The three-machine flowshop scheduling problem to minimise maximum lateness with separate setup times

C.T. Ng

Department of Logistics,
The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong, PR China
E-mail: Daniel.Ng@inet.polyu.edu.hk

Ali Allahverdi*

Department of Industrial and Management Systems Engineering,
Kuwait University, P.O. Box 5969, Safat, Kuwait
E-mail: allahverdi@kuc01.kuniv.edu.kw
*Corresponding author

Fawaz S. Al-Anzi

Department of Computer Engineering,
Kuwait University, P.O. Box 5969, Safat, Kuwait
E-mail: alanzif@eng.kuniv.edu.kw

T.C. Edwin Cheng

Department of Logistics,
The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong, PR China
E-mail: Edwin.Cheng@inet.polyu.edu.hk

Abstract: We address the three-machine flowshop scheduling problem to minimise maximum lateness, where setup times are considered separate from processing times. We establish a dominance relation for the problem where it specifies which of the two adjacent jobs should precede in an optimal solution. Moreover, we propose three heuristics: Earliest Due Date (EDD), an Enhanced Earliest Due Date (EEDD) and a Polynomial Genetic-based Algorithm (PGA). We conduct computational analysis on randomly generated problems to evaluate the performance of the proposed heuristics. The analysis shows that the performance of EEDD is acceptable if the computational time is of the main concern and the number of jobs are large. The analysis also shows that PGA significantly outperforms EDD and EEDD.

Keywords: scheduling; flowshop; maximum lateness; dominance relation; heuristics.

Reference to this paper should be made as follows: Ng, C.T., Allahverdi, A., Al-Anzi, F.S. and Cheng, T.C.E. (2007) 'The three-machine flowshop scheduling problem to minimise maximum lateness with separate setup times', *Int. J. Operational Research*, Vol. 2, No. 2, pp.135–155.

Biographical notes: C.T. Ng is Associate Professor and Associate Head of the Department of Logistics, The Hong Kong Polytechnic University. He obtained his BSc from the University of Hong Kong, and PhD from the Chinese University of Hong Kong. His research interests are in operations management and supply chain management. He has published over 30 papers in international journals, including *Naval Research Logistics*, *Operations Research Letters*, *European Journal of Operational Research*, *International Journal of Production Economics*, *Computers & Operations Research*, and *Journal of Scheduling*. He is now Associate Editor of the *Asia-Pacific Journal of Operational Research*.

Ali Allahverdi received his BS from Istanbul Technical University, his MSc and PhD from Rensselaer Polytechnic Institute. He is a Professor and received Distinguished Researcher Award from Kuwait University and Dissertation Prize from Rensselaer Polytechnic Institute. He has published over 60 papers in well known international journals. One of his papers is cited among 'Highly Cited Papers' by the *ISI Web of Knowledge* (more than 80 citations). He severed as Guest Editor for *European Journal of Operations Research* and *International Journal of Operations Research*. He is currently serving as Associate Editor of *International Journal of Operations Research* and *Central European Journal of Operations Research*.

Fawaz S. Al-Anzi is an Associate Professor in Computer Engineering Department at Kuwait University. He is currently serving as Vice-Dean in College of Engineering at Kuwait University. He holds a BS Degree in Electrical Engineering from Kuwait University and MS and PhD Degrees in Computer Science from Rensselaer Polytechnic Institute, USA. His current research interests includes scheduling, engineering databases and data mining. He received the National Research Award from Kuwait Foundation for Advancement of Sciences in 2004 and the Research Performance Award from Kuwait University in 2005. He published over 20 papers in well-known international journals and presented over 45 papers in international conferences.

T.C. Edwin Cheng is Chair Professor of Management in the Department of Logistics of The Hong Kong Polytechnic University. He has published over 300 journal papers and co-authored three books. He received the Outstanding Young Engineer of the Year Award of the IIE, USA in 1992, and the Croucher Senior Research Fellowship in 2001. He was named one of the 'most cited scientists' in All Fields and in Engineering by the *ISI Web of Knowledge* in 2005. One of his papers published in the *EJOR* in 1989 was selected by the editors for inclusion in a collection of 30 influential articles published in *EJOR* in 2006.

1 Introduction

Treating setup times as separate from processing times is important in many applications. One application of separate setup can be found in the printing industry, where machine-cleaning (setup) time depends on the colour of the order. Similar practical situations arise in the chemical, pharmaceutical, food processing, metal processing and semiconductor industries, e.g., see Bianco et al. (1988), Bitran and Gilbert (1990), Uzsoy et al. (1992) and Kim and Bobrowski (1994). Recent surveys on scheduling problems involving setup times are given by Allahverdi et al. (1999, 2006).

Many real life problems can be modelled as a three-machine flowshop scheduling problem where setup times are treated as separate. Allahverdi and Al-Anzi (2006) give a specific application in the area of distributed computing. Distributed computing is becoming one of the most promising choices for speeding up computations, Elmasri and Navathe (1999). This is especially true for cases where tasks can easily be broken into sequential-related subtasks. In the field of databases, it is common to find that the business logic of a transaction (job) can be divided into queries that run on separate machines sequentially. The only relation between two adjacent tasks would be the transfer of results from one subtask to the next one. For this reason, in an m site distributed database, queries are typically programmed to execute on different sites for efficiency reasons. Consider a configuration of three sites. An execution of a job is first submitted to the machine on site 1 for first stage of processing. The result of machine processing on site 1 is then transferred through the network to the machine on site 2 for more processing, which allows the machine on site 1 to process a new job. When processing of machine 2 is complete, the result is transferred to the machine on site 3 for final processing, which leaves the machine on site 2 ready to process further jobs that have been completed by the machine on site 1. Note that some extra times are needed for submitting a job to the machine on site 1 as well as transferring the results to the machines on sites 2 and 3. These times can be observed as times needed for setting-up jobs for processing on machines on different sites.

Yoshida and Hitomi (1979) extended the well-known two-machine flowshop scheduling problem to the case where setup times are separate from processing times and sequence independent with the objective of minimising makespan. Allahverdi (1995) extended the work of Yoshida and Hitomi to stochastic environments. There are other studies addressing flowshop scheduling problem with the objective function of makespan including Rios-Mercado and Bard (1998, 1999).

Bagga and Khurana (1986) addressed the two-machine flowshop problem to minimise total completion time when considering setup times as separate from processing times and sequence independent. They developed one dominance relation and a lower bound for the problem. Allahverdi (2000) also considered the same problem of Bagga and Khurana. He established two more dominance relations for the problem. He also presented a branch-and-bound algorithm incorporating the dominance relations that he and Bagga and Khurana established. Moreover, he presented three heuristics for the problem. Al-Anzi and Allahverdi (2001) showed that the three-tired client-server database internet connectivity problem is equivalent to the two-machine flowshop problem with separate setup times, and hence the results of Bagga and Khurana (1986) and Allahverdi (2000) can be used for the three-tired client-server database internet connectivity problem when the objective is to minimise total completion time. Al-Anzi and Allahverdi (2001) also proposed heuristics for the problem and discussed the computational complexity of the proposed and the previous heuristics. Al-Anzi and Allahverdi (2001) showed that their heuristics outperform the previous ones. Allahverdi and Al-Anzi (2006) considered the three-machine flowshop scheduling problem to minimise total completion time with separate setup times. They obtained a dominance relation and a lower bound for the problem. They also presented a branch-and-bound algorithm for the problem.

Dileepan and Sen (1991) addressed the same problem, but with the objective of minimising maximum lateness. They presented dominance relations along with a lower bound to be used in a branch-and-bound algorithm. They also developed two heuristic algorithms. Allahverdi and Al-Anzi (2002) indicated that the multimedia data objects scheduling problem for WWW Applications can be modelled as a two-machine flowshop problem of minimising maximum lateness with separate setup times. They established three dominance relations and proposed four heuristics. They showed that their proposed heuristics outperform the ones developed by Dileepan and Sen. Other literature on scheduling problems with maximum lateness objective function and separate setup times include the works of Cheng et al. (2001), Pan et al. (2001) and Lin and Jeng (2004). Cheng et al. (2001) and Pan et al. (2001) consider the single machine-scheduling problem while Lin and Jeng (2004) consider the parallel machine-scheduling problem.

The literature reveals that the flowshop problem with separate setup times and maximum lateness criterion has been limited to the two-machine flowshop. In this paper, we extend the problem to the three-machine flowshop problem. We establish a dominance relation and propose several heuristics. We also evaluate the performance of these heuristics.

Problem formulation is presented in the next section. The dominance relation, heuristics, computational experiments and conclusions are presented in Sections 3–6, respectively.

2 Formulation

Let

$t_{j,k}$: Processing time of job $j(j = 1, 2, \ldots, n)$ on machine $k(k = 1, 2, 3)$.
$s_{j,k}$: Setup time of job j on machine k.
d_j: Due date of job j.
$C_{j,k}$: Completion time of job j on machine k.
L_j: Lateness of job j.
$L_{\max}$: Maximum lateness.

Also let $[j]$ denote the job in position j. Therefore, $t_{[j],k}$ denotes the processing time of the job in position j. Other variables are defined similarly.

Let $ST_{[j],k}$ denote the sum of the setup and processing times of jobs in positions $1, 2, \ldots, j$ on machine k, i.e.,

$$ST_{[j],k} = \sum_{r=1}^{j} (s_{[r],k} + t_{[r],k}), \quad j = 1, 2, ..., n \text{ and } k = 1, 2, 3.$$

Moreover, let

$$\delta_{[j]} = ST_{[j],1} - (ST_{[j-1],2} + s_{[j],2}), \quad j = 1, 2, ..., n \tag{1}$$

where $ST_{[0],2} = 0$. Let $IT_{[j],2}$ denote total idle time on the second machine until the job in position j on the machine is completed. It is known that, Allahverdi (2000),

$$IT_{[j],2} = \max\{0, \delta_{[1]}, \delta_{[2]}, ..., \delta_{[j]}\}. \tag{2}$$

Therefore,

$$C_{[j],2} = ST_{[j],2} + IT_{[j],2}. \tag{3}$$

Now let

$$\phi_{[j]} = IT_{[j],2} + ST_{[j],2} - (ST_{[j-1],3} + s_{[j],3}), \quad j = 1, 2, ..., n \tag{4}$$

where $ST_{[0],3} = 0$. Similarly, if $IT_{[j],3}$ denotes the total idle time on the third machine until the job in position j on the machine is completed it can be shown that

$$IT_{[j],3} = \max\{0, \phi_{[1]}, \phi_{[2]}, ..., \phi_{[j]}\}. \tag{5}$$

Hence,

$$C_{[j],3} = ST_{[j],3} + IT_{[j],3}. \tag{6}$$

Once the completion times of jobs on the last (third) machine are known, then, the lateness of the job in position j is defined as:

$$L_{[j]} = C_{[j],3} - d_{[j]}. \tag{7}$$

Therefore,

$$L_{\max} = \max\{L_{[1]}, L_{[2]}, \dots, L_{[n]}\}.$$

It should be noted that throughout this paper, we only consider permutation flowshops.

3 A dominance relation

Dominance relations are common in the scheduling literature, e.g., Chu (1992), Bagga and Khurana (1986), Allahverdi (2000) and Allahverdi and Al-Anzi (2002). They are mainly used in implicit enumeration techniques such as branch-and-bound algorithms. In this section, a dominance relation is developed for our problem.

Consider exchanging the positions of two adjacent jobs on a three-machine flowshop in a sequence π_1 that has job i in an arbitrary position τ and job j in position $\tau + 1$. Consider another sequence that is obtained from the sequence π_1 by only interchanging jobs i and j. Call the sequence obtained from π_1 as π_2, i.e., $\pi_2 = \dots, i, j, \dots$ and $\pi_2 = \dots, j, i, \dots$. If it is shown that $L_{\max}(\pi_2) \le L_{\max}(\pi_1)$, then, sequence π_2 would be no worse than sequence π_1, and, therefore, job j precedes job i in a sequence that minimises maximum lateness.

The following four lemmas will be used for the proof of Theorem 1, which specifies a dominance relation for our problem. In Lemma 1, it is shown that $\delta_{[r]}$ values for both sequences π_1 and π_2 are the same for all positions except τ and $\tau + 1$.

Lemma 1: *For both sequences π_1 and π_2, $\delta_{[r]}(\pi_1) = \delta_{[r]}(\pi_2)$ for $r = 1, 2, \ldots, \tau - 1, \tau + 2, \tau + 3, \ldots, n$.*

Proof: It is obvious that $\delta_{[r]}(\pi_1) = \delta_{[r]}(\pi_2)$ for $r = 1, 2, \ldots, \tau - 1$ since both sequences have the same jobs in these positions. Furthermore, $\delta_{[r]}(\pi_1) = \delta_{[r]}(\pi_2)$ for $r = \tau + 2, \tau + 3, \ldots, n$ since again both sequences have the same jobs in these positions, and both include the jobs in positions τ and $\tau + 1$. Observe that the sum of processing and setup times is taken into account and hence the order is not important since both jobs in positions τ and $\tau + 1$ are considered for $r = \tau + 2, \tau + 3, \ldots, n$.

In the following Lemma, it is shown that $\phi_{[r]}$ value for sequence π_2 is less than or equal to that of π_1 for all positions except τ and $\tau + 1$, if a certain condition is satisfied.

Lemma 2: *If $\max\{\delta_{[\tau]}(\pi_2), \delta_{[\tau+1]}(\pi_2)\} \le \max\{\delta_{[\tau]}(\pi_1), \delta_{[\tau+1]}(\pi_1)\}$, then $\phi_{[r]}(\pi_1) = \phi_{[r]}(\pi_2)$ for $r = 1, 2, \ldots, \tau - 1$, and furthermore $\phi_{[r]}(\pi_2) \le \phi_{[r]}(\pi_1)$ for $r = \tau + 2, \tau + 3, \ldots, n$.*

Proof: It is clear that $\phi_{[r]}(\pi_1) = \phi_{[r]}(\pi_2)$ for $r = 1, 2, \ldots, \tau - 1$ since both sequences have the same jobs in these positions. Now, it follows by definition of $\phi_{[r]}$ that for $r = \tau + 2, \tau + 3, \ldots, n$.

$$\phi_{[r]}(\pi_2) - \phi_{[r]}(\pi_1) = \max\{0, \delta_{[1]}(\pi_2), \ldots, \delta_{[\tau]}(\pi_2), \delta_{[\tau+1]}(\pi_2), \ldots, \delta_{[r]}(\pi_2)\}$$
$$- \max\{0, \delta_{[1]}(\pi_1), \ldots, \delta_{[\tau]}(\pi_1), \delta_{[\tau+1]}(\pi_1), \ldots, \delta_{[r]}(\pi_1)\}.$$

If $\max\{\delta_{[\tau]}(\pi_2), \delta_{[\tau+1]}(\pi_2)\} \le \max\{\delta_{[\tau]}(\pi_1), \delta_{[\tau+1]}(\pi_1)\}$, then by Lemma 1, $\phi_{[r]}(\pi_2) \le \phi_{[r]}(\pi_1)$.

Lemma 3: *For an r value where $r < \tau$,*

$$L_{[r]}(\pi_2) = L_{[r]}(\pi_1).$$

Proof: Since both sequences have the same jobs in positions $1, 2, \ldots, \tau - 1$, $L_{[r]}(\pi_2) = L_{[r]}(\pi_1)$ for $r = 1, 2, \ldots, \tau - 1$.

Lemma 4: *If $\max\{\phi_{[\tau]}(\pi_2), \phi_{[\tau+1]}(\pi_2)\} \le \max\{\phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\}$, then $L_{[r]}(\pi_2) \le L_{[r]}(\pi_1)$ for $r = \tau + 2, \tau + 3, \ldots, n$.*

Proof: For $r = \tau + 2, \tau + 3, \ldots, n$, it follows from the definition of $L_{[r]}$ that

$$L_{[r]}(\pi_2) - L_{[r]}(\pi_1) = \max\{\phi_{[1]}(\pi_2), \ldots, \phi_{[\tau]}(\pi_2), \phi_{[\tau+1]}(\pi_2), \ldots, \phi_{[r]}(\pi_2)\}$$
$$- \max\{\phi_{[1]}(\pi_1), \ldots, \phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1), \ldots, \phi_{[r]}(\pi_1)\}.$$

Therefore, if $\max\{\phi_{[\tau]}(\pi_2), \phi_{[\tau+1]}(\pi_2)\} \le \max\{\phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\}$, then by Lemma 2 $L_{[r]}(\pi_2) \le L_{[r]}(\pi_1)$.

A dominance relation is given in the following theorem.

Theorem 1: *Consider a three-machine flowshop where setup times are treated as separate from processing times. Suppose that two jobs i and j satisfy the following conditions:*

i $d_j \leq d_i$

ii $s_{j,2} + t_{j,2} + s_{i,3} \leq s_{i,2} + t_{i,2} + s_{j,3}$

iii $s_{j,1} + t_{j,1} + s_{i,2} \leq s_{i,1} + t_{i,1} + s_{j,2}$

iv *either* $\{s_{j,2} + t_{j,2} \leq s_{j,3} + t_{j,3}$ *and either* $s_{i,1} + t_{i,1} \leq s_{i,2} + t_{j,2}$ *or* $s_{j,1} + t_{j,1} \leq s_{j,2} + t_{j,2}\}$ *or* $\{s_{i,2} + t_{i,2} \leq s_{i,3} + t_{j,3}$ *and* $s_{i,1} + t_{i,1} \leq s_{i,2} + t_{j,2}\}$.

Then, there exists an optimal solution that minimises maximum lateness in which job j precedes job i if jobs i and j are adjacent.

Proof: Consider the two sequences described earlier. For the jobs in positions τ and $\tau + 1$ for the two sequences π_1 and π_2 we have,

$$\delta_{[\tau]}(\pi_1) = ST_{[\tau-1],1}(\pi_1) + s_{i,1} + t_{i,1} - ST_{[\tau-1],2}(\pi_1) - s_{i,2}, \tag{8}$$

$$\delta_{[\tau]}(\pi_2) = ST_{[\tau-1],1}(\pi_2) + s_{j,1} + t_{j,1} - ST_{[\tau-1],2}(\pi_2) - s_{j,2}, \tag{9}$$

$$\delta_{[\tau+1]}(\pi_1) = ST_{[\tau-1],1}(\pi_1) + s_{i,1} + t_{i,1} + s_{j,1} + t_{j,1} - ST_{[\tau-1],2}(\pi_1) - s_{i,2} - t_{i,2} - s_{j,2}, \tag{10}$$

$$\delta_{[\tau+1]}(\pi_2) = ST_{[\tau-1],1}(\pi_2) + s_{j,1} + t_{j,1} + s_{i,1} + t_{i,1} - ST_{[\tau-1],2}(\pi_2) - s_{j,2} - t_{j,2} - s_{i,2}. \tag{11}$$

Observe that $ST_{[\tau-1],1}(\pi_1) = ST_{[\tau-1],1}(\pi_2)$ and $ST_{[\tau-1],2}(\pi_1) = ST_{[\tau-1],2}(\pi_2)$ since both sequences have the same jobs in all the positions prior to τ.

By the hypothesis of (iii) and equations (8) and (9),

$$\delta_{[\tau]}(\pi_2) \leq \delta_{[\tau]}(\pi_1). \tag{12}$$

If $s_{i,1} + t_{i,1} \leq s_{i,2} + t_{j,2}$, then by equations (9) and (11),

$$\delta_{[\tau+1]}(\pi_2) \leq \delta_{[\tau]}(\pi_2), \tag{13}$$

and if $s_{j,1} + t_{j,1} \leq s_{j,2} + t_{j,2}$, then by equations (8) and (11),

$$\delta_{[\tau+1]}(\pi_2) \leq \delta_{[\tau]}(\pi_1). \tag{14}$$

Hence, it follows by equation (12) and the hypothesis of (iv) (i.e., if either equation (13) or (14) holds) that

$$\max\{IT_{[\tau-1],2}(\pi_2), \delta_{[\tau]}(\pi_2), \delta_{[\tau+1]}(\pi_2)\} \leq \max\{IT_{[\tau-1],2}(\pi_1), \delta_{[\tau]}(\pi_1), \delta_{[\tau+1]}(\pi_1)\}. \tag{15}$$

The ϕ values for the jobs in positions τ and $\tau + 1$ for the two sequences π_1 and π_2 are given as

$$\begin{aligned}\phi_{[\tau]}(\pi_1) = \max\{IT_{[\tau-1],2}(\pi_1), \delta_{[\tau]}(\pi_1)\} + ST_{[\tau-1],2}(\pi_1) \\ + s_{i,2} + t_{i,2} - ST_{[\tau-1],3}(\pi_1) - s_{i,3},\end{aligned} \tag{16}$$

$$\phi_{[\tau]}(\pi_2) = \max\{IT_{[\tau-1],2}(\pi_2), \delta_{[\tau]}(\pi_2)\} + ST_{[\tau-1],2}(\pi_2) + s_{j,2} + t_{j,2} - ST_{[\tau-1],3}(\pi_2) - s_{j,3}, \tag{17}$$

$$\phi_{[\tau+1]}(\pi_1) = \max\{IT_{[\tau-1],2}(\pi_1), \delta_{[\tau]}(\pi_1), \delta_{[\tau+1]}(\pi_1)\} + ST_{[\tau-1],2}(\pi_1) + s_{i,2} + t_{i,2} + s_{j,2} + t_{j,2} - ST_{[\tau-1],3}(\pi_1) - s_{i,3} - t_{i,3} - s_{j,3}, \tag{18}$$

$$\phi_{[\tau+1]}(\pi_2) = \max\{IT_{[\tau-1],2}(\pi_2), \delta_{[\tau]}(\pi_2), \delta_{[\tau+1]}(\pi_2)\} + ST_{[\tau-1],2}(\pi_2) + s_{j,2} + t_{j,2} + s_{i,2} + t_{i,2} - ST_{[\tau-1],3}(\pi_2) - s_{j,3} - t_{j,3} - s_{i,3}. \tag{19}$$

Remember that both sequences have the same jobs in positions 1 through $\tau - 1$, therefore, $ST_{[\tau-1],2}(\pi_1) = ST_{[\tau-1],2}(\pi_2)$, $ST_{[\tau-1],3}(\pi_1) = ST_{[\tau-1],3}(\pi_2)$, and $IT_{[\tau-1],2}(\pi_1) = IT_{[\tau-1],2}(\pi_2)$.

It follows from equations (12), (16), (17) and the hypothesis of (ii) that

$$\phi_{[\tau]}(\pi_2) \le \phi_{[\tau]}(\pi_1). \tag{20}$$

From equations (8), (9), (11), (16) and (19),

$$\phi_{[\tau+1]}(\pi_2) \le \phi_{[\tau]}(\pi_1) \tag{21}$$

if $s_{j,2} + t_{j,2} \le s_{j,3} + t_{j,3}$ and if either $s_{i,1} + t_{i,1} \le s_{i,2} + t_{j,2}$ (for the case $\delta_{[\tau+1]}(\pi_2) \le \delta_{[\tau]}(\pi_2)$) or $s_{j,1} + t_{j,1} \le s_{j,2} + t_{j,2}$ (for the case $\delta_{[\tau+1]}(\pi_2) \le \delta_{[\tau]}(\pi_1)$).

From equations (9), (11), (17) and (19),

$$\phi_{[\tau+1]}(\pi_2) \le \phi_{[\tau]}(\pi_2) \tag{22}$$

if $s_{i,2} + t_{i,2} \le s_{i,3} + t_{j,3}$ and if $s_{i,1} + t_{i,1} \le s_{i,2} + t_{j,2}$ (for the case $\delta_{[\tau+1]}(\pi_2) \le \delta_{[\tau]}(\pi_2)$).

Clearly if equation (20) and either equations (21) or (22) hold then

$$\max\{IT_{[\tau-1],3}(\pi_2), \phi_{[\tau]}(\pi_2), \phi_{[\tau+1]}(\pi_2)\} \le \max\{IT_{[\tau-1],3}(\pi_1), \phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\}. \tag{23}$$

It can easily be shown that if the conditions in which equation (20) holds, the conditions in which either equations (21) or (22) holds are satisfied, then

$$\max\{IT_{[\tau-1],2}(\pi_2), \delta_{[\tau]}(\pi_2), \delta_{[\tau+1]}(\pi_2)\} \le \max\{IT_{[\tau-1],2}(\pi_1), \delta_{[\tau]}(\pi_1), \delta_{[\tau+1]}(\pi_1)\}. \tag{24}$$

For sequences π_1 and π_2, the latenesses of the jobs in positions τ and $\tau + 1$ on the third machine are given as

$$L_{[\tau]}(\pi_1) = ST_{[\tau-1],3}(\pi_1) + s_{i,3} + t_{i,3} + \max\{IT_{[\tau-1],3}(\pi_1), \phi_{[\tau]}(\pi_1)\} - d_i, \tag{25}$$

$$L_{[\tau]}(\pi_2) = ST_{[\tau-1],3}(\pi_2) + s_{j,3} + t_{j,3} + \max\{IT_{[\tau-1],3}(\pi_2), \phi_{[\tau]}(\pi_2)\} - d_j, \tag{26}$$

$$L_{[\tau+1]}(\pi_1) = ST_{[\tau-1],3}(\pi_1) + s_{i,3} + t_{i,3} + s_{j,3} + t_{j,3} + \max\{IT_{[\tau-1],3}(\pi_1), \phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\} - d_j, \tag{27}$$

$$L_{[\tau+1]}(\pi_2) = ST_{[\tau-1],3}(\pi_2) + s_{j,3} + t_{j,3} + s_{i,3} + t_{i,3} + \max\{IT_{[\tau-1],3}(\pi_2), \phi_{[\tau]}(\pi_2), \phi_{[\tau+1]}(\pi_2)\} - d_i. \tag{28}$$

From equations (26) and (27),

$$L_{[\tau]}(\pi_2) - L_{[\tau+1]}(\pi_1) = -s_{i,3} - t_{i,3} + \max\{IT_{[\tau-1],3}(\pi_2), \phi_{[\tau]}(\pi_2)\} - \max\{IT_{[\tau-1],3}(\pi_1), \phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\}.$$

It follows from equation (23) that

$$\max\{IT_{[\tau-1],3}(\pi_2), \phi_{[\tau]}(\pi_2)\} \leq \max\{IT_{[\tau-1],3}(\pi_1), \phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\}.$$

Therefore,

$$L_{[\tau]}(\pi_2) \leq L_{[\tau+1]}(\pi_1). \tag{29}$$

From equations (27) and (28),

$$L_{[\tau+1]}(\pi_2) - L_{[\tau+1]}(\pi_1) = d_j - d_i + \max\{IT_{[\tau-1],3}(\pi_2), \phi_{[\tau]}(\pi_2), \phi_{[\tau+1]}(\pi_2)\} - \max\{IT_{[\tau-1],3}(\pi_1), \phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\}.$$

It follows from equation (23) that

$$\max\{IT_{[\tau-1],3}(\pi_2), \phi_{[\tau]}(\pi_2), \phi_{[\tau+1]}(\pi_2)\} \leq \max\{IT_{[\tau-1],3}(\pi_1), \phi_{[\tau]}(\pi_1), \phi_{[\tau+1]}(\pi_1)\}.$$

Moreover, $d_j \leq d_i$. Therefore,

$$L_{[\tau+1]}(\pi_2) \leq L_{[\tau+1]}(\pi_1). \tag{30}$$

Hence, from equations (29) and (30),

$$\max\{L_{[\tau]}(\pi_2), L_{[\tau+1]}(\pi_2)\} \leq \max\{L_{[\tau]}(\pi_1), L_{[\tau+1]}(\pi_1)\}. \tag{31}$$

Thus, from equations (23) and (31), and Lemmas 3 and 4, $L_{\max}(\pi_2) \leq L_{\max}(\pi_1)$. This concludes the proof.

4 Heuristics

We consider three heuristics for the problem. One is the EDD heuristic where jobs are arranged based on the increasing order of their due dates. We also propose another version of EDD, we call it EEDD (Enhanced Earliest Due Date), where sum of the setup and processing times on all the three machines are also taken into account. In EEDD, we first calculate ED_i for each job i, where

$$ED_i = \frac{d_i}{\sum_{k=1}^{3}(s_{i,k} + t_{i,k})}.$$

Then, the EEDD heuristic is obtained by arranging the jobs in increasing order of ED_i. While EDD gives higher priority to jobs with smaller d_i's, EEDD gives higher priority to jobs with not only smaller d_i's but also to jobs with smaller sum of setup and processing times. The third heuristic that we consider is a PGA (Polynomial-based Genetic Algorithm).

Genetic algorithms have been used for the scheduling problems by many researchers including Ruiz and Maroto (2005, 2006), and Tavakkoli-Moghaddam et al. (2005). Our proposed PGA considers a population (*POP*) of given sequences, generated randomly and selects two sequences out of *POP* as parents to produce two offsprings. These two offsprings are produced by swapping subsequences of equal length among two parents. Care must be taken that both offsprings are feasible schedule. To understand this process, consider the following two sequences of X and Y where $X = \{x_1, x_2, \ldots, x_i, \ldots, x_j, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_i, \ldots, y_j, \ldots, y_n\}$. The two segments of $x_i, \ldots, x_n$ and $y_i, \ldots, y_n$ are said to be compatible if they include the same subset of jobs, but not necessarily in the same order. Two sequences X and Y are called compatible, if they have two compatible segments. The process of generating offsprings from a given population is repeated *CP* times.

The process of generating the offsprings is repeated for a given number of generations (*GEN*). Then, y schedules of the population (*POP*) are replaced with the best y schedules from the set of offspring schedules. At the same time, each schedule in the population is mutated with a known probability p, where two randomly selected jobs are interchanged. At the end of the given number of generations, a schedule with best value of maximum lateness is accepted as the heuristic solution.

The steps of the PGA are as follows.

Step 1: Initialise a population, *POP*, of random sequences.

Step 2: Compute the $L_{\max}$ of each sequence in *POP*.

Step 3: Order the sequences in *POP* according to $L_{\max}$ from the best to the worst.

Step 4: Repeat Steps (i)–(v) for *GEN* times

(i) Repeat Steps (a)–(d) for *CP* times
  - (a) Randomly choose two different compatible parents to crossover
  - (b) Select compatible segments in the two parents
  - (c) Swap the segments
  - (d) Save the new sequences in *CHILD* and compute $L_{\max}$ of each.

(ii) Order *CHILD* with respect to $L_{\max}$.

(iii) Replace the worst y sequences of *POP* with the best y sequences in *CHILD*.

(iv) Mutate each sequence in *POP* with the probability p.

(v) Compute $L_{\max}$ and order *POP*.

Step 5: Store the best solution from *POP* as the heuristic solution.

It should be noted that the two parents that are used to perform the crossover operation are scanned from left to right. We stop at the earliest position where we can do a swap. That is, the scan process continues until all the positions in both sequences (parents) contain the same set of jobs, not necessarily in the same order. It should also be noted that if y is less than the total number of offsprings, then the remaining offsprings are omitted. On the other hand, if y is greater than the total number of offsprings, we adjust the value of y temporarily to the number of offsprings (this will allow more parents to go into the next generation).

Careful setting of the parameters for our proposed genetic algorithm is essential to achieve a good performance. This is done experimentally. To do so, various parameter settings were tested with the following ranges: *POP*, *GEN* and *CP* from n to $5n$ with the increment of n; y from 1/6 to 5/6 with the increment of 1/6; and p from 0.005 to 0.1 with the increment of 0.005. After an extensive computational analysis, the parameters are set as given in Table 1. Note that according to these parameters, it can easily be shown that the complexity of the PGA is $O(n^3)$.

Table 1 Parameters of the hybrid genetic algorithm

Parameter	*Value*
POP	$2n$
GEN	n
CP	$2n$
y	1/3
p	0.035

5 Computational experiments

In this section, the effectiveness of the proposed heuristics is investigated on randomly generated problem instances. The heuristics were implemented using C programming language on a Sun Sparc 20. The processing and setup times were randomly generated from discrete uniform distributions with $t_{i,j}$ from [1, 100] and $s_{i,j}$ from [0, 100k]. The parameter k is the expected ratio of setup time to processing time ($s_{i,j}/t_{i,j}$).

Due dates are generated from a discrete uniform distribution in a range (PX, PY) where P is set to

$$P = \sum_{i=1}^{n} (s_{i,3} + t_{i,3}) + \min_{i=1}^{n}(s_{i,1} + t_{i,1}) + \min_{i=1}^{n}(s_{i,2} + t_{i,2})$$

which is a lower bound on the makespan. The parameters X and Y are defined as $X = (1–T–R/2)$ and $Y = (1–T + R/2)$, where R is called due date range whereas T is called tardiness factor. This method of generating due dates is common in the literature for different combinations of T and R values, e.g., see Volgenant and Teerhuis (1999), Kim (1993, 1995) and Chu (1992).

Problem data are generated for different number of jobs for the range of 30–80 jobs in an increment of 10. The data are generated for different combinations of k, T and R values (k = 0.1, 0.4, 0.7; T = 0.3, 0.6, 0.9 and R = 0.3, 0.6, 0.9), which results in 27 combinations. Twenty-five replicates are randomly generated for each combination.

We compare the performance of the heuristics using three measures: average percentage error (Error), standard deviation (Std) and the percentage of the number of the best solutions (NOB). The percentage error is defined as $100 \times (L_{\max}(\text{heuristic}) - L_{\max}(\text{best}))/(L_{\max}(\text{worst}) - L_{\max}(\text{best}))$. According to this definition, the worst heuristic will have an error of 100 and the best will have an error of zero. For comparison purposes, we evaluate the performance of EDD, EEDD and PGA against that of a random sequence.

The computational results are summarised in Tables 2–4 for different problem sizes. The error for random sequence was 100% for 150 cases and it was more than 96% for the remaining 12 cases. The overall percentage errors of the heuristics EDD, EEDD and PGA were 3.35, 2.78 and 0.01, respectively, while it was 99.63 for the random sequence. It is clear that all the three heuristics performed very well. Owing to large values of the error for the random sequence, it will not be included in tables and figures, and will not be discussed in the rest of the analysis.

Table 2 Computational results for $n = 30$ and $n = 40$

				EDD			*EEDD*			*PGA*		
n	*k*	*T*	*R*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*
30	0.1	0.3	0.3	48	4.918	0.080	48	3.931	0.061	100	0.000	0.000
			0.6	36	4.126	0.051	44	4.419	0.058	100	0.000	0.000
			0.9	32	5.115	0.064	40	3.039	0.039	100	0.000	0.000
		0.6	0.3	8	9.052	0.173	24	6.406	0.120	96	0.047	0.002
			0.6	40	6.820	0.105	48	4.834	0.101	100	0.000	0.000
			0.9	40	8.465	0.137	32	8.304	0.151	100	0.000	0.000
		0.9	0.3	44	3.393	0.045	56	3.659	0.069	100	0.000	0.000
			0.6	40	3.651	0.048	44	5.753	0.109	100	0.000	0.000
			0.9	32	4.397	0.054	48	7.658	0.174	96	0.136	0.006
	0.4	0.3	0.3	56	8.388	0.187	52	2.782	0.054	100	0.000	0.000
			0.6	52	3.881	0.072	52	3.998	0.078	100	0.000	0.000
			0.9	44	4.200	0.066	48	3.980	0.060	100	0.000	0.000
		0.6	0.3	28	7.761	0.113	36	5.055	0.058	100	0.000	0.000
			0.6	52	6.673	0.200	52	4.716	0.092	100	0.000	0.000
			0.9	48	3.026	0.061	40	3.439	0.051	100	0.000	0.000
		0.9	0.3	64	3.484	0.066	52	8.143	0.203	100	0.000	0.000
			0.6	64	2.554	0.047	68	1.937	0.040	100	0.000	0.000
			0.9	72	4.073	0.083	84	1.569	0.042	100	0.000	0.000
	0.7	0.3	0.3	68	2.402	0.048	80	1.131	0.028	100	0.000	0.000
			0.6	60	3.045	0.066	76	1.651	0.036	100	0.000	0.000
			0.9	56	2.547	0.037	48	4.567	0.057	100	0.000	0.000
		0.6	0.3	44	3.341	0.046	52	4.799	0.097	100	0.000	0.000
			0.6	72	2.939	0.059	60	2.201	0.041	100	0.000	0.000
			0.9	48	8.502	0.199	48	6.316	0.125	100	0.000	0.000
		0.9	0.3	72	1.500	0.032	80	1.097	0.029	100	0.000	0.000
			0.6	72	5.491	0.201	64	3.166	0.069	100	0.000	0.000
			0.9	72	3.232	0.063	64	4.012	0.081	100	0.000	0.000
40	0.1	0.3	0.3	48	3.664	0.046	60	2.091	0.038	100	0.000	0.000
			0.6	36	4.658	0.063	44	3.963	0.073	100	0.000	0.000
			0.9	32	3.415	0.057	36	2.966	0.066	96	0.044	0.002
		0.6	0.3	52	6.161	0.200	56	6.437	0.204	100	0.000	0.000
			0.6	28	5.753	0.071	32	3.888	0.053	100	0.000	0.000
			0.9	28	4.296	0.061	32	2.549	0.030	96	0.106	0.005

Table 2 Computational results for *n* = 30 and *n* = 40 (continued)

n	*k*	*T*	*R*	*EDD*			*EEDD*			*PGA*		
				NOB	*Error*	*Std*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*
		0.9	0.3	40	8.320	0.208	44	12.138	0.286	100	0.000	0.000
			0.6	48	6.617	0.143	56	3.391	0.053	100	0.000	0.000
			0.9	36	4.913	0.070	32	5.147	0.074	100	0.000	0.000
	0.4	0.3	0.3	44	3.182	0.042	72	1.060	0.021	96	0.044	0.002
			0.6	44	4.350	0.073	44	3.720	0.038	100	0.000	0.000
			0.9	44	3.062	0.045	44	4.951	0.080	100	0.000	0.000
		0.6	0.3	40	5.941	0.102	40	2.941	0.053	100	0.000	0.000
			0.6	40	2.739	0.038	44	2.797	0.036	100	0.000	0.000
			0.9	44	2.444	0.026	44	3.159	0.069	100	0.000	0.000
		0.9	0.3	56	4.369	0.067	52	2.712	0.043	100	0.000	0.000
			0.6	36	9.078	0.210	48	9.996	0.216	100	0.000	0.000
			0.9	60	1.726	0.033	68	2.072	0.044	100	0.000	0.000
	0.7	0.3	0.3	48	2.938	0.047	56	2.905	0.049	100	0.000	0.000
			0.6	64	2.202	0.061	64	1.702	0.029	100	0.000	0.000
			0.9	52	3.380	0.050	48	3.417	0.044	100	0.000	0.000
		0.6	0.3	48	4.063	0.082	40	4.214	0.098	100	0.000	0.000
			0.6	48	2.634	0.050	56	2.594	0.049	100	0.000	0.000
			0.9	64	3.550	0.072	48	3.806	0.070	96	0.010	0.000
		0.9	0.3	64	4.768	0.104	76	0.901	0.018	100	0.000	0.000
			0.6	76	1.352	0.037	80	2.679	0.086	100	0.000	0.000
			0.9	76	1.945	0.052	80	0.909	0.026	96	0.080	0.004

Table 3 Computational results for *n* = 50 and *n* = 60

n	*k*	*T*	*R*	*EDD*			*EEDD*			*PGA*		
				NOB	*Error*	*Std*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*
50	0.1	0.3	0.3	52	1.993	0.033	72	1.015	0.022	100	0.000	0.000
			0.6	44	3.018	0.053	20	3.465	0.043	100	0.000	0.000
			0.9	36	6.079	0.146	24	5.570	0.144	100	0.000	0.000
		0.6	0.3	32	3.803	0.053	44	2.691	0.035	100	0.000	0.000
			0.6	24	4.182	0.053	40	2.310	0.033	100	0.000	0.000
			0.9	24	4.933	0.072	24	4.100	0.055	100	0.000	0.000
		0.9	0.3	48	5.066	0.059	52	3.573	0.061	96	0.123	0.006
			0.6	40	4.700	0.059	52	2.528	0.037	100	0.000	0.000
			0.9	40	7.330	0.199	48	6.098	0.199	100	0.000	0.000
	0.4	0.3	0.3	56	4.733	0.102	72	5.517	0.199	100	0.000	0.000
			0.6	48	2.389	0.050	60	1.824	0.047	96	0.010	0.000
			0.9	32	3.051	0.035	36	2.380	0.027	96	0.010	0.000
		0.6	0.3	64	1.705	0.039	64	0.914	0.018	100	0.000	0.000
			0.6	48	4.425	0.061	44	2.778	0.037	100	0.000	0.000
			0.9	20	4.506	0.073	48	1.921	0.031	100	0.000	0.000

Table 3 Computational results for *n* = 50 and *n* = 60 (continued)

				EDD			*EEDD*			*PGA*		
n	*k*	*T*	*R*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*
		0.9	0.3	56	3.030	0.074	72	1.915	0.040	100	0.000	0.000
			0.6	72	2.281	0.067	72	2.009	0.042	100	0.000	0.000
			0.9	64	1.925	0.034	72	1.483	0.032	100	0.000	0.000
	0.7	0.3	0.3	48	4.138	0.081	52	3.156	0.046	100	0.000	0.000
			0.6	36	2.793	0.046	44	3.681	0.067	100	0.000	0.000
			0.9	56	1.393	0.025	72	1.803	0.041	92	0.165	0.005
		0.6	0.3	64	2.252	0.039	48	3.715	0.065	100	0.000	0.000
			0.6	48	5.511	0.103	48	2.414	0.053	96	0.040	0.002
			0.9	48	1.902	0.026	48	2.529	0.044	96	0.073	0.003
		0.9	0.3	52	4.123	0.068	60	2.669	0.051	100	0.000	0.000
			0.6	64	2.707	0.061	64	2.805	0.081	100	0.000	0.000
			0.9	64	2.110	0.038	68	4.806	0.123	100	0.000	0.000
60	0.1	0.3	0.3	44	3.454	0.051	52	2.224	0.042	100	0.000	0.000
			0.6	48	1.678	0.026	56	0.820	0.016	96	0.056	0.002
			0.9	32	2.007	0.024	36	2.921	0.069	100	0.000	0.000
		0.6	0.3	40	2.030	0.023	28	2.323	0.020	100	0.000	0.000
			0.6	24	3.122	0.042	40	2.460	0.027	100	0.000	0.000
			0.9	52	2.526	0.041	48	2.483	0.034	100	0.000	0.000
		0.9	0.3	52	2.725	0.043	68	1.004	0.023	100	0.000	0.000
			0.6	48	1.797	0.029	48	1.325	0.015	100	0.000	0.000
			0.9	48	6.332	0.197	56	5.346	0.199	100	0.000	0.000
	0.4	0.3	0.3	40	3.035	0.084	64	2.073	0.041	100	0.000	0.000
			0.6	36	4.473	0.063	40	2.319	0.028	100	0.000	0.000
			0.9	44	2.770	0.036	44	3.022	0.046	100	0.000	0.000
		0.6	0.3	72	1.479	0.033	64	1.738	0.044	100	0.000	0.000
			0.6	52	1.781	0.027	60	1.942	0.038	96	0.034	0.001
			0.9	44	1.975	0.027	44	1.872	0.031	100	0.000	0.000
		0.9	0.3	48	3.492	0.054	52	3.810	0.083	100	0.000	0.000
			0.6	44	1.611	0.019	52	1.587	0.024	100	0.000	0.000
			0.9	68	1.291	0.022	72	0.907	0.018	100	0.000	0.000
	0.7	0.3	0.3	44	1.572	0.021	56	2.160	0.040	100	0.000	0.000
			0.6	44	2.569	0.033	60	1.217	0.024	96	0.018	0.000
			0.9	48	2.978	0.044	32	4.188	0.048	100	0.000	0.000
		0.6	0.3	64	2.492	0.049	72	1.304	0.034	100	0.000	0.000
			0.6	52	1.196	0.016	60	1.636	0.044	100	0.000	0.000
			0.9	60	1.224	0.025	72	0.610	0.016	100	0.000	0.000
		0.9	0.3	72	1.625	0.036	72	1.018	0.022	100	0.000	0.000
			0.6	68	1.079	0.022	76	1.025	0.027	100	0.000	0.000
			0.9	60	1.700	0.033	68	1.486	0.028	100	0.000	0.000

Table 4 Computational results for n = 70 and n = 80

				EDD			*EEDD*			*PGA*		
n	*k*	*T*	*R*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*
70	0.1	0.3	0.3	36	4.297	0.054	48	2.393	0.029	100	0.000	0.000
			0.6	32	3.723	0.046	36	2.356	0.036	100	0.000	0.000
			0.9	32	3.994	0.064	52	1.602	0.023	96	0.164	0.008
		0.6	0.3	24	7.271	0.200	40	3.456	0.070	100	0.000	0.000
			0.6	16	7.363	0.197	24	2.883	0.035	100	0.000	0.000
			0.9	44	2.067	0.038	48	1.599	0.035	100	0.000	0.000
		0.9	0.3	32	4.521	0.064	48	2.248	0.035	100	0.000	0.000
			0.6	24	4.518	0.052	52	2.054	0.031	100	0.000	0.000
			0.9	56	1.776	0.033	56	1.274	0.026	100	0.000	0.000
	0.4	0.3	0.3	56	2.995	0.074	48	3.170	0.055	100	0.000	0.000
			0.6	52	1.408	0.022	56	2.495	0.040	96	0.075	0.003
			0.9	36	2.465	0.038	52	1.925	0.033	100	0.000	0.000
		0.6	0.3	64	4.831	0.199	72	4.552	0.199	96	0.014	0.000
			0.6	60	2.433	0.048	60	2.167	0.041	100	0.000	0.000
			0.9	48	1.536	0.024	48	1.335	0.020	100	0.000	0.000
		0.9	0.3	52	2.855	0.045	64	1.720	0.040	100	0.000	0.000
			0.6	56	2.031	0.037	60	0.925	0.013	100	0.000	0.000
			0.9	52	2.266	0.039	52	1.291	0.025	100	0.000	0.000
	0.7	0.3	0.3	64	2.097	0.054	68	1.366	0.024	100	0.000	0.000
			0.6	68	2.188	0.056	64	1.091	0.020	100	0.000	0.000
			0.9	56	2.162	0.037	60	1.521	0.027	96	0.012	0.000
		0.6	0.3	72	0.526	0.014	68	2.102	0.046	100	0.000	0.000
			0.6	44	1.899	0.031	48	2.488	0.046	96	0.057	0.002
			0.9	48	2.899	0.048	44	4.093	0.072	96	0.009	0.000
		0.9	0.3	60	3.222	0.079	56	1.951	0.033	100	0.000	0.000
			0.6	72	3.268	0.105	88	0.527	0.015	100	0.000	0.000
			0.9	64	1.360	0.026	76	1.133	0.029	100	0.000	0.000
80	0.1	0.3	0.3	36	2.209	0.030	48	1.404	0.026	100	0.000	0.000
			0.6	28	3.010	0.041	32	1.248	0.016	100	0.000	0.000
			0.9	12	2.616	0.029	32	1.676	0.024	96	0.013	0.000
		0.6	0.3	32	2.131	0.032	40	1.428	0.022	100	0.000	0.000
			0.6	24	3.780	0.047	32	3.062	0.051	100	0.000	0.000
			0.9	28	4.973	0.107	32	4.313	0.142	100	0.000	0.000
		0.9	0.3	24	2.986	0.045	40	2.475	0.042	100	0.000	0.000
			0.6	64	1.774	0.028	52	1.129	0.014	100	0.000	0.000

Table 4 Computational results for n = 70 and n = 80 (continued)

				EDD			*EEDD*			*PGA*		
n	*k*	*T*	*R*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*	*NOB*	*Error*	*Std*
			0.9	44	2.273	0.032	56	1.191	0.017	96	0.012	0.000
	0.4	0.3	0.3	56	1.278	0.022	52	1.620	0.028	100	0.000	0.000
			0.6	28	2.803	0.047	60	1.324	0.023	100	0.000	0.000
			0.9	40	2.734	0.040	44	1.668	0.028	100	0.000	0.000
		0.6	0.3	40	4.292	0.061	44	3.283	0.058	100	0.000	0.000
			0.6	44	1.956	0.035	40	2.010	0.031	96	0.044	0.002
			0.9	52	1.580	0.027	52	1.583	0.029	100	0.000	0.000
		0.9	0.3	60	1.051	0.022	64	1.808	0.035	100	0.000	0.000
			0.6	48	1.899	0.044	68	2.048	0.064	100	0.000	0.000
			0.9	64	2.346	0.059	64	2.025	0.056	100	0.000	0.000
	0.7	0.3	0.3	52	1.503	0.022	48	2.596	0.051	100	0.000	0.000
			0.6	48	2.174	0.033	56	0.917	0.015	96	0.009	0.000
			0.9	48	1.919	0.026	52	0.959	0.015	96	0.113	0.005
		0.6	0.3	76	1.383	0.046	88	0.357	0.011	100	0.000	0.000
			0.6	56	1.356	0.025	64	0.907	0.015	100	0.000	0.000
			0.9	68	1.172	0.032	60	1.568	0.028	96	0.005	0.000
		0.9	0.3	84	0.560	0.015	76	2.408	0.099	100	0.000	0.000
			0.6	60	2.303	0.043	76	1.066	0.022	100	0.000	0.000
			0.9	52	4.385	0.109	64	2.389	0.060	96	0.038	0.001

The results of Tables 2–4 are summarised in Figures 1–3. Figure 1 indicates the overall percentage errors of the heuristics with respect to the number of jobs where each point represents the average of 675 instances, which correspond to 25 randomly generated replicates for each of 27 combinations of k, R and T values. Figures 2 and 3 represent the standard deviation and the number of best solution performances for the same set of instances of the three heuristics. It is clear from the figures that the percentage errors of EDD and EEDD slightly decrease as n increases. The same is also true for Std values as shown in Figure 2. It is also clear from Figures 1–3 that EEDD performs better than EDD. The overall error of EDD is 1.2 times that of EEDD while both have almost the same computational time. Therefore, EEDD is preferable to EDD. The figures also clearly indicate that PGA significantly outperforms EEDD and EDD. Moreover, the overall error of EEDD is 278 times that of PGA. It should be noted that the computational time of PGA was less than half a minute for 50 jobs, and for the extreme case of 80 jobs, it was less than five minutes. Therefore, if the computational time is of main concern, then it is recommended to use PGA for n up to 50 while to use EEDD for larger number of jobs if an error of 2.78 is tolerable.

Figure 1 Error vs. number of jobs

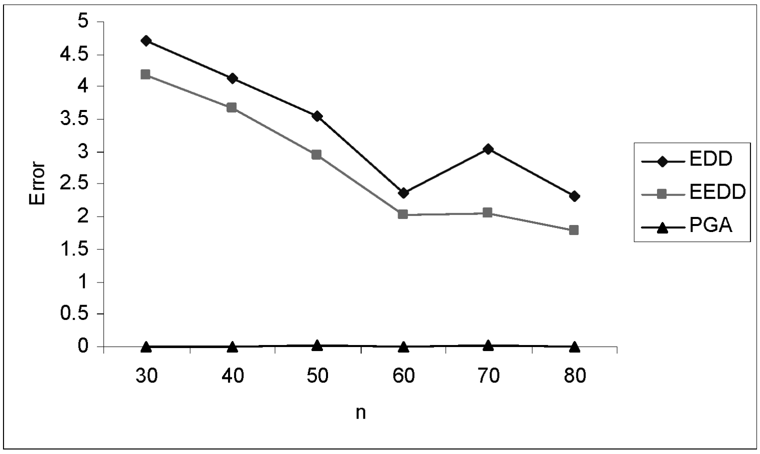


Figure 2 Std vs. number of jobs

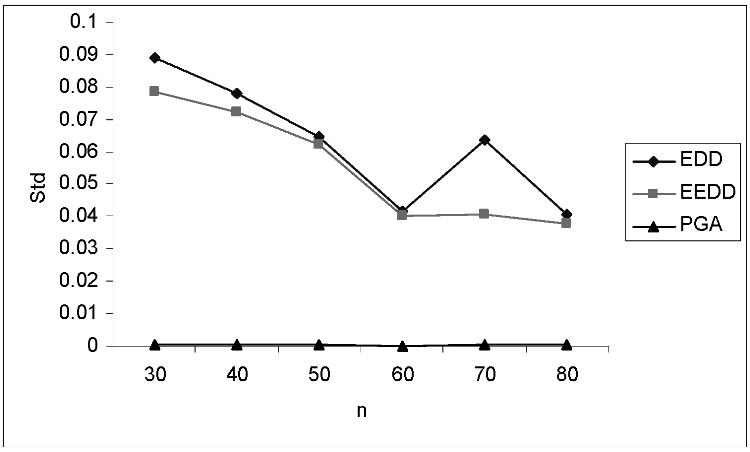
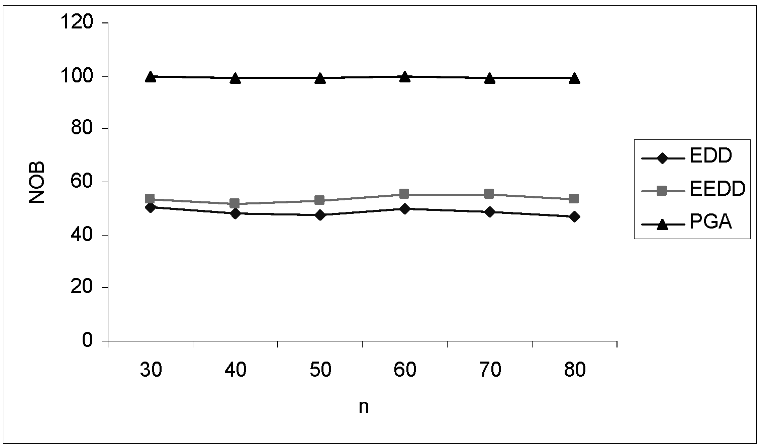


Figure 3 NOB vs. number of jobs



Figures 4–6 examine the error performance of the three heuristics for the three different values of *k*, *T* and *R*. Figure 4 indicates that the performances of both EDD and EEDD slightly improve as the *k* value increases. On the other hand, the performance of PGA was not sensitive to the *k* values. Figure 5 shows that the performances of the heuristics are not sensitive to *T* values. It should be noted that the larger the value of *R*, the wider will be the spread of due dates among the jobs. In this case, the problem becomes relatively easy to solve because of large due dates. On the other hand, the problem becomes difficult to solve for smaller values of *R*, since in this case due dates are smaller. It is clear from Figure 6 that the performances of EDD and EEDD slightly deteriorate as *R* gets smaller. Although it is not clear from Figure 6 owing to scale used, the performance of PGA gets slightly better as *R* gets smaller.

Figure 4 Heuristics' sensitivity to *k*

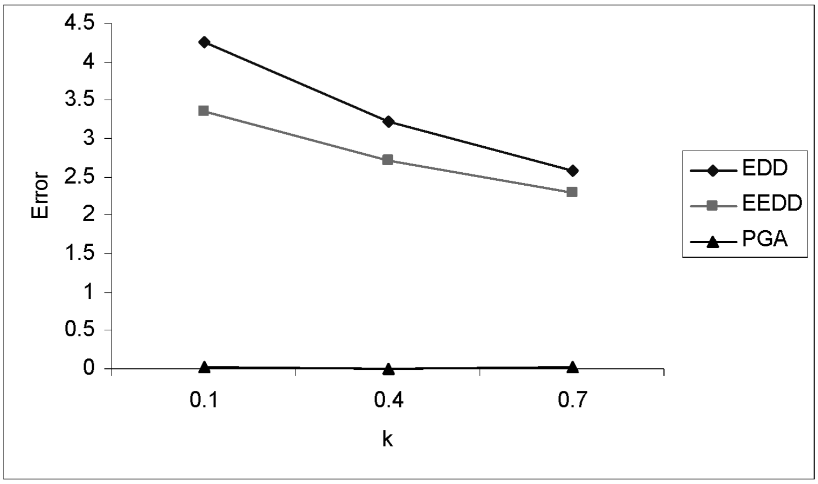


Figure 5 Heuristics' sensitivity to *T*

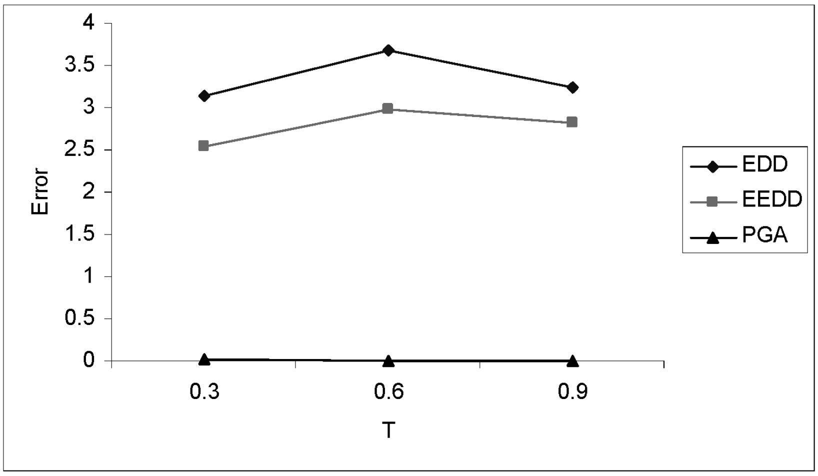
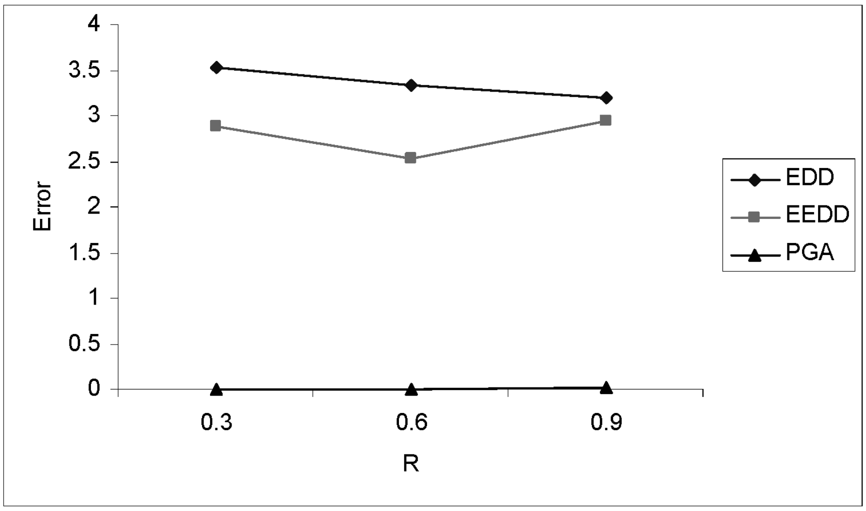


Figure 6 Heuristics' sensitivity to *R*



6 Conclusions

The literature reveals that the flowshop scheduling problem with maximum lateness criterion and separate setup times has been limited to the two-machine case. This paper extends this problem to the three-machine case. The contribution of the paper includes a dominance relation and three heuristics (EDD, EEDD and PGA). Computational experiments are conducted to compare the performances of the proposed heuristics. The computational analysis shows that PGA significantly outperforms EDD and EEDD. If the computational time is of the main concern and the number of jobs are large, EEDD can be an acceptable alternative to PGA.

Evaluation of the dominance relation developed in this paper has not been performed. Dominance relations are usually helpful when used in a branch-and-bound algorithm. Therefore, a possible extension is to develop a branch-and-bound algorithm for the problem by incorporating the dominance relation developed in this paper. Moreover, the performance of a branch-and-bound is also affected by the upper bound used in the algorithm. Hence, the proposed PGA can be used as an upper bound in the branch-and-bound algorithm to be developed since PGA performs very well.

Setup times considered in this paper are assumed to be sequence independent. There are some environments where setup times are sequence-independent. However, there are many other environments where it is not valid to assume that setup times as sequence independent. Therefore, another possible area to study is to consider the same problem with sequence-dependent case. It may not be that easy to come up with some dominance relations for the sequence-dependent case. However, some new heuristics can be developed and compared with the ones proposed in this paper.

Acknowledgements

This research was initiated when Professor Allahverdi was invited to Hong Kong. The research was supported in part by The Hong Kong Polytechnic University under grant number A628 from the Area of Strategic Development in China Business Services. The authors would like to thank the referees for their helpful comments.

References

Al-Anzi, F.S. and Allahverdi, A. (2001) 'The relationship between three-tiered client-server internet database connectivity and two-machine flowshop', *International Journal of Parallel and Distributed Systems and Networks*, Vol. 4, pp.94–101.

Allahverdi, A. (1995) 'Two-stage production scheduling with separated setup times and stochastic breakdowns', *Journal of the Operational Research Society*, Vol. 46, pp.896–904.

Allahverdi, A. (2000) 'Minimizing mean flowtime in a two-machine flowshop with sequence independent setup times', *Computers and Operations Research*, Vol. 27, pp.111–127.

Allahverdi, A. and Al-Anzi, F.S. (2002) 'Using two-machine flowshop with maximum lateness objective to model multimedia data objects scheduling problem for WWW applications', *Computers and Operations Research*, Vol. 29, pp.971–994.

Allahverdi, A. and Al-Anzi, F.S. (2006) 'A branch-and-bound algorithm for three-machine flowshop scheduling problem to minimize total completion time with separate setup times', *European Journal of Operational Research*, Vol. 169, pp.767–780.

Allahverdi, A., Gupta, J.N.D. and Aldowaisan, T.A. (1999) 'A review of scheduling research involving setup considerations', *OMEGA The International Journal of Management Sciences*, Vol. 27, pp.219–239.

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).

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*, Vol. 2, pp.47–57.

Bianco, L., Ricciardelli, S., Rinaldi, G. and Sassano, A. (1988) 'Scheduling tasks with sequence-dependent processing times', *Naval Research Logistics*, Vol. 35, pp.177–184.

Bitran, G.R. and Gilbert, S.M. (1990) 'Sequencing production on parallel machines with two magnitudes of sequence-dependent setup cost', *Journal of Manufacturing and Operations Management*, Vol. 3, pp.24–52.

Cheng, T.C.E., Janiak, A. and Kovalyov, M.Y. (2001) 'Single machine batch scheduling with resource dependent setup and processing times', *European Journal of Operational Research*, Vol. 135, pp.177–183.

Chu, C. (1992) 'A branch-and-bound algorithm to minimize total tardiness with different release dates', *Naval Research Logistics*, Vol. 39, pp.265–283.

Dileepan, P. and Sen, T. (1991) 'Job lateness in a two-machine flowshop with setup times separated', *Computers and Operations Research*, Vol. 18, pp.549–556.

Elmasri, R. and Navathe, S.B. (1999) *Fundamentals of Database Systems*, 3rd ed., Addison-Wesley, New York.

Kim, S.C. and Bobrowski, P.M. (1994) 'Impact of sequence-dependent setup-times on job shop scheduling performance', *International Journal of Production Research*, Vol. 32, pp.1503–1520.

Kim, Y.D. (1993) 'A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops', *Computers and Operations Research*, Vol. 20, pp.391–401.

Kim, Y.D. (1995) 'Minimizing total tardiness in permutation flowshops', *European Journal of Operational Research*, Vol. 85, pp.541–555.

Lin, B.M.T. and Jeng, A.A.K. (2004) 'Parallel-machine batch scheduling to minimize the maximum lateness and the number of tardy jobs', *International Journal of Production Economics*, Vol. 91, pp.121–134.

Pan, J.C.H., Chen, J.S. and Cheng, H.L. (2001) 'A heuristic approach for single-machine scheduling with due dates and class setups', *Computers and Operations Research*, Vol. 28, pp.1111–1130.

Rios-Mercado, R. and Bard, J.F. (1998) 'Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups', *Computers and Operations Research*, Vol. 25, pp.351–366.

Rios-Mercado, R. and Bard, J.F. (1999) 'Enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times', *Journal of Heuristics*, Vol. 5, pp.53–70.

Ruiz, R. and Maroto, C. (2005) 'A comprehensive review and evaluation of permutation flowshop heuristics', *European Journal of Operational Research*, Vol. 165, pp.479–494.

Ruiz, R. and Maroto, C. (2006) 'A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility', *European Journal of Operational Research*, Vol. 169, pp.781–800.

Tavakkoli-Moghaddam, R., Aryanezhad, M.B., Safaei, N. and Azaron, A. (2005) 'Solving a dynamic cell formation problem using metaheuristics', *Applied Mathematics and Computation*, Vol. 170, pp.761–780.

Uzsoy, R., Lee, C. and Martin-Vega, A. (1992) 'Scheduling semiconductor test operations: minimizing maximum lateness and number of tardy jobs on a single machine', *Naval Research Logistics*, Vol. 39, pp.369–388.

Volgenant, A. and Teerhuis, E. (1999) 'Improved heuristics for the *n*-job single machine weighted tardiness problem', *Computers and Operations Research*, Vol. 26, pp.35–44.

Yoshida, T. and Hitomi K. (1979) 'Optimal two-stage production scheduling with setup times separated', *AIIE Transactions*, Vol. 11, pp.261–263.