Chapter 48
Scheduling Unrelated Machines with Costs

David B. Shmoys* Éva Tardos†

Abstract

We consider the problem of scheduling unrelated parallel machines with costs. Each job is to be processed by exactly one machine; processing job j on machine i requires time p_{ij} and incurs a cost of c_{ij}. There are two optimization criteria: minimizing the makespan of the schedule, i.e., the maximum job completion time; and minimizing the total cost. Our main result is as follows. There is a polynomial-time algorithm that, given values C and T, finds a schedule of cost at most C and makespan at most $2T$, if there exists a schedule of cost C and makespan T. We also extend this result to a variant of the problem where, instead of a fixed processing time p_{ij}, there is a range of possible processing times for each machine–job pair, and the cost linearly increases as the processing time decreases. Finally, we show that these results imply a polynomial-time 2-approximation algorithm to minimize a weighted sum of the cost and the makespan.

1 Introduction

Consider the following scheduling problem: each of n independent jobs is to be processed by exactly one of m unrelated parallel machines; job j takes p_{ij} time units when processed by machine i, and incurs a cost c_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$. For notational simplicity, we shall assume that $n \geq m$. We are interested in two optimization criteria: minimizing the *makespan* of the schedule, i.e., the maximum job completion time; and minimizing the total cost. Lenstra, Shmoys, and Tardos [4] give a polynomial-time 2-approximation algorithm for the single criterion problem of minimizing the makespan, where a *ρ-approximation algorithm* is one that is guaranteed to produce a solution with objective function value at most ρ times the optimum. In this paper, we generalize that result to the bicriteria problem mentioned above.

Trick [8,9] and Lin & Vitter [5] consider variants of this bicriteria problem. Lin and Vitter [5] give a polynomial-time algorithm that, given cost C, makespan T, and $\epsilon > 0$, finds a solution of cost at most $(1 + \epsilon)C$ and makespan at most $(2 + 1/\epsilon)T$, if there exists a schedule of cost at most C and makespan at most T. In the variant considered by Trick [8,9], there is an interval of possible processing times, rather than a fixed time p_{ij}, and the cost of processing job j on machine i linearly increases as the processing time decreases. Trick [9] focuses on the single criterion problem of minimizing a linear objective function that is a weighted sum of the cost and the makespan, and gives a polynomial-time 2.618-approximation algorithm.

The main result of our paper is as follows. We present a polynomial-time algorithm that, given values C and T, finds a schedule of cost at most C and makespan at most $2T$, if a schedule of cost C and makespan T exists. As a corollary, we give a polynomial-time 2-approximation algorithm for the variant considered by Trick.

All of the above algorithms are based on solving linear relaxations of a particular integer programming formulation, and then rounding the fractional solution to a nearby integer solution. Whereas the results of Trick [8,9] and Lin & Vitter [5] invoke the rounding theorem of Lenstra, Shmoys & Tardos [4], the main contribution of this paper is the introduction of a new rounding technique. The technique used in [4] requires that the solution to be rounded must be a vertex of the linear relaxation. One interesting aspect of the new technique is that it does not have this restriction.

The most time-consuming part of our approximation algorithms is the solution of the linear relaxations. For our results that separately treat the two criteria, we observe that these linear programs fall into the class of fractional packing problems considered in [7], and therefore a slightly further relaxed schedule can be found by

*School of Operations Research & Industrial Engineering, Cornell University; research partially supported by an NSF PYI award CCR-89-96272 with matching support from UPS, and Sun Microsystems, and by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550.

†School of Operations Research & Industrial Engineering, Cornell University; research partially supported by a Packard Fellowship, a Sloan Fellowship, an NSF PYI award, and by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550.

a randomized algorithm in $O(n^2 \log n)$ expected time, or deterministically, in $O(mn^2 \log n)$ time.

Approximation algorithms for special cases of the scheduling problems considered in this paper have been studied over the last twenty-five years, and for a survey of this literature, the reader is referred to [3]. Finally, we note that it is likely that our results cannot be too substantially improved upon, since Lenstra, Shmoys & Tardos [4] have shown the following result for the single criterion problem of minimizing the makespan: for any $\epsilon < 1/2$, no polynomial-time $(1+\epsilon)$-approximation algorithm exists, unless $P = NP$.

2 The main result

We first consider the simplest version of our scheduling problem, when there is a fixed processing time p_{ij} and a cost c_{ij} associated with each machine $i = 1, \ldots, m$, and each job $j = 1, \ldots, n$. For any $t \geq T$, integer solutions to the following linear program, $LP(t)$, are in one-to-one correspondence with schedules of cost at most C and makespan at most T.

$$
\begin{aligned}
\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} &\leq C, \\
\sum_{i=1}^{m} x_{ij} &= 1, \quad \text{for } j = 1, \quad , n, \\
\sum_{j=1}^{n} p_{ij}x_{ij} &\leq T, \quad \text{for } i = 1, \ldots, m, \qquad LP(t) \\
x_{ij} &\geq 0, \quad \text{for } i = 1, \ldots, m, \\
& \qquad\qquad j = 1, \ldots, n, \\
x_{ij} &= 0, \quad \text{if } p_{ij} > t, \\
& \qquad\qquad i = 1, \ldots, m, \\
& \qquad\qquad j = 1, \ldots, n.
\end{aligned}
$$

THEOREM 2.1. *If $LP(t)$ has a feasible solution, then there exists a schedule that has makespan at most $T + t$ and cost at most C.*

We will prove the theorem by providing an algorithm that converts a feasible solution x of $LP(t)$ to the required schedule. We will construct a bipartite graph $B(x) = (V, W, E)$ and a value $x'(v, w)$ for each edge $(v, w) \in E$. One side of the bipartite graph consists of *job nodes*

$$W = \{w_j : j = 1, \ldots, n\}.$$

The other side consists of *machine nodes*

$$V = \{v_{is} : i = 1, \ldots, m,\ s = 1, \ldots, k_i\},$$

where

$$k_i = \lceil \sum_j x_{ij} \rceil;$$

the k_i nodes $\{v_{is} : s = 1, \ldots, k_i\}$ correspond to machine i, $i = 1, \ldots, m$.

Edges of the graph $B(x)$ will correspond to machine–job pairs (i, j) such that $x_{ij} > 0$. For each positive coordinate of x, there will be one or two corresponding edges in $B(x)$. The vector x' defined on the edges of $B(x)$ will have the property that, for each $i = 1, \ldots, m$, $j = 1, \ldots, n$,

$$x_{ij} = \sum_{s:(v_{is}, w_j) \in E} x'(v_{is}, w_j).$$

The cost of each edge $(v_{is}, w_j) \in E$ is c_{ij}.

The graph $B(x)$ and the vector x' are constructed in the following way. To construct the edges incident to the nodes corresponding to machine i, sort the jobs in order of nonincreasing processing time p_{ij}; for simplicity of notation, assume for the moment that

$$p_{i1} \geq p_{i2} \geq \cdots \geq p_{in}.$$

If $\sum_j x_{ij} \leq 1$, then there is only one node $v_{i1} \in V$ corresponding to machine i: in this case, for each $x_{ij} > 0$, include $(v_{i1}, w_j) \in E$, and set

$$x'(v_{i1}, w_j) := x_{ij}.$$

Otherwise, find the minimum index j_1 such that $\sum_{j=1}^{j_1} x_{ij} \geq 1$. Let E contain those edges (v_{i1}, w_j), $j = 1, \ldots, j_1 - 1$, for which $x_{ij} > 0$, and for each of these set

$$x'(v_{i1}, w_j) := x_{ij}.$$

Furthermore, add edge (v_{i1}, w_{j_1}) to E and set

$$x'(v_{i1}, w_{j_1}) := 1 - \sum_{j=1}^{j_1-1} x'(v_{i1}, w_j).$$

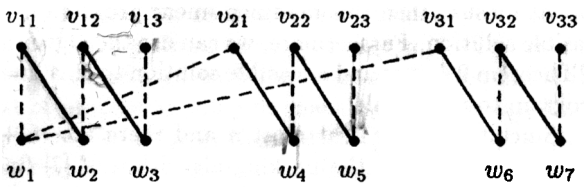
This ensures that the sum of the components of x' for edges incident to v_{i1} is exactly 1. If $\sum_{j=1}^{j_1} x_{ij} > 1$, then a fraction of the value x_{ij_1} is still unassigned, and so create an edge (v_{i2}, w_{j_1}), and set

$$x'(v_{i2}, w_{j_1}) := x_{ij_1} - x'(v_{i1}, w_{j_1}) = (\sum_{j=1}^{j_1} x_{ij}) - 1.$$

We then proceed with jobs $j > j_1$, i.e., those with smaller processing times on machine i, and assign edges to i_2, until a total of exactly one job is assigned to it, and so forth. More precisely, for each $s = 2, \ldots, k_i - 1$, find the minimum index j_s such that $\sum_{j=1}^{j_s} x_{ij} \geq s$. Let E contain those edges (v_{is}, w_j), $j = j_{s-1} + 1, \ldots, j_s - 1$, for which $x_{ij} > 0$, and for each of these set

$$x'(v_{is}, w_j) := x_{ij}$$

$$X = \begin{matrix} 1/3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 1 & 1 \end{matrix}$$



For solid edges, $x'(v_{ik}, w_j) = 2/3$

For dashed edges, $x'(v_{ik}, w_j) = 1/3$

Figure 1: Constructing $B(x)$.

Furthermore, add edge (v_{is}, w_{j_s}) to E and set

$$x'(v_{is}, w_{j_s}) := 1 - \sum_{j=j_{s-1}+1}^{j_s-1} x'(v_{is}, w_j).$$

If $\sum_{j=1}^{j_s} x_{ij} > s$, then also put edge $(v_{i,s+1}, w_{j_s}) \in E$, and set

$$x'(v_{i,s+1}, w_{j_s}) := x_{ij_s} - x'(v_{is}, w_{j_s}) = (\sum_{j=1}^{j_s} x_{ij}) - s.$$

Let j' be the last job assigned in this way; that is, $j' = j_{k_i-1}$. For each $j > j'$ for which $x_{ij} > 0$, create an edge (v_{ik_i}, w_j) and set

$$x'(v_{ik_i}, w_j) := x_{ij}.$$

For each machine node v_{is}, let $p_{is}^{\max}$ denote the maximum of the processing times p_{ij} corresponding to edges $(v_{is}, w_j) \in E$; let $p_{is}^{\min}$ denote the analogous minimum.

We shall use the following instance to give an example of this construction: $m = 3$; $n = m(m-1)+1$; $p_{i1} = m$, $i = 1, \ldots, m$; $p_{ij} = 1$, $i = 1, \ldots, m$, $j = 2, \ldots, n$; $c_{ij} = 0$, $i = 1, \ldots, m$, $j = 1, \ldots, n$; $C = 0$; $T = m$. Figure 1 gives a feasible solution $X = (x_{ij})$ to $LP(T)$, and the corresponding graph $B(x)$.

A non-negative vector z on the edges of a graph is a *fractional matching* if, for each node v, the sum of the components of z corresponding to the edges incident to v is at most 1. The fractional matching *exactly matches a node u* if the corresponding sum is exactly 1. A fractional matching z is a *matching* if each component of z is 0 or 1. The following lemma summarizes some simple properties of the above construction.

LEMMA 2.1. *The vector x' is a fractional matching in $B(x)$ of cost at most C. It exactly matches each node w_j, $j = 1, \ldots, n$, and each node v_{is}, for each $i = 1, \ldots, m$, $s = 1, \ldots, k_i - 1$. Finally, $p_{is}^{\min} \geq p_{i,s+1}^{\max}$ for each $i = 1, \ldots, m$, $s = 1, \ldots, k_i - 1$.* □

The algorithm to construct a schedule from a feasible solution x of $LP(t)$ is as follows.

The algorithm

1. Form the bipartite graph $B(x)$ with costs on its edges.
2. Find a minimum-cost (integer) matching M that exactly matches all job nodes in $B(x)$.
3. For each edge $(v_{is}, w_j) \in M$, schedule job j on machine i.

Proof of Theorem 2.1. We shall prove that the schedule produced by the algorithm satisfies the requirements of the theorem. By Lemma 2.1, x' is a fractional matching in $B(x)$ of cost at most C, which matches all job nodes exactly. This implies there exists an (integral) matching M in $B(x)$ of cost at most C that exactly matches all job nodes (see, for example, [6]). Therefore, the matching required in Step 2 exists and has cost at most C. The cost of the matching is the same as the cost of the schedule constructed. Therefore, the cost of the schedule constructed is at most C.

Next we show that the makespan of the schedule constructed is at most $T + t$. Consider the time required by machine i, $i = 1, \ldots, m$. There are k_i nodes corresponding to machine i in $B(x)$, and for each of these, there will be at most one job scheduled on machine i corresponding to some incident edge. Therefore, the length of the time required by machine i is at most $\sum_{s=1}^{k_i} p_{is}^{\max}$. Clearly, $p_{i1}^{\max} \leq t$. Lemma 2.1 implies that the sum of the remaining terms,

$$\begin{aligned} \sum_{s=2}^{k_i} p_{is}^{\max} &\leq \sum_{s=1}^{k_i-1} p_{is}^{\min} \\ &\leq \sum_{s=1}^{k_i-1} \sum_{j:(v_{is},w_j)\in E} p_{ij} x'(v_{is}, w_j) \\ &\leq \sum_{s=1}^{k_i} \sum_{j:(v_{is},w_j)\in E} p_{ij} x'(v_{is}, w_j) \\ &= \sum_{j=1}^{n} p_{ij} x_{ij} \\ &\leq T, \end{aligned}$$

which proves the theorem. □

Observe that the algorithm ensures that if $x_{ij} = 1$, then job j is assigned to be scheduled on machine i, since each edge incident to w_j in $B(x)$ is of the form (v_{is}, w_j) for some s. Also note that the obvious m-machine generalization of the example given in Figure 1 shows that the analysis of this algorithm is asymptotically tight.

COROLLARY 2.1. *In the problem with fixed processing times p_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$, for any given cost C and makespan T, we can find, in polynomial time, a schedule of cost C and makespan at most $2T$, if one of cost C and makespan T exists.*

Proof. If there exists a schedule of cost at most C and makespan at most T, then $LP(T)$ must have a feasible solution. We can use any polynomial-time linear programming algorithm to find a feasible solution to this linear program. The algorithm used to prove Theorem 2.1 can be implemented to run in polynomial time, which implies the claim. □

COROLLARY 2.2. *In the problem with fixed processing times p_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$, and nonnegative costs, for any given cost C and makespan T, and for any fixed $\epsilon > 0$, we can find a schedule of cost at most $(1+\epsilon)C$ and makespan at most $(2+\epsilon)T$, using a randomized algorithm that runs in expected $O(n^2 \log n)$ time.*

Proof. Plotkin, Shmoys and Tardos [7] developed an algorithm that efficiently finds approximate solutions to a wide class of linear programming problems, known as fractional packing problems. If the costs in $LP(T)$ are nonnegative, then this linear program is a fractional packing problem of the form considered in [7]. The techniques of [7] can be used to determine that $LP(T)$ is infeasible, or else produce a solution that is nearly feasible, in the sense that it is feasible if the right-hand sides of the cost constraint and machine load constraints are relaxed by a factor of $(1+\epsilon)$. Thus, if this algorithm produces such a fractional solution, we can then use Theorem 2.1 to find the claimed schedule.

To view the linear program $LP(T)$ as a fractional packing problem, we partition the constraints into two categories: the m machine load constraints and the cost constraint are the packing constraints, and the remaining constraints are the job assignment constraints. The algorithms of [7] work by maintaining a solution that satisfies the latter, and iteratively moving towards a solution that also satisfies the former.

An important parameter of a fractional packing problem is its *width*, which is the maximum ratio of the right-hand side to the left-hand side of any packing constraint for any solution x that satisfies the remaining constraints. The width of the above formulation can be as high as $\sum_j \max_i c_{ij}/C$. This can be improved as follows: add constraints that set $x_{ij} = 0$ if $c_{ij} > C$. As a consequence, the width is reduced to at most n. If there exists a schedule of makespan at most T and cost at most C then this modified linear program has a feasible solution. Furthermore, we can use the algorithm of Theorem 2.1 to round a feasible solution to this linear program to a schedule.

Since the width is at most n and there are $m+1$ packing constraints, the packing algorithm of [7] finds an approximate solution in $O(n \log n)$ iterations (see Theorem 2.7 of [7]). In each iteration, the algorithm first computes a dual variable corresponding to each packing constraint, which is completely determined by the current primal solution; let y denote the dual variable corresponding to the cost constraint, and let y_i correspond to the load constraint for machine i, $i = 1, \ldots, m$. The algorithm then selects a job j uniformly at random, and finds the machine i on which job j may be scheduled (i.e., $p_{ij} \leq t$ and $c_{ij} \leq C$) for which $yc_{ij} + y_i p_{ij}$ is minimum. A small fraction of job j is rescheduled on this machine. Each iteration takes $O(m)$ time. Therefore, the packing algorithm terminates in $O(mn \log n)$ expected time.

The resulting vector x has $O(n \log n)$ nonzero coordinates. Therefore, the graph $B(x)$ has at most $2n + m = O(n)$ nodes and $O(n \log n)$ edges. The minimum-cost matching that exactly matches the n job nodes can be found via n shortest path computations; using the Fredman-Tarjan implementation of Dijkstra's algorithm, the algorithm runs in $O(n^2 \log n)$ time. □

There is also a deterministic version of the algorithm of [7] that yields a running time of $O(mn^2 \log n)$.

Next consider the version of the problem where job j can be processed by machine i in t_{ij} time units, where $l_{ij} \leq t_{ij} \leq u_{ij}$ and the cost linearly increases as the processing time decreases. In this model, we are given the minimum cost c^u_{ij} and the maximum cost c^l_{ij} of assigning job j to machine i, for each $i = 1, \ldots, m$, $j = 1, \ldots, n$. The cost associated with processing job j on machine i in time t_{ij} is $\mu c^l_{ij} + (1-\mu)c^u_{ij}$ if the time can be written as $t_{ij} = \mu l_{ij} + (1-\mu)u_{ij}$, where $0 \leq \mu \leq 1$.

In the linear programming relaxation $LP_{speed}(T)$ of this problem there are two variables x^l_{ij} and x^u_{ij} associated with each machine–job pair (i,j), $i = 1, \ldots, m$, $j = 1, \ldots, n$. A feasible solution to the linear program directly corresponds to a feasible schedule if $x^l_{ij} + x^u_{ij}$ is integral for each machine–job pair (i,j), $i = 1, \ldots, m$, $j = 1, \ldots, n$. Job j is assigned to machine i if $x^u_{ij} + x^l_{ij} = 1$, where the assigned time is $t_{ij} = x^u_{ij} u_{ij} + x^l_{ij} l_{ij}$ at a cost of $c^u_{ij} x^u_{ij} + c^l_{ij} x^l_{ij}$.

In the linear program $LP(t)$, we forced the variable x_{ij} to zero if $p_{ij} > t$. Analogously, we want to make sure that no job is processed on a machine at a speed on which it would require more than t time units to process the whole job. To do this, we revise the upper bound of the processing times to $\hat{u}_{ij} = \min\{t, u_{ij}\}$. The revised cost $\hat{c}^u_{ij}$ associated with the revised upper bound is the cost of processing job j on machine i in time $\hat{u}_{ij}$, i.e., if $\hat{u}_{ij} = \mu u_{ij} + (1-\mu) l_{ij}$ then we set $\hat{c}^u_{ij} = \mu c^u_{ij} + (1-\mu) c^l_{ij}$. The resulting linear program $LP_{speed}(t)$ is as follows.

$$\begin{aligned}
\sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{c}^u_{ij}x^u_{ij} + c^l_{ij}x^l_{ij}) &\leq C, \\
\sum_{i=1}^{m}(x^u_{ij} + x^l_{ij}) &= 1, \quad \text{for } j = 1,\ldots,n, \\
\sum_{j=1}^{n}(\hat{u}_{ij}x^u_{ij} + l_{ij}x^l_{ij}) &\leq T, \quad \text{for } i = 1,\ldots,m, \\
x^u_{ij}, x^l_{ij} &\geq 0, \quad \text{for } i = 1,\ldots,m, \\
& \qquad\qquad j = 1,\ldots,n, \\
x^u_{ij} = x^l_{ij} &= 0, \quad \text{if } l_{ij} > t, \\
& \qquad\qquad i = 1,\ldots,m, \\
& \qquad\qquad j = 1,\ldots,n.
\end{aligned}$$

THEOREM 2.2. *If the linear program $LP_{speed}(t)$ has a feasible solution, then there exists a schedule with makespan at most $T + t$ and cost at most C.*

Proof. We will prove this theorem by constructing a feasible solution to a related linear program $LP(t)$ and then applying Theorem 2.1. Consider a feasible solution x^l and x^u to $LP_{speed}(t)$. Define the corresponding feasible solution x, scheduling times p, and costs c as follows. Let

$$x_{ij} = x^u_{ij} + x^l_{ij};$$

that is, x_{ij} is the fraction of job j that is scheduled on machine i. For any machine–job pair (i,j) such that $x_{ij} > 0$, define its processing time as

$$p_{ij} = (x^u_{ij}\hat{u}_{ij} + x^l_{ij}l_{ij})/x_{ij};$$

that is, p_{ij} is the time it would take to process all of job j on machine i at the speed used in the fractional schedule; the corresponding cost is defined to be

$$c_{ij} = (x^u_{ij}\hat{c}^u_{ij} + x^l_{ij}c^l_{ij})/x_{ij}.$$

For machine–job pairs (i,j) such that $x_{ij} = 0$, set $p_{ij} = +\infty$ (where any value greater than $T + t$ will suffice) and $c_{ij} = 0$.

Observe that $x_{ij} > 0$ implies that $p_{ij} \leq t$, $l_{ij} \leq p_{ij} \leq u_{ij}$, and c_{ij} is the cost of assigning job j to machine i for time p_{ij}. Notice that x is a solution to the linear program $LP(t)$ defined by T, C, and p_{ij} and c_{ij} for $i = 1,\ldots,m$, $j = 1,\ldots,n$. Therefore, Theorem 2.1 implies that the claimed schedule exists. □

Notice that the algorithm ensures that integral assignments (i.e., pairs (i,j) with $x^l_{ij} + x^u_{ij}$ integral) are used in the schedule constructed.

COROLLARY 2.3. *In the problem with variable processing times, for any given cost C and makespan T, we can find, in polynomial time, a schedule of cost C and makespan at most $2T$, if one of cost C and makespan T exists.*

Proof. If there exists a schedule of cost at most C and makespan at most T, then $LP_{speed}(T)$ must have a feasible solution. We can use any polynomial-time linear programming algorithm to find a feasible solution to this linear program. By applying Theorem 2.2, we obtain the corollary. □

COROLLARY 2.4. *In the problem with variable processing times p_{ij}, $i = 1,\ldots,m$, $j = 1,\ldots,n$, and nonnegative costs, for any given cost C and makespan T, and for any fixed $\epsilon > 0$, we can find a schedule of cost at most $(1+\epsilon)C$ and makespan at most $(2+\epsilon)T$, using a randomized algorithm that runs in expected $O(n^2 \log n)$ time.*

Proof. This proof of this result relies on techniques from [7] in a way analogous to the proof of Corollary 2.2. To make the width of the corresponding packing problem small, we must further restrict the allowed speeds for the assignments. We increase the lower bounds l_{ij} to a modified lower bound $\hat{l}_{ij}$, if necessary, to ensure that the corresponding cost $\hat{c}^l_{ij}$ is at most C. □

3 Minimizing a combined objective function

In this section, we consider the problem of minimizing the objective function consisting of a weighted sum of the makespan and the operating cost,

$$\mu T + \sum_{ij}(c^l_{ij}x^l_{ij} + c^u_{ij}x^u_{ij}),$$

for some parameter $\mu > 0$, where the desired makespan is no longer a part of the input. We assume that the operating costs

$$c^l_{ij} \geq 0 \text{ and } c^u_{ij} \geq 0, \text{ for } i = 1,\ldots,m,\ j = 1,\ldots,n.$$

We shall also assume that the lower and upper bounds on the processing times, l_{ij} and u_{ij}, respectively, are integral, for each machine–job pair (i,j). Let P denote the maximum of the upper bounds on the processing times. Trick [9] gave a ρ-approximation algorithm for this problem, where ρ is roughly 2.618. Here we use Theorem 2.2 to give a 2-approximation algorithm.

For each value $t > 0$, consider the following linear program $LP_{opt}(t)$, where $\hat{c}^u_{ij}$, as in the previous section,

is the cost of scheduling job j on machine i so that it is processed in $\hat{u}_{ij} = \min\{t, u_{ij}\}$ time units.

$$f(t) = \min\ \mu T + \sum_{i=1}^{m}\sum_{j=1}^{n}(\hat{c}^u_{ij}x^u_{ij} + c^l_{ij}x^l_{ij})$$

subject to

$$\begin{aligned}
\sum_{i=1}^{m}(x^u_{ij} + x^l_{ij}) &= 1, && \text{for } j = 1,\ldots,n,\\
\sum_{j=1}^{n}(\hat{u}_{ij}x^u_{ij} + l_{ij}x^l_{ij}) &\le T, && \text{for } i = 1,\ldots,m,\\
x^u_{ij}, x^l_{ij} &\ge 0, && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n,\\
x^u_{ij} = x^l_{ij} &= 0, && \text{if } l_{ij} > t,\ i = 1,\ldots,m,\ j = 1,\ldots,n.
\end{aligned}$$

$LP_{opt}(t)$

To find a schedule with objective function value at most twice optimal, we will perform a bisection search on the range of possible makespan values, and maintain the following invariant: all schedules with objective function value less than half the objective function value of the best schedule found thus far must have makespan within the current range. The number of iterations of this bisection search can be bounded by using the following lemma of Trick [9], which follows from a simple perturbation argument.

LEMMA 3.1. *Among all schedules with minimum objective function value, consider one with minimum makespan; the makespan of this schedule is integral.* □

Since the makespan of any plausible schedule is at most nP, it follows that we can initialize the search by setting the interval to $[0, nP]$. The core of the bisection search is given by the following lemma.

LEMMA 3.2. *For each value $t > 0$, one can, in polynomial time, find a schedule with objective function value $\bar{f}$, and conclude that one of the following holds: (i) each schedule with objective function value less than $\bar{f}/2$ has makespan less than t, or (ii) each schedule with objective function value less than $\bar{f}/2$ has makespan greater than t.*

Proof. The algorithm works as follows. First find an optimal solution x to $LP_{opt}(t)$ and compute $f(t)$. Let C and T denote the cost and makespan of this fractional solution; that is, $f(t) = C + \mu T$. Since x is a feasible solution to $LP_{speed}(t)$ for these values C and T, we can apply Theorem 2.2 to obtain a schedule. The objective function value of this schedule is at most $C + \mu(T + t) = f(t) + \mu t$.

We consider two cases: $f(t) \le \mu t$ or $f(t) \ge \mu t$. Suppose that $f(t) \le \mu t$. The schedule obtained has objective function value $\bar{f} \le f(t) + \mu t \le 2\mu t$. Any schedule with objective function value less than $\bar{f}/2 \le \mu t$ must clearly have makespan below t. Hence, we can conclude that alternative (i) holds. Suppose instead that $f(t) \ge \mu t$. We will show that each schedule with makespan at most t has objective function value at least $\bar{f}/2$. The schedule constructed has objective function value $\bar{f} \le f(t) + \mu t \le 2f(t)$. Since $f(t)$ is the optimal value of $LP_{opt}(t)$, each integral schedule with makespan at most t must have objective function value at least $f(t) \ge \bar{f}/2$. Hence, we can conclude that alternative (ii) applies. □

Observe that if $f(t) = \mu t$, then the algorithm can halt, since (i) and (ii) together imply that there does not exist a schedule with objective function value less than $\bar{f}/2$. By combining Lemma 3.1 and Lemma 3.2, we obtain the following theorem.

THEOREM 3.1. *For the problem of minimizing a weighted sum of the cost and the makespan in scheduling with variable speeds, there exists a 2-approximation algorithm, which needs to solve the linear program $LP_{opt}(t)$ at most $\log nP$ times.* □

4 Further extensions

The rounding technique used to obtain Theorem 2.1 can also be used for integer programs of a slightly more general form. In particular, in the *generalized assignment problem*, we wish to find a solution x to minimize the cost

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
\sum_{i=1}^{m} x_{ij} &= n_j, && \text{for } j = 1,\ldots,n,\\
\sum_{j=1}^{n} p_{ij}x_{ij} &\le \rho b_i, && \text{for } i = 1,\ldots,m,\\
x_{ij} &\ge 0, \text{ integer}, && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n.
\end{aligned}$$

When $\rho = 1$, this problem can be viewed as extending the problem of scheduling unrelated machines by allowing n_j jobs of type j, $j = 1,\ldots,n$, and by having machine i available for processing for only b_i time units, $i = 1,\ldots,m$. The natural extension of the proof of Theorem 2.1 shows that if C^* denotes the minimum cost with $\rho = 1$, then a solution that is feasible with $\rho = 2$ and of cost at most C^* can be found in polynomial time.

Theorem 2.1 can also be applied to another bicriteria scheduling problem. In addition to the makespan

of the schedule, another important objective is to minimize the average completion time of the jobs. Let $C^*_{\max}$ denote the optimal value of the makespan, and let C^*_{ave} denote the optimal value of the average completion time. Horn [2] and Bruno, Coffmann, and Sethi [1] showed that an optimal solution for the latter objective can be found by reducing the problem to the minimum-cost bipartite matching problem. The bipartite graph formed by the reduction is quite similar to the one used in our approximation algorithm. There are n nodes corresponding to each machine i, $i = 1, \ldots, m$; a job matched to the kth node corresponding to machine i is thought of as scheduled kth-to-last on that machine. If job j is the kth-to-last job on machine i, then it contributes p_{ij} to the completion time of k jobs; hence, the cost of the edge (v_{ik}, w_j) is kp_{ij}.

Observe that in an optimal schedule, the jobs processed on each machine i, $i = 1, \ldots, m$, are sequenced in order of nondecreasing processing time. Thus, in the construction used in the approximation algorithm, we can similarly view v_{ik} as the kth-to-last position on machine i. Among all schedules with average completion time C^*_{ave}, let $\bar{C}_{\max}$ denote the makespan of the schedule with minimum makespan. By applying Theorem 2.1, we can efficiently find a schedule with average completion time at most C^*_{ave} and makespan at most $2\bar{C}_{\max}$.

A much stronger result would state that it is possible to find a schedule with average completion time at most C^*_{ave} and makespan at most $2C^*_{\max}$. This result is known for the special case when the machines are identical: an optimal average completion time schedule can be found by sequencing the jobs in nondecreasing processing time order, and iteratively scheduling the next job on the machine on which it would finish earliest; the same algorithm is known to be a 2-approximation algorithm for the minimum makespan problem. We leave the following question as an interesting open problem: is the makespan of any optimal average completion time schedule at most $2C^*_{\max}$, or, equivalently, is this true for any schedule produced by minimum-cost matching algorithm as described above?

Acknowledgments

We would like to thank Leslie Hall for prompting this research. Her discovery of an error in our earlier, trivial, but fallacious proof of Corollary 2.1 led us to find this more interesting, correct solution.

References

[1] J. L. Bruno, E. G. Coffmann, Jr. and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17:382–387, 1974.

[2] W. A. Horn. Minimizing average flow time with parallel machines. *Operations Research*, 21:846–847, 1973.

[3] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *Sequencing and Scheduling: Algorithms and Complexity.* Designing Decision Support Systems Notes NFI 11.89/03, Eindhoven University of Technology, 1989.

[4] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming A*, 46:259–271, 1990.

[5] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 771–782, 1992.

[6] L. Lovász and M. Plummer. *Matching Theory.* Akademiai Kiado, Budapest and North-Holland, Amsterdam, 1986.

[7] S. A. Plotkin, D. B. Shmoys, and É. Tardos. *Fast approximation algorithms for fractional packing and covering problems.* Technical Report 999, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, 1992.

[8] M. A. Trick. Scheduling multiple variable-speed machines. In *Proceedings of the 1st Conference on Integer Programming and Combinatorial Optimization*, pages 485–494, 1990.

[9] M. A. Trick. Scheduling multiple variable-speed machines. Unpublished manuscript, 1991.