Sequence-Dependent Setup Times in a Two-Machine Job-Shop with Minimizing the Schedule Length

Yuri N. Sotskov
United Institute Of Informatics Problems
135 PUBLICATIONS 1,482 CITATIONS
SEE PROFILE

Tsung-Chyan Lai
Harbin Engineering University
34 PUBLICATIONS 396 CITATIONS
SEE PROFILE

Frank Werner
Otto-von-Guericke-Universität Magdeburg
346 PUBLICATIONS 2,588 CITATIONS
SEE PROFILE

Sequence-Dependent Setup and Removal Times in a Two-Machine Job-Shop with Minimizing the Schedule Length

Yuri N. Sotskov[1], Natalja G. Egorova[1], Tsung-Chyan Lai[2], and Frank Werner[3]

[1]United Institute of Informatics Problems,
National Academy of Sciences of Belarus,
Surganova Str. 6, Minsk, Belarus,

[2]College of Management,
National Taiwan University,
50, Lane 144, Sec. 4, Keelung Rd., Taipei, Taiwan

[3]Faculty of Mathematics,
Otto-von-Guericke University,
Postfach 4120, D-39016, Magdeburg, Germany

Abstract: This article addresses the job-shop problem of minimizing the schedule length (makespan) for processing *n* jobs on two machines with sequence-dependent setup and removal times. The processing of each job includes at most two operations that have to be non-preemptive. Machine routes may differ from job to job. If all setup and removal times are equal to zero, this problem is polynomially solvable via Jackson's pair of job permutations, otherwise it is NP-hard even if each of *n* jobs consists of one operation on the same machine. We present sufficient conditions when Jackson's pair of permutations may be used for solving the two-machine job-shop problem with sequence-dependent setup and removal times. For the general case of this problem, the results obtained provide polynomial lower and upper bounds for the objective function value which are used in a branch-and-bound algorithm. Computational experiments show that an exact solution for this problem may be obtained in a suitable time for $n \leq 280$. We also develop a heuristic algorithm and present a worst case analysis for it.

Keyword: Scheduling theory, setup, job-shop.

1. INTRODUCTION

The majority of scheduling research assumes the *setup time* as negligible or as a part of the job processing time. This assumption adversely affects the solution quality for different applications which require an explicit treatment of setup times. Practical situations in which machine setup times must be considered separately from the job processing times arise in chemical, pharmaceutical, food, printing, metal processing and semiconductor industries. During the last decade, these applications have motivated an increasing interest to include separate setups in the scheduling environment. Allahverdi, Gupta, and Aldowaisan (1999) surveyed about 190 papers on scheduling with separate setup times published over a period of 25 years until 1999, while Allahverdi et al. (2006) surveyed more than 300 such papers published in the period 1999 – 2005. Most papers surveyed deal with single machine scheduling, and a lot of papers address the flow-shop problem with setups. In particular, Khurana and Bagga (1984) and Yoshida and Hitomi (1979) addressed the two-machine flow-shop problem of minimizing C_{max}, the schedule length (makespan), by considering setup times separately. In (Allahverdi, 2000; Bagga and Khurana, 1986), the two-machine separate setup time problem of minimizing mean job completion time $\sum C_i$ has been addressed. Allahverdi, Aldowaisan, and Sotskov (2003) addressed the two-machine flow-shop problem to minimize C_{max} or $\sum C_i$ when setup times are relaxed to be distribution-free random variables with only lower and upper bounds being given before scheduling.

As follows from the survey papers (Allahverdi, Gupta, and Aldowaisan, 1999; Allahverdi et al, 2006; Yang and Liao, 1999) on scheduling problems with separate setups and from other surveys (Cheng, Gupta, and Wang, 2000; Potts and Kovalyov, 2000), shop-scheduling problems involving *sequence-independent* setup times have been mainly treated in the OR literature so far, and there is only little research on a job-shop involving *sequence-dependent* setup times. While the assumption that setup times are sequence-independent simplifies the analysis of a shop-scheduling problem and reflects certain applications, it negatively affects the solution quality for many other applications such as those arising in semiconductor manufacturing, see (Zant, 1997), which require a treatment of sequence-dependent setup times.

To our knowledge, all papers that addressed a job-shop involving *sequence-dependent* setup times are included in the references listed at the end of this paper. In particular, using a simulation study Wilbrecht and Prescott (1969) have shown that sequence-dependent setup times play a critical role in the performance of a job-shop operating near the full capacity. Kim and Bobrowski (1994) used a simulation to study the effect of sequence-dependent setup times and discovered that setup times must indeed be explicitly considered while solving scheduling problems when they are sequence-dependent. Using also simulation, Low (1995) compared the performance of a heuristic algorithm for sequence-dependent setup times under various criteria against non-sequence-dependent setup times. O'Grady and Harrison (1988) proposed a search sequencing rule which prioritizes jobs using a linear combination of the due dates, processing and sequence-dependent setup times. Choi and Korkmaz (1997) provided a mixed integer programming to minimize the makespan in a flexible manufacturing system, and they developed a polynomial heuristic that yields a better performance than that proposed by Artigues, Belmokhtar, and Feillet (2004) and Zhou and Egbelu (1989). Gupta (1982) and Brucker and Thiele (1996) provided branch-and-bound algorithms for a job-shop with sequence-dependent setup times. Ovacik and Uzsoy (1994) developed an algorithm with myopic dispatching rules in a job-shop environment, a semiconductor testing facility and a reentrant flow-shop. In (Zoghby, Barnes, and Hasenbein, 2005), feasibility conditions were investigated in the context of metaheuristic searches (such as tabu search or simulated annealing), and an algorithm was proposed for obtaining an initial feasible solution using the disjunctive graph model for a job-shop. Cheung and Zhou (2001) developed a genetic algorithm for a job-shop problem with sequence-dependent setup times. Choi and Choi (2002) and Ballicu, Giua, and Seatzu (2002) derived mixed integer programming models for the same problem. A tabu search heuristic was proposed by Artigues and Buscaylet (2003). Artigues, Lopez, and Ayache (2005) obtained upper bounds by a heuristic algorithm. Sun and Yee (2003) addressed the job-shop with the additional characteristic of reentrant work flows. They utilized the disjunctive graph model and proposed heuristics including a genetic algorithm. In (Artigues and Roubellat, 2002; Sotskov, Tautenhahn, and Werner, 1999), polynomial insertion techniques were used for a job-shop problem with separate setup times. Tahar et al. (2005) proposed an ant colony algorithm and showed by computational analysis that it performs better than a genetic algorithm.

In this paper, we consider the two-machine job-shop problem of minimizing the length of a schedule including sequence-dependent setup times and removal times. We prove sufficient conditions when Jackson's pair of job permutations (Jackson, 1956) may be used for solving this problem polynomially. The results obtained provide a polynomial heuristic algorithm and a lower bound for the schedule length which are used in a branch-and-bound algorithm. Computational experiments show that an exact solution for the randomly generated job-shop problem with sequence-dependent setup times and removal times may be obtained in a suitable CPU-time for $n \leq 280$. We also develop a worst case analysis for the heuristic algorithm. The paper is organized as follows. Notations are given in section 2. Modifications of the setup, removal, and processing times are described in section 3. In section 4, it is shown when these modifications allow us to obtain an optimal solution to the problem with sequence-dependent setup times in polynomial time. A worst case analysis of the heuristic algorithm based on these modifications is developed in section 5. A branch-and-bound algorithm and computational results are described in section 6. Finally, the paper concludes with some generalizations of the obtained results in section 7.

2. PROBLEM SETTING AND NOTATIONS

Assume that a set of jobs $J = \{1, 2, ..., n\}$ has to be processed in a job-shop with two machines $M = \{1, 2\}$ provided that each machine $m \in M$ processes any job $j \in J$ at most once. The subset J_{12} of set J is the set of jobs with the machine route (1, 2), the subset $J_{21} \in J$ is the set of jobs with an opposite machine route (2, 1), and the subset $J_m \in J$ is the set of jobs which have to be processed only on one machine $m \in M$. Thus, $J = J_{12} \cup J_1 \cup J_2 \cup J_{21}$. Let the cardinality of set J_k be denoted as $n_k = |J_k|$, where $k \in \{1, 2, 12, 21\}$. O_{jm} denotes the operation of job $j \in J$ on machine $m \in M$. The processing time p_{jm} of operation O_{jm} is known before scheduling. All n jobs are available for processing from time $t = 0$. Operation preemptions are forbidden.

In practice, machines often have to be reconfigured before starting a job and cleaned after completing the last job. These processes are called *setup* and *removal*, respectively. We assume that the given setup time of a machine depends on the job just completed and the job to be started, i.e., the setup times are sequence-dependent. If job $i \in J$ is directly followed by job $k \in J$ on machine $m \in M$, then the setup time is equal to a non-negative real number s_{ik}^{m}. The notation s_{0k}^{m} is used for the non-negative setup time needed on machine $m \in M$ before starting job k, if k is the first job processed on machine m. Similarly, s_{k0}^{m} denotes the non-negative removal time after job i provided that i is the last job processed on machine $m \in M$. The setup and removal times for machine 1 are given by a real non-negative square matrix $S^1 = \| s_{ij}^{1} \|$ of order $r_1 \times r_1$, where $r_1 = n - n_2 + 1$. Hereafter, in contrast to usual matrix notations when the subindex i (subindex j) of the element s_{ij}^1 of matrix S^1 denotes the row index (column index, respectively), we define that the first subindex i in s_{ij}^1 denotes the job $i \in J \setminus J_2$ and the second subindex j in s_{ij}^1 denotes the job $j \in J \setminus J_2$. As usual, it is assumed that the columns (rows) in matrix S^1 are ordered with respect to an increasing second subindex (first subindex) of their elements s_{ij}^1. In particular, each element s_{0i}^1 of the first row in matrix S^1 defines the setup time for job $i \in J \setminus J_2$ on machine 1, if i is the first job processed on machine 1. Each element s_{j0}^1 of the first column in matrix S^1 defines the removal time for job $j \in J \setminus J_2$, if j is the last job processed on machine 1. The diagonal elements in matrix S^1 are not used. Similarly, the setup and removal times for machine 2 are given by a real non-negative square matrix $S^2 = \| s_{ij}^2 \|$ of order $r_2 \times r_2$, where $r_2 = n - n_1 + 1$.

Since the minimization of the schedule length is a *regular* criterion, we can consider only the set of *semiactive schedules*, each of which is uniquely defined by a permutation of the jobs on machine 1 and by one on machine 2. So, the problem is to find a permutation $\pi' = (i'_1, i'_2, \ldots, i'_{r_1})$ of the jobs $i'_k \in J_{12} \cup J_1 \cup J_{21}$ on machine 1 and a permutation $\pi'' = (i''_1, i''_2, \ldots, i''_{r_2})$ of the jobs $i''_k \in J_{12} \cup J_2 \cup J_{21}$ on machine 2 which minimize the objective function

$$C_{\max}(\pi', \pi'') = \max\{ C_{i'_{r_1}}(\pi', \pi'') + s^1_{i'_{r_1}0},\ C_{i''_{r_2}}(\pi', \pi'') + s^2_{i''_{r_2}0} \}, \tag{1}$$

where $C_i(\pi', \pi'')$ denotes the completion time of job $i \in J$ in the semiactive schedule $s(\pi', \pi'')$ defined by the pair of permutations (π', π''). Objective function (1) is equal to the *schedule length* including the removal time of a machine after processing the last jobs. This problem is denoted as $J2|s_{jk}|C_{max}$.

3. MODIFICATION OF SETUP, REMOVAL AND PROCESSING TIMES

The value of objective function (1) depends on two essentially different parts of the numerical input data. The first part includes the processing times p_{ij} of the jobs $i \in J$ on the machines $j \in M$, while the second part includes the setup and removal times given by the matrices S^1 and S^2. Generally speaking, the former part is easier to treat optimally than the latter part. Indeed, if all setup times and removal times are equal to zero, then problem $J2|s_{jk}|C_{max}$ turns into the classical job-shop problem $J2||C_{max}$ which is polynomially solvable by Jackson's pair of job permutations (Jackson, 1956). Otherwise, problem $J2|s_{jk}|C_{max}$ is NP-hard even if each of the n jobs consists of only one operation on the same machine (e.g., if $n = n_1$) since of the latter problem turns into the NP-hard traveling salesman problem.

If there exist non-zero setup or removal times, then the schedule length $C_{\max}(\pi', \pi'')$ depends on the choice of $r_1 + r_2$ setup and removal times (from the set of $r_1^2 + r_2^2$ possible setup and removal times given by the matrices S^1 and S^2) which have to be involved into the schedule. In this section, we show how it is possible to transfer at least a part of the "hard" numerical input data to the "easy" numerical input data.

Let job i belong to set $J_1 \cup J_{12}$. We calculate the non-negative value

$$s^1(\rightarrow i) = \min\{ s^1_{ki} \mid k \in \{0\} \cup J \setminus J_2, k \neq i\}. \tag{2}$$

Since each setup time before processing operation O_{i1} includes a part equal to $s^1(\rightarrow i)$, we can add the value $s^1(\rightarrow i)$ to the processing time p_{i1} of operation O_{i1} provided that the same value $s^1(\rightarrow i)$ will be subtracted from each setup time s^1_{ki} with $i \neq k \in \{0\} \cup J \setminus J_2$. Thus for each job $i \in J_1 \cup J_{12}$, we obtain the following *modified* processing time:

$$p'_{i1} = s^1(\rightarrow i) + p_{i1} \tag{3}$$

and the *modified* setup and removal times:

$$s^{(1)}_{ki} = s^1_{ki} - s^1(\rightarrow i), \tag{4}$$

where $k \in \{0\} \cup J \setminus J_2$, $k \neq i$. Due to (2) and (4), we obtain inequality $s^{(1)}_{ki} \geq 0$ for each job $i \in J_1 \cup J_{12}$ and each job $k \in \{0\} \cup J \setminus J_2$ with $k \neq i$. Next, we prove that the *original* instance of problem $J2|s_{jk}|C_{max}$ and the *modified* instance that differs from the *original* instance only by the setup and processing times of the jobs $i \in J_1 \cup J_{12}$ modified due to equalities (3) and (4) are *equivalent* in the following sense.

Definition 3.1 *Two instances of a scheduling problem are equivalent if there exists a one-to-one correspondence between their semiactive schedules such that the corresponding two schedules have the same schedule length.*

Indeed, the desired correspondence of the semiactive schedules is defined by the same pair (π', π'') of permutation $\pi' = (i'_1, i'_2, \dots, i'_{r_1})$ of the jobs $i'_k \in J_{12} \cup J_1 \cup J_{21}$ on machine 1 and permutation $\pi'' = (i''_1, i''_2, \dots, i''_{r_2})$ of the jobs $i''_k \in J_{12} \cup J_2 \cup J_{21}$ on machine 2. It is easy to convince that for both instances of problem $J2|s_{jk}|C_{max}$, machines 1 and 2 are occupied (either by processing jobs or by setups or by removals) during the same time intervals since in each semiactive schedule constructed for the *modified* instance each non-negative value $s^1(\to i)$ is added exactly once to the processing time p_{i1} and subtracted exactly once from the setup time (or removal time) which is involved in the schedule. Moreover, the processing time p_{i1} of each job $i \in J_{12}$ may be increased only “from the left-hand side” by the value $s^1(\to i)$ of the setup time. Hence, the processing of job $i \in J_{12}$ on machine 2 may be started just from the same time as in the corresponding semiactive schedule constructed for the *original* instance of problem $J2|s_{jk}|C_{max}$.

Due to machine symmetry, one can also obtain an equivalent *modified* instance of problem $J2|s_{jk}|C_{max}$ via modifying the setup and processing times of the jobs $i \in J_2 \cup J_{21}$ on machine 2:

$$p'_{i2} = s^2(\to i) + p_{i2}, \tag{5}$$

$$s^{(2)}_{ki} = s^2_{ki} - s^2(\to i), \tag{6}$$

where $k \in \{0\} \cup J \setminus J_1$, $k \neq i$, and the above value $s^2(\to i)$ is defined as follows:

$$s^2(\to i) = \min\{ s^2_{ki} \mid k \in \{0\} \cup J \setminus J_1, k \neq i\}. \tag{7}$$

Similarly, one can increase the processing times of the jobs of the set J_{21} on machine 1 “from the right-hand side” due to the decrease of the corresponding setup and removal times as follows. Let job j belong to set $J_1 \cup J_{21}$. We calculate the non-negative value

$$s^1(j \to) = \min\{ s^1_{jk} \mid k \in \{0\} \cup J \setminus J_2, k \neq j\}. \tag{8}$$

Since the removal time and each possible setup time before operation O_{j1} includes a part equal to $s^1(j \to)$, we can add the value $s^1(j \to)$ to the processing time p_{j1} of operation O_{j1} provided that the same value $s^1(j \to)$ will be subtracted from the removal time s^1_{j0} and from each setup time s^1_{jk} with $j \neq k \in J \setminus J_2$. Thus for each job $j \in J_1 \setminus J_{21}$, we obtain the *modified* processing time

$$p'_{j1} = p_{j1} + s^1(j \to), \tag{9}$$

the *modified* setup times

$$s^{(1)}_{jk} = s^1_{jk} - s^1(j \to),\ k \in J \setminus J_2,\ k \neq j, \tag{10}$$

and the *modified* removal time

$$s^{(1)}_{j0} = s^1_{j0} - s^1(j \to). \tag{11}$$

Note that the processing time p_{j1} of job $j \in J_{21}$ may be increased only “from the right-hand side” by the value $s^1(j \to)$ defined by equality (8). Due to this and equalities (2) – (3), the non-negative common part of each setup time may be added to the modified processing time exactly once.

Note that the processing times of jobs $i = j \in J_1$ may be modified both “from the left-hand side” due to equalities (2) – (3) used for job $i \in J_1$ and “from the right-hand side” due to equalities (8) – (9) used for job $j \in J_1$.

Due to machine symmetry, one can modify the setup, removal, and processing times of the jobs $i \in J_2 \cup J_{12}$ on machine 2 using the following formulas (12) – (14):

$$p'_{i2} = p_{i2} + s^2(i \to), \tag{12}$$

$$s^{(2)}_{ik} = s^2_{ik} - s^2(i \to),\ k \in \{0\} \cup J \setminus J_2,\ k \neq i, \tag{13}$$

$$s^{(2)}_{j0} = s^2_{j0} - s^2(j \to), \tag{14}$$

where the value $s^2(i \to)$ is defined as follows:

$$s^2(i \to) = \min\{ s^2_{ik} \mid k \in \{0\} \cup J \setminus J_2, k \neq i\}. \tag{15}$$

In order to transfer further “hard” numerical input data to the “easy” numerical input data, we can introduce a *dummy* job 0 (a *dummy* job $n + 1$, respectively) before starting the first (after completing the last) job on each of the two machines. The processing times p_{0m} and the modified setup times $s^{(m)}_{0j}$ are defined as follows:

$$p_{0m} = s^m(0), \tag{16}$$

$$s^{(m)}_{0j} = s^m_{0j} - s^m(0),\ j \in J \cup J_{3-m} \tag{17}$$

provided that

$s^m(0) = \min\{ s^m_{0j} \mid j \in J \cup J_{3-m} \}$. (18)

The processing times $p_{n+1,m}$ and the modified removal times $s^{(m)}_{j0}$ are defined as follows:

$p_{n+1,m} = s^m(n+1)$, (19)

$s^{(m)}_{j0} = s^m_{j0} - s^m(n+1),\ j \in J \setminus J_{3-m}$, (20)

provided that

$s^m(n+1) = \min\{ s^m_{j0} \mid j \in J \setminus J_{3-m} \}$. (21)

Using Definition 2.1, we can summarize the above arguments in the following claim.

Theorem 3.1 *An instance of problem $J2|s_{jk}|C_{max}$ is equivalent to the modified instance that differs from the original one by the setup, removal, and processing times of the jobs from $J \cup \{0, n+1\}$ modified due to formulas (2) – (21).*

4. SUFFICIENT CONDITIONS FOR OPTIMALITY OF JACKSON'S PERMUTATIONS

In order to obtain the simplest *modified* instance using formulas (2) – (21), which is equivalent to the *original* instance of problem $J2|s_{jk}|C_{max}$, it is necessary to decrease the elements of the matrices S^1 and S^2 as much as possible using formulas (2) – (21). Therefore, the simplest equivalent *modified* instance will be obtained due to Theorem 3.1 when no further modification of these matrices based on formulas (2) – (21) will be possible. Let matrix $S^{(1)} = \| s^{(1)}_{ij} \|$ and matrix $S^{(2)} = \| s^{(2)}_{ij} \|$ denote such *minimal* matrices (their elements have minimal values) obtained from S^1 and S^2, respectively, due to formulas (2) – (21). Note that the matrices $S^{(1)}$ and $S^{(2)}$ are uniquely defined, while there may exist several *modified* instances of the *original* instance of problem $J2|s_{jk}|C_{max}$ because of different orders that may be used for the modification of the rows and columns of the matrices S^1 and S^2. We use the following definition of an instance correspondence.

Definition 4.2 *An instance of problem $J2||C_{max}$ corresponds to that of problem $J2|s_{jk}|C_{max}$ (and vice versa), if their input data are the same except non-zero setup and removal times given for the instance of problem $J2|s_{jk}|C_{max}$. Such an instance of problem $J2||C_{max}$ is called a relaxed one for the corresponding instance of problem $J2|s_{jk}|C_{max}$.*

We consider the following three semiactive schedules defined by the pair (π', π'') of job permutations:

$s(\pi', \pi'')$ denotes the semiactive schedule defined by the pair (π', π'') for the *original* instance of problem $J2|s_{jk}|C_{max}$;

$s'(\pi', \pi'')$ denotes that for the *modified* instance of problem $J2|s_{jk}|C_{max}$ with the processing times p'_{ij}, $i \in J$, $j \in M$, and the minimal matrices $S^{(1)}$ and $S^{(2)}$ of setup times;

$s^o(\pi', \pi'')$ denotes that for the *relaxed* instance of problem $J2||C_{max}$ corresponding to the *modified* instance.

Machine $m \in M$ is called the *main machine* for schedule $s(\pi', \pi'')$, if the following equality holds: $C_{max}(\pi', \pi'') = C_j(\pi', \pi'') + s^m_{j0}$, where $j = i'_{r_1}$ if $m = 1$, and $j = i''_{r_2}$ if $m = 2$. Let $c^m_j(\pi', \pi'')$ denote the completion time of operation O_{jm} in the schedule $s(\pi', \pi'')$.

Corollary 4.1 *If the main machine for schedule $s'(\pi', \pi'')$ has no idle times and has only zero modified setup and removal times, the schedule $s(\pi', \pi'')$ is optimal for the original instance of problem $J2|s_{jk}|C_{max}$.*

Proof. It is clear that the length of schedule $s'(\pi', \pi'')$ constructed for the *modified* instance of problem $J2|s_{jk}|C_{max}$ is no less than that of schedule $s^o(\pi', \pi'')$ constructed for the corresponding *relaxed* instance. As shown by Jackson (1956), the schedule $s^o(\pi', \pi'')$ is optimal for the latter instance. Since the main machine for schedule $s'(\pi', \pi'')$ has zero modified setup times and a zero removal time, the length of schedule $s'(\pi', \pi'')$ is equal to that of schedule $s^o(\pi', \pi'')$. Therefore, the schedule $s'(\pi', \pi'')$ is optimal for the *modified* instance of problem $J2|s_{jk}|C_{max}$. Due to Theorem 3.1, the *modified* instance of problem $J2|s_{jk}|C_{max}$ is equivalent to the *original* instance of problem $J2|s_{jk}|C_{max}$ and the schedule $s(\pi', \pi'')$ defined by the permutations (π', π'') is optimal for the *original* problem $J2|s_{jk}|C_{max}$. ■

The condition of Corollary 4.1 definitely holds, if the minimal matrices $S^{(1)} = \| s_{ij}^{(1)} \|$ and $S^{(2)} = \| s_{ij}^{(2)} \|$ have only zero elements. Thus, we obtain the following sufficient condition for the optimality of schedule $s(\pi', \pi'')$ for the corresponding instance of problem $J2|s_{jk}|C_{max}$:

(j) Matrices $S^{(1)} = \| s_{ij}^{(1)} \|$ *and* $S^{(2)} = \| s_{ij}^{(1)} \|$ *have only zero elements:* $s_{ij}^{(1)} = 0 = s_{ij}^{(2)}$, $i \neq j$.

If it is a priory clear which machine $m \in M$ has to be the main machine in schedule $s(\pi', \pi'')$ without idle times on machine *m*, then the above sufficient condition is reduced to the following one:

(jj) Matrix $S^{(m)}$ *has only zero elements*.

Corollaries 4.1 and the above sufficient conditions *(j)* and *(jj)* provide special cases of problem $J2|s_{jk}|C_{max}$ which are solvable in polynomial time using Jackson's pair of job permutations.

5. WORST CASE ANALYSIS OF THE HEURISTIC ALGORITHM

Using the results proven in section 3 and 4, we propose the following polynomial algorithm for finding an exact solution to problem $J2|s_{jk}|C_{max}$ (if at least one of the sufficient conditions holds) or its heuristic solution (otherwise).

Algorithm HEUR

Step 1: Construct a *modified* instance that is equivalent (due to Theorem 2.1) to the *original* instance of the given problem $J2|s_{jk}|C_{max}$.

Step 2: Find Jackson's pair (π', π'') of job permutations constructed for the problem $J2||C_{max}$ corresponding to the *modified* instance of problem $J2|s_{jk}|C_{max}$.

Step 3: Test the sufficient conditions (given in Corollary 4.1, or conditions *(j)* or *(jj)*) for the optimality of schedule $s'(\pi', \pi'')$ for the *modified* instance of problem $J2|s_{jk}|C_{max}$.

Step 4: If at least one of the above sufficient conditions holds, the schedule $s(\pi', \pi'')$ is optimal for the *original* instance of problem $J2|s_{jk}|C_{max}$. **Stop.** Otherwise **go to** step 5.

Step 5: The schedule $s(\pi', \pi'')$ provides a heuristic solution to the *original* instance of problem $J2|s_{jk}|C_{max}$. **Stop.**

If algorithm HEUR terminates at step 4, it provides an exact solution to the *original* instance of problem $J2|s_{jk}|C_{max}$. If algorithm HEUR terminates at step 5, the schedule $s^o(\pi', \pi'')$ constructed for the corresponding instance of problem $J2||C_{max}$ (schedule $s'(\pi', \pi'')$ constructed for the corresponding modified instance of problem $J2|s_{jk}|C_{max}$) provides a polynomial lower bound LB (upper bound UB, respectively) for the minimal schedule length for problem $J2|s_{jk}|C_{max}$. Both these bounds LB and UB are used in the branch-and-bound algorithm developed for problem $J2|s_{jk}|C_{max}$.

Next, we perform a worst case analysis of the solution $s(\pi', \pi'')$ obtained using algorithm HEUR. First, we consider the case when the following condition holds:

$$s_{ij}^{(m)} \leq p_j^{(m)}, i \in J, i \neq j \in J. \tag{22}$$

Let C^*_{max} denote the optimal value of the objective function (1) for the *original* problem $J2|s_{jk}|C_{max}$, and $C_{max}(\pi', \pi'')$ denote the value of the objective function (1) obtained for the schedule $s(\pi', \pi'')$ calculated using the algorithm HEUR. We use the following notations:

$n_{min} = \min\{\min\{|J \setminus J_1|, |J \setminus J_2|\}, \min\{|J_{12}| + 1, |J_{21}| + 1\}\}$,
$n_{max} = \max\{\max\{|J \setminus J_1|, |J \setminus J_2|\}, \max\{|J_{12}| + 1, |J_{21}| + 1\}\}$,
$s_{min} = \min\{ s_{ij}^{(m)} \mid m \in M, i \in J, i \neq j \in J \}$,
$s_{max} = \max\{ s_{ij}^{(m)} \mid m \in M, i \in J, i \neq j \in J \}$.

The above value n_{min} (n_{max}, respectively) defines the minimal (maximal) cardinality of the critical set of operations which defines the objective value $C_{max}(\pi', \pi'')$.

If condition (22) holds, the obvious bound $C_{max}(\pi', \pi'') \le 2C^*_{max}$ is valid for any semiactive schedule generated from Jackson's pair of permutations constructed for the corresponding *relaxed* instance of the *original* problem $J2|s_{jk}|C_{max}$. Due to Theorem 3.1, we can strengthen this bound as follows:

$$C_{max}(\pi', \pi'') \le 2C^*_{max} - n_{min}s_{min}$$

since at least n_{min} setup and removal times are compensated by the value s_{min} in the *modified* instance of problem $J2|s_{jk}|C_{max}$.

Thus, if condition (22) holds, the bound $C_{max}(\pi', \pi'') \le 2C^*_{max} - n_{min}s_{min}$ holds for the schedule $s(\pi', \pi'')$ constructed by algorithm HEUR. Using similar arguments, we can prove the following bounds.

If

$$p_j^{(m)} \le s_{ij}^{(m)} \le 2p_j^{(m)}, i \in J, i \ne j \in J, \qquad (23)$$

then

$$C_{max}(\pi', \pi'') \le 3/2C^*_{max}.$$

In the general case (when both conditions (22) and (23) do not hold), we obtain the following worst-case bound:

$$C_{max}(\pi', \pi'') \le C^*_{max} + n_{max}(s_{max} - s_{min}).$$

In the latter case, the heuristic rule based on the setup and removal times may be more effective than that based on the modified processing times considered in section 3.

6. BRANCH-AND-BOUND ALGORITHM AND COMPUTATIONAL RESULTS

We develop a branch-and-bound algorithm, called SETUP, for solving problem $J2|s_{jk}|C_{max}$ exactly. Algorithm SETUP is based on the lower bound LB and the upper bound UB obtained by algorithm HEUR, and the stopping rules for branching based on Theorem 3.1 and Corollary 4.1.

The branching procedure is based on fixing an operation at the first place from the left-hand side which is currently free either in the sequence π' on machine 1 or in the sequence π'' on machine 2. After fixing the position of an operation, the size of the subproblem of the *original* problem $J2|s_{jk}|C_{max}$ is decreased by one.

A solution-tree is constructed in order to enumerate feasible semiactive schedules implicitly. At each vertex v_i of the solution tree *T=(V,A)*, the polynomial algorithm HEUR is realized to calculate the lower bound LB$_i$ for the objective function (1) equal to the length of the schedule $s^o(\pi', \pi'')$ and the upper bound UB$_i$ for the objective function (1) equal to the length of the schedule $s'(\pi', \pi'')$. To this end, the corresponding subproblem has to be modified using the modification mentioned in Theorem 3.1. All calculations are realized for the *modified* problem $J2|s_{jk}|C_{max}$ and the *relaxed* problem $J2||C_{max}$.

Algorithm HEUR allows us to cut branching from vertex v_i if at least one of the sufficient conditions proven in section 4 or inequality

$$\text{LB}_i \le \text{UB} \qquad (24)$$

holds, where UB denotes the smallest upper bound on the objective function value (1) for the best schedule for the *original* instance of problem $J2|s_{jk}|C_{max}$ currently constructed in the solution tree.

Algorithm SETUP was coded in C++ and tested on a PC Pentium (2800 MHz) for solving randomly generated problems $J2|s_{jk}|C_{max}$ with $n \le 300$. Table 1 shows the results of the computational experiments for the case when the numbers of jobs in the sets J_{12}, J_1, J_2, and J_{21} are the same and equal to ¼ $|J|$. The number of jobs $n = |J|$ is given in the first column of Table 1.

Table 2 shows the results of the computational experiments for the case when the numbers of jobs in the subsets J_{12}, J_1, J_2, and J_{21} of the set J are different and $n = 100$ (the cardinalities of these subsets are given in the first column of Table 2). The interval for the possible job processing times (setups times) are given in columns 2 and 3 (columns 4 and 5, respectively).

Each line in Tables 1 and 2 presents the results for a series of 10 randomly generated instances. For each series of instances, the number of instances unsolved within the given limit of CPU time or the limit of vertices $|V|$ constructed in the solution tree *T=(V,A)* is given in column 6. In our experiments, we used at most 900 seconds of CPU time and at most 15,000,000 vertices $|V|$ for solving each problem instance. The average and maximal running times used for solving one instance in seconds on a PC Pentium IV processor are given in columns 7 and 9. Column 8 gives the average number of vertices in the solution tree *T=(V,A)* constructed for solving one instance.

Table 1. Computational results for problems with *n* jobs and $60 \le n \le 300$

Number of jobs	Processing times		Setup times		Number of unsolved problems	Average CPU time	Average number $\|V\|$ of vertices	Maximal CPU time
1	2	3	4	5	6	7	8	9
60	10	100	0	10	0	1.8	32898.8	5
80	10	100	0	10	1	86.3	1725784	466
100	10	100	0	10	0	20.7	448734.7	105
120	10	100	0	10	0	28.7	450656.2	70
140	10	100	0	10	1	45.3	402628.8	91
160	10	100	0	10	0	97.6	1039838	244
180	10	100	0	10	1	132.1	844546.7	153
200	10	100	0	10	1	145.6	396204.6	172
220	10	100	0	10	0	267.8	1065730	453
240	10	100	0	10	0	388.9	929035.3	541
260	10	100	0	10	3	533.3	894747.7	538
280	10	100	0	10	1	798.9	1337519	872
300	10	100	0	10	10			
60	1	100	0	10	0	18.2	593697.8	83
80	1	100	0	10	2	262.9	4549846	742
100	1	100	0	10	1	91.3	1759915	516
120	1	100	0	10	2	20	239937.3	58
140	1	100	0	10	0	153.8	2074839	656
160	1	100	0	10	1	96.9	1317195	399
180	1	100	0	10	2	219.9	1139278	813
200	1	100	0	10	4	220.2	822969,5	569
220	1	100	0	10	2	317.3	1291139.9	569
240	1	100	0	10	2	413.3	1442192.8	604
260	1	100	0	10	3	542	837036.6	668
280	1	100	0	10	4	818.8	1644840.8	882
300	1	100	0	10	10			
60	20	100	0	20	2	171.9	3517887	440
80	20	100	0	20	1	84.1	1345522	417
100	20	100	0	20	4	150.4	1429831	374
120	20	100	0	20	3	186.7	1523421.5	483
140	20	100	0	20	7	41	241643.3	56
160	20	100	0	20	6	297	2993533.5	443
180	20	100	0	20	5	270.2	1434625	623
200	20	100	0	20	6	562	2848247	899
220	20	100	0	20	3	589	2012981	812
240	20	100	0	20	4	538.8	1437953	855
260	20	100	0	20	7	642.7	1297065	800
280	20	100	0	20	9	857	1375839	857
60	30	100	0	30	2	147	2765492	538
80	30	100	0	30	6	76.3	1102367.3	131
100	30	100	0	30	6	393.8	3952795.8	577
120	30	100	0	30	6	73.8	566798.3	194
140	30	100	0	30	9	30	161116	30
160	30	100	0	30	8	523,5	2847564	607
60	40	100	0	40	2	54.9	1122884	263
80	40	100	0	40	2	117.9	1653783.3	530
100	40	100	0	40	4	271	2683008	798
120	40	100	0	40	9	15	108613	15
60	50	100	0	50	1	103.4	1956272	559
80	50	100	0	50	2	262.8	3544800.5	891
100	50	100	0	50	6	74.8	748976	138
120	50	100	0	50	8	139	983004	221

The numbers in columns 7, 8, and 9 are calculated only for the portion of instances which were solved exactly within the given limits of CPU time and $|V|$.

Table 2. Computational results for problems with 100 jobs

Number of jobs: $\lvert J\rvert=\lvert J_{12}\rvert+\lvert J_1\rvert+\lvert J_2\rvert+\lvert J_{21}\rvert$	Processing times		Setup times		Number of unsolved problems	Average CPU time	Average number of vertices	Maximal CPU time
1	2	3	4	5	6	7	8	9
100 = 30+20+20+30	10	100	0	10	0	152.7	1719753	712
100 = 35+15+15+35	10	100	0	10	1	180.3	2109264	782
100 = 40+10+10+40	10	100	0	10	2	220.6	1931838	499
100 = 45+5+5+45	10	100	0	10	1	22.7	203081.8	58
100 = 20+30+30+20	10	100	0	10	1	45.7	799548.8	174
100 = 15+35+35+15	10	100	0	10	0	14.8	240780.9	39
100 = 10+40+40+10	10	100	0	10	2	112.25	2003443	509
100 = 5+45+45+5	10	100	0	10	0	22.1	745053.2	78

7. CONCLUDING REMARKS

In most of the shop-scheduling models considered in the OR literature, it is assumed that an individual processing time incorporates all other time parameters (lags) attached to a job. In practice, however, such parameters often have to be considered separately from the actual job processing times. For example, if for an operation some pre-processing and post-processing are required, then it is necessary to use a scheduling model with setup and removal times separated. Moreover, setup times are often sequence-dependent. In sections 3 and 4, we derived sufficient conditions when Jackson's pair of job permutations may be used for solving the two-machine job-shop problem with sequence-dependent setup times and removal times.

The main issue of this paper was to test the significance of the modifications based on Theorem 3.1 for problem $J2|s_{jk}|C_{max}$. The results based on a modification of the setup and removal times may be used for calculating lower bounds for the minimal length of a schedule for problem $Jm|s_{jk}|C_{max}$ with $m > 2$ machines. To this end, one can use a decomposition of problem $Jm|s_{jk}|C_{max}$ into a series of problems $J2|s_{jk}|C_{max}$.

In a forthcoming paper, we will present computational results for heuristic and exact algorithms based on Corollary 4.1 and some other sufficient conditions for the optimality of Jackson's permutations for problem $J2|s_{jk}|C_{max}$. As shown in (Braun, Leshchenko, and Sotskov, 2006), a stability analysis used for the job-shop with limited machine availability allows one to solve randomly generated problems with thousands of jobs exactly. We plan to test similar conditions for problem $J2|s_{jk}|C_{max}$.

This research was partially supported by the National Science Council of Taiwan, by INTAS (project 03-51-5501), and by ISTC (project B-986).

REFERENCES

1. Allahverdi, A. (2000). Minimizing mean flowtime in a two-machine flowshop with sequence independent setup times. *Computers & Operations Research,* 27: 111-127.
2. Allahverdi, A., Aldowaisan, T., and Sotskov, Yu. N. (2003). Two-machine flowshop scheduling problem to minimize makespan or total completion time with random and bounded setup times. *International Journal of Mathematics and Mathematical Sciences,* 39 (11): 2475-2486.
3. Allahverdi, A., Gupta, J.N.D, and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *OMEGA The International Journal of Management Sciences,* 27: 219-239.
4. Allahverdi, A., Ng, C.T, Cheng, T.C.E., and Kovalyov, M.Y. (2006). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* (to appear).
5. Artigues, C., Belmokhtar, S., and Feillet, D. (2004). A new exact solution algorithm for the job shop problem with sequence-dependent setup times. In *J.C. Regin and M. Rueher, editors, 1st International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems.* Lecture Note in Computer Science. 3011, Springer: 37-49.

6. Artigues, C., and Buscaylet, F. (2003). A fast tabu search method for the job-shop problem with sequence-dependent setup times. *Proceedings of Metaheuristic International Conference.* Kyoto, Japan, August 23-28: 1-6.
7. Artigues, C., Lopez, P., and Ayache, P. (2005). Scheduling generation schemes for the job-shop problem with sequence-dependent setup times: dominance properties and computational analysis. *Annals of Operations Research* (to appear).
8. Artigues, C., and Roubellat, F. (2002). An efficient algorithm for operation insertion in a multi-resource job-shop scheduling with sequence-dependent setup times. *Production Planning and Control,* 13: 175-186.
9. Bagga, P.C., and Khurana, K. (1986). Two-machine flowshop with separated sequence-independent setup times: Mean completion time criterion. *Indian Journal of Management and Systems,* 2: 47-57.
10. Ballicu, M., Giua, A., and Seatzu, C. (2002). Job-shop scheduling models with set-up times. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 5: 95-100.
11. Braun, O., Leshchenko, N.M., and Sotskov, Yu.N. (2006). Optimality of Jackson's permutations with respect to limited machine availability. *International Transactions in Operational Research,* 13: 59-74.
12. Brucker, P., and Thiele, O. (1996). A branch and bound algorithm for the general job shop with sequence dependent setup times. *OR Spektrum,* 18: 145-161.
13. Cheng, T.C.E., Gupta, J.N.D, and Wang, G. (2000). A review of flowshop scheduling research with setup times. *Production and Operations Management,* 9: 262-282.
14. Cheung, W., and Zhou, H. (2001). Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times. *Annals of Operations Research*, 107: 65-81.
15. Choi, I.C., and Choi, D.S. (2002). A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups. *Computers & Industrial Engineering*, 42: 43-58.
16. Choi, I.C., and Korkmaz, O. (1997). Job shop scheduling with separable sequence-dependent setups. *Annals of Operations Research*, 70: 155-170.
17. Gupta, S.K. (1982). *n* jobs and *m* machines job-shop problem with sequence-dependent setup times. *International Journal of Production Research,* 20: 643-656.
18. Jackson, J.R. (1956). An extension of Johnson's results on job lot scheduling. *Naval Research Logistics Quarterly,* 3: 201-203.
19. Kim, S.C., and Bobrowski, P.M. (1994). Impact of sequence-dependent setup times on job shop scheduling performance. *International Journal of Production Research,* 32: 1503-1520.
20. Khurana, K., and Bagga, P.C. (1984). Minimizing the makespan in a two-machine flowshop with time lags and setup conditions. *Zeitschrift Operations Research,* 28: 163-174.
21. Low, C.Y. (1995). Job shop scheduling heuristics for sequence dependent setups. *Computers & Industrial Engineering*, 29: 279-283.
22. O'Grady, P.J., and Harrison, C. (1988). Search based job scheduling and sequencing with setup times. *OMEGA The International Journal of Management Sciences,* 16: 547-552.
23. Ovacik, I.M., and Uzsoy, R. (1994). Exploiting shop floor status information to schedule complex job shops. *Journal of Manufacturing Systems*, 13: 73-84.
24. Potts, C.N., and Kovalyov, M.Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, 120: 228-349.
25. Sotskov, Yu.N., Tautenhahn, T, and Werner, F. (1999). On the application of insertion techniques for job shop problems with setup times. *RAIRO Recherche Operationnelle*, 33 (2): 209-245.
26. Sun, J.U., and Yee, S.R. (2003). Job shop scheduling with sequence dependent setup times to minimize makespan. *International Journal of Industrial Engineering: Theory Applications and Practice,* 10: 455-461.
27. Tahar, D.N., Yalaoiui, F., Amodeo, L., and Chu, C. (2005). An ant colony system minimizing total tardiness for hybrid job-shop scheduling problem with sequence-dependent setup times and release dates. *Proceedings of the International Conference on Industrial Engineering and Systems Management.* Marrakech, Morocco, May 16-19: 469-478.
28. Wilbrecht, J.K., and Prescott, W.B. (1969). The influence of setup time on job shop performance. *Management Sciences,* 16: B274-B280.
29. Yang, W.H., and Liao, C.J. (1999). Survey of scheduling research involving setup times. *International Journal of System s Science*, 30: 143-155.
30. Yoshida, T., and Hitomi, K. (1979). Optimal two-stage production scheduling with setup times separated. *AIIE Transactions,* 11: 261-263.
31. Zant, P.V. (1997). *Microchip Fabrication – A Practical Guide to Semiconductor Processing*, 3rd ed., McGraw-Hill, New York, The USA.
32. Zhou, C., Egbelu, P.J. (1989). Scheduling in a manufacturing shop with sequence-dependent setups. *Robotics Comput-Integr. Manufacturing,* 5: 73-81.
33. Zoghby, J., Barnes, J.W., and Hasenbein, J.J. (2005). Modeling the reentrant job shop scheduling problem with setups for metaheuristic searches. *European Journal of Operational Research*, 167: 336-348.