Su, Ling-Huey; Hsiao, Ming-Chih

Article

Two-agent scheduling in open shops subject to machine availability and eligibility constraints

Journal of Industrial Engineering and Management (JIEM)

Provided in Cooperation with:
The School of Industrial, Aerospace and Audiovisual Engineering of Terrassa (ESEIAAT), Universitat Politècnica de Catalunya (UPC)

Suggested Citation: Su, Ling-Huey; Hsiao, Ming-Chih (2015) : Two-agent scheduling in open shops subject to machine availability and eligibility constraints, Journal of Industrial Engineering and Management (JIEM), ISSN 2013-0953, OmniaScience, Barcelona, Vol. 8, Iss. 4, pp. 1103-1124, https://doi.org/10.3926/jiem.1352

This Version is available at:
https://hdl.handle.net/10419/188726

OmniaScience

Journal of Industrial Engineering and Management
JIEM, 2015 – 8(4): 1103-1124 – Online ISSN: 2013-0953 – Print ISSN: 2013-8423
http://dx.doi.org/10.3926/jiem.1352

Two-agent Scheduling in Open Shops Subject to Machine Availability and Eligibility Constraints

Ling-Huey Su, Ming-Chih Hsiao

Chung Yuan Christian University (Taiwan)

linghuey@cycu.edu.tw, Miller_Hsiao@aseglobal.com

Received: December 2014
Accepted: July 2015

Abstract:

Purpose: The aims of this article are to develop a new mathematical formulation and a new heuristic for the problem of preemptive two-agent scheduling in open shops subject to machine maintenance and eligibility constraints.

Design/methodology: Using the ideas of minimum cost flow network and constraint programming, a heuristic and a network based linear programming are proposed to solve the problem.

Findings: Computational experiments show that the heuristic generates a good quality schedule with a deviation of 0.25% on average from the optimum and the network based linear programming model can solve problems up to 110 jobs combined with 10 machines without considering the constraint that each operation can be processed on at most one machine at a time. In order to satisfy this constraint, a time consuming Constraint Programming is proposed. For $n = 80$ and $m = 10$, the average execution time for the combined models (linear programming model combined with Constraint programming) exceeds two hours. Therefore, the heuristic algorithm we developed is very efficient and is in need.

Practical implications: Its practical implication occurs in TFT-LCD and E-paper manufacturing wherein units go through a series of diagnostic tests that do not have to be performed in any specified order.

Originality/value: The main contribution of the article is to split the time horizon into many time intervals and use the dispatching rule for each time interval in the heuristic algorithm, and also to combine the minimum cost flow network with the Constraint Programming to solve the problem optimally.

Keywords: scheduling, two-agent, open shop, machine availability and eligibility, constraint programming

1. Introduction

Consider two agents who have to schedule two sets of jobs on an *m*-machine open shop. Each job has k operations, $k \leq m$, and each operation must be performed on the corresponding specialized machine. The order in which the operations of each job are performed is irrelevant. Operation preemption is allowed and the machine availability and eligibility constraints are considered. The machine availability arises when machines are subject to breakdowns, maintenance, or perhaps high priority tasks are prescheduled in certain time intervals. The machine eligibility constraints are imposed when the number of operations of each job i can be less than m. Each machine can handle at most one operation at a time and each operation can be processed on at most one machine at a time. Two agents are called Agent A and B. Agents A and B have $n_a(n_b)$ jobs. Let n denote the total number of jobs, i.e., $n = n_a + n_b$, and each job j of agent A (B) is denoted by J_j^a (J_j^b). The processing time of a job j of agent A (B) on machine i is denoted by P_{ji}^a (P_{ji}^b). The release time of a job j of agent A (B) is denoted by r_j^a (r_j^b). Each machine i is available for processing in the given $N(i)$ intervals, which are $[b_i^k, f_i^k]$, $i = 1, ..., m$, $k = 1, ..., N(i)$ and $b_i^{k+1} > f_i^k$, where b_i^k and f_i^k are the start time and end time of the kth availability interval of machine i, respectively. The operations of job j of agent A (B) are processed on a specified subset M_j^a (M_j^b) of the machines in an arbitrary order. We use C_j^a (C_j^b) to denote the completion time of J_j for agent A (B) and $C_{\max} = \max\{\max_{1 \leq j \leq n_a} C_j^a, \max_{1 \leq j \leq n_b} C_j^b\}$ to denote the makespan. The objective is to minimize makespan, given that agent B will accept a schedule of cost up to Q. According to the notation for machine scheduling, the problem is denoted as $O, NC_{win} \left| pmtn^a, r_j^a, M_j^a : pmtn^b, r_j^b, M_j^b \right| C_{\max} : C_{\max}^b \leq Q$, where O indicates open machines, NC_{win} means that the machines are not available in certain time intervals, $pmtn^{a(b)}$ signifies job preemption, $r_j^{a(b)}$ implies that each job has a release date, $M_j^{a(b)}$ denotes the specific subset of machines to process job j, and $C_{\max}^b \leq Q$ denotes agent B will accept a schedule of time up to Q.

The multi-agent scheduling problems have received increasing attention recently. Baker and Smith (2003) perhaps the first to consider the problem in which two agents compete on the use of a single machine. They demonstrated that although determining a minimum cost schedule according to any of three criteria: makespan, minimizing maximum lateness, and minimizing total weighted completion time for a single machine, is polynomial, the problem of minimizing a mix of these criteria is NP-hard. Agnetis, Mirchandani, Pacciarelli and Pacifici (2004) studied a two-agent setting for a single machine, two-machine flowshop and two-machine open shop environments. The objective function value of the primary customer is minimized subject to the requirement that the objective function value of the second customer cannot exceed a given number. The objective functions are the maximum of regular functions, the number of late jobs, and the total weighted completion times. The problem in a similar two-agent single machine was further studied by Cheng, Ng and Yuan (2006, 2008), Ng, Cheng and Yuan (2006), Agnetis, Pacciarelli and Pacifici (2007), Agnetis, Pascale and Pacciarelli (2009), Leung, Pinedo and Wan (2010) and Lee, Chung and Huang (2013). When release times are further considered, see (Lee, Chung & Hu, 2012; Yin, Wu, Cheng and Wu, 2012; Yin, Wu, Cheng, Wu & Wu, 2014; Wu, Wu, Chen, Yin & Wu, 2013). The multi-agent problems are extended by considering variations in job processing time such as controllable processing time (Wan, Vakati, Leung & Pinedo, 2010), deteriorating job processing time (Cheng, Wu, Cheng and Wu, 2011; Liu, Yi & Zhou, 2011), and learning effect (Cheng, Cheng, Wu, Hsu and Wu, 2011; Lee & Hsu, 2012; Wu, Huang & Lee, 2011; Yin, Cheng and Wu, 2012). Yu, Zhang, Xu and Yin (2013) considered a two agents problem to minimize an aggregate increasing objection function of two agents' objective function and a piece-rate maintenance which is implemented once a fixed number of jobs is completed. Mor and Mosheiov (2010) considered minimizing the maximum earliness cost or total weighted earliness cost of one agent, subject to an upper bound on the maximum earliness cost of the other agent. They showed that both minimax and minsum cases are polynomially solvable while the weighted minsum case is NP-hard.

As for the preemptive open shop problem, Gonzalez and Sahni (1976) proposed a polynomial time algorithm to obtain the minimum makespan. Breit, Schmidt and Strusevich (2001) studied a two-machine open shop where one machine is not available for processing during a given time interval. The objective is to minimize the makespan. They showed that the problem is NP-hard and presented a heuristic with a worst-case ratio of 4/3. When time-windows is considered for each job on an open shop and the objective is to minimize makespan, Sedeno-Noda, Alcaide and Gonzalez-Martin (2006) introduced a network flow procedure to check feasibility and a max-flow parametrical algorithm to minimize the makespan. Sedeno-Noda, Pablo and Gonzalez-Martin (2009) extended the same problem by considering performance costs including resource and personnel involvement.

This two-agent problem under consideration arises in TFT-LCD and E-Paper manufacturing wherein units go through a series of diagnostic tests that do not have to be performed in any specified order.

This paper is organized as follows: Section 2 formulates the problem and provides a heuristic. Section 3 is devoted to describing the minimum cost flow network and its corresponding linear programming model. Section 4 illustrates the effectiveness of the heuristic algorithm and computational efficiency of the minimum cost flow network, and finally in Section 5, we provide conclusions.

2. Problem Formulation and the Heuristic Algorithm

First, we rank all r_j^a, r_j^b, b_i^k, f_i^k, and Q in nondecreasing order and put them into the time epoch set E. Let e_l be the lth time epoch in E, i.e., $E = \{e_1, e_2, e_3, \ldots, e_v\}$, where $e_l < e_{l+1}$ and $e_v = \max_{i=1,\ldots,m} f_i^{N(i)}$. Let T_l be the lth time interval between two adjacent time epochs e_l and e_{l+1}, where $l = 1, \ldots, |E|$-1. Let $m(l)$ be the set of machines that are available in T_l, i.e., $m(l) = \{i | \exists \text{integer } k,\ 1 \le k \le N(i),\ \text{such that } b_i^k \le e_l \text{ and } e_{l+1} \le f_i^k\}$.

Numerical Example 1. There are two jobs for each agent to be processed on three machines. The job and machine data are listed in Tables 1 and 2 and agent B will accept a schedule of time up to $Q = 17$.

Job (j)	J_1^a	J_2^a	J_1^b	J_2^b
Release time (r_j^x)	1	3	2	6
Processing time (P_{ji}^x)				
i = 1	4	5	3	4
i = 2	3	0	6	0
i = 3	0	2	1	5
Eligible subset (M_j^x)	{1,2}	{1,3}	{1,2,3}	{1,3}

Table 1. Job data

Machine (i)	i = 1	i = 2	i = 3
Availability interval ($[b_i^k, f_i^k]$)	$[b_1^1, f_1^1]$=[0,8]	$[b_2^1, f_2^1]$=[2,9]	$[b_3^1, f_3^1]$=[1,8]
	$[b_1^2, f_1^2]$=[9,20]	$[b_2^2, f_2^2]$=[10,20]	$[b_3^2, f_3^2]$=[10,18]

Table 2. Machine data

All values of r_j^x, b_i^k, f_i^k and Q are ranked in ascending order. The corresponding time epoch set $E = \{e_1, e_2, e_3, \ldots, e_{11}\} = \{0, 1, 2, 3, 6, 8, 9, 10, 17, 18, 20\}$, are shown in Figure 1.

We call operation O_{ji}^x for job j of agent x, $x = a$ or b, is *available* on its corresponding machine i in the time interval T_l if $r_j^x < e_l$, $JR_{ji}^x > 0$ and $i \in m(l)$, where JR_{ji}^x denotes the remaining unscheduled processing time of operation O_{ji}^x.

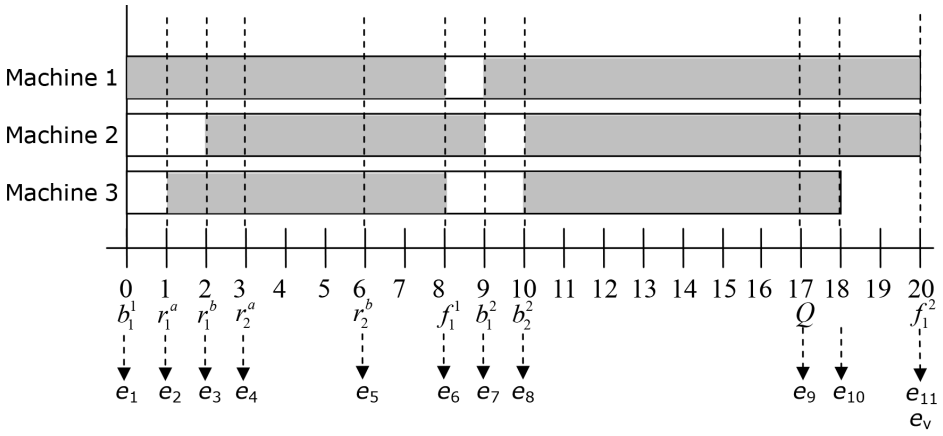


Figure 1. Distinct time epochs in set E

The basic idea of the heuristic is to combine the rule of giving priority to agent B and the rule of the largest remaining processing time first (LRPT) for assigning operations to machines in each time interval. The rationale of giving priority to agent B is that agent B only accepts a schedule of cost up to Q. The LRPT rule first selects job j of agent x, JR_j^x, which has the maximum remaining processing time and among the available operations of this job, choosing the one which is to be processed on the machine that has maximum remaining processing time to process. Within each time interval, say T_l, if the completion time y of the selected operation is less than the time epoch e_{l+1}, a new time epoch is set equal to y and a new time interval is created accordingly, i.e., T_l is split into two time intervals. That is, if $y < e_{l+1}$, set $e_{l+1} = y$, $e_{l+2} = e_{l+1}$, $e_{l+3} = e_{l+2}$, …, $e_{v+1} = e_v$. The steps of the heuristic algorithm are as follows.

Step 1. Rank r_j^x, b_i^k, f_i^k and Q in ascending order, and put into the time epoch E. Let $E = \{e_1, e_2, e_3, \ldots, e_v\}$, $e_v = \max_{i=1,\ldots,m} f_i^{N(i)}$. Set $l = 1$, $JR_j^x = \sum_{i \in M_j^x} P_{ji}^x$, and $JR_{ji}^x = P_{ji}^x$.

Step 2. If no operation is available to the time interval T_l, set $l = l + 1$ and return to Step 2. If any operation of agent *b* available to the time interval $T_l([e_l, e_{l+1}])$, set $x = b$, otherwise set $x = a$. Use the LRPT rule to find the operation O_{ji}^x of agent *x* to be processed on the machine, say machine *i*, in the time interval T_l. Set $y = \min(P_{ji}^x, T_l)$, $P_{ji}^x = P_{ji}^x - y$, $JR_j^x = JR_j^x - y$, $JR_{ji}^x = JR_{ji}^x - y$ and $r_j^x = e_l + y$. If $JR_j^x = 0$ for $\forall j = 1, 2, \ldots, n_x$, $x = a, b$; the heuristic algorithm is terminated and $C_{\max} = \max(r_j^x)$. If $r_j^x \neq e_{l+1}$, go to Step 3 to create a new interval; otherwise if $r_j^x = e_{l+1}$, then go to Step 2 until the available operations of agent x in this time interval T_l have been considered. If $x = b$, then continue the assignment of operations in the same manner for $x = a$ in time interval T_l. Set $l = l + 1$ and go to Step 2.

Step 3. Let $e_{l+1} = r_j^x$, $e_{l+2} = e_{l+1}$, $e_{l+3} = e_{l+2}$, ..., $e_{v+1} = e_v$, $E = \{e_{l+1}, e_{l+2}, \ldots, e_{v+1,}\}$, and go to Step 2.

We demonstrate the heuristic algorithm using the data of Numerical Example 1.

Numerical Example 2. The values of the total remaining unscheduled processing time of job *j* for both agents are $JR_1^a = 7$, $JR_2^a = 7$, $JR_1^b = 10$ and $JR_2^b = 9$.

Iteration 1

Step 1. The corresponding time epoch set $E = \{e_1, e_2, e_3, \ldots, e_{11}\} = \{0, 1, 2, 3, 6, 8, 9, 10, 17, 18, 20\}$ is shown in Figure 1. Set $l = 1$.

Step 2. Since there is no available operation in $[e_1, e_2]$, set $l = 2$. In $[e_2, e_3]$, no operation for agent *B* is available and thus O_{11}^a is selected to be processed on m_1 due to the maximum value of JR_1^a and the maximum machine load of m_1. Since $y = 1$, update the values of P_{11}^a, JR_1^a and r_1^a as 3, 6 and 2, respectively. Since there is no more available operation in the time interval $[e_2, e_3]$, we set $l = 3$. Using the same manner, the time interval $[e_3, e_4]$, $[e_4, e_5]$, $[e_5, e_6]$, $[e_6, e_7]$ and $[e_7, e_8]$ are scheduled. In the time interval $[e_8, e_9]$, the available jobs for agent *B* are $\{J_1^b, J_2^b\}$ and J_2^b is selected to be processed on m_3. Since $y = 3$, update the values of P_{23}^b, JR_2^b and r_2^b as 0, 3 and 13, respectively. Since $r_2^b \notin E$, therefore return to Step 3.

Step 3. Let e_9 equal to $r_2^b = 13$ and re-index time epochs in *E*. That is $[e_{10}, e_9]$, $[e_{11}, e_{10}]$, $[e_{12}, e_{11}]$, $E = \{13, 17, 18, 20\}$ and go to Step 2.

Iteration 2

Step 2. In $[e_8, e_9]$, J_1^b is selected to be processed on m_2, however, r_1^b is not in set E, therefore go to Step 3.

Step 3. Let e_9 equal to $r_1^b = 11$, $E = \{11, 13, 17, 18, 20\}$, and return to Step 2.

Proceed in the same manner, the heuristic solution with $C_{\max} = 18$ is obtained and shown in Figure 2.

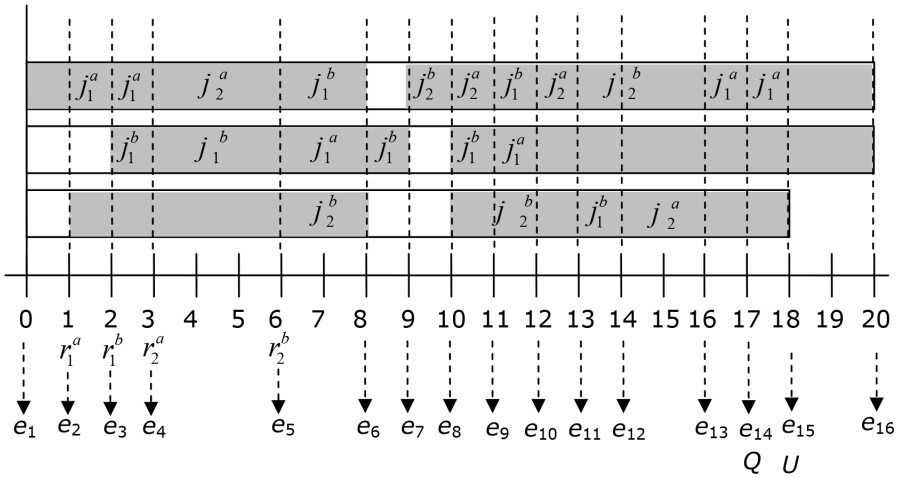


Figure 2. Gantt diagram for heuristic solution

3. The Exact Algorithm

In this section, we develop a linear programming model based on minimum cost flow network to test the effectiveness of the heuristic. For the problem of the parallel-machine scheduling problem, $P,\ NC_{win}|pmtn,\ r_j,\ M_j|C_{\max}$, Liao and Sheen (2008) solved the problem by first formulating a base problem with respect to a given $C_{\max}$ as a maximum flow problem. They then proposed a binary search algorithm to test the feasibility of the problem and to determine the optimal makespan by solving a series of maximum network flow problems. Su, Cheng and Chou (2013) revised their maximum flow model to the minimum cost flow model for the same problem with makespan and maximum lateness criteria. We further extend their network for the two-agent open shop problem to test the effectiveness of our heuristic algorithm.

First, we set U, the solution of the heuristic algorithm, as an upper bound on the makespan. Next, we rank all r_j^a, r_j^b, b_i^k, f_i^k, Q and U in increasing order and put them into the time epoch set E as previously described. $E = \{e_1, e_2, e_3, \ldots, e_v\}$, where $e_l < e_{l+1}$ and $e_v = U$. The time epochs in the time interval from U to $\max_{i=1,\ldots,m} f_i^{N(i)}$ are redundant and can be discarded. Using the data of the Numerical Example 1, the diagram of the time epoch set E and the time intervals T_l is shown in Figure 3.

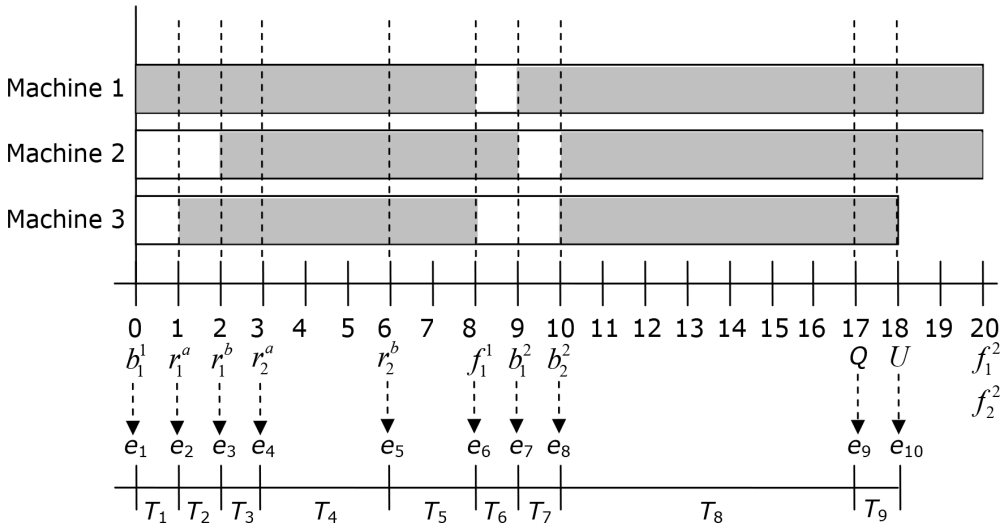


Figure 3. Distinct time epochs in set *E* for Numerical Example 1

Upon obtaining E, T_l and $m(l)$, we formulate the problem with U as a minimum cost flow network problem on a tripartite network $G(U)$.

The network consists of a set of nodes and a set of arcs connecting certain pairs of the nodes.

The node set in $G(U)$ includes a source node s, a sink node t, and the following three node sets.

(i) Job nodes, $J_1^x, J_2^x, ..., J_j^x$, $j = 1, 2, \ldots, n_x$, $x = a, b$.

(ii) Combination nodes (J_j^x, T_l) consisting of job J_j^x, $j = 1, 2, \ldots, n_x$, $x = a, b$ and time interval T_l, where T_l satisfies $e_{l+1} \leq U$ and $r_j^a \leq e_l$ for agent A and $e_{l+1} \leq Q$ and $r_j^b \leq e_l$ for agent B.

(iii) Combination nodes (i, T_l) consisting of machine i and time interval T_l, where T_l satisfies $e_{l+1} \leq U$ and $i \in m(l)$.

The arc set consists of directed arcs that are generated as follows.

(i) An arc (s, J_j^x) with capacity $\sum_{i \in M_j^x}^{m} p_{ji}^x$ for $j = 1, 2, \ldots, n_x$, $x = a, b$.

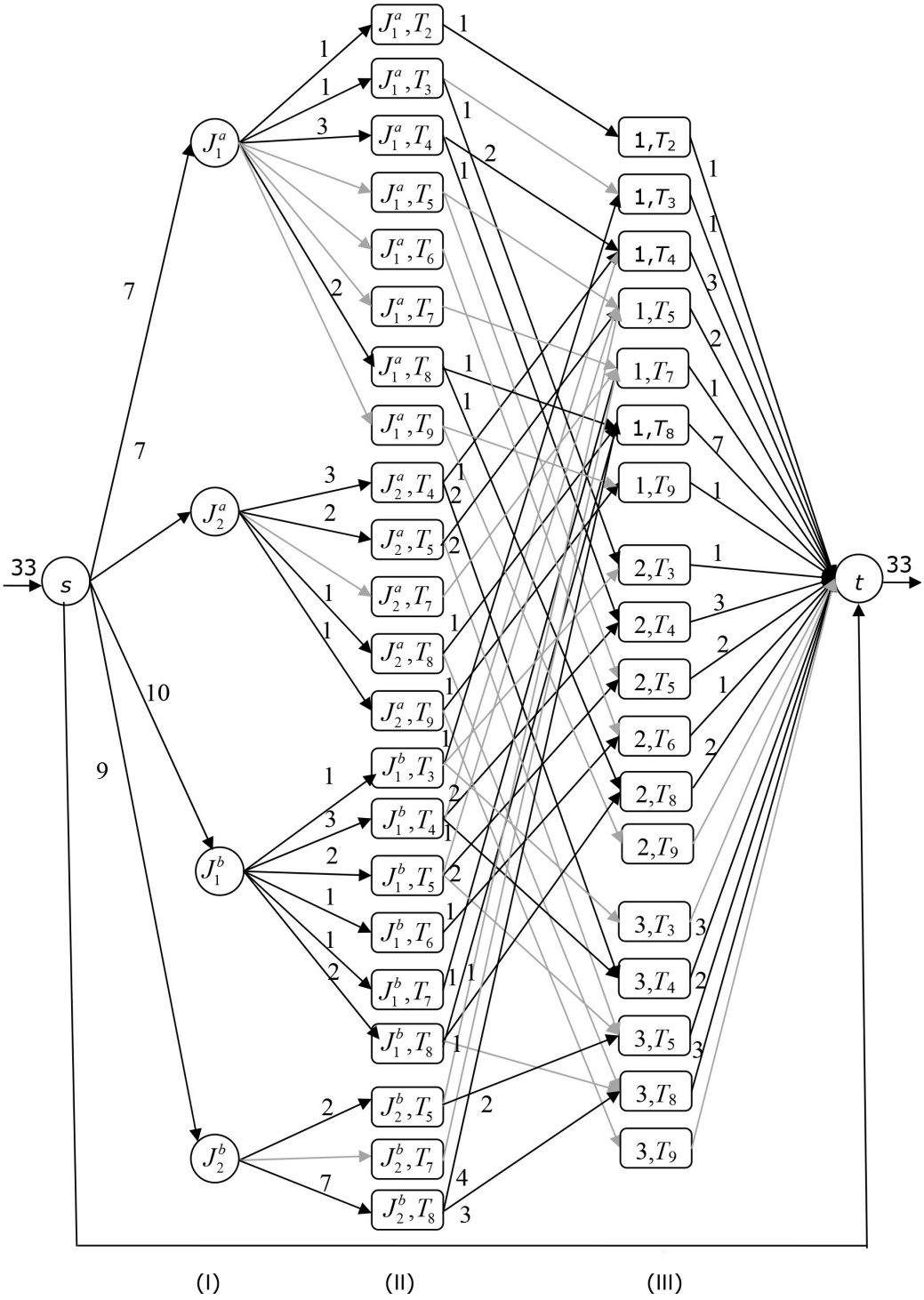
(ii) An arc $(J_j^x, (J_j^x, T_l))$ from node J_j^x to node (J_j^x, T_l) with capacity $(e_{l+1} - e_l)$.

(iii) An arc $((J_j^x, T_l), (i, T_l))$ from node (J_j^x, T_l) to (i, T_l) with capacity $(e_{l+1} - e_l)$ if and only if $i \in M_j^x \cap m(l)$.

(iv) An arc $((i, T_l), t)$ from node (i, T_l) to the sink t with capacity $(e_{l+1} - e_l)$, $i = 1, 2, \ldots, m$.

(v) Finally, there is an arc (s, t) from node s to node t with capacity ∞.

The source node s consists of $n + 1$ emanating arcs. Let $f(x, y)$ denote the flow in the arc from node x to node y, i.e. node x and node y are a pair of nodes connecting this arc. A positive $f(s, t)$ indicates that an amount of flow equal to $f(s, t)$ cannot be assigned to the machines under the upper bound U, i.e. there exists no feasible schedule in which the completion time of each job is no greater than U. This happens if U is a trial value which is incorrectly set or the heuristic algorithm is incorrectly calculated. Denote the latest arrival times of agent A and B and the time epochs Q and U as e_u, e_w, e_z and e_v, respectively.



The corresponding directed tripartite network $G(U)$ using the data of Numerical Example 1 is shown in Figure 4. Figure 4. The optimal solution for Numerical Example 1

Using the convention that summations are taken only over existing arcs, we formulate $O,\ NC_{win}|pmtn^a,\ r_j^a,\ M_j^a : pmtn^b,\ r_j^b,\ M_j^b|C_{\max} : C_{\max}^b \le Q$ as a linear programming problem, called LP, as follows:

Minimize

$$M \times f(s,t) + \varepsilon \times \{ \sum_{h=1}^{v-\max(u,w)-1} \sum_{i \in m(\max(u,w)+h)} h \times f((i, T_{\max(u,w)+h}), t) \} \quad (1)$$

Subject to

$$\sum_{j=1}^{n_a} f(s, J_j^a) + \sum_{j=1}^{n_b} f(s, J_j^b) + f(s,t) = \sum_{j=1}^{n_a} \sum_{i=1}^{m} p_{ji}^a + \sum_{j=1}^{n_b} \sum_{i=1}^{m} p_{ji}^b \quad (2)$$

$$\begin{aligned} f(s, J_j^a) &= \sum_{l=1}^{v-1} f(J_j^a, (J_j^a, T_l)), \quad j = 1,2,...,n_a \\ f(s, J_j^b) &= \sum_{l=1}^{v-1} f(J_j^b, (J_j^b, T_l)), \quad j = 1,2,...,n_b \end{aligned} \quad (3)$$

$$\begin{aligned} f(J_j^a, (J_j^a, T_l)) &= \sum_{i \in S_j^a \cap m(l)} f((J_j^a, T_l), (i, T_l)), \text{ for } \forall (J_j^a, T_l) \text{ node} \\ f(J_j^b, (J_j^b, T_l)) &= \sum_{i \in S_j^b \cap m(l)} f((J_j^b, T_l), (i, T_l)), \text{ for } \forall (J_j^b, T_l) \text{ node} \end{aligned} \quad (4)$$

$$\sum_{j=1}^{n_a} f((J_j^a, T_l), (i, T_l)) + \sum_{j=1}^{n_b} f((J_j^b, T_l), (i, T_l)) = f((i, T_l), t), \text{ for } \forall (i, T_l) \text{ node} \quad (5)$$

$$\sum_{i=1}^{m} \sum_{l=1}^{v-1} f((i, T_l), t) + f(s,t) = \sum_{j=1}^{n_a} \sum_{i=1}^{m} p_{ji}^a + \sum_{j=1}^{n_b} \sum_{i=1}^{m} p_{ji}^b \quad (6)$$

$$\begin{aligned} \sum_{l=1}^{v-1} f((J_j^a, T_l), (i, T_l)) &\le p_{ji}^a, \text{ for } j = 1,2,...,n_a, \quad i \in S_j^a \cap m(l) \\ \sum_{l=1}^{z-1} f((J_j^b, T_l), (i, T_l)) &\le p_{ji}^b, \text{ for } j = 1,2,...,n_b, \quad i \in S_j^b \cap m(l) \end{aligned} \quad (7)$$

$$\begin{aligned} f(s, J_j^a) &\le \sum_{i=1}^{m} p_{ji}^a, \quad j = 1,2,...,n_a \\ f(s, J_j^b) &\le \sum_{i=1}^{m} p_{ji}^b, \quad j = 1,2,...,n_b \end{aligned} \quad (8)$$

$$\begin{aligned} f(J_j^a, (J_j^a, T_l)) &\le (e_{l+1} - e_l), \quad j = 1,2,...,n_a, \quad l = 1,2,...,v-1 \\ f(J_j^b, (J_j^b, T_l)) &\le (e_{l+1} - e_l), \quad j = 1,2,...,n_b, \quad l = 1,2,...,z-1 \end{aligned} \quad (9)$$

$$\begin{aligned} f((J_j^a, T_l), (i, T_l)) &\le (e_{l+1} - e_l), \quad j = 1,2,...,n_a, \quad l = 1,2,...,v-1, \quad i = 1,2,...,m \\ f((J_j^b, T_l), (i, T_l)) &\le (e_{l+1} - e_l), \quad j = 1,2,...,n_b, \quad l = 1,2,...,z-1, \quad i = 1,2,...,m \end{aligned} \quad (10)$$

$$f((i, T_l), t) \le (e_{l+1} - e_l), \quad l = 1,2,...,v-1, \quad i = 1,2,...,m \quad (11)$$

and all variables ≥ 0.

In the first term of the objective Function (1), we assign an arbitrarily large cost M to the arc $f(s, t)$ to ensure that the maximum feasible flow goes through all the other arcs (s, J_j^x), $j = 1, 2, \ldots, n_x$, $x = a, b$. The summations in the latter terms ensure the assignment of jobs to time intervals as early as possible, where ε denotes an arbitrarily small number. As an illustration, refer to Numerical Example 1 and Figure 4. The latter term of the objective Function (1) for Numerical Example 1 is $\varepsilon \times \{\sum_{h=1}^{4} \sum_{i \in m(5+h)} h \times f((i, T_{5+h}), t)\}$, in which the cost is given to the time intervals from T_6 to T_9. The time epochs from the last arrival time to the upper bound of $C_{\max}^*$ is e_5 to e_{10}, associated with time intervals from T_5 to T_9. Since e_5 is the last arrival time and some operations will definitely be scheduled in T_5, therefore the cost is imposed to the time interval from T_6 to T_9. We give cost ε, 2ε, 3ε and 4ε to the time interval T_6, T_7, T_8 and T_9, respectively, to ensure the assignment of operations to these time intervals as early as possible. Constraints (2) guarantee that the total processing times of all operations of both agents should be assigned to machines in different time intervals. Constraints (3)-(6) are the mass balance constraints. Constraints (7)-(11) represent the capacity constraints.

If $f(s, t)$ is zero, then the optimal makespan is $C_{\max}^* = e_l + \max_{i \in m(l)} f((i, T_l), t)$, where $l = \{k \mid \max_{k=\max(u,w),\ldots,v-1} (f(i, T_k), t)$ such that $f((i, T_k), t) > 0$ for $i = 1, 2, \ldots, m\}$. For illustration, as Figure 4 shows, the basic variables of the optimal solution are depicted by the thick lines. $l = \{k \mid \max_{k=5,\ldots,9} (f(i, T_k), t)$ such that $f((i, T_k), t) > 0$ for $i = 1, 2, 3\} = 9$, and $C_{\max}^* = e_l + \max_{i \in m(l)} f((i, T_l), t) = e_9 + 1 = 17 + 1 = 18$.

The linear programming model with $1 + n \times v + m \times (n + 1) \times (v - 1)$ variables, including n for arcs (s, J_j^x), $n \times (v - 1)$ for arcs $(J_j^x, (J_j^x, T_l))$, $m \times (v - 1)$ for arcs $((i, T_l), t)$, and one for arc (s, t), and $3 + 2 \times v\,(n + m) + m \times (n \times v - 2)$ constraints for all functional constraints from (2) to (11).

The corresponding optimal operations in each time interval for Numerical Example 1 are shown in Figure 5.

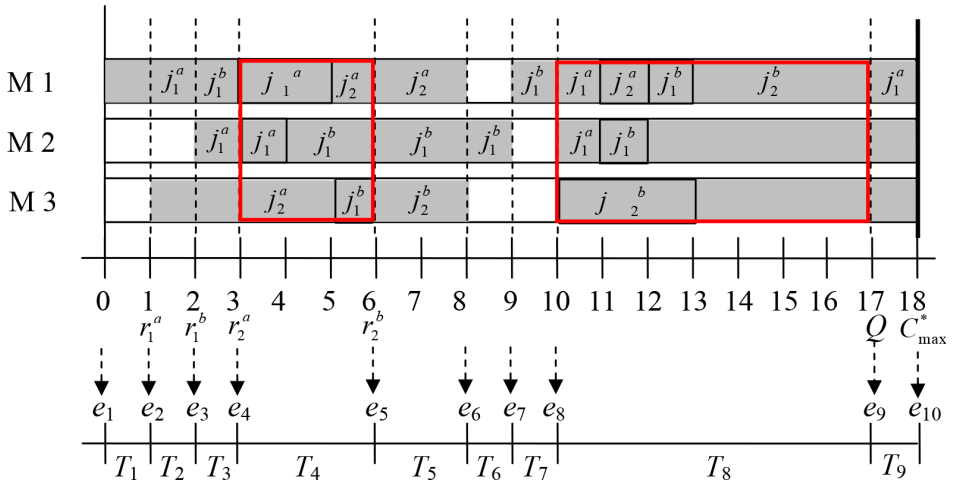


Figure 5. Gantt diagram of the optimal solution for Example 1

As Figure 5 shows, operation J_1^a is processed on machines 1 and 2, operation J_2^a is processed on machines 1 and 3, and operation J_1^b is processed on machine 2 and 3 in time interval T_4. In order to satisfy the constraint that each operation can be processed on at most one machine at a time, we use Constraint Programming (Brailsford, Potts & Smith, 1999; Van Hentenryck, 2002) to solve this problem. Constraint Programming generates a schedule that satisfies the above constraint in each time interval. The model in Figure 6 is coded for this purpose.

```
Model:
        int Interval=...;
        int nbJobs=...;
        int nbMachines=...;
        range Jobs=1..nbJobs;
        range Machines=1..nbMachines;
        range intervals=1..Interval; // number of unit time in the current interval
        int resource[Jobs, Machines]=...; // total processing time of each jobs on each
        // machine in the current interval
        dvar int X[Machines, intervals] in 0.. 3; // input the number of unit time in the
        // current interval, for example, the unit time of T4 in the example is 3
        subject to {
        // each operation cannot be processed on more than one machine at a time
        forall(a in Machines, b in Machines, k in intervals:a!=b)
        X[a,k]!=0 => X[a,k]!=X[b,k];
        // total processing times of each job to be processed on each machine equal to
        // the corresponding time obtained by the LP model
        forall(j in Jobs, i in Machines)
        sum(k in intervals)(X[i,k]==j) == resource[j,i];
        }
```

Figure 6. The CPL model for rescheduling the operations of each job

We coded the algorithm in ILOG OPL Studio 6.1 and ran it on a PC with AMD 2.91GHz. The Gantt diagram of the optimal solution is shown in Figure 7, where the time interval T_8 is implemented using the same model.

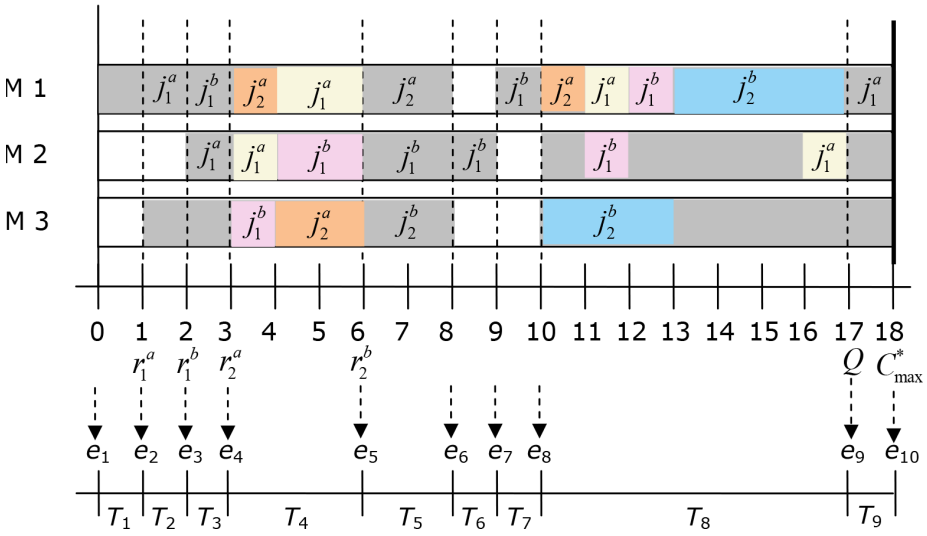


Figure 7. Gantt diagram of the optimal solution for Example 1

4. Computational Results

The objective of the computational experiments described in this section is to evaluate both the performances of the heuristic and the exact algorithms. All experimental tests were run on a personal computer with AMD 2.91GHz CPU. The heuristic algorithm was coded in Visual Basic, and the linear programming model and the constraint programming model were solved by LINGO11.0 and ILOG OPL 6.1, respectively. The experiment involves the instances with the number of jobs $n = 40$, 60, 80, 100 and 110 in which three pairs of n_a and n_b are set as

$$\{(n_a = 20, n_b = 20), (n_a = 25, n_b = 15), \text{and } (n_a = 15, n_b = 25)\},$$

$$\{(n_a = 30, n_b = 30), (n_a = 35, n_b = 25), \text{and } (n_a = 25, n_b = 35)\},$$

$$\{(n_a = 40, n_b = 40), (n_a = 45, n_b = 35), \text{and } (n_a = 35, n_b = 45)\},$$

$$\{(n_a = 50, n_b = 50), (n_a = 60, n_b = 40), \text{and } (n_a = 40, n_b = 60)\}, \text{ and}$$

$$\{(n_a = 55, n_b = 55), (n_a = 60, n_b = 50), \text{and } (n_a = 50, n_b = 60)\}, \text{ respectively.}$$

The processing time of p_{ji}^{x} are randomly generated from an uniform distribution $U[0, 10]$. The job arrival time r_j^x refers to Chu (1992) and are generated from $U[0, \overline{P}]$, where $\overline{P} = (\sum_{i=1}^{m}\sum_{j=1}^{n_a} p_{ji}^{a} + \sum_{i=1}^{m}\sum_{j=1}^{n_b} p_{ji}^{b}) \Big/ m$ is the mean processing time on each machine. The number of machine m is set as $m = 6$, 8 and 10. To generate the upper bound of makespan for agent b, we refer to Bank and Werner (2001) with

$$Q = \beta \left[\frac{n}{m} \left(p + \frac{\sum_{j=1}^{n_a} r_j^a + \sum_{j=1}^{n_b} r_j^b}{n} \right) \right] \frac{n_b}{n},$$

where $\overline{P} = \dfrac{\sum_{i=1}^{m}\sum_{j=1}^{n_a} p_{ji}^{a} + \sum_{i=1}^{m}\sum_{j=1}^{n_b} p_{ji}^{b}}{n}$ and $\beta \in \{1.2,\ 1.5,\ 2,\ 2.5\}$. The rate of machine availability θ is set as 0.7, 0.9 and 1.0. Five instances are generated for each combination of n, m, p_{ji}^{x}, r_j^x, θ, Q, yielding 900 instances. In comparing heuristic performance, the following formula is used to determine the deviation of the heuristic solution over the optimal solution. Deviation (%) = [(heuristic-optimum)/optimum] × 100%.

Table 3 shows the solution quality of the heuristic. The influences of m, n, n_a/n_b and θ on the solution quality of the heuristic are analyzed.

			40			60			80			100			110		
			na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb	na,nb
			20,20	25,15	15,25	30,30	35,25	25,35	40,40	45,35	35,45	50,50	60,40	40,60	55,55	60,50	50,60
θ	m	β	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)	Dev(%)
0.7	6	1.2	0.00%	4.78%	0.28%	0.31%	0.00%	0.00%	0.00%	0.00%	0.13%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		1.5	0.00%	4.52%	0.00%	0.00%	0.36%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2	0.39%	0.48%	0.90%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	8	1.2	0.62%	-	0.78%	2.04%	0.18%	0.00%	0.00%	0.73%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		1.5	0.00%	-	0.00%	0.00%	0.57%	0.81%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2	0.00%	0.65%	0.00%	0.00%	0.00%	0.51%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	10	1.2	2.09%	-	0.60%	2.88%	-	1.19%	1.00%	1.49%	2.70%	0.63%	0.75%	0.00%	0.00%	0.10%	0.00%
		1.5	0.57%	-	0.00%	0.25%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2	0.25%	0.42%	1.31%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2.5	0.00%	1.81%	0.61%	0.17%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
0.9	6	1.2	0.00%	0.75%	0.00%	1.12%	0.00%	0.48%	0.00%	0.28%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		1.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.62%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.39%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2.5	0.18%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	8	1.2	0.00%	-	0.00%	0.38%	0.00%	0.00%	0.00%	0.61%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		1.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.56%	0.00%	0.00%	0.00%	1.67%	0.00%	0.00%	0.00%	0.43%
		2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2.5	0.78%	1.28%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.64%
	10	1.2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		1.5	3.28%	0.00%	0.00%	0.00%	0.12%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2	0.00%	0.92%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2.5	0.73%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1	6	1.2	0.00%	1.64%	0.00%	0.00%	0.00%	1.99%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.50%	0.00%
		1.5	0.00%	1.45%	0.00%	0.00%	0.00%	0.00%	0.00%	0.97%	0.00%	0.00%	0.84%	1.47%	1.38%	0.00%	0.00%
		2	1.56%	1.91%	1.06%	0.00%	0.00%	0.00%	0.00%	0.48%	0.11%	0.00%	0.00%	0.00%	0.00%	0.82%	0.00%
		2.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.93%	0.00%	0.00%
	8	1.2	2.92%	-	0.00%	0.14%	3.16%	1.85%	1.31%	0.00%	0.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		1.5	0.00%	1.58%	2.01%	0.00%	0.00%	0.00%	0.65%	0.00%	0.87%	0.00%	0.00%	0.7%	0.00%	0.00%	0.00%
		2	1.40%	0.00%	0.00%	1.15%	0.00%	1.67%	0.00%	0.00%	0.00%	0.00%	2.43%	0.00%	0.00%	0.00%	0.00%
		2.5	4.64%	1.40%	0.00%	2.56%	0.00%	0.00%	0.00%	1.03%	1.41%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	10	1.2	4.30%	-	1.37%	0.00%	2.36%	3.22%	1.05%	0.00%	1.35%	0.00%	0.00%	0.00%	2.89%	0.00%	2.53%
		1.5	0.00%	-	0.00%	1.20%	1.39%	0.00%	0.86%	0.00%	1.37%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		2	0.00%	2.72%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.70%	0.00%	0.00%	0.00%	0.00%	0.00%
		2.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
												"-"no feasible solution is incurred due to that the					
Total average deviation:0.25%												time limit Q for the second agent is too small.					

Table 3. The average solution quality of the heuristic algorithm

Table 3 illustrates that the deviation of the heuristic from the optimal solution appears in descending trend as the value of m increases. The reason is that the heuristic selects the maximum total remaining processing time of each job, instead of that of each machine, to be processed on the corresponding machine. For the rate of machine availability θ, the value of 1 showing that all machines are available at any times. Since none of $[b_i^k, f_i^k]$, $i = 1, ..., m$, $k = 1, ..., N(i)$ incurred, the number of time intervals decreases and the span of time interval increases, therefore more jobs are competing to be scheduled in each time interval and hence the error increases. As to the number of jobs, the higher value that n is, the better the performance of the heuristic. One reason is that the higher value for n implies more time epochs r_j^x incurred and a smaller time span for each T_l making the heuristic easier to assigning the operations correctly. Another reason is that the denominator increases, whereas the deviation of the heuristic solution from the optimal solution may not increase in proportion to the denominator. There is no significant difference in the performance of the heuristic on the value of n_a/n_b, but $n_a/n_b = 0.5$ gives the best performance. When n_a/n_b increases, the deviation increases due to the fact that the heuristic gives priority to agent B and thus more operations of agent A should compete to schedule in each time interval.

Based on this analysis, we found that decreasing machine number m and the value of n_a/n_b reduced the deviation of the heuristic, while decreasing job number n increased the deviation of the heuristic. The heuristic generates a good quality schedule with a deviation from the optimum of 0.25% on average.

As to the execution time of the heuristic, Table 4 shows the average execution time of the network based linear programming and the heuristic. The average execution time of the heuristic is small compared to that of the linear programming.

m	*n*	Lingo(ss)	Heuristic(ss)
6	90	19.25	1.25
6	100	37.75	1.75
6	110	30.25	1.75
8	90	32.25	1.75
8	100	165.25	2.25
8	110	71.25	2.50
10	90	181.25	2.00
10	100	629.00	3.00
10	110	1002.75	3.00

Table 4. Execution times of Lingo and Heuristic for small-size instances

In Table 5, the average execution times in seconds for Linear Programming model, Constraint Programming model, and Combined model (Linear programming and Constraint Programming) are shown in columns lingo, OPL, and Total, respectively. For $n = 80$ and $m = 10$, the average execution time for the combined model exceeds two hours. Therefore, an efficient heuristic algorithm is in great need.

			n								
			40			60			80		
θ	m	β	lingo	OPL	Total	lingo	OPL	Total	lingo	OPL	Total
0.7	6	1.2	1.67	3471.55	3473.22	5.83	5632.6	5638.43	12.50	5383.2	5395.70
		1.5	2.50	1621.4	1623.90	5.83	4866.14	4871.97	12.33	6217.37	6229.70
		2	1.83	4443.77	4445.60	4.83	3658.6	3663.43	12.83	8374.42	8387.25
		2.5	1.67	2189.61	2191.28	5.50	4569.55	4575.05	11.00	7941.41	7952.41
	8	1.2	4.83	3563.08	3567.91	21.50	4424.8	4446.30	78.67	6268.49	6347.16
		1.5	7.00	1443.82	1450.82	18.50	3954.75	3973.25	38.17	7704.92	7743.09
		2	7.67	5066.59	5074.26	18.83	6614.74	6633.57	27.33	2103.56	2130.89
		2.5	3.33	4791.44	4794.77	27.33	5982.09	6009.42	40.17	7497.72	7537.89
	10	1.2	6.00	4189.73	4195.73	54.83	6148.1	6202.93	143.00	5235.4	5378.40
		1.5	8.50	3821.1	3829.60	49.50	5743.97	5793.47	114.33	6818.86	6933.19
		2	19.00	4693.21	4712.21	49.17	7330.26	7379.43	183.17	8758.87	8942.04
		2.5	15.50	2043.46	2058.96	46.17	6236.85	6283.02	62.17	8695.65	8757.82
0.9	6	1.2	3.67	3039.84	3043.51	6.67	2850.65	2857.32	26.33	5551.2	5577.53
		1.5	1.67	2232.66	2234.33	9.50	3644.31	3653.81	20.17	5946.28	5966.45
		2	3.00	2767.03	2770.03	8.50	1418.79	1427.29	15.83	7841.27	7857.10
		2.5	3.00	1283.03	1286.03	13.67	3345.55	3359.22	17.50	6982.52	7000.02
	8	1.2	4.33	2918.39	2922.72	44.17	386.79	430.96	121.17	6002.08	6123.25
		1.5	3.83	1051.23	1055.06	11.67	2515.51	2527.18	50.83	6147.72	6198.55
		2	12.33	3845.41	3857.74	51.50	4309.71	4361.21	84.00	6349.53	6433.53
		2.5	15.50	2504.07	2519.57	17.17	4862.58	4879.75	45.83	6600.61	6646.44
	10	1.2	9.17	3579.19	3588.36	25.50	6039.01	6064.51	168.33	7074.55	7242.88
		1.5	14.83	298.89	313.72	58.00	5946.64	6004.64	183.00	6661.49	6844.49
		2	14.83	4352.03	4366.86	51.67	7150.68	7202.35	123.00	8079.32	8202.32
		2.5	11.67	2143.27	2154.94	25.33	6270.04	6295.37	99.00	7973.84	8072.84
1	6	1.2	1.50	3215.97	3217.47	15.83	3695.52	3711.35	12.17	6414.66	6426.83
		1.5	1.83	3013.4	3015.23	4.50	5124.51	5129.01	16.83	6301.5	6318.33
		2	2.17	2111.1	2113.27	4.50	4288.64	4293.14	25.50	6042.42	6067.92
		2.5	1.67	1390.48	1392.15	5.50	5092.22	5097.72	12.17	1745.63	1757.80
	8	1.2	2.67	2791.07	2793.74	17.00	2167.23	2184.23	48.50	5986.16	6034.66
		1.5	4.50	2551.8	2556.30	12.17	2216.43	2228.60	53.00	6284.06	6337.06
		2	2.67	3868.56	3871.23	15.00	4533.6	4548.60	20.00	5285.74	5305.74
		2.5	9.17	4052.98	4062.15	13.67	5139.19	5152.86	86.33	6349.87	6436.20
	10	1.2	6.50	3989.02	3995.52	35.17	6148.1	6183.27	101.17	9280.7	9381.87
		1.5	6.33	3484	3490.33	39.50	5743.97	5783.47	139.83	6818.86	6958.69
		2	4.83	3693.21	3698.04	13.00	7330.26	7343.26	36.17	8758.87	8795.04
		2.5	3.33	3043.1	3046.43	14.67	6236.85	6251.52	54.00	8695.65	8749.65

Table 5. Execution times of LP model, OPL model and those of both models

5. Conclusion

In this paper, we have analyzed the preemptive open-shop with machine availability and eligibility constraints for two-agent scheduling problem. The objective is to minimize makespan, given that one agent will accept a schedule of time up to Q. This problem arises in TFT-LCD and E-Paper manufacturing wherein units go through a series of diagnostic tests that do not have to be performed in any specified order. We proposed an effective heuristic to find a nearly optimal solution and a linear programming model based on minimum cost flow network to optimally solve the problem. Computational experiments show that the heuristic generates a good quality schedule with a deviation of 0.25% on average from the optimum.

References

Agnetis, A., Mirchandani, P.B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research,* 52(2), 229-242. http://dx.doi.org/10.1287/opre.1030.0092

Agnetis, A., Pacciarelli, D., & Pacifici, A. (2007). Multi-agent single machine scheduling. *Annals of Operations Research,* 150(1), 3-15. http://dx.doi.org/10.1007/s10479-006-0164-y

Agnetis, A., Pascale, de G., & Pacciarelli, D. (2009). A Lagrangian approach to single-machine scheduling problems with two competing agents. *Journal of Scheduling,* 12(4), 401-415. http://dx.doi.org/10.1007/s10951-008-0098-0

Brailsford, S.C., Potts, C.N., & Smith, B.M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research,* 119(3), 557-581. http://dx.doi.org/10.1016/S0377-2217(98)00364-6

Bank, J., & Werner, F. (2001). Heuristic Algorithms for Unrelated Parallel Machine Scheduling with a Common Due Date, Release Dates, and Linear Earliness and Tardiness Penalties. *Mathematical and Computer Modelling,* 33(4-5), 363-383. http://dx.doi.org/10.1016/S0895-7177(00)00250-8

Breit, J., Schmidt, G., & Strusevich, V.A. (2001). Two-machine open shop scheduling with an availability constraint. *Operations Research Letters,* 29(2), 65-77. http://dx.doi.org/10.1016/S0167-6377(01)00079-7

Baker, K.R., & Smith, J.C. (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling,* 6(1), 7-16. http://dx.doi.org/10.1023/A:1022231419049

Chu, C.B. (1992). A Branch-and-Bound Algorithm to Minimize Total Tardiness with Different Release Dates. *Naval Research Logistics,* 39(2), 265-283. http://dx.doi.org/10.1002/1520-6750(199203)39:2<265::AID-NAV3220390209>3.0.CO;2-L

Cheng, T.C.E., Ng, C.T., & Yuan, J.J. (2006). Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical Computer Science,* 362(1-3), 273-281. http://dx.doi.org/10.1016/j.tcs.2006.07.011

Cheng, T.C.E., Ng, C.T., & Yuan, J.J. (2008). Multi-agent scheduling on a single machine with max-form criteria. *European Journal of Operational Research,* 188(2), 603-609. http://dx.doi.org/10.1016/j.ejor.2007.04.040

Cheng, T.C.E., Cheng, S.R., Wu, W.H., Hsu, P.H., & Wu, C.C. (2011). A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations. *Computers & Industrial Engineering,* 60(4), 534-541. http://dx.doi.org/10.1016/j.cie.2010.12.008

Cheng, T.C.E., Wu, W.H., Cheng, S.R., & Wu, C.C. (2011). Two-agent scheduling with position-based deteriorating jobs and learning effects. *Applied Mathematics and Computation,* 217(21), 8804-8824. http://dx.doi.org/10.1016/j.amc.2011.04.005

Gonzalez, T., & Sahni, S. (1976). Open shop scheduling to minimize finish time. *Annals of Operations Research,* 23(4), 665-679. http://dx.doi.org/10.1145/321978.321985

Liao, L.W., & Sheen, G.J. (2008). Parallel machine scheduling with machine availability and eligibility constraints. *European Journal of Operational Research,* 184(2), 458-467. http://dx.doi.org/10.1016/j.ejor.2006.11.027

Leung, J.Y.T., Pinedo, M., & Wan, G.H. (2010). Competitive Two-Agent Scheduling and Its Applications. *Operations Research,* 58(2), 458-469. http://dx.doi.org/10.1287/opre.1090.0744

Lee, D.C., & Hsu, P.H. (2012). Solving a two-agent single-machine scheduling problem considering learning effect. *Computers & Operations Research*, 39, 1644-1651. http://dx.doi.org/10.1016/j.cor.2011.09.018

Lee, W.C., Chung, Y.H., & Hu, M.C. (2012). Genetic algorithms for a two-agent single-machine problem with release time. *Applied Soft Computing*, 12, 3580-3589. http://dx.doi.org/10.1016/j.asoc.2012.06.015

Lee, W.C., Chung, Y.H., & Huang, Z.R. (2013). A single-machine bi-criterion scheduling problem with two agents. *Applied Mathematics and Computation*, 219, 10831-10841. http://dx.doi.org/10.1016/j.amc.2013.05.025

Liu, P., Yi, N., & Zhou, X. (2011). Two-agent single-machine scheduling problems under increasing linear deterioration. *Applied Mathematical Modelling*, 35, 2290-2296. http://dx.doi.org/10.1016/j.apm.2010.11.026

Mor, B., & Mosheiov, G. (2010). Scheduling problems with two competing agents to minimize minmax and minsum earliness measures. *European Journal of Operational Research,* 206(3), 540-546. http://dx.doi.org/10.1016/j.ejor.2010.03.003

Ng, C.T., Cheng, T.C.E., & Yuan, J.J. (2006). A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization,* 12(4), 386-393. http://dx.doi.org/10.1007/s10878-006-9001-0

Sedeno-Noda, A., Alcaide, D., & Gonzalez-Martin, C. (2006). Network flow approaches to pre-emptive open-shop scheduling problems with time-windows. *European Journal of Operational Research,* 174(3), 1501-1518. http://dx.doi.org/10.1016/j.ejor.2005.01.062

Sedeno-Noda, A., Pablo, de D.A.L., & Gonzalez-Martin, C. (2009). A network flow-based method to solve performance cost and makespan open-shop scheduling problems with time-windows. *European Journal of Operational Research,* 196(1), 140-154. http://dx.doi.org/10.1016/j.ejor.2008.02.031

Su, L.H., Cheng, T.C.E., & Chou, F.D. (2013). A minimum-cost network flow approach to preemptive parallel-machine scheduling. *Computers and Industrial Engineering,* 64, 453-458. http://dx.doi.org/10.1016/j.cie.2012.04.020

Van Hentenryck, P. (2002). Constraint and integer programming in OPL. *Informs Journal on Computing,* 14(4), 345-372. http://dx.doi.org/10.1287/ijoc.14.4.345.2826

Wan, G.H., Vakati, S.R., Leung, J.Y.T., & Pinedo, M. (2010). Scheduling two agents with controllable processing times. *European Journal of Operational Research,* 205(3), 528-539. http://dx.doi.org/10.1016/j.ejor.2010.01.005

Wu, C.C., Huang, S.K., & Lee, W.C. (2011). Two-agent scheduling with learning consideration. *Computers & Industrial Engineering*, 61, 1324-1335. http://dx.doi.org/10.1016/j.cie.2011.08.007

Wu, C.C., Wu, W.H., Chen, J.C., Yin, Y., & Wu, W.H. (2013). A study of the single-machine two-agent scheduling with release times. *Applied Soft Computing,* 13, 998-1006. http://dx.doi.org/10.1016/j.asoc.2012.10.003

Yin, Y., Cheng, S.R., & Wu, C.C. (2012). Scheduling problems with two agents and a linear non-increasing deterioration to minimize earliness penalties. *Information Sciences*, 189, 282-292. http://dx.doi.org/10.1016/j.ins.2011.11.035

Yin, Y., Wu, W.H., Cheng, S.R., & Wu, C.C. (2012). An investigation on a two-agent single-machine scheduling problem with unequal release dates. *Computers & Operations Research*, 39, 3062-3073. http://dx.doi.org/10.1016/j.cor.2012.03.012

Yin, Y., Wu, W.H., Cheng, T.C.E., Wu, C.C., & Wu, W.H. (2014). A honey-bees optimization algorithm for a two-agent single-machine scheduling problem with ready times. *Applied Mathematical Modelling*. http://dx.doi.org/10.1016/j.apm.2014.10.044

Yu, X., Zhang, Y., Xu, D., & Yin, Y. (2013). Single machine scheduling problem with two synergetic agents and piece-rate maintenance. *Applied Mathematical Modelling*, 37, 1390-1399. http://dx.doi.org/10.1016/j.apm.2012.04.009

Journal of Industrial Engineering and Management, 2015 (www.jiem.org)