6

Methodology To Develop Alternative Makespan Algorithm For Re-entrant Flow Shop Using Bottleneck Approach

Salleh Ahmad Bareduan and Sulaiman Hj Hasan
Universiti Tun Hussein Onn Malaysia,
Malaysia

1. Introduction

The Flow shop manufacturing is a very common production system found in many manufacturing facilities, assembly lines and industrial processes. It is known that finding an optimal solution for a flow shop scheduling problem is a difficult task (Lian *et al*., 2008) and even a basic problem of $F3 \mid\mid C_{max}$ is already strongly NP-hard (Pinedo, 2002). Therefore, many researchers have concentrated their efforts on finding near optimal solution within acceptable computation time using heuristics.

One of the important subclass of flow shop which is quite prominent in industries is re-entrant flow shop. The special feature of a re-entrant flow shop compared to ordinary flow shop is that the job routing may return one or more times to any facility. Among the researchers on re-entrant flow shop, Graves et al. (1983) has developed a cyclic scheduling method that takes advantage of the flow character of the re-entrant process. This work illustrated a re-entrant flow shop model of a semiconductor wafer manufacturing process and developed a heuristic algorithm to minimize average throughput time using cyclic scheduling method at specified production rate. The decomposition technique in solving maximum lateness problem for re-entrant flow shop with sequence dependent setup times was suggested by Dermirkol & Uzsoy (2000). Mixed integer heuristic algorithms was later elaborated by Pan & Chen (2003) in minimizing makespan of a permutation flow shop scheduling problem. Significant works on re-entrant hybrid flow shop can be found in Yura (1999), Pearn et al. (2004) and Choi et al. (2005) while hybrid techniques which combine lower bound-based algorithm and idle time-based algorithm was reported by Choi & Kim (2008).

In scheduling literature, heuristic that utilize the bottleneck approach is known to be among the most successful methods in solving shop scheduling problem. This includes shifting bottleneck heuristic (Adams et al., 1988),(Mukherjee & Chatterjee, 2006) and bottleneck minimal idleness heuristic (Kalir & Sarin, 2001)(Wang et al., 2006). However, not much progress is reported on bottleneck approach in solving re-entrant flow shop problem.

Among the few researches are Dermirkol & Uzsoy (2000) who developed a specific version of shifting bottleneck heuristic to solve the re-entrant flow shop sequence problem.

This chapter explores and investigates an Internet based collaborative design and manufacturing process scheduling which resembles a four machine permutation re-entrant flow shop. It presents the methodology to develop an effective makespan computation algorithm using bottleneck analysis. This computation is specifically intended for the cyber manufacturing centre at Universiti Tun Hussein Onn Malaysia (UTHM).

2. Cyber Manufacturing Centre

UTHM has developed a web-based system that allows the university to share the sophisticated and advanced machinery and software available at the university with the small and medium enterprises using Internet technology (Bareduan et al., 2006). The heart of the system is the cyber manufacturing centre (CMC) which consists of an advanced computer numerical control (CNC) machining centre fully equipped with cyber manufacturing system software that includes computer aided design and computer aided manufacturing (CAD/CAM) system, scheduling system, tool management system and machine monitoring system.

The Petri net (PN) model that describes a typical design and manufacturing activities at the CMC is shown in Fig. 1. The places denoted by P22, P23, P24 and P25 in Fig. 1 are the resources utilized at the CMC. These resources are the CAD system, CAM system, CNC postprocessor and CNC machine centre respectively. At the CMC, all jobs must go through all processes following the sequence represented in the PN model. This flow pattern is very much similar with flow shop manufacturing (Onwubolu, 1996 and Pinedo, 2002). However, it can be noticed from the PN model that there a few processes that share common resources. The process of generating CNC program for prototyping (T3) and the process of generating CNC program for customer (T5) are executed on the same CNC postprocessor (P24). Similarly, the processes of prototype machining (T4) and parts machining (T6) are executed on the same CNC machine centre. These indicate that there are re-entries at both CNC postprocessor (P24) and CNC machine centre (P25) for each job going through the CMC process activities. Thus, this process flow is considered as a re-entrant flow shop as described by Graves et al. (1983). It can also be noticed that both shared resources (P24 and P25) must completely finish the processing of a particular job at T5 and T6 before starting to process any new job at T3 and T4. This means that the CMC scheduling will always follow the permutation rule in executing its functions. In other words, considering all characteristics of the CMC, this problem can be identified as four machine permutation re-entrant flow shop with the processing route of M1,M2,M3,M4,M3,M4 as similarly described by Yang et al. (2008).

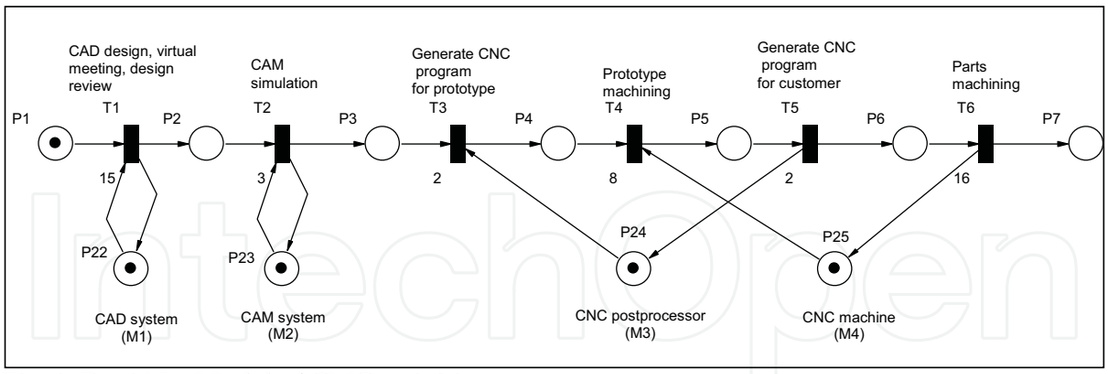

Fig. 1. Petri Net Model of CMC activities

3. Absolute Bottleneck Analysis

Let say, the CMC is currently having four jobs that need to be processed. Typical processing time ranges for all processes are shown in Table 1. From this table, it is obvious that most probably $P(1,j)$ is the bottleneck for the overall process because it is having the longest processing time range. By using the time ranges in Table 1, sets of random data was generated for four jobs that need to be processed. These data is shown in Table 2. Assuming that the data in Table 2 is arranged in the order of First-come-first-served (FCFS), then a Gantt chart representing a FCFS schedule is built as illustrated in Fig. 2. The Gantt chart is built by strictly referring to the PN model in Fig. 1 together with strict permutation rule.

	$P(1, j)$	$P(2, j)$	$P(3, j)$	$P(4, j)$	$P(5, j)$	$P(6, j)$
Minimum time	70	2	2	8	2	8
Maximum time	100	8	8	40	8	40

Table 1. Processing Time Range (hr)

	$P(1, j)$	$P(2, j)$	$P(3, j)$	$P(4, j)$	$P(5, j)$	$P(6, j)$
Job A	73	8	3	8	5	30
Job B	90	2	5	32	5	32
Job C	98	2	3	8	8	17
Job D	75	6	3	36	4	35

Table 2. Processing Time Data (hr)

Referring to Table 2, Figure 1 and Figure 2, the scheduling algorithm for the CMC can be written as the followings and is identified as Algorithm 1 (Bareduan & Hasan, 2008):

Algorithm 1

Let i = Process number or work centre number (i=1,2,3,….)

j = Job number (j=1,2,3,…)

Start (i,j) = start time of the j^{th} job at i^{th} work centre.

Stop (i,j) = stop time of the j^{th} job at i^{th} work centre.

$P(i,j)$ = processing time of the j^{th} job at i^{th} work centre.

For i=1,2,5,6 and j=1,2,3,…n
Start (i,j) = Max [Stop (i,j-1), Stop (i-1,j)] except Start (1,1) = initial starting time
Stop (i,j) = Start (i,j) + P (i,j)

For i =3,4 and j=1,2,3,…n
Start (i,j) = Max [Stop (i,j-1), Stop (i-1,j), Stop (i+2,j-1)]
Stop (i,j) = Start (i,j) + P(i,j)

Work centre (resource)	Job	Time (hours)																		
		0	24	48	72	96	120	144	168	192	216	240	264	288	312	336	360	384	408	432
1 (M1)	JA	24	24	24	1															
	JB				23	24	24	19												
	JC							5	24	24	24	21								
	JD											3	24	24	24					
2 (M2)	JA				8															
	JB							2												
	JC											2								
	JD															6				
3 (M3)	JA				3															
	JB							3	2											
	JC											1	2							
	JD															3				
4 (M4)	JA				8															
	JB								22	10										
	JC												8							
	JD															15	21			
5 (M3)	JA				4	1														
	JB									5										
	JC												8							
	JD																3	1		
6 (M4)	JA					23	7													
	JB									9	23									
	JC												6	11						
	JD																	23	12	

Fig. 2. Gantt Chart for ABCD Job Sequence

Upon thorough study on the schedule Gantt chart at Fig. 2 inwhich $P(1,j)$ appears to be the clear bottleneck, a general makespan computation algorithm for the example problems can be described as below:

$$C_{max} = \sum_{j=1}^{n} P(1,j) \quad + \quad \sum_{i=2}^{6} P(i,n) \tag{1}$$

The bottleneck of $P(1, j)$ in Fig. 2 is represented by the value of $\sum_{j=1}^{n} P(1,j)$ in Equation (1).

Since $\sum_{j=1}^{n} P(1,j)$ will always result to the same value at any job sequence, then the makespan is directly influenced by $\{P(2, n) + P(3, n) + P(4, n) + P(5, n) + P(6, n)\}$ which is the sum of the second, third, fourth, fifth and sixth task processing time for the last job.

Equation (1) is similar with completion time algorithm described by Ho & Gupta (1995) and Cepek *et al*. (2002) for the problem $Fm|ddm|\gamma$. They illustrated the scheduling sequence of decreasing dominant machine (*ddm*) which follows the criteria of Min{j=1,2…n}[$P(k, j)$] ≥ Max{j=1,2…n}[$P(r, j)$]. While these two researches concentrated on strict *ddm* flow shop case study, this chapter focuses on the problem of a special type of flow shop which is known as re-entrant flow shop that exhibits dominant or bottleneck machine characteristics.

To illustrate the usage of Equation (1), the data in Table 2 is used to compute the makespan for the scheduling sequence of DCBA.

$$\begin{aligned} C_{max}\,(\text{DCBA}) &= \{P(1,1) + P(1,2) + P(1,3) + P(1,4)\} \\ &\quad + \{P(2,4) + P(3,4) + P(4,4) + P(5,4) + P(6,4)\} \\ &= \{75 + 98 + 90 + 73\} + \{8 + 3 + 8 + 5 + 30\} \\ &= 390 \text{ hours} \end{aligned}$$

Equation (1) can also be used to obtain the optimum job sequence. This is achieved by assigning the last job sequence to the job that has the smallest value of $\{P(2, j) + P(3, j) + P(4, j) + P(5, j) + P(6, j)\}$ (Bareduan et al., 2008). From Table 2, it can be noticed that Job C has the smallest value of $\{P(2, j) + P(3, j) + P(4, j) + P(5, j) + P(6, j)\}$. Therefore, by assigning Job C as the last sequence, an optimal schedule can be expected. The job sequences that end up with Job C as the last sequence are ABDC, ADBC, BADC, BDAC, DABC and DBAC. The makespan computation for ABDC sequence using Equation (1) is as the followings:

$$\begin{aligned} C_{max}\,(\text{ABDC}) &= \{P(1,1) + P(1,2) + P(1,3) + P(1,4)\} \\ &\quad + \{P(2,4) + P(3,4) + P(4,4) + P(5,4) + P(6,4)\} \\ &= \{73 + 90 + 75 + 98\} + \{2 + 3 + 8 + 8 + 17\} \\ &= 374 \text{ hours} \end{aligned}$$

Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)
ABCD	420	ABDC	374	ACDB	412	BCDA	390
ACBD	420	ADBC	374	ADCB	412	BDCA	390
BACD	420	BADC	374	CADB	412	CBDA	390
BCAD	420	BDAC	374	CDAB	412	CDBA	390
CABD	420	DABC	374	DACB	412	DBCA	390
CBAD	420	DBAC	374	DCAB	412	DCBA	390

Table 3. Makespan for different job sequences using Algorithm 1

In searching for the optimum solution for the 4 job problems in Table 2, a complete enumeration consisting of 24 schedule sequences representing ABCD, ABDC, BADC, etc arrangements were investigated. Algorithm 1 was used to obtain the makespan for all possible sequences and the results are shown in Table 3. From this table, it is noticed that 374 hours is the minimum makespan value and this value belongs to all sequences that end up with Job C as the last sequence. This indicates that minimum makespan solution

provided by Equation (1) in the specific example problem of Table 2 is correct. A further detail observation of Equation (1) suggests that this equation works well for CMC makespan computation under a strict bottleneck conditions. The conditions are imposed to ensure that all first tasks of the CMC activities ($P(1, j)$) are always the bottleneck. These conditions are identified as absolute bottleneck in which regardless of any job sequence arrangement, $P(1,j)$ without fails are always the schedule bottleneck. The absolute bottleneck conditions of Equation (1) are written as the followings:

$$\text{Max}\{j=1\ldots n\}[\,P(2, j)] \le \text{Min}\{j=1\ldots n\}[\,P(1, j)] \quad (2)$$

$$\text{Max}\{j=1\ldots n\}[P(2,j)+P(3,j)+P(4,j)+P(5,j)] \le \text{Min}\{j=1\ldots n\}[P(1,j)+P(2,j)] \quad (3)$$

$$\text{Max}\{j=1\ldots n\}[P(2,j)+P(3,j)+P(4,j)+P(5,j)+P(6,j)] \le \text{Min}\{j=1\ldots n\}[P(1,j)+P(2,j)+P(3,j)] \quad (4)$$

Condition (2) implies that every job takes less time going through task 2 compares to any job going through task 1. It functions to make sure that no matter how the jobs are arranged, $P(2,j)$ will never imposed a bottleneck to the system. $P(2,j)$ will always complete earlier than $P(1,j+1)$ and this guarantees that task 2 of any job can immediately begins after the completion of its task 1 process. This environment is illustrated in Fig. 3 in which X2 completes earlier than Y1. As a result, Y2 can begin immediately after the completion of Y1.

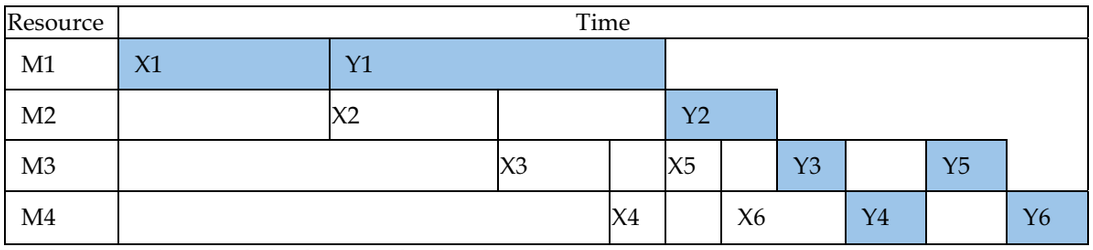

Fig. 3. Example schedule that fulfils (2), (3) and (4)

Condition (3) states that every job takes less time going through task 2, 3, 4 and 5 than any job going through task 1 and 2. This means that for a job X starting on task 2, and job Y starting at task 1 at the same time, this condition guarantees that job X would have released the CNC postprocessor M3 (refer Fig. 1) after its task 5 no later than when job Y needs the postprocessor for its task 3. This is due to the fact that tasks 3 and 5 are sharing the same M3 CNC postprocessor. However, this achievement is only guaranteed if (2) is also satisfied. Fig. 3 also illustrates the example of a schedule that fulfils (3). In this figure, Y3 can begin immediately after completion of Y2 because Y1+Y2 is longer than or equal to X2+X3+X4+X5.

Similarly, condition (4) implies that every job takes less time going through task 2, 3, 4, 5 and 6 than any job going through task 1, 2 and 3. This means that for a job X starting on task 2, and job Y starting at task 1 at the same time, this condition guarantees that job X would have released CNC machine M4 (refer Fig. 1) after its task 6 no later than when job Y needs the CNC machine for its task 4. However, this achievement is only guaranteed if both Condition (2) and (3) are also satisfied. This is also shown in Fig. 3 in which

X2+X3+X4+X5+X6 is less or equal to Y1+Y2+Y3. As a result, it is guaranteed that Y4 can begin immediately after the completion of Y3.

Condition (2), (3) and (4) were then tested on the data from Table 2. The result of this test is shown in Table 4. It can be seen clearly that Condition (2) is satisfied by the values of all $P(2, j)$ which are always smaller than the minimum value of $P(1, j)$. At the same time Condition (3) is also satisfied by the value of $\sum_{i=2}^{5} P(i, j)$ for each job "j" which is never greater than the minimum value of $\sum_{i=1}^{2} P(i, j)$ over all jobs. Similarly, Condition (4) is not violated by looking at the fact that the value of $\sum_{i=2}^{6} P(i, j)$ for each job "j" is always smaller or equal to the minimum value of $\sum_{i=1}^{3} P(i, j)$.

	Condition (2)						Condition (3)		Condition (4)	
Job	$P(1,j)$	$P(2,j)$	$P(3,j)$	$P(4,j)$	$P(5,j)$	$P(6,j)$	$\sum_{i=1}^{2} P(i, j)$	$\sum_{i=2}^{5} P(i, j)$	$\sum_{i=1}^{3} P(i, j)$	$\sum_{i=2}^{6} P(i, j)$
Job A	73	8	3	8	5	30	81	24	84	54
Job B	90	2	5	32	5	32	92	44	97	76
Job C	98	2	3	8	8	17	100	21	103	38
Job D	75	6	3	36	4	35	81	49	84	84

Table 4. Condition (2), Condition (3) and Condition (4) observations

If a set of scheduling data fulfils Condition (2), (3) and (4), then Equation (1) can be used to calculate the schedule makespan as well as to find the job sequences that provide the optimum makespan. Strictly depending on the job sequence, the completion time for each job (C_j) then can be computed as the followings:

$$C_j = \sum_{k=1}^{j} P(1,k) + \sum_{i=2}^{6} P(i,j) \quad (5)$$

Using the data at Table 2, the completion time of Job B under DCBA job sequence can be computed using Equation (5) as the followings:

C_B = $\{P(1,1) + P(1,2) + P(1,3)\}$
$+ \{P(2,3) + P(3,3) + P(4,3) + P(5,3) + P(6,3)\}$
$= \{75 + 98 + 90\} + \{2 + 5 + 32 + 5 + 32\}$
$= 339$ hours

The explanations on the absolute bottleneck conditions indicate that Equation (1) produces accurate makespan computation result if Condition (2), Condition (3) and Condition (4) were met. In order to test this statement, a total of 10000 simulations for four job sequence were conducted using random data between 1 to 20 hours for each of $P(2,j)$, $P(3,j)$, $P(4,j)$, $P(5,j)$ and $P(6,j)$. The value of $P(1,j)$ were set to be between 1 to 100 in order to have a more evenly distributed matching of either Condition (2), Condition (3) and Condition (4) or any of their combinations. These simulations were conducted in Microsoft® Excel using Microsoft® Visual Basic for Application programming.

Each set of random data obtained was tested with a total of 24 different sequences that resembles the sequence arrangements of ABCD, ABDC, ACBD etc. This means that with 10000 simulations, a total of 240,000 job sequence arrangements were tested. The makespan results from Algorithm 1 were compared with the makespan value obtained from Equation (1). In analyzing the test result, a set of data is said to produce perfect result if all the 24 different job sequences makespan from Equation (1) are the same with Algorithm 1. The test analysis result is shown in Table 5. The results from the simulations showed that all makespan value from both Equation (1) and Algorithm 1 are equal for random data sets that fulfill all the three conditions. This indicates that Equation (1) produces accurate makespan computation if Condition (2), Condition (3) and Condition (4) were met. With all these three conditions met, we can conclude that $P(1, j)$ is the absolute bottleneck of the scheduling system. Then, Equation (1) can also be used to determine the optimum job sequences and Equation (5) can be used to calculate the completion time for each job.

Conditions Met	Sets of Data Fulfill The Conditions	Percentage of Perfect Result
4.7+4.8+4.9	900	100%
4.7+4.8	286	8.74%
4.7+4.9	185	19.46%
4.7	3729	1.13%
-	4898	0%

Table 5. Accuracy of Equation (1) at various conditions

4. Bottleneck Correction Factor

During the simulation process to investigate the accuracy of makespan computation using Equation (1) in relation with Condition (2), (3) and (4), it was observed that within a data set that violates these conditions, Equation (1) still produces accurate makespan result at some sequence arrangements but not on all of them. It is worth to investigate further whether there are rules or conditions that might be used to describe this phenomenon. This can only be done by comparing the makespan computation for all possible job sequences within the specific data set using both Algorithm 1 and Equation (1). A sample of data set that exhibits this phenomenon is illustrated in Table 6 in which, Condition (4) is violated. Table 7 shows the makespan computation of data from Table 6 using Algorithm 1 while Table 8 is the makespan computation using Equation (1).

	Condition (2)						Condition (3)		Condition (4)	
Job	$P(1,j)$	$P(2,j)$	$P(3,j)$	$P(4,j)$	$P(5,j)$	$P(6,j)$	$\sum_{i=1}^{2} P(i,j)$	$\sum_{i=2}^{5} P(i,j)$	$\sum_{i=1}^{3} P(i,j)$	$\sum_{i=2}^{6} P(i,j)$
Job A	73	8	3	8	5	30	81	24	84	54
Job B	90	2	5	32	5	32	92	44	97	76
Job C	98	2	3	35	8	39	100	48	103	87
Job D	75	6	3	36	4	35	81	49	84	84

Table 6. Condition (4) violations

Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)
ABCD	423	ABDC	423	ACDB	412	BCDA	393
ACBD	420	ADBC	423	ADCB	412	BDCA	393
BACD	423	BADC	423	CADB	412	CBDA	390
BCAD	420	BDAC	423	CDAB	412	CDBA	390
CABD	420	DABC	423	DACB	412	DBCA	393
CBAD	420	DBAC	423	DCAB	412	DCBA	390

Table 7. Makespan computation using Algorithm 1

Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)	Job Sequence	Makespan (hr)
ABCD	420	ABDC	423	ACDB	412	BCDA	390
ACBD	420	ADBC	423	ADCB	412	BDCA	390
BACD	420	BADC	423	CADB	412	CBDA	390
BCAD	420	BDAC	423	CDAB	412	CDBA	390
CABD	420	DABC	423	DACB	412	DBCA	390
CBAD	420	DBAC	423	DCAB	412	DCBA	390

Table 8. Makespan computation using Equation (1)

Comparing Table 7 and Table 8, it can be noticed that majority of the makespan value using Equation (1) are equal with the makespan value from Algorithm 1. This means that Equation (1) works very well with majority of the job sequences except ABCD, BACD, BCDA, BDCA and DBCA. In other words, even though Condition (4) is violated, Equation (1) still produces accurate makespan value for majority of the job sequences. To further investigate the failure characteristics of Equation (1) in computing the makespan of some job sequences, a detail analysis of the Gantt charts representing the job arrangements that were wrongly computed by the equation was conducted. During the analysis, it was observed that Equation (1) is still valid for makespan computation if some localized sequence dependent conditions were met. These localized sequence dependent conditions for the 4-job example case in Table 6 are (Bareduan & Hasan, 2008):

$$P(1,2) + P(1,3) + P(1,4) \geq VP(2,1) + VP(2,2) + VP(2,3) \quad (6)$$

$$P(1,2) + P(1,3) + P(1,4) + P(2,4) \geq P(2,1) + VP(3,1) + VP(3,2) + VP(3,3) \quad (7)$$

$$P(1,2) + P(1,3) + P(1,4) + P(2,4) + P(3,4) \geq P(2,1) + P(3,1) + VP(4,1) + VP(4,2) + VP(4,3) \quad (8)$$

where VP = Virtual Processing Time.

Condition (6) is meant to make sure that for the last job sequence, task 2 can immediately be started as soon as task 1 completed its process. For example, if Condition (6) is violated, $P(2,n\text{-}1)$ completion time is later than the completion time of $P(1,n)$, this means that $P(2,n)$ cannot start immediately after the completion of $P(1,n)$. It can only begin after the completion of $P(2,n\text{-}1)$ which is also indicated by the completion time of $VP(2,n\text{-}1)$. This introduces a delay between $P(1,n)$ and $P(2,n)$ thus affecting the accuracy of Equation (1). Fig. 4 shows an example schedule that violates Condition (6). The completion time of *P23* which is later than the completion time of *P14* prevents *P24* from starting immediately after the completion of *P14*.

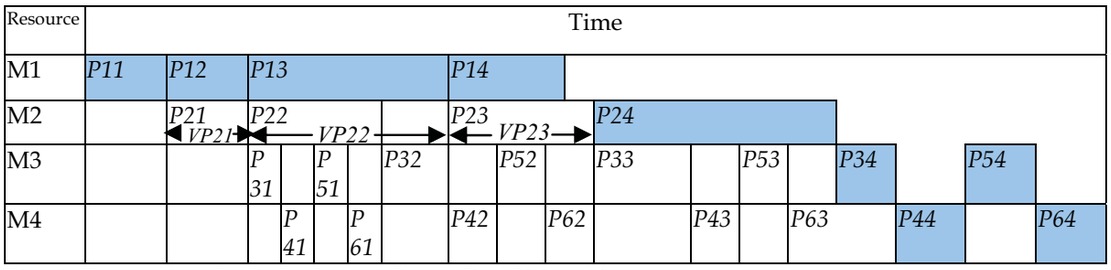

Fig. 4. Example schedule that violates Condition (6)

The virtual processing times for task 2 in Condition (6) are assigned as the followings:

$$\text{For } j = 1,\ VP(2,1) = \text{Max}\,[\,P(2,1),\,P(1,2)\,]$$

$$\text{For } j = 2,3\ldots n\text{-}1,\ VP(2,j) = Max\left[\left[\sum_{k=1}^{j-1} VP(2,k)\right] + P(2,j), \left[\sum_{k=2}^{j+1} P(1,k)\right]\right] - \sum_{k=1}^{j-1} VP(2,k) \quad (9)$$

Condition (7) functions to ensure that for the last job sequence, task 3 can immediately be started as soon as task 2 completed its process. For example, if Condition (7) is violated, this means that the right side value of this condition is larger than its left side value. Since *P*3 and *P*5 are sharing the same M3 processor (refer Fig. 1), the violation of Condition (7) will result to a later completion time of $P(5,n\text{-}1)$ compares to the completion time of $P(2,n)$. Consequently, $P(3,n)$ cannot start immediately after the completion of $P(2,n)$. It can only begin after the completion of $P(5,n\text{-}1)$ which is the completion time of $VP(3,n\text{-}1)$. This introduces a delay between $P(2,n)$ and $P(3,n)$ thus affecting the accuracy of Equation (1). Fig. 5 shows an example schedule that violates Condition (7). The completion time of *P53* which is later than the completion time of *P24* prevents *P34* from starting immediately after the completion of *P24*.

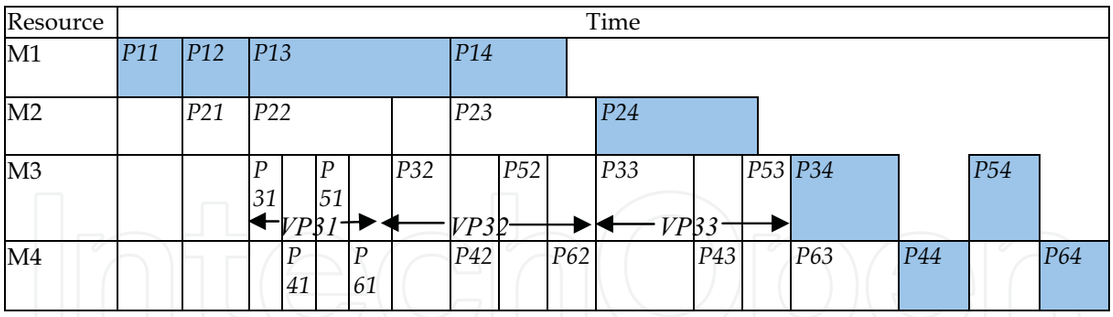

Fig. 5. Example schedule that violates Condition (7)

The virtual processing times for task 3 are assigned as the followings:

$$\text{For } j = 1,\ VP(3,1) = \text{Max}\left[\{VP(2,1) + P(2,2)\},\{P(2,1) + P(3,1) + P(4,1) + P(5,1)\}\right] - P(2,1)$$

$$\text{For } j = 2,3\ldots n\text{-}1,\quad VP(3,j) = Max\left[\begin{array}{l}\left[\sum_{k=1}^{j} VP(2,k)\right] + P(2,j+1), \left[P(2,1) + \sum_{k=1}^{j-1} VP(3,k) + \sum_{i=3}^{5} P(i,j)\right], \\ P(2,1) + P(3,1) + \sum_{k=1}^{j-1} VP(4,k) + \sum_{i=4}^{5} P(i,j)\end{array}\right] - \left[P(2,1) + \sum_{k=1}^{j-1} VP(3,k)\right] \quad (10)$$

Condition (8) functions to guarantee that for the last job sequence, task 4 can immediately be started as soon as task 3 completed its process. For example, if Condition (8) is violated, this means that the right side value of this condition is larger than its left side value. Since *P*4 and *P*6 are sharing the same M4 CNC machine (refer Fig. 1), the violation of Condition (8) will result to a later completion time of *P*(6,*n*-1) compares to the completion time of *P*(3,*n*). Consequently, *P*(4,*n*) cannot start immediately after the completion of *P*(3,*n*). It can only begin after the completion of *P*(6,*n*-1) which is indicated by the completion time of *VP*(4,*n*-1). This introduces a delay between *P*(3,*n*) and *P*(4,*n*) thus affecting the accuracy of Equation (1). Fig. 6 shows an example schedule that violates Condition (8). The completion time of *P63* which is later than the completion time of *P34* prevents *P44* from starting immediately after the completion of *P34*.

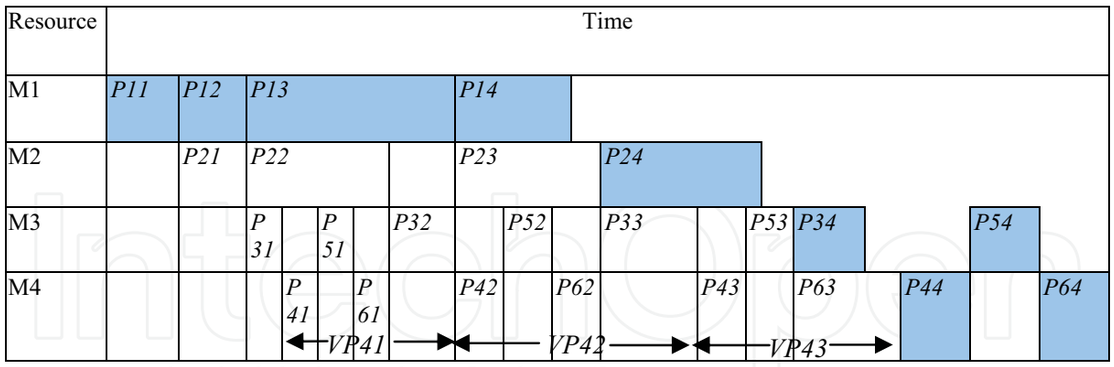

Fig. 6. Example schedule that violates Condition (8)

The virtual processing time for task 4 are assigned as the followings:

$$\text{For } j = 1,\ VP(4,1) =$$

$$\text{Max}\left[\{VP(3,1) + P(3,2)\},\{P(3,1) + P(4,1) + P(5,1) + P(6,1)\}\right] - P(3,1)$$

$$\text{For } j = 2,3\ldots n\text{-}1,\quad VP(4,j) = \tag{11}$$

$$Max\left[\left[\sum_{k=1}^{j} VP(3,k)\right] + P(3,j+1), \left[P(3,1) + \sum_{k=1}^{j-1} VP(4,k) + \sum_{i=4}^{6} P(i,j)\right]\right] - \left[P(3,1) + \sum_{k=1}^{j-1} VP(4,k)\right]$$

If any of the Condition (6), (7) and (8) is violated, Equation (1) is no longer valid for the makespan computation. This equation has to be modified and improved by introducing a dedicated correction factor in order to absorb the violated conditions if it is still to be used for makespan computation beyond the above stipulated conditions.

Detail observations of Condition (6), (7) and (8) reveal that the inaccuracy of Equation (1) due to the violation of Condition (6) is inclusive in Condition (7). Similarly, the error of Equation (1) resulted from the violation of Condition (7) is also inclusive in Condition (8). These are due to the fact that the computations of $VP(4, j)$ in Condition (8) are inclusive of all errors resulting from violations of both Condition (6) and (7). As such, by evaluating and monitoring only Condition (8), all the errors of Equation (1) resulted from the violations of either Conditions (6), (7) and (8) or their combinations are actually covered.

Job	j	$P(1, j)$	$P(2, j)$	$P(3, j)$	$P(4, j)$	$P(5, j)$	$P(6, j)$
Job A	1	73	8	3	8	5	30
Job B	2	90	2	5	32	5	32
Job C	3	98	2	3	35	8	39
Job D	4	75	6	3	36	4	35

	A	B	C	D	E	F	G	H	K
j	Sum $P(1,k)$ k=2, j+1 For j=1,2..n-1	$VP(2, j)$ For j=1,2..n-1	Sum $VP(2,k)$ k=1, j-1 For j=2,3..n-1	Sum $VP(2,k)$ k=1, j For j=1,2..n-1	$VP(3, j)$ For j=1,2..n-1	Sum $VP(3,k)$ k=1, j-1 For j=2,3..n-1	Sum $VP(3,k)$ k=1, j For j=1,2..n-1	$VP(4, j)$ For j=1,2..n-1	Sum $VP(4,k)$ k=1, j-1 For j=2,3..n-1
1	90	90		90	84		84	86	
2	188	98	90	188	98	84	182	96	86
3	263	75	188	263	79	182	261	82	182

Table 9. Table for makespan computation

Table 9 is developed in order to determine the values of $VP(2,j)$, $VP(3,j)$ and $VP(4,j)$ from the data in Table 6. These values will be used to monitor Condition (8). Referring to Table 9, Cell B1 represents $VP(2,1)$. This is computed using Equation (9) as the followings:

For $j = 1$, $VP(2,1) = \text{Max}\,[P(2,1), P(1,2)]$
$= \text{Max}\,[8,90]$
$= 90$

Cell B2 represents $VP(2,2)$. This is also computed using Equation (9) as the followings:

For $j = 2,3\ldots n\text{-}1$,

$$VP(2,j) = Max\left[\left[\sum_{k=1}^{j-1} VP(2,k)\right] + P(2,j), \left[\sum_{k=2}^{j+1} P(1,k)\right]\right] - \sum_{k=1}^{j-1} VP(2,k)$$

Therefore,

$$VP(2,2) = Max\left[\left[\sum_{k=1}^{2-1} VP(2,k)\right] + P(2,2), \left[\sum_{k=2}^{2+1} P(1,k)\right]\right] - \sum_{k=1}^{2-1} VP(2,k)$$

$= \text{Max}\,[C2 + P(2,2), A2] - D1$
$= \text{Max}\,[90 + 2, 188] - 90$
$= 188 - 90$
$= 98$

Using the same equation, Cell B3 is computed as follows:

$VP(2,3) = \text{Max}\,[C3 + P(2,3), A3] - D2$
$= \text{Max}\,[188 + 2, 263] - 188$
$= 263 - 188$
$= 75$

The value for $VP(3,1)$ which belongs to Cell E1 is computed using Equation (10) as the followings:

For j = 1, $VP(3,1)$ = Max [{$VP(2,1) + P(2,2)$}, {$P(2,1) + P(3,1) + P(4,1) + P(5,1)$}] - $P(2,1)$
= Max [{90 + 2}, {8+3+8+5}] - 8
= Max [92, 24] - 8
= 92 - 8
= 84

Cell E2 represents $VP(3,2)$. This is also computed using Equation (10) as the followings:
For j = 2,3…n-1, $VP(3,j)$ =

$$Max\left[\begin{array}{l}\left[\left[\sum_{k=1}^{j} VP(2,k)\right] + P(2,j+1), \left[P(2,1) + \sum_{k=1}^{j-1} VP(3,k) + \sum_{i=3}^{5} P(i,j)\right], \\ P(2,1) + P(3,1) + \sum_{k=1}^{j-1} VP(4,k) + \sum_{i=4}^{5} P(i,j)\end{array}\right] - \left[P(2,1) + \sum_{k=1}^{j-1} VP(3,k)\right]$$

Therefore,

$$VP(3,2) = Max\left[\begin{array}{l}\left[\left[\sum_{k=1}^{2} VP(2,k)\right] + P(2,2+1), \left[P(2,1) + \sum_{k=1}^{2-1} VP(3,k) + \sum_{i=3}^{5} P(i,2)\right], \\ P(2,1) + P(3,1) + \sum_{k=1}^{2-1} VP(4,k) + \sum_{i=4}^{5} P(i,2)\end{array}\right] - \left[P(2,1) + \sum_{k=1}^{2-1} VP(3,k)\right]$$

= Max [{D2 + $P(2,3)$}, {$P(2,1)$ + F2 + $P(3,2)$ + $P(4,2)$ + $P(5,2)$},
{$P(2,1)$ + $P(3,1)$ + K2 + $P(4,2)$ + $P(5,2)$}] - {$P(2,1)$ + F2}
= Max [{188 + 2}, {8 + 84 + 5 + 32 + 5}, {8 + 3 + 86 + 32 + 5}] - {8 + 84}
= Max [190, 134, 134] - 92
= 190 - 92
= 98

$VP(3,3)$
= Max [{D3 + $P(2,4)$}, {$P(2,1)$ + F3 + $P(3,3)$ + $P(4,3)$ + $P(5,3)$},
{$P(2,1)$ + $P(3,1)$ + K3 + $P(4,3)$ + $P(5,3)$}] - {$P(2,1)$ + F3}
= Max [{263 + 6}, {8 + 182 + 3 + 35 + 8}, {8 + 3 + 182 + 35 + 8}] - {8 + 182}
= Max [269, 236, 236] - 190
= 269 - 190
= 79

Referring again to Table 9, Cell H1 represents $VP(4,1)$. This is computed using Equation (11) as the followings:

For j = 1, $VP(4,1)$ = Max [{$VP(3,1) + P(3,2)$}, {$P(3,1) + P(4,1) + P(5,1) + P(6,1)$}] - $P(3,1)$
= Max [{84 + 5}, {3 + 8 + 5 + 30}] - 3
= Max [89, 46] - 3
= 89 - 3
= 86

Cell H2 represents $VP(4,2)$. This is also computed using Equation (11) as the followings:

For $j = 2,3\ldots n\text{-}1$, $VP(4,j)$ =

$$Max\left[\left[\sum_{k=1}^{j} VP(3,k)\right] + P(3,j+1), \left[P(3,1) + \sum_{k=1}^{j-1} VP(4,k) + \sum_{i=4}^{6} P(i,j)\right]\right] - \left[P(3,1) + \sum_{k=1}^{j-1} VP(4,k)\right]$$

Therefore, $VP(4,2)$ =

$$Max\left[\left[\sum_{k=1}^{2} VP(3,k)\right] + P(3,2+1), \left[P(3,1) + \sum_{k=1}^{2-1} VP(4,k) + \sum_{i=4}^{6} P(i,2)\right]\right] - \left[P(3,1) + \sum_{k=1}^{2-1} VP(4,k)\right]$$

= Max [{G2 + $P(3,3)$}, {$P(3,1)$ + K2 + $P(4,2)$ + $P(5,2)$ + $P(6,2)$}] - {$P(3,1)$ + K2}
= Max [{182 + 3}, {3 + 86 + 32 + 5 + 32}] - {3 + 86}
= Max [185, 158] - 89
= 185 - 89
= 96

$VP(4,3)$
= Max [{G3 + $P(3,4)$}, {$P(3,1)$ + K3 + $P(4,3)$ + $P(5,3)$ + $P(6,3)$}] - { $P(3,1)$ + K3}
= Max [{261 +3}, {3 + 182 + 35 + 8 + 39}] - {3 + 182}
= Max [264, 267] - 185
= 267 - 185
= 82

The values of $VP(2, j)$, $VP(3, j)$ and $VP(4, j)$ from Table 9 are used to detect the occurrences of bottleneck at processes other than $P(1, j)$. In other words, this table will be used to suggest the correction factor need to be added to Equation (1) if the previously described Condition (8) is violated. This correction factor is computed as the followings:

From Condition (8):

$$P(1,2) + P(1,3) + P(1,4) + P(2,4) + P(3,4) \geq P(2,1) + P(3,1) + VP(4,1) + VP(4,2) + VP(4,3)$$

If Condition (8) is violated, it means:

$$P(1,2) + P(1,3) + P(1,4) + P(2,4) + P(3,4) < P(2,1) + P(3,1) + VP(4,1) + VP(4,2) + VP(4,3)$$

Therefore, the correction factor ($P1BCF$) is:

$$P1BCF = \{P(2,1) + P(3,1) + VP(4,1) + VP(4,2) + VP(4,3)\} - \{P(1,2) + P(1,3) + P(1,4) + P(2,4) + P(3,4)\}$$

If $\{P(2,1) + P(3,1) + VP(4,1) + VP(4,2) + VP(4,3)\} - \{P(1,2) + P(1,3) + P(1,4) + P(2,4) + P(3,4)\} < 0$ then, $P1BCF = 0$

The general formulation of the correction factor is written as the following:

$$P1BCF = \text{Max}\left[0, \sum_{i=2}^{3} P(i,1) \quad + \quad \sum_{j=1}^{n-1} VP(4,j) \quad - \quad \sum_{j=2}^{n} P(1,j) \quad - \quad \sum_{i=2}^{3} P(i,n)\right] \tag{12}$$

where, *P1BCF* = Process 1 Bottleneck Correction Factor

With the introduction of *P1BCF*, then the generalized makespan computation algorithm for the CMC is:

$$C_{max} = \sum_{j=1}^{n} P(1,j) \quad + \quad \sum_{i=2}^{6} P(i,n) \quad + P1BCF \tag{13}$$

By using Equation 4.11 and 4.12, the makespan computation for the ABCD job sequence in Table 9 is as the followings:

P1BCF = Max [0, {*P*(2, 1) + *P*(3, 1)} + {*VP*(4, 1) + *VP*(4, 2) + *VP*(4, 3)}
- {*P*(1, 2) + *P*(1, 3) + *P*(1, 4)} - {*P*(2, 4) + *P*(3, 4)}]
= Max [0, {8 + 3} + {86 + 96 + 82} - {90 + 98 + 75} - {6 + 3}]
= Max [0, 3]
= 3 hours

Therefore, the makespan is equal to:

C_{max} (ABCD) = {*P*(1, 1) + *P*(1, 2) + *P*(1, 3) + *P*(1, 4)}
+ { *P*(2, 4) + *P*(3, 4) + *P*(4, 4) + *P*(5, 4) + *P*(6, 4) } + *P1BCF*
= {73 + 90 + 98 + 75} + {6 + 3 + 36 + 4 + 35} + 3
= 336 + 84 + 3 = 423 hours

The makespan of 423 hours is equal to the results using Algorithm 1 as in Table 7 for ABCD job sequence. This shows that Equation (13) is capable to perform accurate makespan computation even if the absolute bottleneck condition is violated. To verify the accuracy of Equation (13) in performing the makespan computations, a total of 10,000 simulations were conducted using random data of between 1 to 80 hours for each of *P*(1,*j*), *P*(2,*j*), *P*(3,*j*), *P*(4,*j*), *P*(5,*j*) and *P*(6,*j*) with four job sequence for each simulation. These simulations were conducted in Microsoft® Excel using Microsoft® Visual Basic for Application programming. Each set of random data obtained was also tested with a total of 24 different sequences that resembles the sequence arrangement of ABCD, ABDC, ACBD etc. This means that with 10000 sets of random data, a total of 240,000 job sequence arrangements were tested. The makespan results from using Equation (13) were compared with the makespan value obtained from Algorithm 1. The results from the comparisons showed that all makespan value from both Equation (13) and Algorithm 1 are equal. This indicates that Equation (13) produces accurate makespan results for 4-job CMC scheduling problem. Equation (13) was also tested for computing the makespan for 6-job, 10-job and 20-job CMC scheduling. Each test was conducted with 10,000 sets of random data between 1 to 80 hours for each of *P*(1,*j*), *P*(2,*j*), *P*(3,*j*), *P*(4,*j*), *P*(5,*j*) and *P*(6,*j*). Each set of random data obtained was also tested with different sequences that resemble the sequence arrangement of ABCDEF, ABCDFE, ABCEDF etc. All results indicate that Equation (13) produces makespan value equal the

results of Algorithm 1. This shows the accuracy of Equation (13) in computing the makespan of the CMC scheduling arrangements.

5. Conclusion

This chapter presented the methodology to develop an effective makespan computation algorithm using bottleneck analysis for M1,M2,M3,M4,M3,M4 permutation re-entrant flow shop in which M1 has high tendency of exhibiting bottleneck characteristics. It was shown that under the absolute bottleneck characteristics, both the makespan and optimum job sequence can be accurately determined by the makespan algorithms developed using bottleneck analysis. In cases where the absolute bottleneck condition is violated, the makespan can still be accurately determined by the introduction of bottleneck correction factors. The bottleneck-based methodology or approach presented in this chapter is not only valid for the specific case study alone, but can also be utilised to develop alternative makespan algorithms for other flow shop operation systems that shows significant bottleneck characteristics. With the successful development of makespan computation method using bottleneck analysis, future work can be concentrated to further utilize the bottleneck approach in developing heuristic for optimizing the CMC re-entrant scheduling problems.

6. Acknowledgement

This work was partially supported by the Fundamental Research Grant Scheme (Ministry of Higher Education), Malaysia (Vot 0368, 2007).

7. References

Adams J, Balas E & Zawack D. (1988). The shifting bottleneck procedure for job shop scheduling, *Management Science,* 34, 391-401

Bareduan SA, Hasan SH, Rafai NH & Shaari MF. (2006). Cyber manufacturing system for small and medium enterprises: a conceptual framework, *Transactions of North American Manufacturing Research Institution for Society of Manufacturing Engineers,* 34, 365-372

Bareduan SA & Hasan SH. (2008). Makespan Computation for Cyber Manufacturing Centre Using Bottleneck Analysis: A Re-entrat Flow Shop Problem, Proceedings of International Multiconference of Engineers & Computer Scientists (IMECS 2008) Hong Kong, 19-21/3/2008, pp. 1644-1648

Bareduan SA, Hasan SH & Ariffin S. (2008). Finite scheduling of collaborative design and manufacturing activity: a Petri net approach, *Journal of Manufacturing Technology Management*, 19(2), 274-288

Cepek O, Okada M & Vlach M. (2002). Nonpreemptive Flowshop Scheduling With Machine Dominance, *European Journal of Operational Research*. 139, 245-261

Choi SW & Kim YD. (2008). Minimizing makespan on an *m*-machine re-entrant flowshop, *Computers & Operations Research*, 35(5), 1684-1696

Choi SW, Kim YD & Lee GC. (2005). Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop, *International Journal of Production Research*, 43(11), 2149-2167

Demirkol E & Uzsoy R. (2000). Decomposition methods for reentrant flow shops with sequence dependent setup times, *Journal of Scheduling*, 3, 115-177

Graves SC, Meal HC, Stefek D & Zeghmi AH. (1983) Scheduling of re-entrant flow shops, *Journal of Operations Management*, 3(4), 197-207

Ho JC & Gupta JND. (1995). Flowshop Scheduling With Dominant Machines, *Computers and Operations Research,* 22(2), 237-246

Kalir AA & Sarin SC. (2001). A near optimal heuristic for the sequencing problem in multiple-batch flow-shops with small equal sublots. *Omega*, 29, 577-584

Lian Z, Gu X & Jiao B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan, *Chaos, Solitons and Fractals*, 35(5), 851-861

Mukherjee S & Chatterjee AK. (2006). Applying machine based decomposition in 2-machine flow shops, *European Journal of Operational Research*, 169, 723-741

Onwubolu GC. (1996). A flow-shop manufacturing scheduling system with interactive computer graphics, *International Journal of Operations & Production Management*, 16(9), 74-84

Pan JC & Chen JS. (2003). Minimizing makespan in re-entrant permutation flow-shops, *Journal of Operation Research Society*, 54(6), 642-653

Pearn WL, Chung SH, Chen AY & Yang MH. (2004). A case study on the multistage IC final testing scheduling problem with reentry, *International Journal of Production Economics*, 88(3), 257-267

Pinedo M. (2002). *Scheduling: Theory, algorithms, and systems*, 2nd ed., Upper Saddle River, N.J., Prentice-Hall

Wang JB, Shan F, Jiang B & Wang LY. (2006). Permutation flow shop scheduling with dominant machines to minimize discounted total weighted completion time, *Applied Mathematics and Computation*, 182(1), 947-954

Yang DL, Kuo WH & Chern MS. (2008). Multi-family scheduling in a two-machine re-entrant flow shop with setups, *European Journal of Operational Research*, 187(3), 1160-1170

Yura K. (1999). Cyclic scheduling for re-entrant manufacturing systems, *International Journal of Production Economics*, 60, 523-528

Engineering the Computer Science and IT

Edited by Safeeullah Soomro

ISBN 978-953-307-012-4

Hard cover, 506 pages

Publisher InTech

Published online 01, October, 2009

Published in print edition October, 2009

It has been many decades, since Computer Science has been able to achieve tremendous recognition and has been applied in various fields, mainly computer programming and software engineering. Many efforts have been taken to improve knowledge of researchers, educationists and others in the field of computer science and engineering. This book provides a further insight in this direction. It provides innovative ideas in the field of computer science and engineering with a view to face new challenges of the current and future centuries. This book comprises of 25 chapters focusing on the basic and applied research in the field of computer science and information technology. It increases knowledge in the topics such as web programming, logic programming, software debugging, real-time systems, statistical modeling, networking, program analysis, mathematical models and natural language processing.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen