# Comparative Evaluation of Heuristic Algorithms for the Single Machine Scheduling Problem with Two Operations per Job and Time-Lags

**Jatinder N. D. Gupta**

*Department of Management*
*Ball State University*
*Muncie, IN 47306, USA*

Revised: March 25, 1996

**Abstract**

This paper shows that the single machine scheduling problem with multiple operations per job separated by minimum specified time-lags is NP-hard in the strong sense. Seven simple and polynomially bounded heuristic algorithms are developed for its solution when each job requires only two operations. Empirical evaluation shows that the percentage deviation of the heuristic solutions from their lower bounds is quite low and the effectiveness of these heuristic algorithms in finding optimal schedules increases with an increase in the number of jobs.

Keywords: *Single machine scheduling, multiple operations, time-lags, NP-hardness, heuristics, worst-case analysis, computational results*

Acknowledgements: This work started while the author was a Visiting Professor at the University of Twente, Enschede, Holland. The author thanks Mauro Dell'Amico, Jan Karel Lenstra, William M. Nawijn, and Steef Van de Velde for valuable discussions on this problem which greatly enhanced the quality of the results presented in this paper. Thoughtful comments of an anonymous reviewer that greatly enhanced the quality and clarity of the paper are appreciated.

© 1996 *Kluwer Academic Publishers. Printed in the Netherlands.*

## 1 Introduction

Traditional scheduling models assume that each job requires only one operation per machine. However, in several practical situations, a machine is capable of performing more than one type of operation. Further, a finite time must elapse between the processing of two consecutive operations of the same job on the same machine. Depending on the practical applications being considered, a machine may remain idle during this time or may perform operations of other jobs.

As an illustration of the scheduling problems with multiple job operations per machine, consider a simple Flexible Manufacturing System (FMS) with only one work-station. This work-station is capable of performing more than one type of operation. All operations of a job need not be performed consecutively.

In order to perform two consecutive operations of a job on the same work-station, refixturing of the part has to take place in a load/unload station accompanied by (manual) transportation activities [1]. These activities (called preparing a stage to perform the operation) do not consume any time at the FMS work-station as they are done off-line, but do take a finite amount of time. This time is independent of the job sequence and need not be the same for all operations or jobs. During the time a job is being prepared for processing, the FMS work-station could be used to process one of the two operations of another job and need not remain idle.

One way to approach the above problem is to assume that the off-line operations can be done as and when required without any resource conflicts. With this assumption, the scheduling problem is formulated as a single machine problem with multiple job operations where the time consumed for outside activities is represented as a *finish-to-start time-lag* between operations. Once a schedule for the FMS work-station has been constructed, it can be checked whether or not the other activities can be performed within the available time.

As another example of the scheduling with multiple operations and time-lags, consider the scheduling of batch processing in a chemical plant where the same processor is used to perform various operations on the same job. However, because of several chemical reactions (like the need for a job to cool down to a desired temperature), a finite amount of time must elapse between the finish of an operation of a job and the start of the next operation of the same job.

The scheduling problems with multiple operations and time-lags are considered by Kern and Nawijn [6] and Riezebos et al [8]. Kern and Nawijn [6] consider a single machine scheduling problem where each job requires two operations separated by stop-to-start time-lags between operations and show that the problem is NP-hard in the ordinary sense. Riezebos et al [8] consider a multi-stage flowshop problem with multiple operations per job at the same machine and stop-to-start time-lags between various operations and develop heuristic algorithms to solve the problem.

The problem with multiple operations and time-lags can also be considered as a scheduling problem with communication delays and precedence constraints considered by Balas et al [2]. Even though their problem includes non-zero release and due dates, the problem can be formulated as a single machine problem with multiple operations and time-lags. Looked at this way, a solution of the single machine scheduling problem with multiple operations and time-lags may provide some insights into the solution of the flowshop or job shop scheduling problems as well [2, 8].

This paper considers the single machine scheduling problem with multiple operations and time-lags where each job requires only two operations and develops heuristic algorithms to find minimum makespan schedules. The rest of the paper is organized as follows: Section 2 shows that the single machine scheduling problems with multiple (and exactly two) operations per job and time-lags are NP-hard in the strong sense and describes simple lower bounds on the makespan of a schedule for the single machine scheduling problem with two operations per job and time-lags. Seven heuristic algorithms to minimize makespan are developed in section 3. The effectiveness of these heuristic algorithms in finding optimal schedules is empirically evaluated in Section 4 and found to increase with the increase in the number of jobs. Finally section 5 concludes the paper with some fruitful directions for future research.

## 2 Problem Complexity and Lower Bounds

Following the standard notations of Lawler et al [7], we will represent the single machine problem with multiple operations per job and time-lags as a $1|O_i > 1, time-lags|C_{\max}$ problem, where $O_i$ represents the total number of operations for job $i$, *time-lags* denotes the condition of time-lags between operations, and $C_{\max}$ implies that the objective is one of minimizing the maximum flowtime (also called the makespan). Further, the notation $O_i > 1$ in the problem representation implies that at least some jobs require more than one operation per job.

Let the processing times of the two operations of job $i \in N$ be $a_i$ and $b_i$ where $N = \{1, 2, ...., n\}$ is the set of all $n$ jobs. Further, let $c_i$ denote the minimum time-lag between the completion of the first operation of job $i$ and start of its second operation. Then, we sate the following property:

**Property 1** : *For solving the $1|O_i = 2, time-lags|C_{\max}$ problem, it is sufficient to consider schedules where the second operation of any job is processed only after the completion of the first operation of all jobs.*

Let $C_{i,1}$ be the completion time of the first operation of job $i \in N$. Further, let $\sum a_i$, $\sum c_i$, and $\sum b_i$ denote the sum of the processing times of first operations, time-lags, and processing times of second operations of all jobs respectively. For each job $i$, define the release date, $r_i$, as the earliest time at which the second operation of job $i$ can begin processing. Thus, using property 1 above, $r_i$ can be expressed as follows:

$$r_i = max\left\{C_{i,1} + c_i; \sum a_i\right\} \tag{1}$$

Then an optimal schedule of the second operations with respect to a given sequence of the first operations of the jobs can be found using the following property:

**Property 2** : *For a specified schedule of first operations, an optimal schedule of the second operations of all jobs is obtained by processing second operations of all jobs in non-decreasing order of their release dates.*

## 2.1 Problem Complexity

Kern and Nawijn [6] used a reduction from a simple partition problem to show that the $1|O_i = 2, time - lags|C_{\max}$ problem is NP-hard in the ordinary sense. However, the issue of whether the $1|O_i > 1, time - lags|C_{\max}$ and $1|O_i = 2, time - lags|C_{\max}$ problems are pseudopolynomially solvable is unresolved. We now provide a negative answer for this question.

**Theorem 1** : *The $1|O_i > 1, time - lags|C_{\max}$ and $1|O_i = 2, time - lags|C_{\max}$ problems are NP-hard in the strong sense.*

**Proof**: Using a reduction from a $F2|time - lags|C_{\max}$ problem, we first show that the $1|O_i = 2, time - lags|C_{\max}$ problem is NP-hard in the strong sense. To do so, consider a $F2|time - lags|C_{\max}$ problem (called problem $P1$) with $n$ jobs where the processing times of job $i$ are represented as $\alpha_i$ and $\beta_i$, and its stop-to-start time lag is $t_i$. Now define a $1|O_i = 2, time - lags|C_{\max}$ problem $P2$ with $n$ jobs where the processing times of job $i$ are $a_i = \alpha_i$ and $b_i = \beta_i$, and its time-lag is $c_i = t_i + K$ where $K = \sum a_i$.

Since the optimal sequence of second operations is given by property 2, it follows that the optimal schedules of problems $P1$ and $P2$ are identical. Since the $F2|time-lags|C_{\max}$ problem is known to be NP-hard in the strong sense [3, 4], it follows that the $1|O_i = 2, time - lags|C_{\max}$ problem, is NP-hard in the strong sense.

Since the $1|O_i = 2, time - lags|C_{\max}$ problem is a special case of the $1|O_i > 1, time - lags|C_{\max}$ problem, the above results also imply that the $1|O_i > 1, time - lags|C_{\max}$ problem is NP-hard in the strong sense □.

## 2.2 Lower Bounds

We now describe lower bounds on the makespan of schedule for the $1|O_i = 2, time - lags|C_{\max}$ problem. First, the makespan cannot be less than the sum of the processing times of all operations. Thus,

$$LB1 = \sum a_i + \sum b_i \tag{2}$$

Second, we assume that the first operation of all jobs can be done simultaneously. This relaxed problem is equivalent to a $1|r_i = a_i + c_i|C_{\max}$ problem which can be solved by finding a schedule $\sigma = (\sigma(1), \sigma(2), ...., \sigma(n))$ such that jobs are arranged in an ascending order of $r_i$ values. Then, the makespan of this schedule, $M1$, is a lower bound (LB2) for the $1|O_i = 2, time - lags|C_{\max}$ problem. Thus,

$$LB2 \;=\; M1 = max_{1\leq j\leq n}\left\{a_{\sigma(j)} + c_{\sigma(j)} + \sum_{s=j}^{n} b_{\sigma(s)}\right\} \tag{3}$$

Third, let the schedule $\pi = (\pi(1), \pi(2), ..., \pi(n))$ be such that $\{b_{\pi(i)} + c_{\pi(i)}\} \geq \{b_{\pi(i+1)} + c_{\pi(i+1)}\}$ for all $2 \leq i \leq n$. Then, by symmetry of the problem, makespan of this schedule, $M2$, is a lower bound for the $1|O_i = 2, time - lags|C_{\max}$ problem. Thus,

$$LB3 = M2 = max_{1\leq j\leq n}\left\{\sum_{s=1}^{j} a_{\pi(s)} + c_{\pi(j)} + b_{\pi(j)}\right\} \tag{4}$$

Combining equations (2), (3), and (4) above, a lower bound, $LB$, for the makespan of a $1|O_i = 2, time - lags|C_{\max}$ problem can be stated as: $LB = \max\{LB1, LB2, LB3\}$.

## 3 Heuristic Algorithms

Since the problem is NP-hard in the strong sense, this section develops effective and efficient heuristic algorithms for the $1|O_i = 2, time - lags|C_{\max}$ problem.

### 3.1 Heuristics with Permutation Schedules

We first assume that the first and second operations of all jobs are processed in the same order and describe three $O(nlogn)$ heuristic algorithms that generate permutation schedules.

#### Heuristics Algorithm H1

Our first heuristic algorithm processes jobs in descending order of their time-lags as follows:

*Input:* $c_i$ for $i = 1, \ldots, n$.

*Step 1.* Find a schedule $\pi = (\pi(1), \pi(2), ....., \pi(n))$ such that $c_{\pi(i)} \geq c_{\pi(i+1)}$ for $2 \leq i \leq n$.

**Heuristic algorithm H2**

The performance of the algorithm $H1$ can be improved by considering different scheduling rules depending on whether the processing times of the first operations dominate those of the second operations. Thus, algorithm $H2$ may be described as follows:

*Input:* $a_i$, $b_i$, and $c_i$ for $i = 1, \ldots, n$.

*Step 1.* If $\sum b_i \leq \sum a_i$, find a schedule $\pi = (\pi(1), \ldots, \pi(n))$ such that $\{b_{\pi(i)} + c_{\pi(i)}\} \geq \{b_{\pi(i+1)} + c_{\pi(i+1)}\}$ for $2 \leq i \leq n$; otherwise enter step 2.

*Step 2.* Find a schedule $\sigma = (\sigma(1), \ldots, \sigma(n))$ such that $\{a_{\sigma(i)} + c_{\sigma(i)}\} \leq \{a_{\sigma(i+1)} + c_{\sigma(i+1)}\}$ for $2 \leq i \leq n$.

**Heuristic algorithm H3**

Algorithm $H3$ uses Johnson's algorithm [5] to find a permutation schedule by defining a $F2||C_{\max}$ problem with processing times $\alpha_i = a_i + c_i$ and $\beta_i = c_i + b_i$ as follows:

*Input:* $\alpha_i$, $\beta_i$ for $i = 1, \ldots, n$.

*Step 1.* For jobs $j$ with $\alpha_j \leq \beta_j$, form a partial sequence $\pi$ such that $\alpha_{\pi(1)} \leq \ldots \leq \alpha_{\pi(n')}$, where $n'$ is the number of jobs $j$ with $\alpha_j \leq \beta_j$.

*Step 2.* Append the jobs $j$ with $\alpha_j > \beta_j$ to $\pi$ so that $\beta_{\pi(n'+1)} \geq \cdots \geq \beta_{\pi(n)}$.

Considering permutation schedules, heuristic algorithm $H3$ provides the best results since algorithm $H3$ above finds an *optimal permutation schedule* for the $F2|time-lags|C_{\max}$ and hence the $1|O_i = 2, time-lags|C_{\max}$ problem.

## 3.2 Improvement Heuristics

The effectiveness of the above heuristic algorithms can be improved by considering non-permutation schedules. Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ and $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be the schedules of first and second operations. Using property 2 to reorder the second (or first) operations, two basic $O(nlogn)$ improvement algorithms are as follows:

**Algorithm F: Forward Improvement**

*Input:* $\pi$, $a_i$, $c_i$ for $i = 1, \ldots, n$.

*Step 1.* For each job $\pi(i) \in \pi$, calculate $r_{\pi(i)} = \sum_{s=1}^{i} a_{\pi(s)} + c_{\pi(i)}$. Let $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be the schedule such that $r_{\sigma(i)} \leq r_{\sigma(i+1)}$ for $2 \leq i \leq n$.

**Algorithm B: Backward Improvement**

*Input:* $\sigma$, $b_i$, $c_i$ for $i = 1, \ldots, n$.

*Step 1.* For each job $\sigma(i) \in \sigma$ in step 1 above, calculate $q_{\sigma(i)} = c_{\sigma(i)} + \sum_{s=i}^{n} b_{\sigma(s)}$. Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ be the schedule $q_{\pi(i)} \geq q_{\pi(i+1)}$ for $2 \leq i \leq n$.

The combinations of three permutation and two improvement heurstics results in several possible heuristic algorithms. In this paper, we consider seven heuristics algorithms as follows:

- Three permutation heuristics: $H1$, $H2$, $H3$.
- Four improved heuristics: $H4$ (= $H2F$ if $\sum b_i \leq \sum a_i$; $H2B$ otherwise), $H5$ (= $H3F$), $H6$ (= $H3BF$), and $H7$ (= best of $H6$ and $H6BF$) where, heuristic algorithm $H5$, for example, represents the use of forward improvement heuristic $F$ when the basic schedule is generated by heuristic $H3$.

The above heuristic algorithms were selected as they provided the best combinations to minimize makespan as confirmed by the empirical tests deccribed in the next section.

## 4 Computational Results

This section reports the results of computational experience with a large number of randomly generated problems which show that the proposed lower bounds and heuristic algorithms are quite effective in solving the $1|O_i = 2, time-lags|C_{\max}$ problem. For this purpose, the proposed heuristic algorithms $H1$ through $H7$ were coded in $FORTRAN$ on a $VAX6620$ computer to solve a large number problems.

## 4.1 Generation of Problems

The processing times of the problems were generated from a uniform distribution in the range $(1, 99)$. In order to examine the effect of differences in processing times and the time-lags, time-lags were allowed to vary in the range $(1, 0.2k \sum_{i=1}^{n} a_i)$ where $1 \leq k \leq 9$. This large range of time-lags was used to ensure that the results obtained from the heuristic algorithms are not biased towards small time-lags. However, for problems where $k \leq 5$, each of the heuristic algorithms was able to find optimal solutions for almost all problems. This is due to the fact that permutation schedules are optimal for most of these problems as the formation of queues makes $LB1$ achievable. For this reason, the time-lags used in these computational experiments were generated by assuming that $6 \leq k \leq 9$. The number of jobs in each data set varied from 4 to 150 with 20 problems for each problem size and each value of $k$. Thus, for each problem size, 80 problems were solved using each of the heuristic algorithms $H1$ through $H7$.

## 4.2 Measures of Algorithm Effectiveness

For each problem, the makespan was found using the proposed heuristic algorithms and compared to its lower bound. Percentage deviation of the heuristic makespan from its lower bound was calculated. Similarly, the number of times the heuristic makespan equals the lower bound was also noted. Altogether, for each of the heuristic algorithms $H1$ through $H7$, the following statistics were computed:

| | |
|---|---|
| $n_1$ | Number of times heuristic makespan equalled its lower bound. |
| $AVR$ | Average percentage deviation of the heuristic makespan from its lower bound. |
| $MAX$ | Maximum percentage deviation of the heuristic makespan from its lower bound. |

For each of the seven algorithms $H1$ through $H7$, Tables 1 through 3 show the results obtained by solving eighty problem for each problem size. The heuristic algorithms $H1$ and $H2$ are not very attractive to solve a problem since the number of non-optimal solutions, average and maximum percentage deviations of the heuristic solutions from their respective makespan are quite large and do not decrease with the increase in the size of the problem. Heuristic algorithm $H3$, while better than algorithms $H1$ and $H2$, is still not very effective in minimizing makespan. The effectiveness of heuristic algorithms $H5$ through $H7$ in finding a minimum makespan schedule increases with the number of jobs. For heuristic algorithms $H4$ through $H7$, results in Tables 1 through 3 show that the average and maximum percent deviation of heuristic makespan from its lower bound consistently decrease with the increase in the size of the problem. This indicates that the heuristic algorithms that generate non-permutation schedules perform much better than those that generate only the permutation schedules. It is also evident that heuristic algorithm $Hj$ performs better than algorithm $Hi$ for $1 \leq i \leq j \leq 7$. Even though algorithm $H7$ is an improvement over algorithm $H6$, the improvement is not as significant as that from $H5$ to $H6$.

**Table 1 Number of Times Heuristic Makespan Equals Lower Bound**

| # of Jobs | Number of times GLB achieved | | | | | | |
|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H5$ | $H6$ | $H7$ |
| 4 | 10 | 24 | 24 | 33 | 30 | 35 | 35 |
| 5 | 6 | 13 | 26 | 35 | 34 | 43 | 45 |
| 6 | 3 | 8 | 20 | 30 | 35 | 44 | 48 |
| 7 | 3 | 12 | 22 | 33 | 34 | 47 | 50 |
| 10 | 0 | 10 | 24 | 42 | 48 | 62 | 64 |
| 15 | 0 | 5 | 23 | 38 | 48 | 60 | 64 |
| 20 | 0 | 8 | 23 | 31 | 51 | 66 | 71 |
| 25 | 0 | 2 | 23 | 39 | 50 | 74 | 75 |
| 30 | 0 | 5 | 24 | 35 | 53 | 71 | 76 |
| 40 | 0 | 2 | 22 | 32 | 54 | 74 | 78 |
| 50 | 0 | 0 | 23 | 36 | 50 | 77 | 78 |
| 70 | 0 | 0 | 26 | 28 | 52 | 75 | 78 |
| 90 | 0 | 1 | 21 | 32 | 59 | 80 | 80 |
| 100 | 0 | 0 | 22 | 30 | 56 | 78 | 80 |
| 120 | 0 | 0 | 24 | 33 | 59 | 78 | 80 |
| 140 | 0 | 0 | 22 | 32 | 58 | 78 | 79 |
| 150 | 0 | 0 | 22 | 30 | 59 | 80 | 80 |
| Total | 22 | 90 | 391 | 569 | 830 | 1122 | 1161 |
| Percent | 1.62 | 6.62 | 28.75 | 41.84 | 61.03 | 82.5 | 85.37 |

From Tables 1 through 3, it is clear that the effectiveness of the proposed heuristic algorithms $H4$ through $H7$ increases as the number of jobs increases. In view of the fact that lower bounds were used instead of the optimal makespans, actual percent deviations from optimal makespan will be less than those reported in Tables 1 through 3, especially for small sized problems. Further, since the lower bounds may be weak for some problem categories, the percentage deviation of heuristic makespan from lower bounds for these problems may be much larger than the actual percentage deviation from their optimal solutions. Thus, we may conclude that the performance of the heuristic algorithms $H4$ through $H7$ does not significantly deteriorate with changes in the problem data characteristics.

**Table 2 Average Percentage Deviation of Heuristic Makespan from Lower Bound**

| # of Jobs | Average percent deviation from lower bound | | | | | | |
|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H5$ | $H6$ | $H7$ |
| 4 | 16.78 | 13.39 | 10.04 | 5.06 | 4.89 | 3.79 | 3.47 |
| 5 | 18.93 | 13.67 | 8.78 | 4.30 | 4.71 | 2.39 | 2.11 |
| 6 | 21.81 | 16.26 | 10.22 | 4.76 | 4.25 | 2.48 | 2.00 |
| 7 | 21.38 | 16.01 | 9.78 | 4.43 | 4.92 | 2.32 | 1.79 |
| 10 | 22.29 | 17.20 | 9.01 | 3.17 | 3.23 | 1.23 | 0.92 |
| 15 | 22.99 | 19.60 | 9.80 | 3.99 | 3.22 | 1.53 | 1.07 |
| 20 | 24.07 | 19.99 | 10.50 | 3.75 | 3.09 | 0.82 | 0.55 |
| 25 | 23.78 | 20.19 | 9.30 | 2.63 | 1.92 | 0.40 | 0.24 |
| 30 | 23.87 | 20.60 | 9.24 | 2.13 | 2.02 | 0.41 | 0.20 |
| 40 | 24.25 | 21.83 | 10.08 | 2.18 | 2.08 | 0.29 | 0.17 |
| 50 | 24.45 | 22.10 | 10.02 | 1.47 | 2.15 | 0.19 | 0.10 |
| 70 | 24.76 | 22.80 | 10.02 | 1.59 | 1.85 | 0.09 | 0.01 |
| 90 | 24.59 | 22.97 | 9.84 | 0.93 | 1.59 | 0.00 | 0.00 |
| 100 | 24.59 | 22.70 | 9.57 | 1.05 | 1.49 | 0.03 | 0.00 |
| 120 | 24.83 | 23.47 | 10.10 | 1.21 | 1.85 | 0.02 | 0.00 |
| 140 | 24.77 | 23.25 | 9.75 | 0.89 | 1.66 | 0.05 | 0.00 |
| 150 | 25.00 | 23.71 | 10.34 | 0.81 | 1.88 | 0.00 | 0.00 |
| Overall | 23.13 | 19.99 | 9.79 | 2.61 | 2.75 | 0.94 | 0.74 |

The computational times required to solve the problems were not measured. The solutions by heuristic algorithms required very small amount of $CPU$ time on the VAX 6620 computer. In any case, each of the proposed heuristic algorithm is polynomially bounded as the computational effort required is $O(nlogn)$. Within the polynomial bounded limits of the computational times, the order of computational time required by various algorithms is as follows: $H1 - H2(H3) - H4(H5) - H6 - H7$ where the later algorithms require more computation time than the earlier heuristic algorithms.

From a practical viewpoint, use of heuristic algorithms $H1$ through $H3$ is not recommended for solving a problem. The selection of a specific heuristic algorithm from $H4$ through $H7$ depends on the specific problem characteristics. However, the computational results above suggest that use of heuristic algorithms $H6$ or $H7$ would be the best choice to solve a given problem.

**Table 3 Maximum Percentage Deviation of Heuristic Makespan from Lower Bound**

| # of Jobs | Maximum percent deviation from lower bound | | | | | | |
|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H5$ | $H6$ | $H7$ |
| 4 | 39.64 | 36.23 | 28.19 | 24.01 | 19.46 | 19.46 | 19.46 |
| 5 | 45.42 | 33.55 | 28.19 | 29.34 | 27.05 | 17.76 | 17.76 |
| 6 | 42.60 | 41.14 | 35.74 | 23.40 | 23.27 | 17.50 | 15.41 |
| 7 | 41.49 | 39.95 | 31.04 | 21.38 | 20.03 | 18.42 | 14.40 |
| 10 | 40.28 | 42.48 | 32.08 | 25.31 | 23.51 | 12.14 | 11.02 |
| 15 | 39.98 | 41.97 | 33.89 | 19.69 | 15.54 | 14.86 | 11.70 |
| 20 | 43.97 | 43.97 | 35.04 | 22.22 | 19.71 | 14.10 | 10.83 |
| 25 | 41.72 | 41.72 | 29.16 | 20.83 | 18.06 | 13.41 | 10.02 |
| 30 | 44.81 | 44.81 | 32.51 | 13.10 | 18.35 | 7.03 | 7.03 |
| 40 | 44.72 | 44.72 | 29.72 | 16.91 | 13.12 | 11.33 | 9.09 |
| 50 | 43.04 | 43.04 | 32.14 | 14.37 | 16.23 | 10.99 | 7.65 |
| 70 | 43.88 | 43.88 | 33.06 | 11.48 | 13.45 | 1.94 | 0.67 |
| 90 | 42.49 | 42.49 | 27.92 | 7.65 | 11.75 | 0.00 | 0.00 |
| 100 | 41.36 | 41.36 | 29.31 | 7.17 | 12.68 | 1.48 | 0.00 |
| 120 | 42.43 | 42.43 | 29.11 | 8.34 | 12.47 | 1.44 | 0.00 |
| 140 | 42.14 | 42.14 | 28.83 | 6.40 | 14.35 | 3.13 | 0.16 |
| 150 | 42.01 | 42.01 | 27.32 | 5.04 | 10.20 | 0.00 | 0.00 |
| Overall | 45.42 | 44.81 | 35.74 | 29.34 | 27.05 | 19.46 | 19.46 |

## 4.3 Effectiveness of Lower Bounds

To investigate the closeness of these lower bounds to their respective optimal makespans, problems involving four to seven jobs were solved using complete enumeration procedure as follows: for each of the $n!$ schedules of the first operations, an optimal schedule for the second operations was found using property 2. The best of these schedules gives the optimal makespan. Based on 80 problems of each problem size, the following statistics were collected:

$n_1$ Number of times the optimal makespan equalled its lower bound.

$n_2$ Number of times the deviation of the optimal makespan from its lower bound was greater than zero but less than or equal to 3 percent.

$n_3$ Number of times the deviation of the optimal makespan from its lower bound was greater than 3 per cent but less than or equal to 5 percent.

$n_4$ Number of times the deviation of the optimal makespan from its lower bound was greater than 5 percent.

$AVR$ Average percentage deviation of the optimal makespan from its lower bound

$MAX$ Maximum percentage deviation of the optimal makespan from its lower bound.

The results in Table 4 above show that the proposed lower bounds are quite effective for a large number of problems. In 61.87 percent of cases, the lower bound is equal to its optimal makespan. Only in 13.75 percent of the problems the optimal makespan deviated from its lower bound by more than 5 percent.

Table 4: Effectiveness of Lower Bounds

| $n$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $AVR$ | $MAX$ |
|---|---|---|---|---|---|---|
| 4 | 40 | 11 | 11 | 18 | 2.4 | 12.01 |
| 5 | 48 | 13 | 11 | 8 | 1.64 | 14.13 |
| 6 | 56 | 10 | 5 | 9 | 1.26 | 13.08 |
| 7 | 54 | 12 | 5 | 9 | 1.33 | 10.15 |
| Aggregate | 198 | 46 | 32 | 44 | 1.66 | 14.13 |
| Percentage | 61.87 | 14.38 | 10.00 | 13.75 | | |

In general, an increase in the range of time-lags causes the lower bounds to deviate more from their optimal makespan values as was confirmed by the empirical results for various heuristic algorithms reported above. Further, the computational results in Tables 1 through 3 indicate that if heuristic algorithm $H7$ is used to find an upper bound on makespan to be used in a branch and bound algorithm employing the proposed lower bounds, in over 85 % of the problems, an optimal schedule will be found at its root node. This indicates that a large percentage of problems can be optimized using the proposed heuristics and lower bounds.

## 5 Conclusions

This paper has considered the single machine scheduling problem with multiple operations and time-lags and has shown that the $1|O_i = 2, time - lags|C_{\max}$ and the $1|O_i > 1, time - lags|C_{\max}$ problems are NP-hard in the strong sense. In view of the NP-hard nature of these problems, polynomial heuristic algorithms are proposed for finding a minimum makespan schedule for the $1|O_i = 2, time - lags|C_{\max}$ problem. Computational experience with randomly generated problems indicates that the proposed heuristic algorithms are quite effective in minimizing makespan. Further, empirical results show that the effectiveness of the proposed heuristic algorithms increases as the number of jobs increases. Algorithms $H6$ and $H7$ give the best results.

Several unresolved issues are worthy of future research. Firstly, finding worst-case performance ratio of the proposed heuristic algorithms would be useful. Secondly, development of the solution techniques for the $1|O_i > 1, time - lags|C_{\max}$ problem is both interesting and desirable. Thirdly, consideration of precedence relationships between operations to develop heuristic algorithms will be useful. Fourthly, extension of the above results to multiple machines and multiple stages cases will provide more practical uses of the techniques proposed here. Finally, development of solution techniques for objective functions other than makespan will be worthwhile.

## References

[1] Aanen, E., *Planning and scheduling in a flexible manufacturing system*, PhD thesis, Faculty of Mechanical Engineering, University of Twente, Enschede, the Netherlands, 1988.

[2] Balas, E., Lenstra, J. K., and Vazacopoulos, A., "The One Machine Problem with Delayed Precedence Constraints and its use in Job Shop Scheduling," *Management Science*, Vol. 41, No. 1, 1995, pp. 94-109.

[3] Dell'Amico, M., "Shop Problems with Two Machines and Time Lags," *Internal Report No. 93/20*, Department of Electronics and Information, Milan Polytechnic, Milan, Italy, 1993.

[4] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G., "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Annals of Discrete Mathematics*, Vol. 5, 1979, pp. 287-326.

[5] Johnson, S. M., "Optimal two and three stage production schedules with setup times included," *Naval Research Logistics Quarterly*, Vol. 1, No. 1, 1954, pp. 61- 68.

[6] Kern, W., and Nawijn, W.M., "Scheduling Multi-operation Jobs with Time-lags on a Single Machine," *Working paper*, University of Twente, Enschede, the Netherlands, 1991.

[7] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B., "Sequencing and scheduling: algorithms and complexity," in Graves, S. C., et al., *Handbook of Operations Research*, Vol. 4, Elsevier, Amsterdam, 1993, pp. 445-552.

[8] Riezebos, J., Gaalman, G., and Gupta, J. N. D., "Flowshop scheduling with multiple operations and time-lags," *Journal of Intelligent Manufacturing*, Vol. 6, 1995, pp. 105-115.