Short shop schedules

D. P. Williamson
IBM Watson

L. A. Hall
The Johns Hopkins University

J. A. Hoogeveen
C. A. J. Hurkens
Eindhoven University of Technology

J. K. Lenstra
Eindhoven University of Technology
CWI, Amsterdam

S. V. Sevast'janov
Institute of Mathematics, Novosibirsk

D. B. Shmoys
Cornell University

May 12, 1995

Abstract

We consider the open shop, job shop, and flow shop scheduling problems with integral processing times. We give polynomial-time algorithms to determine if an instance has a schedule of length at most 3, and show that deciding if there is a schedule of length at most 4 is $\mathcal{NP}$-complete. The latter result implies that, unless $\mathcal{P} = \mathcal{NP}$, there does not exist a polynomial-time approximation algorithm for any of these problems that constructs a schedule with length guaranteed to be strictly less than 5/4 times the optimal length. This work constitutes the first nontrivial theoretical evidence that shop scheduling problems are hard to solve even approximately.

Subject classification:

- Analysis of algorithms, computational complexity: NP-completeness results
- Production/scheduling, approximations: impossibility results
- Production/scheduling, multiple machine deterministic scheduling: non-preemptive shop scheduling

Shop scheduling problems form a class of scheduling models in which each job consists of several operations. In particular, we are given a set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$, a set of machines $\mathcal{M} = \{M_1, \ldots, M_m\}$, and a set of operations $\mathcal{O} = \{O_1, \ldots, O_t\}$; each operation $O_k \in \mathcal{O}$ belongs to a specific job $J_j \in \mathcal{J}$ and must be processed on a specific machine $M_i \in \mathcal{M}$ for a given amount of time p_k, which is a nonnegative integer. At any time, at most one operation can be processed on each machine, and at most one operation of each job can be processed. In this paper we consider nonpreemptive models: each operation must be processed to completion without interruption.

Shop models are further classified based on ordering restrictions for the operations of a job. In an *open shop*, the operations of each job may be processed in any order. In a *job shop*, the operations of each job must be processed in a given order specific to that job. A *flow shop* is a job shop in which each job has exactly one operation on each machine and the order in which each job is processed by the machines is the same for all jobs. In all three models, we define the *length* $C_{\max}$ of a schedule as the time at which all operations are completed, and $C^*_{\max}$ as the length of the shortest feasible schedule. Due to the integrality of the processing times, any schedule can easily be converted into one at least as good in which all completion times are integral; from now on, we shall restrict attention to such schedules.

Shop scheduling models have long been identified as having a number of important practical applications, dating back to the work of Johnson (1954). Job shop models, in particular, have become notorious for their computational difficulty, since even quite small instances have resisted solution by the full range of computational techniques that have been developed for combinatorial optimization problems over the past quarter century. Because of these characteristics, shop scheduling models have received a great deal of attention in the computational scheduling literature. Among the approaches that have been used to obtain good solutions are simple greedy construction methods, iterated local search procedures (Adams, Balas, & Zawack, 1988; Applegate & Cook, 1991), simulated annealing (Van Laarhoven, Aarts, & Lenstra, 1992), and taboo search (Dell'Amico & Trubian, 1993); branch-and-bound optimization algorithms have been based on 1-machine lower bounds (Bratley, Florian, & Robillard, 1973; Lageweg, Lenstra, & Rinnooy Kan, 1977; Carlier & Pinson, 1989, 1990), surrogate duality (Fisher, Lageweg, Lenstra, & Rinnooy Kan, 1983), and polyhedral methods (Balas, 1985; Applegate & Cook, 1991).

For each of the three shop scheduling models, the problem of finding a schedule of minimum length is strongly $\mathcal{NP}$-hard, even for severely restricted versions of these problems. For a summary of these results, the reader is referred to the survey of Lawler, Lenstra, Rinnooy Kan, and Shmoys (1993).

In contrast, nothing was known up to now about the computational complexity of deciding whether a given instance has a schedule of at most a given *constant* length. An example of such a problem is the question, 'Given an instance, does there exist a schedule of length no more than 3?'

We have resolved this issue: in each of the open shop, job shop, and flow shop scheduling models, deciding if there is a schedule of length at most 3 is in $\mathcal{P}$, whereas deciding if there is a schedule of length at most 4 is $\mathcal{NP}$-complete.

The $\mathcal{NP}$-completeness results imply that finding near-optimal shop schedules is $\mathcal{NP}$-hard. Suppose that, for some $\rho < 5/4$, we have a polynomial-time ρ-approximation algorithm for one of these problems, that is, an algorithm that runs in polynomial time and is guaranteed to produce a schedule of length at most $\rho C^*_{\max}$. If there exists a schedule of length at most 4, then our algorithm will return a schedule of length less than $5/4 \cdot 4 = 5$, that is, of length at most 4; otherwise, it will, of course, produce a schedule of length at least 5. Hence, a schedule of length at most 4 exists if and only if our approximation algorithm finds such a schedule. In other words, the supposed algorithm solves an $\mathcal{NP}$-complete problem in polynomial time. It follows that, for any $\rho < 5/4$, there is no polynomial-time ρ-approximation algorithm, unless $\mathcal{P} = \mathcal{NP}$.

Our results constitute the first nontrivial theoretical evidence that shop scheduling problems are hard to solve even approximately. The best result known previously is that no fully polynomial approximation scheme could exist for any such problem, unless $\mathcal{P} = \mathcal{NP}$. (A fully polynomial approximation scheme takes a problem instance and an $\epsilon > 0$ as input, and produces a schedule of length at most $(1+\epsilon)C^*_{\max}$ in time polynomial in the size of the instance and $1/\epsilon$.) This is a straightforward consequence of the strong $\mathcal{NP}$-hardness of these problems (Garey and Johnson, 1979, p. 141).

Some positive results are known about approximate shop scheduling. A result of Racsmány (see Bárány and Fiala, 1982; Shmoys, Stein, and Wein, 1994) implies that a simple list scheduling rule for open shops produces a schedule of length less than $2C^*_{\max}$. For job shops and flow shops, the best known polynomial-time approximation algorithm delivers a schedule of length $O(\log^2 m / \log\log m)C^*_{\max}$ (Shmoys, Stein, and Wein, 1994; Schmidt, Siegel, and Srinivasan, 1993). Any attempt to close the gaps by proving stronger negative results would require a new approach, since deciding if there is a schedule of length at most 3 is in $\mathcal{P}$.

1 Open shop scheduling

We shall present a polynomial-time algorithm to decide if a given instance of the open shop scheduling problem has a schedule of length at most 3. Furthermore, we shall prove that it is $\mathcal{NP}$-complete to decide if there is an open shop schedule of length at most 4.

It is easy to give a polynomial-time algorithm in case all operations have length 0 or 1. This special case can be reduced to the problem of coloring the edges of a bipartite graph G so that no two edges with a common endpoint have the same color. Let $G = (V_1, V_2, E)$ where $V_1 = \mathcal{M}$, $V_2 = \mathcal{J}$, and $E = \{e_k = (M_i, J_j) :$ operation O_k of J_j is on M_i and $p_k = 1\}$; note that if some job has more than one operation of length 1 on a machine, then G is a multigraph. Any edge-coloring of G with the colors $\{1, 2, \ldots, t\}$ can be viewed as a schedule in the following way: if e_k is colored c, then O_k is scheduled to start at time $c-1$ and to complete at time c; all operations of length 0 are scheduled at time 0. Conversely, any open shop schedule of length $C_{\max}$ can be similarly translated to yield a coloring that uses $C_{\max}$ colors. Since an optimal edge-coloring of a bipartite graph can be computed in polynomial time (see, e.g., Bondy and Murty, 1976), we see that this special case of the open shop scheduling problem is in $\mathcal{P}$.

If we wish to decide if an arbitrary instance of the problem has a schedule of length at most 3, we need only be concerned with operations of length at most 3. Moreover, since operations of length 0 or 3 are trivial to schedule, we can focus on instances with operations of length 1 or 2 only. We will show that this problem can be reduced to a constrained bipartite edge-coloring problem, which can be solved using an algorithm for the weighted bipartite matching problem.

Consider a job J_j for which one of its operations, O_k, is to be processed on M_i for two time units. The main idea behind the algorithm is that in any schedule of length 3, O_k is always processed throughout the time interval [1,2], plus either [0,1] or [2,3]. Hence, any unit-length operation of J_j cannot be processed in the interval [1,2]. Similarly, any unit-length operation to be processed on M_i cannot be processed in the interval [1,2]. Thus, a schedule is equivalent to an edge-coloring of the graph G as defined above, where any edge corresponding to an operation of length 2 is colored either 1 or 3 to reflect whether it is processed during [0,2] or [1,3], respectively, and any edge corresponding to a unit-length operation constrained to be processed in either [0,1] or [2,3] by an operation of length 2 is colored either 1 or 3 as well. The following is a more precise statement of the algorithm:

1. For each machine in $\mathcal{M}$, check if the total processing requirement of its operations is at most 3; if not, output 'no schedule' and halt.

2. For each job in $\mathcal{J}$, check if the total processing requirement of its operations is at most 3; if not, output 'no schedule' and halt.

3. Schedule all operations of length 0 or 3 to start at time 0.

4. Form the bipartite graph $G = (V_1, V_2, E)$ where $V_1 = \mathcal{M}$, $V_2 = \mathcal{J}$, and $E = \{e_k = (M_i, J_j) :$ operation O_k of J_j is on M_i and $p_k = 1$ or $2\}$. To each edge $e_k \in E$ assign a weight p_k.

5. Define a set $S \subseteq E$ of *special* edges containing each edge of weight 2 and each edge that has a common endpoint with an edge of weight 2.

6. Decide if G can be edge-colored with the colors $\{1, 2, 3\}$ such that each special edge is colored either 1 or 3. If no such coloring exists, then output 'no schedule'. Otherwise, for each edge colored c that corresponds to an operation of length 1, schedule that operation to be processed

from time $c-1$ to c. The remaining edges, which correspond to operations of length 2, are colored either 1 or 3; schedule the operations corresponding to color 1 to be processed from time 0 to 2, and the others from time 1 to 3.

Only step 6 is nontrivial to implement. The key idea is to first identify those edges assigned color 2. Let T be the set of nodes in G of degree 3. In any suitable edge-coloring of G, the edges assigned color 2 form a matching M in $G' = (V_1, V_2, E - S)$ that *covers* T; that is, each node in T is an endpoint of some edge in M. We show that, conversely, the existence of such a matching M yields a suitable coloring of G: Color each edge in M with color 2, and consider the graph of uncolored edges $G'' = (V_1, V_2, E - M)$. G'' is a bipartite graph of maximum degree 2, and hence it can be edge-colored with two colors, 1 and 3. This yields the desired coloring. It is easy to give a polynomial-time algorithm to decide if G' has a matching that covers T. For example, if each edge is assigned a weight equal to the number of its endpoints in T, then we can simply apply any algorithm that finds a maximum weight matching (see, e.g., Lovász and Plummer, 1986).

Theorem 1 *The problem of deciding if there is an open shop schedule of length at most* 3 *is in* $\mathcal{P}$.
■

We shall now prove the $\mathcal{NP}$-completeness of deciding if a schedule of length at most 4 exists by a reduction from the following $\mathcal{NP}$-complete problem:

> MONOTONE-NOT-ALL-EQUAL-3SAT
> *Instance:* Set U of variables, collection C of clauses over U such that each clause has size 3 and contains only unnegated variables.
> *Question:* Is there a truth assignment for U such that each clause in C has at least one true variable and at least one false variable?

MONOTONE-NOT-ALL-EQUAL-3SAT can be shown to be $\mathcal{NP}$-complete by a reduction from NOT-ALL-EQUAL-3SAT (Garey and Johnson, 1979, p. 259) in which all literals $\overline{x}_i$ are replaced by new variables y_i, and clauses of the form $y_i \vee x_i \vee x_i$ are added.

We give a polynomial-time reduction from this problem to open shop scheduling such that the optimal schedule length for an instance is 4 if and only if the MONOTONE-NOT-ALL-EQUAL-3SAT instance is satisfiable. Suppose we are given an instance of MONOTONE-NOT-ALL-EQUAL-3SAT with $U = \{x_1, \ldots, x_u\}$ and $C = \{c_1, \ldots, c_v\}$, in which each variable x_i appears t_i times. For notational convenience, we view the kth occurrence of x_i as the variable x_{ik}. Furthermore, let $\sigma(x_{ik})$ denote the next occurrence of x_i, cyclically ordered; that is, $\sigma(x_{ik}) = x_{il}$, where $l = k \bmod t_i + 1$. We transform this instance into the following instance of the open shop scheduling problem. For each variable x_{ik}, we construct two machines, $M_A(x_{ik})$ and $M_B(x_{ik})$. We construct three types of jobs:

1. For each variable x_{ik}, we construct an *assignment job* with operations $A(x_{ik})$ and $B(x_{ik})$, each of length 2, which are to be processed by $M_A(x_{ik})$ and $M_B(x_{ik})$, respectively.

2. For each variable x_{ik}, we construct a *consistency job* to ensure that its value is equal to the value of its next occurrence, $\sigma(x_{ik})$. It has two operations $\hat{B}(x_{ik})$ and $\hat{A}(x_{ik})$ of length 2 and 1, respectively, which must be processed by $M_B(x_{ik})$ and $M_A(\sigma(x_{ik}))$.

3. For each clause $c = (x \vee y \vee z)$, we construct a *clause job* with three unit-length operations, $T(x)$, $T(y)$, and $T(z)$, to be processed on $M_A(x)$, $M_A(y)$, and $M_A(z)$, respectively.

The optimal schedule must have length at least 4 in order to run the assignment jobs. In the following discussion, we will refer to the operation of an assignment job (consistency job, clause job) for a particular machine as the assignment operation (consistency operation, clause operation) for that machine.

The intuition behind the reduction is that each assignment job will denote the truth assignment of an occurrence of a variable. Consider the assignment job corresponding to x_{ik}. It has operations of length 2 on $M_A(x_{ik})$ and $M_B(x_{ik})$. In a schedule of length 4, one of these assignment operations must run on one machine from time 0 to 2 and the other operation must run on the other machine from time 2 to 4. Hence, we can consider each assignment job as a switch, which can be set in one of two positions depending on whether the job runs first on $M_A(x_{ik})$ or on $M_B(x_{ik})$. We will say that x_{ik} is true if the job runs first on $M_A(x_{ik})$, and false if it runs first on $M_B(x_{ik})$. The consistency job for x_{ik} prevents assignment jobs from being scheduled at the same time on machines $M_B(x_{ik})$ and $M_A(\sigma(x_{ik}))$, thus ensuring that the truth assignment of the occurrences x_{ik} and $\sigma(x_{ik})$ will be the same. Finally, given the assignment and consistency jobs, no clause job will be able to have all of its three operations scheduled on machines that process variables with the same value. This property will enforce the not-all-equal constraint. Figure 1 illustrates the reduction.

Theorem 2 *The problem of deciding if there is an open shop schedule of length at most* 4 *is* $\mathcal{NP}$*-complete.*

Proof: We show that the instance of MONOTONE-NOT-ALL-EQUAL-3SAT is satisfiable if and only if the open shop instance constructed has a schedule of length 4.

Suppose that there is a schedule of length 4. We first prove that in any such schedule, for $i = 1, \ldots, u$, either every machine $M_A(x_{ik})$ $(k = 1, \ldots, t_i)$ processes its assignment operation from time 0 to 2, or every machine $M_A(x_{ik})$ processes its assignment operation from time 2 to 4. If this is not the case, then there exist i and k such that $M_A(x_{ik})$ processes its assignment operation from time 0 to 2, and $M_A(\sigma(x_{ik}))$ processes its assignment operation from time 2 to 4. But $M_B(x_{ik})$ processes its assignment operation from time 2 to 4 as well. The consistency job for x_{ik} must be processed on both $M_B(x_{ik})$ and $M_A(\sigma(x_{ik}))$, and both of these machines are processing other operations from time 2 to 4. Hence, this schedule does not complete by time 4, which is a contradiction.

We now construct a satisfying assignment. For each variable x_i, set x_i to be true if the assignment operation for $M_A(x_{i1})$ runs from time 0 to 2, and false otherwise. By the argument above, a clause operation has been scheduled sometime between time 2 and 4 in case the variable corresponding to that operation has been set true, and sometime between time 0 and 2 in case the variable has been set false. Because each clause job has three unit-length operations which have been scheduled in nonoverlapping time periods, not all of its operations can correspond to true variables and not all of its operations can correspond to false variables. Hence, at least one variable of each clause must be true and at least one variable must be false.

Now suppose that the instance of MONOTONE-NOT-ALL-EQUAL-3SAT is satisfiable. We construct a schedule of length 4 in the following way. If x_i is true in the satisfying assignment, then we schedule the assignment operations for all machines $M_A(x_{ik})$ from time 0 to 2 and the ones for $M_B(x_{ik})$ from time 2 to 4; if x_i is false, then we do it the other way around. Therefore, for each occurrence x_{ik}, $M_A(x_{ik})$ is idle from time 2 to 4 if x_{ik} is true, whereas it is idle from time 0 to 2 if x_{ik} is false. For each clause, the clause operation corresponding to the first true variable of the clause can be scheduled from time 2 to 3, and the operation corresponding to the first false variable can be scheduled from time 0 to 1; the third clause operation can be scheduled from time 3 to 4 if the corresponding variable is true, and from time 1 to 2 if it is false. To schedule the consistency jobs, suppose, without loss of generality, that x_i is true. Pick any machine $M_B(x_{ik})$; by our construction thus far, $M_B(x_{ik})$ is idle from time 0 to 2. Schedule the consistency operation $\hat{B}(x_{ik})$ in this interval. In addition, $M_A(\sigma(x_{ik}))$ is idle either from time 2 to 3, or from time 3 to 4. Hence we can schedule the consistency operation $\hat{A}(x_{ik})$ without conflict in one of these intervals. All consistency jobs can be scheduled in this way. ■

Corollary 3 *For any $\rho < 5/4$, there does not exist a polynomial-time ρ-approximation algorithm for the open shop scheduling problem, unless $\mathcal{P} = \mathcal{NP}$.* ■

2 Job shop and flow shop scheduling

First we present a polynomial-time algorithm to decide if a given instance of the job shop scheduling problem has a schedule of length at most 3. Since the flow shop is a special case of the job shop, the algorithm can be applied to the flow shop scheduling problem as well. We then prove that deciding if there is a job shop schedule of length at most 4 is $\mathcal{NP}$-complete, and finally show how to extend this result to flow shop scheduling.

We shall show that deciding if there is a job shop schedule of length 3 can be reduced to the 2SAT problem, which is solvable in polynomial time (see, e.g., Garey and Johnson, 1979, p. 259).

Recall that the 2SAT problem is to decide whether a logical formula in which each clause contains at most two variables has a satisfying assignment. For ease of exposition, we use clauses of the form $(x_j \Rightarrow x_k)$, which is equivalent to $(\bar{x}_j \vee x_k)$.

The key to solving the problem is to schedule jobs with total processing time 2 or 3; once these are scheduled, the remaining jobs of total processing time 0 or 1 can easily be scheduled, as long as the total processing requirement of each machine is at most 3. For the longer jobs, each operation has at most two possible starting times. We construct a 2SAT formula F with variables of the form x_{jt}, where setting x_{jt} to be true will have the interpretation that operation O_j is scheduled to start at time t. An operation is said to be a *beginning* operation if it is not preceded in its job by a positive-length operation; an operation is said to be an *ending* operation if it is not a beginning operation and is not succeeded in its job by a positive-length operation. The reduction works as follows:

1. For each machine in $\mathcal{M}$, check if the total processing requirement of its operations is at most 3; if not, let F be the unsatisfiable formula $(x)(\bar{x})$, and halt.

2. For each job in $\mathcal{J}$, check if the total processing requirement of its operations is at most 3; if not, let F be the unsatisfiable formula $(x)(\bar{x})$, and halt.

3. For each beginning operation O_j of length 0, add the singleton clauses $x_{j0}, \bar{x}_{j1}, \bar{x}_{j2}, \bar{x}_{j3}$ to F.

4. For each ending operation O_j of length 0, add the singleton clauses $\bar{x}_{j0}, \bar{x}_{j1}, \bar{x}_{j2}, x_{j3}$ to F.

5. For each operation O_j in a job of total length 3, the starting time t is determined for any schedule of length 3; add to F the singleton clause x_{jt} and, for all $t' \neq t$, the singleton clauses $\bar{x}_{jt'}$.

6. Next consider all remaining operations in jobs of total length 2. We construct clauses to ensure that, for each job, its operations are scheduled in the correct order.

   (a) For each such operation O_j of length 2, add the clauses $(x_{j0} \vee x_{j1}), (x_{j0} \Rightarrow \bar{x}_{j1}), \bar{x}_{j2}, \bar{x}_{j3}$ to F.

   (b) For each such operation O_j of length 1, if O_j is a beginning operation, add the clauses $(x_{j0} \vee x_{j1}), (x_{j0} \Rightarrow \bar{x}_{j1}), \bar{x}_{j2}, \bar{x}_{j3}$, as well as $(x_{j1} \Rightarrow \bar{x}_{k1})$, where operation O_k is the other unit-length operation of its job.

   (c) For each such operation O_j of length 1, if O_j is an ending operation, add the clauses $(x_{j1} \vee x_{j2}), (x_{j1} \Rightarrow \bar{x}_{j2}), \bar{x}_{j0}, \bar{x}_{j3}$.

   (d) For each such operation O_j of length 0 (so that O_j is neither a beginning nor an ending operation), add the clauses $(x_{j1} \vee x_{j2}), (x_{j1} \Rightarrow \bar{x}_{j2}), \bar{x}_{j0}, \bar{x}_{j3}$, as well as $(x_{j1} \Rightarrow$

$x_{k0}), (x_{j2} \Rightarrow x_{l2})$, where operations O_k and O_l are the unit-length operations of this job that must precede and follow O_j, respectively; if the immediate successor $O_{j'}$ of O_j is of length 0, then add the clauses $(x_{j'1} \Rightarrow x_{j1})$ and $(x_{j2} \Rightarrow x_{j'2})$.

7. We finally add clauses to ensure that each machine processes at most one operation at a time.

   (a) Let O_j be any operation of length $p_j > 0$ in a job of length 2 or 3; suppose that it is to be processed on M_i. If it starts at time t, then $S_t = \{t' \in \mathbf{Z} : t \leq t' < \min\{t + p_j, 3\}\}$ is the set of disallowed starting times for other positive-length operations on M_i. Add the clause $(x_{jt} \Rightarrow \bar{x}_{kt'})$ for each $t' \in S_t$ and each other positive-length operation O_k on M_i.

   (b) Let O_j be any operation of length $p_j \geq 2$; suppose that it is to be processed on M_i. If it starts at time t, then $T_t = \{t' \in \mathbf{Z} : t < t' < \min\{t + p_j, 3\}\}$ is the set of disallowed starting times for operations of length 0 on M_i. Add the clause $(x_{jt} \Rightarrow \bar{x}_{kt'})$ for each $t' \in T_t$ and each operation O_k of length 0 on M_i.

Theorem 4 *The* 2SAT *formula F is satisfiable if and only if there is a job shop schedule of length 3.*

Proof: Suppose that there is a schedule of length 3. We can modify the schedule so that each beginning operation of length 0 is scheduled at time 0, and each ending operation of length 0 is scheduled at time 3. For each variable x_{jt} that occurs in F, set it to be true if operation O_j begins at time t in the modified schedule, and false otherwise. It is immediate that each clause in F is satisfied.

Suppose that F has a satisfying assignment. We first observe that this yields a feasible schedule of length 3 for all jobs of total length 0, 2 or 3. As suggested above, if x_{jt} is true in the satisfying assignment, then operation O_j is scheduled to start at time t. The clauses formed in steps 3 to 6 ensure that, for each operation O_j, at most one of x_{j0}, x_{j1}, x_{j2} and x_{j3} is true, and that, for each job, its operations are processed in the specified order. The clauses formed in step 7 ensure that no machine is assigned to process two operations simultaneously. We next extend this schedule to include all jobs of total length 1. For each such job, each operation of length 0 is either a beginning or an ending operation, and hence can be scheduled at either time 0 or time 3; for the unit-length operation, one unit of time must be available on its machine, since the operations on each machine have total length at most 3. ■

Corollary 5 *The problem of deciding if there is a job shop schedule of length at most 3 is in $\mathcal{P}$.* ■

To prove that deciding if there is a job shop schedule of length at most 4 is $\mathcal{NP}$-complete, we construct a reduction from a restricted version of 3SAT in which each clause contains at most three literals and each variable occurs (negated or not) at most three times in the logical formula. We call this problem 3-BOUNDED-3SAT; it is $\mathcal{NP}$-complete (Garey and Johnson, 1979, p. 259).

With each instance of 3-BOUNDED-3SAT we associate an instance of the job shop scheduling problem with the property that a schedule of length 4 exists if and only if the 3-BOUNDED-3SAT instance is satisfiable. Without loss of generality, we assume that each clause contains at least two variables and that each variable occurs at least once negated and at least once unnegated: if a clause contains only one literal, that literal must be true, and if a literal does not appear in the formula, then the complementary literal may be set true; this process yields a smaller formula that satisfies our assumptions. In constructing the scheduling instance, we need to distinguish between the first and second unnegated (or negated) occurrence of each *literal*; thus we refer to the kth occurrence of the literal x_i as x_{ik}, and of $\bar{x}_i$ as $\bar{x}_{ik}$, for $k = 1, 2$.

We specify the instance constructed by giving, for each job, its sequence of operations, and, for each machine, the set of operations that it must process; all operations are of unit length. With each variable x_i we associate four jobs $J(x) = (B(x), M(x), E(x))$, for $x \in \{x_{i1}, x_{i2}, \bar{x}_{i1}, \bar{x}_{i2}\}$; that is, $B(x)$, $M(x)$, and $E(x)$ are, respectively, the first, second, and third operations of $J(x)$, and are called its *beginning*, *middle*, and *ending* operations. There will be the following classes of machines.

1. For each variable x_i, there are two *assignment machines*: the first processes operations $B(x_{i1})$ and $B(\bar{x}_{i1})$, whereas the second processes $B(x_{i2})$ and $B(\bar{x}_{i2})$.

2. For each variable x_i, there are two *consistency machines*: the first processes operations $M(x_{i1})$ and $M(\bar{x}_{i2})$, whereas the second processes $M(x_{i2})$ and $M(\bar{x}_{i1})$.

3. For each clause c, there is a *clause machine*; its construction depends on whether c has 2 or 3 literals. If $c = (x \vee y)$, we introduce a clause machine that processes $E(x)$ and $E(y)$. If $c = (x \vee y \vee z)$, we also introduce *dummy jobs* and *dummy machines*. There are three dummy jobs $\hat{J}(w) = (\hat{B}(w), \hat{E}(w))$, for $w \in \{x, y, z\}$. The clause machine for c processes $\hat{B}(x)$, $\hat{B}(y)$, and $\hat{B}(z)$. For each $w \in \{x, y, z\}$, there is a dummy machine that processes $E(w)$ and $\hat{E}(w)$.

4. For each literal $x \in \{x_{i2}, \bar{x}_{i2}\}$ *not* occurring in the formula, we construct a *garbage machine* that processes only the operation $E(x)$.

The intuition behind the reduction is that schedules of length 4 are constrained in the following way. Each beginning operation must start at either time 0 or 1. The assignment machines ensure that, for each x_{ik}, one of the operations $B(x_{ik})$ and $B(\bar{x}_{ik})$ is scheduled at time 0 and the other at time 1; this provides the means to set x_{ik} to be either true or false. The consistency machines ensure that the two copies of each literal are set identically. Each ending operation must start at either time 2 or 3. Only those corresponding to true literals can be scheduled to start at time 2, and the clause machines ensure that each clause has at least one literal whose ending operation starts at time 2. Figure 2 summarizes the elements of the reduction.

Theorem 6 *The problem of deciding if there is a job shop schedule of length at most* 4 *is* $\mathcal{NP}$*-complete.*

Proof: We show that the instance of 3-BOUNDED-3SAT is satisfiable if and only if the job shop instance constructed has a schedule of length 4.

Suppose that the instance of 3-BOUNDED-3SAT is satisfiable. We construct a schedule of length 4 in the following way. For each true literal, its corresponding beginning and middle operations are scheduled to start at times 0 and 1, respectively, whereas the beginning and middle operations corresponding to false literals are started at times 1 and 2, respectively. For each clause c, we select one of its true literals; such a literal exists since the entire formula is true. We schedule the ending operations corresponding to these literals to start at time 2, whereas all other ending operations are started at time 3. The dummy operations can then be scheduled appropriately.

Now suppose that there is a job shop schedule of length 4. We will show that the schedule must be essentially of the form just constructed, and hence we will be able to extract a satisfying assignment for the original formula. Each beginning operation must start at either time 0 or 1, each middle operation must start at either time 1 or 2, and each ending operation must start at either time 2 or 3. Furthermore, for any literal x, if $E(x)$ starts at time 2, then $B(x)$ starts at time 0. For each x_{ik}, $B(x_{ik})$ and $B(\bar{x}_{ik})$ are processed on the same machine, and hence one of these must start at time 0 and the other at time 1.

We show next that $B(x_{i1})$ and $B(x_{i2})$ are processed simultaneously. Assume, without loss of generality, that $B(x_{i1})$ starts at time 1. This implies that $M(x_{i1})$ starts at time 2, so that, on the same consistency machine, $M(\bar{x}_{i2})$ starts at time 1. This in turn implies that $B(\bar{x}_{i2})$ starts at time 0, and hence $B(x_{i2})$ starts at time 1, at the same time as $B(x_{i1})$. Notice that because $B(x_{i1})$ and $B(x_{i2})$ are scheduled at the same time, $B(\bar{x}_{i1})$ and $B(\bar{x}_{i2})$ are scheduled at the same time too.

We set the variable x_i true if and only if $B(x_{i1})$ starts at time 0; we wish to show that this is a satisfying assignment. Consider a clause c. If $c = (x \vee y)$, then there is a clause machine that processes $E(x)$ and $E(y)$. One of these must start at time 2, which implies that the corresponding beginning operation starts at time 0, and the associated literal has been set true. If $c = (x \vee y \vee z)$, then there is a clause machine that processes the dummy operations $\hat{B}(x)$, $\hat{B}(y)$, and $\hat{B}(z)$. Since these operations are succeeded by $\hat{E}(x)$, $\hat{E}(y)$, and $\hat{E}(z)$, respectively, none of them can start at time 3. Hence, for some $w \in \{x, y, z\}$, $\hat{B}(w)$ starts at time 2, so that $\hat{E}(w)$ starts at time 3. But then, on the same dummy machine, $E(w)$ starts at time 2. This implies that we have set the literal w to be true. ■

Corollary 7 *For any $\rho < 5/4$, there does not exist a polynomial-time ρ-approximation algorithm for the job shop scheduling problem, unless $\mathcal{P} = \mathcal{NP}$.* ■

The construction of Theorem 6 is easily strengthened to yield the same result for flow shop scheduling. As is suggested in Figure 2, the machines can be ordered so that the operations of each job are consistent with that order: first take all assignment machines, followed by all consistency machines, then all clause machines for clauses of size 2, the remaining clause machines, the dummy machines, and finally the garbage machines. In a flow shop, each job must have one operation on each machine. The instance just specified can easily be filled out with additional zero-length operations; since all of the original operations are of unit length, these new operations can be scheduled trivially, without affecting the overall schedule.

Theorem 8 *The problem of deciding if there is a flow shop schedule of length at most* 4 *is $\mathcal{NP}$-complete.* ■

Corollary 9 *For any $\rho < 5/4$, there does not exist a polynomial-time ρ-approximation algorithm for the flow shop scheduling problem, unless $\mathcal{P} = \mathcal{NP}$.* ■

Acknowledgements

We would like to thank Lászlo Lovász for a helpful discussion. The research of the first author was supported by an NSF Graduate Fellowship, DARPA Contract N00014-89-J-1988, and an NSF Postdoctoral Fellowship. The research by the third author was supported by a grant from the Netherlands Organization for Scientific Research (NWO). The research of the second, fifth, and last authors was partially supported by NSF PYI grant CCR-8896272 with matching support from UPS, Sun, Proctor & Gamble, and DuPont, and by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550. The second author was additionally supported by NSF grant DDM-9210979 and the last author by NSF grant CCR-9307391.

References

J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job-shop scheduling. *Management Sci.*, 34:391–401, 1988.

D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA J. Comput.*, 3:149–156, 1991.

E. Balas. On the facial structure of scheduling polyhedra. *Math. Programming Stud.*, 24:179–218, 1985.

I. Bárány and T. Fiala. Többgépes ütemezési problémák közel optimális megoldása. *Szigma–Mat.–Közgazdasági Folyóirat*, 15:177–191, 1982.

J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* American Elsevier, New York, and MacMillan, London, 1976.

P. Bratley, M. Florian, and P. Robillard. On sequencing with earliest starts and due dates with application to computing bounds for the $(n/m/G/F_{\max})$ problem. *Naval Res. Logist. Quart.*, 20:57–67, 1973.

J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Sci.*, 35:164–176, 1989.

J. Carlier and E. Pinson. A practical use of Jackson's preemptive schedule for solving the job-shop problem. *Ann. Oper. Res.*, 26:269–287, 1990.

M. Dell'Amico and M. Trubian. Applying tabu search to the job-shop scheduling problem. *Ann. Oper. Res.*, 41:231–252, 1993.

M. L. Fisher, B. J. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan. Surrogate duality relaxation for job shop scheduling. *Discrete Appl. Math.*, 5:65–75, 1983.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., New York, 1979.

S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Quart.*, 1:61–68, 1954.

B. J. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan. Job-shop scheduling by implicit enumeration. *Management Sci.*, 24:441–450, 1977.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, editors, *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory*, pages 445–522. North-Holland, 1993.

L. Lovász and M. D. Plummer. *Matching Theory.* Akadémiai Kiadó, Budapest, and North-Holland, Amsterdam, 1986.

J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. In *Proceedings of the 4th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 331–340, 1993.

D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM J. Comput.*, 23:617–632, 1994.

P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Oper. Res.*, 40:113–125, 1992.

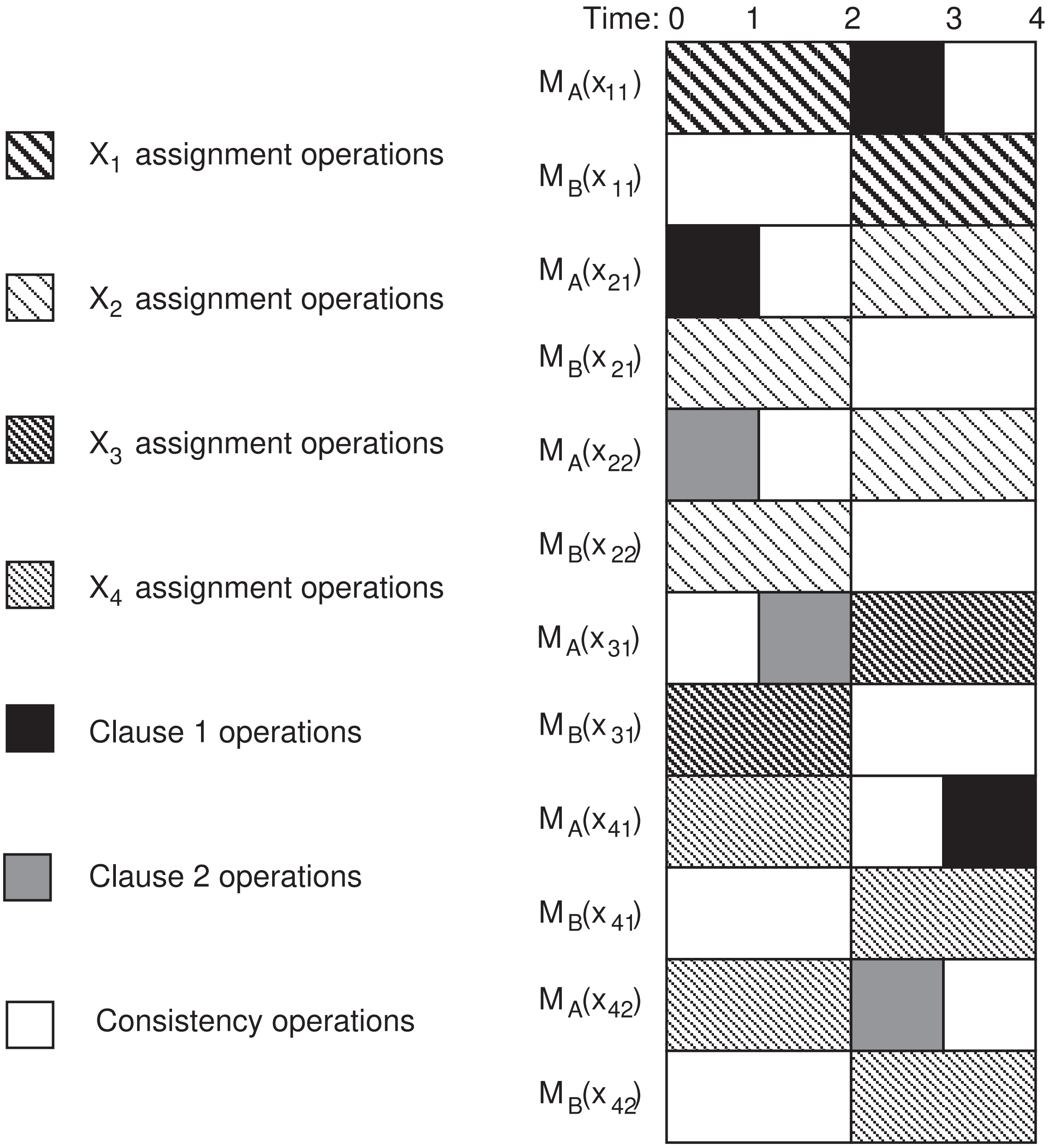

Figure 1: Open shop schedule corresponding to $(x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_3 \vee x_4)$

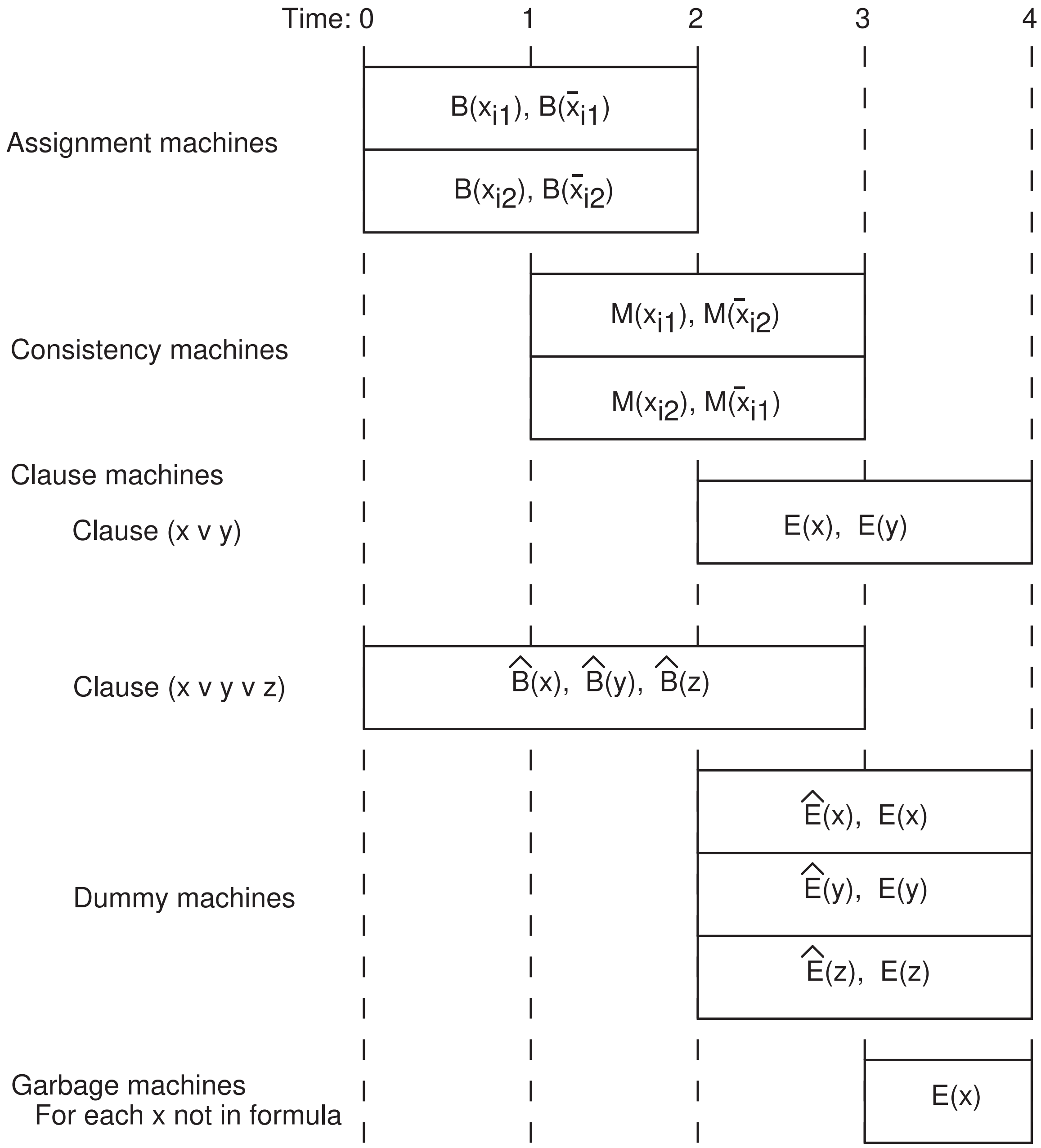

Figure 2: Reduction from 3-BOUNDED-3SAT to job shop schedule