|\right).$$

Here, we exploit the fact that set cardinalities are integral: the right-hand side is minimized when the number of jobs assigned to each machine is as balanced as possible. That is, given $|\mathcal{S}| = am+b$ where $a, b \in \mathbb{Z}_+$ and $b < m$, then b machines will be assigned $a+1$ jobs and the remaining $m-b$ machines will be assigned a jobs. Thus, the following inequalities remain valid for $P|p_j = p|\sum w_jC_j$, resulting in (2):

$$\begin{aligned}\sum_{j\in\mathcal{S}} C_j \geq \frac{p}{2}\Bigg(&\left\lceil\frac{|\mathcal{S}|}{m}\right\rceil^2 \cdot (|\mathcal{S}| \bmod m) \\ &+ \left\lfloor\frac{|\mathcal{S}|}{m}\right\rfloor^2 \cdot (m - |\mathcal{S}| \bmod m) + |\mathcal{S}|\Bigg)\end{aligned}$$

for every $\mathcal{S} \subseteq \mathcal{N}$, where $\lfloor x \rfloor$ is the largest integer less than or equal to x and $\lceil x \rceil$ is the smallest integer greater than or equal to x. In other words, $f(\mathcal{S})$ is the sum of completion times of $|\mathcal{S}|$ jobs that are balanced on m machines.

Definition 3.1. A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is supermodular if, for every $A \subseteq B \subseteq \mathcal{N}$ and $j \notin B$,

$$f(B \cup \{j\}) - f(B) \geq f(A \cup \{j\}) - f(A).$$

Let $\mathcal{P}$ be the polyhedron defined by the valid inequalities in (2).

Lemma 3.2. *$\mathcal{P}$ is a supermodular polyhedron with integer vertices.*

Proof. The set function f is integer-valued by definition. For completeness, suppose $|\mathcal{S}| = am + b$ where $a, b \in \mathbb{Z}_+$ and $b < m$. Then,

$$\begin{aligned}f(\mathcal{S}) &= \frac{p}{2}\Bigg(\left\lceil\frac{|\mathcal{S}|}{m}\right\rceil^2 \cdot (|\mathcal{S}| \bmod m) \\ &\qquad + \left\lfloor\frac{|\mathcal{S}|}{m}\right\rfloor^2 \cdot (m - |\mathcal{S}| \bmod m) + |\mathcal{S}|\Bigg) \\ &= \frac{p}{2}\left((a+1)^2 \cdot b + a^2(m-b) + am + b\right) \\ &= \frac{p}{2}\left(a^2m + am + 2ab + 2b\right) = \frac{p}{2}\left(a(a+1)m + 2(a+1)b\right)\end{aligned}$$

which establishes integrality.

Next, we show that f is supermodular according to Definition 3.1. First observe that $f(\emptyset) = 0$. The balancing nature of f ensures that $f(A \cup \{j\})$ places job j into a machine with $\lfloor |A|/m \rfloor$ jobs. All other machines remain unaffected. Job j then has completion time $p(\lfloor |A|/m \rfloor + 1)$. By assumption, $|B| \geq |A|$, so

$$\begin{aligned}f(B \cup \{j\}) - f(B) = p(\lfloor |B|/m \rfloor + 1) &\geq p(\lfloor |A|/m \rfloor + 1) \\ &= f(A \cup \{j\}) - f(A),\end{aligned}$$

which establishes supermodularity. □

Theorem 3.3. *The valid inequalities in (2) completely describe the scheduling polyhedron for $P|p_j = p|\sum w_jC_j$.*

Proof. Since we already know the validity of the inequalities in question, what remains to show is that $\mathcal{P}$ is contained in the scheduling polyhedron for $P|p_j = p|\sum w_jC_j$. Let C^* be an arbitrary vertex of $\mathcal{P}$, and let w be a vector such that C^* is the unique solution to the linear program $\min\{w^\intercal C : C \in \mathcal{P}\}$. Without loss of generality, suppose jobs are sorted in nonincreasing order of weights. Given Lemma 3.2, the greedy algorithm for supermodular polyhedra implies that

$$\begin{aligned} C^*_j &= f(\{1,\dots,j\}) - f(\{1,\dots,j-1\}) \\ &= p\left(\left\lfloor \frac{j-1}{m} \right\rfloor + 1\right) \end{aligned} \tag{4}$$

for all $j \in \mathcal{N}$. Given our equal-processing-time assumption, all completion times must occur at multiples of p. We conclude that C^* is a completion time vector in $P|p_j = p|\sum w_jC_j$. $\square$

A direct consequence of Theorem 3.3 is that the solutions in (4) give the following parallel-machine extension to Smith's WSPT rule for $P|p_j = p|\sum w_jC_j$.

Corollary 3.4. *An optimal solution for $P|p_j = p|\sum w_jC_j$ can be constructed by processing a job not yet processed with the largest weight whenever a machine is idle.*

Correctness of this extended WSPT rule is an established result, and it is easy to construct a proof by interchange. This paper takes a polyhedral approach to this problem instead and offers a second proof via linear programming, where the proof of Theorem 3.3 implies Corollary 3.4. In particular, the linear inequalities in (2) make explicit a strengthened class of inequalities that give a complete characterization of the scheduling polyhedron for $P|p_j = p|\sum w_jC_j$. Thus, our LP-based derivation of the extended WSPT rule, as we show next, comes with a built-in certificate of optimality that an interchange argument does not.

We now construct a feasible solution to the dual of the linear program (1)-(2), and show that our dual solution obeys complementary slackness conditions with respect to the completion time vector given by Corollary 3.4. The dual LP is

$$\begin{aligned} \max \sum_{\mathcal{S}\subseteq\mathcal{N}} & f(\mathcal{S})y_{\mathcal{S}} \\ \text{s.t.} \sum_{\mathcal{S}\subseteq\mathcal{N}:j\in\mathcal{S}} & y_{\mathcal{S}} = w_j \quad \forall j \in \mathcal{N} \\ & y_{\mathcal{S}} \ge 0 \quad \forall \mathcal{S} \subseteq \mathcal{N} \end{aligned}$$

with dual variables $y_{\mathcal{S}}$ for each primal constraint $\mathcal{S} \subseteq \mathcal{N}$. Let $n = |\mathcal{N}|$ and suppose without loss of generality that jobs are sorted in nonincreasing order of w_j. The dual solution

$$\begin{aligned} y_{\{1\}} &= w_1 - w_2 \\ y_{\{1,2\}} &= w_2 - w_3 \\ &\vdots \\ y_{\{1,\dots,n-1\}} &= w_{n-1} - w_n \\ y_{\{1,\dots,n\}} &= w_n, \end{aligned}$$

with all other variables set to zero, is feasible. The corresponding primal constraints that hold with equality are

$$\sum_{j=1}^{k} C_j = f(\{1,\dots,k\}) \quad \text{for each } k = 1,\dots,n$$

which, as in our earlier discussion, implies that

$$\begin{aligned} C_j &= f(\{1,\dots,j\}) - f(\{1,\dots,j-1\}) \\ &= p\left(\left\lfloor \frac{j-1}{m} \right\rfloor + 1\right) \end{aligned}$$

for each job $j \in \mathcal{N}$.

Finally, we use the equivalence between an input for $P|p_j = p|\sum w_jC_j$ and a flipped input for $P||\sum C_j$, in which p_j and w_j are interchanged for every job j and the order in which jobs are processed is also reversed. This sorts jobs in *nondecreasing* order of *processing times*. SPT optimality follows directly from Theorem 3.3 and Corollary 3.4.

Corollary 3.5. *An optimal solution for $P||\sum C_j$ can be constructed by processing a job not yet processed with the shortest processing time whenever a machine is idle.*

4. Extensions

Our methods and the geometric insights therein may find further uses. We apply the same principle in two generalizations that are both well known to be polynomial-time solvable [2,8,9].

4.1. Uniform machines

The first extension considers *uniform machines*, where each machine i has speed $s_i > 0$ and so processing job j on machine i takes p_j/s_i time units. This problem is denoted $Q||\sum C_j$. Much of the same principle applies in establishing an equivalence between $Q||\sum C_j$ and $Q|p_j = p|\sum w_jC_j$. A polyhedral approach similar to Theorem 3.3 can be used to solve the latter problem.

Observe that machines in $Q|p_j = p|\sum w_jC_j$ become idle at the following multiset of possible job completion times:

$$\left\{\frac{p}{s_1},\dots,\frac{p}{s_m},\frac{2p}{s_1},\dots,\frac{2p}{s_m},\dots,\frac{np}{s_1},\dots,\frac{np}{s_m}\right\}.$$

Let $t_1 \le t_2 \le \dots \le t_n$ be the n smallest numbers in the above multiset. We use the following linear program for $Q|p_j = p|\sum w_jC_j$:

$$\begin{aligned} \min & \sum_{j\in\mathcal{N}} w_jC_j \\ \text{s.t.} & \sum_{j\in\mathcal{S}} C_j \ge \sum_{j=1}^{|\mathcal{S}|} t_j \quad \text{for all } \mathcal{S} \subseteq \mathcal{N}. \end{aligned} \tag{5}$$

The validity of the inequalities in (5) is immediate.

Theorem 4.1. *The valid inequalities in (5) define a supermodular polyhedron that completely describes $Q|p_j = p|\sum w_jC_j$.*

Proof. For ease of exposition, let $g(\mathcal{S}) = \sum_{j=1}^{|\mathcal{S}|} t_j$. We first show that g is supermodular according to Definition 3.1. Observe that $g(\emptyset) = 0$. By assumption, $|B| \ge |A|$, so

$$g(B \cup \{j\}) - g(B) = t_{|B|+1} \ge t_{|A|+1} = g(A \cup \{j\}) - g(A),$$

which establishes supermodularity.

Without loss of generality, suppose jobs are sorted in nonincreasing order of weights. The greedy algorithm for supermodular polyhedra implies that

$$\begin{aligned} C_j &= g(\{1,\dots,j\}) - g(\{1,\dots,j-1\}) \\ &= t_j \end{aligned}$$

for all $j \in \mathcal{N}$, so C is indeed a completion time vector in $Q|p_j = p|\sum w_jC_j$. $\square$

By Theorem 4.1, an optimal schedule for $Q|p_j = p|\sum w_jC_j$ sorts jobs in nonincreasing order of weights and processes job k for completion at time t_k. When t_k takes the form $t_k = \ell p/s_i$, job k is the ℓth job scheduled on a machine with speed s_i. We therefore conclude that job k is the ℓth *last* job scheduled on machine i in an optimal schedule for $Q||\sum C_j$.

4.2. Eligibility constraints

Consider a generalization of $P||\sum C_j$ in which each job j is compatible only with a subset of machines $\mathcal{M}_j$. We denote this problem $P|\mathcal{M}_j|\sum C_j$ (this problem may also be denoted $R|p_{ij} \in \{p_j, \infty\}|\sum C_j$ as a special case of scheduling on *unrelated machines* where processing job j on machine i takes p_j units if $i \in \mathcal{M}_j$ and ∞ otherwise). Then, the same principle from Section 2 can be used to establish an equivalence with $P|\mathcal{M}_j, p_j = p|\sum w_jC_j$.

We present a new result on SPT optimality given the following highly structured set of inputs where machine eligibility sets $\mathcal{M}_j$ are *nested*, and the highest-weight jobs are also the least restrictive, i.e., $w_1 \le w_2 \le \cdots \le w_n$ and $|\mathcal{M}_1| \le |\mathcal{M}_2| \le \cdots \le |\mathcal{M}_n|$ hold.

Theorem 4.2. *Suppose machine eligibility sets $\mathcal{M}_j$ are nested and jobs are sorted such that $w_1 \le w_2 \le \cdots \le w_n$ and $|\mathcal{M}_1| \le |\mathcal{M}_2| \le \cdots \le |\mathcal{M}_n|$ hold. For this highly structured set of inputs, an optimal solution for $P|\mathcal{M}_j, p_j = p|\sum w_jC_j$ can be constructed by inserting jobs over time, in sorted order, into the first slot in an eligible machine with the smallest sum of job weights.*

Proof. For a proof by contradiction, consider an optimal schedule that cannot be produced by this procedure. We show that we can always construct a schedule that follows this procedure that is as good as the optimal schedule.

Let W_{ij} denote the sum of job weights in machine i when job j is about to be scheduled. Let job j be the maximum-weight job in an optimal schedule that could not have been placed there in a schedule generated by the procedure. More precisely, we assume that job j is assigned to machine i when there exists some machine $\hat{\imath} \neq i$ such that $\hat{\imath} = \arg\min_{k \in \mathcal{M}_j} W_{kj}$. Let job $\hat{\jmath}$ be the job scheduled where job j should have been, that is, the first job scheduled in machine $\hat{\imath}$ after job j. If no such job $\hat{\jmath}$ exists, then job j must be the maximum-weight job in machine i: job j is the maximum-weight job that violates the procedure, and since $\mathcal{M}_j$ is nested, any job that comes after job j that is eligible for machine i is also eligible for machine $\hat{\imath}$. Finally, $W_{\hat{\imath}j} \le W_{ij} \le W_{ij} + w_j$, so a job must be scheduled in $\hat{\imath}$ before another can be scheduled in machine i. Reassigning job j into the first slot of machine $\hat{\imath}$ places job j into a position compatible with the procedure and changes the objective by $(-W_{ij} + W_{\hat{\imath}j})p \le 0$, which establishes a contradiction.

Suppose job $\hat{\jmath}$ exists, and let C_j and $C_{\hat{\jmath}}$ be the completion times of jobs j and $\hat{\jmath}$ in an optimal schedule, respectively. If $C_j \le C_{\hat{\jmath}}$, swapping jobs j and $\hat{\jmath}$ changes the objective by

$$w_jC_{\hat{\jmath}} + w_{\hat{\jmath}}C_j - (w_jC_j + w_{\hat{\jmath}}C_{\hat{\jmath}}) = (w_j - w_{\hat{\jmath}})(C_{\hat{\jmath}} - C_j) \le 0.$$

Otherwise, if $C_j > C_{\hat{\jmath}}$, we can swap the segment $[0, C_j)$ in machine i with the segment $[0, C_{\hat{\jmath}})$ in machine $\hat{\imath}$, which changes the objective by $(-W_{ij} + W_{\hat{\imath}j})(C_j - C_{\hat{\jmath}}) \le 0$. In both cases, we place job j into a position compatible with the procedure and obtain a contradiction. Repeating this process for every job not compatible with the procedure gives an optimal solution. □

By the equivalence created by flipped inputs, an optimal solution for $P|\mathcal{M}_j|\sum C_j$ can be constructed by processing jobs in sorted order in an eligible machine with the shortest total processing time.

4.2.1. A primal-dual interpretation

We conclude by outlining an LP-based approach for solving $P|\mathcal{M}_j, p_j = p|\sum w_jC_j$ for general inputs. This problem requires a new LP formulation that explicitly considers job-to-machine assignments. Define a binary variable x_{ijk} where $x_{ijk} = 1$ if job j is the kth job processed on machine i, and 0 otherwise. Let c_{ijk} denote the cost of this assignment such that $c_{ijk} = w_jkp$. Then the integer program for $P|\mathcal{M}_j, p_j = p|\sum_j w_jC_j$ is

$$\min \sum_{j=1}^{n} \sum_{i \in \mathcal{M}_j} \sum_{k=1}^{n} c_{ijk}x_{ijk}$$

$$\text{s.t.} \sum_{i \in \mathcal{M}_j} \sum_{k=1}^{n} x_{ijk} = 1 \qquad \forall j = 1, \ldots, n \tag{6}$$

$$\sum_{j: i \in \mathcal{M}_j} x_{ijk} \le 1 \qquad \forall i = 1, \ldots, m;\ k = 1 \ldots, n \tag{7}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall j = 1, \ldots, n;\ i \in \mathcal{M}_j;\ k = 1 \ldots, n.$$

Constraint (6) ensures that every job is scheduled. By constraint (7), a machine can process at most one job at any given time. This is a bipartite matching problem with n jobs on one hand and nm machine-slot pairs on the other. It is well known that integrality constraints may be relaxed without altering the feasible region. The dual of the LP relaxation is

$$\max \sum_{j=1}^{n} u_j - \sum_{i=1}^{m} \sum_{k=1}^{n} v_{ik}$$

$$\text{s.t. } u_j \le c_{ijk} + v_{ik} \qquad \forall j = 1, \ldots, n;\ i \in \mathcal{M}_j;\ k = 1 \ldots, n \tag{8}$$

$$v_{ik} \ge 0 \qquad \forall i = 1, \ldots, m;\ k = 1 \ldots, n.$$

Dual variables u_j and v_{ik} both have natural pricing interpretations: u_j is the total cost of assignment for job j, which includes both a baseline cost c_{ijk} and a premium v_{ik} attached to the k^{th} slot in machine i. Naturally, job j ultimately chooses an assignment that minimizes its total cost.

Primal-dual algorithms that solve minimum cost bipartite matching problems have been widely studied in the literature [10]. In what follows, we describe an iterative approach that led to the insights behind Theorem 4.2.

For each $j = 1, \ldots, n$, define a bipartite graph $G_j = (L_j, R, E_j)$ where $L_j = \{1, \ldots, j\}$ is a subset of jobs, $R = \mathcal{M} \times \mathcal{N}$ is the set of machine-slot pairs, and $E_j = \{(\ell, ik) |\ \ell \in L_j, (i, k) \in R\}$. We initialize with an empty set of assignments $M = \emptyset$ and a dual feasible solution $u = v = 0$, and run a primal-dual matching algorithm on G_1. At each iteration $j = 2, \ldots, n$, we run the same algorithm on G_j with solutions obtained in the previous iteration as our initial feasible solutions. Upon termination, correctness follows automatically if $|M| = n$.

Acknowledgements

This paper is dedicated to the memory of our friend and colleague Gerhard Woeginger. It would have been nice to share it with him, since he worked on many closely related problems. We would like to believe that this is the sort of result that would prompt his wry smile in response.

References

[1] E. Balas, On the facial structure of scheduling polyhedra, in: Mathematical Programming Essays in Honor of George B. Dantzig Part I, Springer Berlin Heidelberg, 1985, pp. 179–218.
[2] P. Brucker, Scheduling Algorithms, Springer Berlin Heidelberg, 2007.

[3] R.W. Conway, W.L. Maxwell, L.W. Miller, Theory of Scheduling, Addison-Wesley Publishing Company, 1967.
[4] W.L. Eastman, S. Even, I.M. Isaacs, Bounds for the optimal scheduling of n jobs on m processors, Manag. Sci. 11 (1964) 268–279.
[5] M.X. Goemans, D.P. Williamson, Two-dimensional Gantt charts and a scheduling algorithm of Lawler, SIAM J. Discrete Math. 13 (2000) 281–294.
[6] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Discrete Optimization II, in: Annals of Discrete Mathematics, vol. 5, Elsevier, 1979, pp. 287–326.
[7] L.A. Hall, A.S. Schulz, D.B. Shmoys, J. Wein, Scheduling to minimize average completion time: off-line and on-line approximation algorithms, Math. Oper. Res. 22 (1997) 513–544.
[8] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Sequencing and scheduling: algorithms and complexity, in: Logistics of Production and Inventory, in: Handbooks in Operations Research and Management Science, vol. 4, Elsevier, 1993, pp. 445–522, Chapter 9.
[9] J.K. Lenstra, D.B. Shmoys, Elements of scheduling, CoRR, 2020.
[10] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice Hall, 1982.
[11] M. Queyranne, Structure of a simple scheduling polyhedron, Math. Program. 58 (1993) 263–285.
[12] M. Queyranne, A.S. Schulz, Polyhedral approaches to machine scheduling, Tech. Rep., 1994.
[13] M. Queyranne, Y. Wang, Single-machine scheduling polyhedra with precedence constraints, Math. Oper. Res. 16 (1991) 1–20.
[14] A.S. Schulz, Polytopes and Scheduling, PhD thesis, TU Berlin, 1996.
[15] W.E. Smith, Various optimizers for single-stage production, Nav. Res. Logist. Q. 3 (1956) 59–66.
[16] L.A. Wolsey, Mixed integer programming formulations for production planning and scheduling problems, in: 12th International Symposium on Mathematical Programming, 1985.