Approximation Algorithms for Scheduling Unrelated Parallel Machines

Jan Karel Lenstra
Centre for Mathematics and Computer Science, Amsterdam

David B. Shmoys
Massachusetts Institute of Technology, Cambridge

Éva Tardos
Eötvös University, Budapest

ABSTRACT

We consider the following scheduling problem. There are m parallel machines and n independent jobs. Each job is to be assigned to one of the machines. The processing of job j on machine i requires time p_{ij}. The objective is to find a schedule that minimizes the makespan.

Our main result is a polynomial algorithm which constructs a schedule that is guaranteed to be no longer than twice the optimum. We also present a polynomial approximation scheme for the case that the number of machines is fixed. Both approximation results are corollaries of a theorem about the relationship of a class of integer programming problems and their linear programming relaxations. In particular, we give a polynomial method to round the fractional extreme points of the linear program to integral points that nearly satisfy the constraints.

In contrast to our main result, we prove that no polynomial algorithm can achieve a worst-case ratio less than 3/2 unless $P = NP$. We finally obtain a complexity classification for all special cases with a fixed number of processing times.

1. INTRODUCTION

Although the performance of approximation algorithms has been studied for over twenty years, very little is understood about the structural properties of a problem that permit good performance guarantees. In fact, there are practically no tools to distinguish those problems for which there does exist a polynomial algorithm for any performance bound, and those for which this is not the case. One problem area in which these questions have received much attention is that of scheduling and bin packing. We examine a scheduling problem for which all previously analyzed polynomial algorithms have particularly poor performance guarantees. We present a polynomial algorithm that delivers a solution guaranteed to be within a factor of 2 of the optimum, and prove that this is nearly best possible, in the sense that no polynomial algorithm can guarantee a factor less than 3/2 unless $P = NP$. Our algorithm is based on a result concerning the relationship of certain integer programming problems and their linear relaxations that is of interest in its own right.

One of the most natural strategies to obtain good solutions to an integer linear program is to drop the integrality constraints, solve the resulting linear programming problem, and then round the solution to an integral solution. There are many difficulties with this approach. The rounded solution may be suboptimal, and it may even be impossible to round the solution to a feasible solution. For restricted classes of integer programs, however, the behavior might not be quite as bad. Certainly, if the extreme points of the linear programming relaxation are all integral, then the optimal solution is obtained without even rounding, as is the case, for example, for the bipartite matching, maximum flow, and minimum cost flow problems.

It is an interesting question to study those classes of integer programs for which the linear relaxations provide a good approximation, in that rounded solutions can be found that are nearly feasible or nearly optimal. Much work along these lines has been done for integer programs where the coefficients of the constraints are restricted to {0,1} [Lovász 1975; Chvátal 1979; Bartholdi, Orlin, and Ratliff 1980; Bartholdi 1981; Baum and Trotter 1981; Marcotte 1983; Aharoni, Erdös, and Linial 1985; Raghavan and Thompson 1985; Raghavan 1986]. We present a rounding theorem of this sort for a natural class of integer programs with arbitrary coefficients.

The scheduling problem to be considered is as follows. There are n independent jobs that must be scheduled without preemption on a collection of m parallel machines. If job j is scheduled on machine i, the processing time required is p_{ij}, which we assume to be a positive integer. The total time used by machine i is the

0272-5428/87/0000/0217$01.00 © 1987 IEEE

sum of the p_{ij} for the jobs that are assigned to machine i, and the makespan of a schedule is the maximum total time used by any machine. The objective is to find a schedule that minimizes the makespan. Graham, Lawler, Lenstra, and Rinnooy Kan [1979] denote this problem by $R||C_{\max}$. Davis and Jaffe [1981] presented a list scheduling algorithm and proved that it delivers a schedule with makespan no more than $2\sqrt{m}$ times the optimum. So far, this was the best performance bound known for any polynomial algorithm for the problem. We present a polynomial algorithm that guarantees a factor of 2.

Approximation algorithms for this problem and several of its special cases have been studied for over two decades. Much of this work has focused on the case where the machines are *identical*; that is, $p_{hj} = p_{ij}$ for any job j and any two machines h,i. The area of worst-case analysis of approximation algorithms for *NP*-hard optimization problems can be traced to Graham [1966], who showed that for this special case with identical machines, a list scheduling algorithm always delivers a schedule with makespan no more than $(2-1/m)$ times the optimum. We shall refer to an algorithm that is guaranteed to produce a solution of length no more than ρ times the optimum as a *ρ-approximation algorithm*. Note that we do not require such an algorithm to be polynomial, although our primary focus will be on this subclass.

An important family of further restricted cases is obtained by considering a fixed number of identical machines. Graham [1969] showed that for any specified number m of machines, it is possible to obtain a polynomial $(1+\epsilon)$-approximation algorithm for any fixed $\epsilon > 0$, but the running time depends exponentially on $1/\epsilon$ (and on m). Such a family of algorithms is called a *polynomial approximation scheme*. This result was improved by Sahni [1976], who reduced the dependence of the running time on $1/\epsilon$ to a polynomial. Such a family of algorithms is called a *fully polynomial approximation scheme*.

If the number of machines is specified as part of the problem instance, results by Garey and Johnson [1975, 1978] imply that no fully polynomial approximation scheme can exist, even if the machines are identical, unless $P = NP$. However, Hochbaum and Shmoys [1987] presented a polynomial approximation scheme for the problem with identical machines.

A natural generalization of identical machines is the case of machines that run at different speeds but do so *uniformly*. Thus, for each machine i there is a speed factor s_i, and $p_{ij} = p_j/s_i$ where p_j is the inherent processing requirement of job j. Results analogous to the case of identical machines have been obtained for uniform machines. Gonzalez, Ibarra, and Sahni [1977] gave a polynomial 2-approximation algorithm. For any fixed number of machines, Horowitz and Sahni [1976] presented a fully polynomial approximation scheme, whereas Hochbaum and Shmoys [1988] gave a polynomial approximation scheme for the case that the number of machines is a part of the problem instance.

Given these strong results for special cases, there was no apparent reason to suspect that analogous results did not hold for the general setting of *unrelated* machines. In fact, Horowitz and Sahni [1976] also presented a fully polynomial approximation scheme for any fixed number of unrelated machines. However, for the case that the number of machines is specified as part of the problem instance, a polynomial approximation scheme is unlikely to exist. We prove that the existence of a polynomial $(1+\epsilon)$-approximation algorithm for any $\epsilon < 1/2$ would imply that $P = NP$.

An interesting algorithm for this problem was presented by Potts [1985]. It is a 2-approximation algorithm with running time bounded by m^{m-1} times a polynomial in the input size. At first glance, his result does not appear to be particularly interesting, since for fixed m, a fully polynomial approximation scheme was already known. However, that scheme not only requires time $O(nm(nm/\epsilon)^{m-1})$ but also space $O((nm/\epsilon)^{m-1})$, while Potts' algorithm requires only polynomial space. Thus, from both a practical and a theoretical viewpoint, Potts' algorithm is a valuable contribution. It is based on extending the integral part of a linear programming solution by an enumerative process. We extend his work by proving that the fractional solution to the linear program can be rounded to a good integral approximation in polynomial time, thereby obviating the need for enumeration and removing the exponential dependence on m. We also consider the problem with any fixed number of machines and present a polynomial approximation scheme for this case, where the space required is bounded by a polynomial in the input, m, and $\log(1/\epsilon)$.

Another natural way to restrict the problem is to consider instances where the number of different processing times is bounded. For example, if all processing times are equal, then the optimum schedule is computable in polynomial time. As a byproduct of our investigation, we obtain a complete characterization of the polynomially solvable special cases with a fixed number of processing times under the assumption that $P \neq NP$.

2. A Rounding Theorem

We first present the key tool for our approximation algorithms. Let $J_i(t)$ denote the set of jobs that require no more than t time units on machine i, and let $M_j(t)$ denote the set of machines that can process job j in no more than t time units. Consider a decision version of our scheduling problem, where for each machine i there

is a deadline d_i and where we are further constrained to schedule jobs so that each uses processing time at most t; we wish to decide if there is a feasible schedule.

THEOREM 1 (ROUNDING THEOREM). *Let* $P = (p_{ij}) \in \mathbf{Z}_+^{m \times n}$, $(d_1, \ldots, d_m) \in \mathbf{Z}_+^m$, *and* $t \in \mathbf{Z}_+$. *If the linear program*

$$\begin{aligned} &\textstyle\sum_{i \in M_j(t)} x_{ij} = 1 && \text{for } j = 1, \ldots, n, \\ &\textstyle\sum_{j \in J_i(t)} p_{ij} x_{ij} \leqslant d_i && \text{for } i = 1, \ldots, m, \\ &x_{ij} \geqslant 0 && \text{for } j \in J_i(t),\ i = 1, \ldots, m, \end{aligned} \tag{LP}$$

has a feasible solution, then any vertex $\tilde{x}$ *of this polytope can be rounded to a feasible solution* $\bar{x}$ *of the integer program*

$$\begin{aligned} &\textstyle\sum_{i \in M_j(t)} x_{ij} = 1 && \text{for } j = 1, \ldots, n, \\ &\textstyle\sum_{j \in J_i(t)} p_{ij} x_{ij} \leqslant d_i + t && \text{for } i = 1, \ldots, m, \\ &x_{ij} \in \{0, 1\} && \text{for } j \in J_i(t),\ i = 1, \ldots, m, \end{aligned} \tag{IP}$$

and this rounding can be done in polynomial time.

Proof. Let v denote the number of variables in the linear program (LP). This polyhedron is defined by $v + m + n$ constraints and is contained in the unit hypercube. Each vertex of such a *pointed* polyhedron is determined by v linearly independent rows of the constraint matrix such that each of these constraints is satisfied with equality [Schrijver 1986]. As a result, for any vertex $\tilde{x}$ all but $m + n$ of the variables must have value 0, and a straightforward counting argument shows that all but $2m$ must have integral values. In the remainder of the proof, we first show a somewhat stronger structural property, and then use it to round the solution.

It will be convenient to associate the rows and columns of P with machines and jobs, as is true in our application. Suppose that (LP) is feasible and let $\tilde{x}$ be a vertex of (LP). Form a bipartite graph $G = (M, J, E)$, where $M = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ correspond to the sets of machines and jobs, respectively, and $E = \{(i,j) \mid \tilde{x}_{ij} > 0\}$. We have already indicated that G has no more edges than nodes. We now show that each connected component of G has this property; that is, G is a *pseudoforest*. (This result was already stated in a slightly different form by Dantzig [1963].)

Suppose that G has c connected components. We partition the constraints and the variables x_{ij} that are assigned positive values according to connected components. Let X_k denote the set of variables corresponding to edges in the kth component ($k = 1, \ldots, c$), and let Z be the set of variables with $\tilde{x}_{ij} = 0$. Aside from the nonnegativity constraints, each constraint in (LP) can be associated with a machine or a job. Let A be the matrix formed with rows corresponding to the coefficients of the left-hand sides of precisely those constraints satisfied by $\tilde{x}$ with equality. Let R_k denote the set of rows in A that correspond to constraints associated with job and machine nodes contained in the kth component ($k = 1, \ldots, c$). We use the R_k and X_k to reorder the rows and columns of A to obtain the permuted matrix A' depicted in Figure 1.

	X_1	X_2	. . .	X_c	Z
R_1	C_1	0	. . .	0	B_1
R_2	0	C_2	. . .	0	B_2
. . .	. . .	. . .	. . .	. . .	. . .
R_c	0	0	. . .	C_c	B_c
	0	0	. . .	0	I

FIGURE 1. The permuted matrix A'.

Since A has full column rank, the permuted matrix A' must be of full column rank, and if we perform elementary row operations to replace each B_k by 0, we see that the resulting matrix A'' must also have full column rank. However, the rank of such a block matrix must be equal to the sum of the column ranks of the C_k and I. We conclude that each C_k must be a matrix of full column rank. In other words, the number of tight constraints corresponding to job and machine nodes in the kth component is at least equal to the number of edges in it, and thus the number of nodes in each component is at least equal to the number of edges. Each component of G is therefore either a tree or a tree plus one additional edge, so that G is a pseudoforest.

We now use the fact that G is a pseudoforest to round the corresponding vertex $\tilde{x}$. Consider each edge (i,j) with $\tilde{x}_{ij} = 1$. For each such edge, we adopt this integral schedule for job j and set $\bar{x}_{ij} = 1$. These jobs correspond to the job nodes of degree 1, so that by deleting all of these nodes we get a pseudoforest G' with the additional property that each job node has degree at least 2.

We show that G' has a matching that covers all of the job nodes. For each component that is a tree, root the tree at any node, and match each job node with any one of its children. (Note that each job node must have at

least one child and that, since each machine node has at most one parent, no machine is matched with more than one job.) For each component that contains a cycle, take alternate edges of the cycle in the matching. (Note that the cycle must be of even length.) If the edges of the cycle are deleted, we get a collection of trees which we think of as rooted at the node that had been contained in the cycle. For each job node that is not already matched, pair it with one of its children. This gives us the desired matching. If (i,j) is in the matching, set $\bar{x}_{ij} = 1$. Each remaining $\bar{x}_{ij}$ that has not been assigned is set to 0.

It is straightforward to verify that $\bar{x}$ is a feasible solution to (IP). Each job has been scheduled on exactly one machine, so that

$$\sum_{i \in M_j(t)} \bar{x}_{ij} = 1$$

for $j = 1, \ldots, n$. For each machine i, there is at most one job j such that $\tilde{x}_{ij} < \bar{x}_{ij} = 1$; since $p_{ij} \leq t$ for each x_{ij} in (IP), we have

$$\sum_{j \in J_i(t)} p_{ij} \bar{x}_{ij} \leq \sum_{j \in J_i(t)} p_{ij} \tilde{x}_{ij} + t \leq d_i + t$$

for $i = 1, \ldots, m$. □

3. Approximation Algorithms

We are now ready to present approximation algorithms for the minimum makespan problem on unrelated parallel machines. A notion that will play a central role in these algorithms is that of a *ρ-relaxed decision procedure*. Consider the following decision version of the problem: given a matrix P of processing times and a deadline d, does there exist a schedule with makespan at most d? On input (P,d), an ordinary decision procedure would output 'yes' or 'no', depending on whether there was in fact such a schedule. A ρ-relaxed decision procedure outputs 'no' or 'almost'; more precisely, on input (P,d),

(1) it either outputs 'no' or produces a schedule with makespan at most ρd, and

(2) if the output is 'no', then there is no schedule with makespan at most d.

Variations of the following lemma have been used in several recent results on approximation algorithms for scheduling problems [Hochbaum and Shmoys 1987].

Lemma 1. *If there is a polynomial ρ-relaxed decision procedure for the minimum makespan problem on unrelated parallel machines, then there is a polynomial ρ-approximation algorithm for this problem.*

Proof. On input P, construct a greedy schedule, where each job is assigned to the machine on which it runs fastest. If the makespan of this schedule is t, then t is an upper bound on the optimum makespan, whereas t/m is a lower bound. Using these initial bounds, we run a binary search procedure. If u and l are the current upper and lower bounds, set $d = \lfloor (u+l)/2 \rfloor$ and apply the ρ-relaxed decision procedure to (P,d). If the answer is yes, then reset u to d, and otherwise reset l to $d+1$, while storing the best solution obtained so far. When the upper and lower bounds are equal, output the best solution found.

It is easy to see that this procedure has the appropriate performance guarantee. A trivial inductive argument shows that l is always a lower bound on the optimum makespan and that the best solution encountered has makespan at most ρu. Since $u = l$ at termination, we get the desired bound. Furthermore, the algorithm runs clearly in polynomial time. The difference between the upper and lower bounds after $k + \log m$ iterations is bounded by 2^{-k} times the optimum. Thus after a polynomial number of iterations, the difference is less than 1 and the algorithm terminates. □

To obtain a polynomial 2-approximation algorithm is quite simple, given the rounding theorem and Lemma 1. We construct a 2-relaxed decision procedure for the decision version of the problem. Let (P,d) be a problem instance. Consider the linear program (LP) of the rounding theorem, with $d_1 = \cdots = d_m = t = d$. If the instance is a 'yes' instance, then the schedule that completes by time d gives a feasible solution to the linear program: simply set x_{ij} to 1 if job j is assigned to machine i and 0 otherwise. In this case, the feasible region of the linear program is nonempty and pointed, and so it is possible to find a vertex $\tilde{x}$ in polynomial time [Khachian 1979; Grötschel, Lovász, and Schrijver 1987]. Thus, if no vertex of (LP) is found, the instance must be a 'no' instance; otherwise, the procedure given in the proof of the rounding theorem produces a solution to the integer program (IP). This 0-1 solution can be interpreted as a schedule in the obvious way, and it has makespan at most $2d$. Hence, the procedure is a 2-relaxed decision procedure. We have proved the following result.

Theorem 2. *There is a 2-approximation algorithm for the minimum makespan problem on unrelated parallel machines that runs in time bounded by a polynomial in the input size.*

The analysis of the algorithm cannot be improved to yield a better bound. Consider the following instance. There are $m^2 - m + 1$ jobs and m identical machines. The first job takes m time units on all machines, and all other jobs take one time unit on all machines. Clearly, the optimal schedule has makespan m: assign the first

job to one machine and m of the remaining jobs to each of the other machines. No deadline less than m has a feasible fractional schedule. Suppose that the vertex of (LP) that is found corresponds to the schedule where one unit of the job of length m and $m-1$ unit length jobs are assigned to each machine. Rounding this fractional solution produces a schedule of length $2m-1$.

As a second application of the rounding theorem, we give a polynomial approximation scheme for any fixed number m of machines. The running time of the procedure A_ϵ, which produces a schedule guaranteed to be within a factor $1+\epsilon$ of the optimum, will be bounded by a function that is the product of $(n+1)^{m/\epsilon}$ and a polynomial in the size of the input P. Given the fully polynomial approximation scheme of Horowitz and Sahni [1976], one may question the novelty of such a scheme. The significance of the new result lies in the fact that the space required by the old scheme is $(nm/\epsilon)^m$ whereas the new scheme uses space that is polynomial in both $\log(1/\epsilon)$ and m (and the input size).

Again, all we have to do is to construct a $(1+\epsilon)$-relaxed decision procedure for the decision version of the problem. Given (P,d), the algorithm attempts to find solutions to $(n+1)^{m/\epsilon}$ linear programs. If there is a schedule with makespan at most d, then one of these linear programs has a feasible solution, and this will correspond to a schedule with makespan at most $(1+\epsilon)d$. This suffices to guarantee the properties of a $(1+\epsilon)$-relaxed decision procedure.

For any schedule for the instance (P,d), we classify the assignment of a job to a machine as either *long* or *short*, depending on whether or not the processing time in question is greater than ϵd. No machine can handle $1/\epsilon$ or more long assignments before time d. Thus, for any instance there are less than $(n+1)^{m/\epsilon}$ schedules of long assignments.

If the instance (P,d) has a feasible schedule, then this includes a partial schedule of long assignments (which may be empty). Suppose that for machine i the long assignments amount to a total processing time t_i, and thus the remaining jobs are completed within time $d_i = d - t_i$. If we then set $t = \epsilon d$, we see that the linear program (LP) must once again have a feasible solution, so that we can apply the rounding theorem. The resulting integral solution yields a schedule of short assignments such that the total time taken by short assignments to machine i is at most $d - t_i + \epsilon d$. Combining this with the schedule of long assignments, we get a schedule where the total time used by machine i is at most $t_i + d - t_i + \epsilon d = (1+\epsilon)d$.

We try all possible schedules of long assignments in this way, computing the remaining available time on each machine and applying the rounding procedure. Either we conclude that the instance is a 'no' instance, or we produce a schedule with makespan at most $(1+\epsilon)d$. (Notice that if $\epsilon = 1$, there are no long assignments and the algorithm reduces to the previous one.) We have shown the following result.

THEOREM 3. *Let m be a fixed integer. There is a family $\{A_\epsilon\}$ of algorithms such that, for each $\epsilon > 0$, A_ϵ is a $(1+\epsilon)$-approximation algorithm for the minimum makespan problem on m unrelated parallel machines that requires time bounded by a polynomial in the input size and space bounded by a polynomial in m, $\log(1/\epsilon)$, and the input size.*

An interesting open question is whether this result can be strengthened to give a *fully* polynomial approximation scheme for fixed values of m, where the space required is bounded by a polynomial in m, $\log(1/\epsilon)$, and the input size.

4. LIMITS TO APPROXIMATION

We now present results which show that certain polynomial approximation algorithms cannot exist unless $P = NP$. To this end, we investigate the computational complexity of decision versions of our scheduling problem with small integral deadlines.

THEOREM 4. *For the minimum makespan problem on unrelated parallel machines, the question of deciding if there exists a schedule with makespan at most 3 is NP-complete.*

Proof. We prove this result by a reduction from the 3-dimensional matching problem, which is known to be *NP*-complete:

3-DIMENSIONAL MATCHING
Instance: Disjoint sets $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, $C = \{c_1, \ldots, c_n\}$, and a family $F = \{T_1, \ldots, T_m\}$ of triples with $|T_i \cap A| = |T_i \cap B| = |T_i \cap C| = 1$ for $i = 1, \ldots, m$.
Question: Does F contain a matching, i.e., a subfamily F' for which $|F'| = n$ and $\bigcup_{T_i \in F'} T_i = A \cup B \cup C$?

Given an instance of this problem, we construct an instance of the scheduling problem with m machines and $2n+m$ jobs. Machine i corresponds to the triple T_i, for $i = 1, \ldots, m$. There are $3n$ 'element jobs' that correspond to the $3n$ elements of $A \cup B \cup C$ in the natural way. In addition, there are $m-n$ 'dummy jobs'. (If $m < n$, we construct some trivial 'no' instance of the scheduling problem.) Machine i corresponding to $T_i = (a_j, b_k, c_l)$ can process each of the jobs corresponding to a_j, b_k and c_l in one time unit and each other job in three time units. Note that the dummy jobs require

three time units on each machine.

It is quite simple to show that there is a schedule with makespan at most 3 if and only if there is a 3-dimensional matching. Suppose there is a matching. For each $T_i = (a_j, b_k, c_l)$ in the matching, schedule the element jobs corresponding to a_j, b_k and c_l on machine i. Schedule the dummy jobs on the $m-n$ machines corresponding to the triples that are not in the matching. This gives a schedule with makespan 3. Conversely, suppose that there is such a schedule. Each of the dummy jobs requires three time units on any machine and is thus scheduled by itself on some machine. Consider the set of n machines that are not processing dummy jobs. Since these are processing all of the $3n$ element jobs, each of these jobs is processed in one time unit. Each three jobs that are assigned to one machine must therefore correspond to elements that form the triple corresponding to that machine. Since each element job is scheduled exactly once, the n triples corresponding to the machines that are not processing dummy jobs form a matching. □

As an immediate corollary of Theorem 4, we get the following result.

COROLLARY 1. *For every $\rho < 4/3$, there does not exist a polynomial ρ-approximation algorithm for the minimum makespan problem on unrelated parallel machines, unless $P = NP$.*

The technique employed in Theorem 4 can be refined to yield a stronger result.

THEOREM 5. *For the minimum makespan problem on unrelated parallel machines, the question of deciding if there exists a schedule with makespan at most 2 is NP-complete.*

Proof. We again start from the 3-dimensional matching problem. We call the triples that contain a_j *triples of type* j. Let t_j be the number of triples of type j, for $j = 1, \ldots, n$. As before, machine i corresponds to the triple T_i, for $i = 1, \ldots, m$. There are now only $2n$ element jobs, corresponding to the $2n$ elements of $B \cup C$. We refine the construction of the dummy jobs: there are $t_j - 1$ dummy jobs of type j, for $j = 1, \ldots, n$. (Note that the total number of dummy jobs is $m-n$, as before.) Machine i corresponding to a triple of type j, say, $T_i = (a_j, b_k, c_l)$, can process each of the element jobs corresponding to b_k and c_l in one time unit and each of the dummy jobs of type j in two time units; all other jobs require three time units on machine i.

Suppose there is a matching. For each $T_i = (a_j, b_k, c_l)$ in the matching, schedule the element jobs corresponding to b_k and c_l on machine i. For each j, this leaves $t_j - 1$ idle machines corresponding to triples of type j that are not in the matching; schedule the $t_j - 1$ dummy jobs of type j on these machines. This completes a schedule with makespan 2. Conversely, suppose that there is such a schedule. Each dummy job of type j is scheduled on a machine corresponding to a triple of type j. Therefore, there is exactly one machine corresponding to a triple of type j that is not processing dummy jobs, for $j = 1, \ldots, n$. Each such machine is processing two element jobs in one time unit each. If the machine corresponds to a triple of type j and its two unit-time jobs correspond to b_k and c_l, then (a_j, b_k, c_l) must be the triple corresponding to that machine. Since each element job is scheduled exactly once, the n triples corresponding to the machines that are not processing dummy jobs form a matching. □

COROLLARY 2. *For every $\rho < 3/2$, there does not exist a polynomial ρ-approximation algorithm for the minimum makespan problem on unrelated parallel machines, unless $P = NP$.*

5. Restricted Processing Times

We conclude this paper with a few remarks about special cases of our scheduling problem in which the number of different processing times is bounded. If all $p_{ij} = 1$, the problem is clearly solvable in polynomial time, and even if all $p_{ij} \in \{1, \infty\}$, the problem can be solved by bipartite cardinality matching. Theorem 6 shows that, if all $p_{ij} \in \{1,2\}$, the problem is still solvable in polynomial time by matching techniques.

THEOREM 6. *The minimum makespan problem on unrelated parallel machines is solvable in polynomial time in the case that all $p_{ij} \in \{1,2\}$.*

Proof. The problem of deciding if there exists a schedule with makespan at most d can be transformed into the following problem, which is known to be solvable in polynomial time by matching techniques [Schrijver 1983]: Given a bipartite graph $G = (S, T, E)$, find a subgraph with a maximum number of edges in which each node in S has degree 0 or 2 and each node in T has degree 1.

In case $d = 2k$, construct G as follows: there are k nodes in S for each machine; there is one node in T for each job; and edges correspond to unit processing times. Now solve the above problem on G. Construct a partial schedule of unit-time assignments corresponding to the edges in the subgraph, and try to extend it to a complete schedule by assigning the remaining jobs for two time units each while respecting the deadline. If a schedule with makespan $2k$ exists, this procedure will find such a

schedule.

In case $d = 2k-1$, create an additional unit-time job for each machine, which requires two time units on any other machine, and apply the above procedure for $d = 2k$. It follows from a straightforward interchange argument that the original problem has a schedule with makespan $2k-1$ if and only if the modified problem has a schedule with makespan $2k$. □

Our final theorem implies that all other cases with a fixed number of processing times cannot be solved in polynomial time, unless $P = NP$.

THEOREM 7. *The minimum makespan problem on unrelated parallel machines is NP-hard in the case that all* $p_{ij} \in \{p,q\}$ *with* $p < q$, $2p \neq q$.

Proof. We assume without loss of generality that p and q are relatively prime. Recall that the result has already been proved in Theorem 4 for the case that all $p_{ij} \in \{1,3\}$ by a reduction from 3-dimensional matching. We now reduce from q-dimensional matching. Given a matching instance with m q-tuples over a ground set of qn elements, we construct a scheduling instance with qn element jobs, $p(m-n)$ dummy jobs, and m machines. An element job can be processed in p time units by each machine that corresponds to a tuple containing the corresponding element; all other processing times are q time units. It is trivial to see that for each matching there is a schedule with makespan pq. To prove the opposite implication, no schedule with makespan pq can have idle time, and it now follows from an easy number theoretic argument that each machine processes either q element jobs of length p or p dummy jobs of length q. □

ACKNOWLEDGEMENTS

The research of the first author was supported in part by the National Science Foundation under grant MCS-83-11422. The research by the second author was supported in part by the National Science Foundation under grant ECS-85-01988 and by Air Force contract AFOSR-86-0078. The research by the third author was supported in part by the National Science Foundation under grant MCS-81-20790 and by the Hungarian National Foundation for Scientific Research under grant 1812.

REFERENCES

R. AHARONI, P. ERDÖS, N. LINIAL (1985). Dual integer linear programs and the relationship between their optima. *Proc. 17th Annual ACM Symp. Theory of Computing*, 476-483.

J.J. BARTHOLDI III, J.B. ORLIN, H.D. RATLIFF (1980). Cyclic scheduling via integer programs with circular ones. *Oper. Res. 28*, 1074-1085.

J.J. BARTHOLDI III (1981). A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Oper. Res. 29*, 501-510.

S. BAUM, L.E. TROTTER, JR. (1981). Integer rounding for polymatroid and branching optimization problems. *SIAM J. Algebraic Discrete Methods 2*, 416-425.

V. CHVÁTAL (1979). A greedy heuristic for the set-covering problem. *Math. Oper. Res. 4*, 233-235.

G.B. DANTZIG (1963). *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ.

E. DAVIS, J.M. JAFFE (1981). Algorithms for scheduling tasks on unrelated processors. *J. Assoc. Comput. Mach. 28*, 721-736.

M.R. GAREY, D.S. JOHNSON (1975). Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput. 4*, 397-411.

M.R. GAREY, D.S. JOHNSON (1978). Strong *NP*-completeness results: motivation, examples and implications. *J. Assoc. Comput. Mach. 25*, 499-508.

M.R. GAREY, D.S. JOHNSON (1979). *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

T. GONZALEZ, O.H. IBARRA, S. SAHNI (1977). Bounds for LPT schedules on uniform processors, *SIAM J. Comput. 6*, 155-166.

R.L. GRAHAM (1966). Bounds for certain multiprocessing anomalies. *Bell System Tech. J. 45*, 1563-1581.

R.L. GRAHAM (1969). Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math. 17*, 416-429.

R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math. 5*, 287-326.

M. GRÖTSCHEL, L. LOVÁSZ, A. SCHRIJVER (1987). *Geometric Algorithms and Combinatorial Optimization*, Springer, Berlin.

D.S. HOCHBAUM, D.B. SHMOYS (1987). Using dual approximation algorithms for scheduling problems: practical and theoretical results. *J. Assoc. Comput. Mach. 34*, 144-162.

D.S. HOCHBAUM, D.B. SHMOYS (1988). A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approach. *SIAM J. Comput.*, to appear.

E. HOROWITZ, S. SAHNI (1976). Exact and approximate algorithms for scheduling nonidentical processors. *J. Assoc. Comput. Mach. 23*, 317-327.

L.G. KHACHIAN (1979). A polynomial time algorithm in linear programming. *Soviet Math. Dokl. 20*, 191-194.

L. LOVÁSZ (1975). On the ratio of optimal and fractional covers. *Discrete Math. 13*, 383-390.

O.M.-C. MARCOTTE (1983). *Topics in Combinatorial*

Packing and Covering, Ph.D. thesis, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY.

C.N. POTTS (1985). Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Appl. Math. 10*, 155-164.

P. RAGHAVAN (1986). Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Proc. 27th Annual IEEE Symp. Foundations of Computer Science*, 10-18.

P. RAGHAVAN, C.D. THOMPSON (1985). Provably good routing in graphs: regular arrays. *Proc. 17th Annual ACM Symp. Theory of Computing*, 79-87.

S. SAHNI (1976). Algorithms for scheduling independent tasks. *J. Assoc. Comput. Mach. 23*, 116-127.

A. SCHRIJVER (1983). Min-max results in combinatorial optimization. A. Bachem, M. Grotschel, B. Korte (eds.) (1983). *Mathematical Programming: the State of the Art - Bonn 1982*, 439-500.

A. SCHRIJVER (1986). *Theory of Linear and Integer Programming*, Wiley, Chichester.