Two-machine flow shop scheduling with synchronous material movement

Kwei-Long Huang

Institute of Industrial Engineering,
102 Guo-Cing Building,
No. 1, Sec. 4, Roosevelt Road,
National Taiwan University,
Taipei, 106, Taiwan
E-mail: craighuang@ntu.edu.tw

José A. Ventura*

The Harold and Inge Marcus Department of Industrial and Manufacturing Engineering,
356 Leonhard Building,
The Pennsylvania State University,
University Park, PA 16802, USA
Fax: (814)863-4745
E-mail: jav1@psu.edu
*Corresponding author

Abstract: This study considers a new flow shop scheduling problem with synchronous material movement. Specifically, we consider an automated machining centre that consists of a loading/unloading station, two CNC machining stations, and a material handling device. The material handling device is a rotary table that moves jobs between stations simultaneously. Given a set of jobs that need to be processed in the machining centre, the objective of the problem is to find the sequence that minimises the makespan. This problem can be shown to be NP-hard in the strong sense. A dynamic programming algorithm is proposed to obtain an optimal sequence for small- and medium- size problems. For large-scale problems, two heuristic algorithms are developed to obtain a near-optimal solution in a short time. Computational results for the proposed algorithms are provided.

Keywords: flow shop scheduling; makespan; synchronous material handling; dynamic programming; heuristic algorithm.

Reference to this paper should be made as follows: Huang, K-L. and Ventura, J.A. (2013) 'Two-machine flow shop scheduling with synchronous material movement', *Int. J. Planning and Scheduling*, Vol. 1, No. 4, pp.301–315.

Biographical notes: Kwei-Long Huang is an Assistant Professor of the Institute of Industrial Engineering at the National Taiwan University. He received his PhD degree from the Department of Industrial and Manufacturing Engineering at The Pennsylvania State University. He was employed as a Systems Engineer at Compal Electronics for two years. He was involved in several projects sponsored by the National Science Council of Taiwan, in

which quantitative methods are applied to solve problems related to operations scheduling in manufacturing and service industries. His research interests include production and operation scheduling, supply chain management and applied operations research.

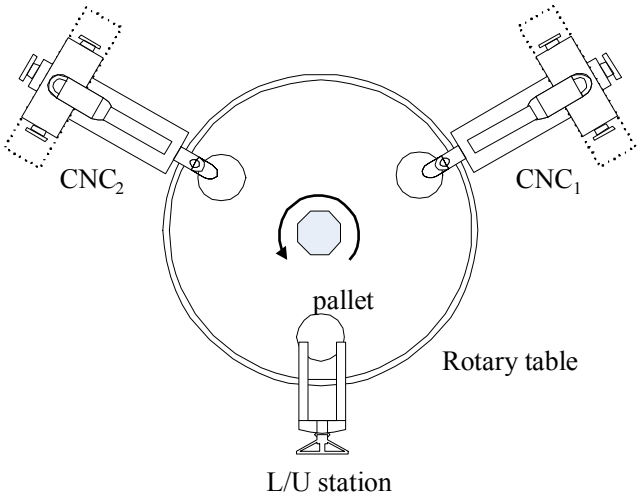
José A. Ventura is a Professor of the Department of Industrial and Manufacturing Engineering at The Pennsylvania State University. He has developed a number of decomposition and Lagrangian dual algorithms to solve large-scale network design and equilibrium models in telecommunication systems design, congestion analysis in traffic networks, and planning of production-distribution systems. He has also worked on several logistics problems and supply chain inventory network models. He has published over 90 refereed journal papers. He is the past Associate Editor of the *Journal of Manufacturing Systems* and *IIE Transactions*, and past Editor of *IIE Computer and Information Systems Newsletter*.

This paper is a revised and expanded version of a paper entitled 'Flow shop scheduling with synchronous material movement' presented at the 2008 Industrial Engineering Research Conference, Vancouver, CA, May 2008.

1 Introduction

Automated manufacturing systems which integrate material handling and processing devices are commonly employed in manufacturing industries to gain a competitive advantage. Typically, an integrated cell or machining centre consists of a loading/unloading (L/U) station, one or more processing machines, and material handling devices to efficiently process and move a group of similar parts. Examples of this type of integrated cells are the T-line machining centres developed by Cincinnati Milacron (1989).

Figure 1 A T-line machining centre with two CNC machines (see online version for colours)



In a T-line machining centre, a job is loaded at an L/U station, processed sequentially through a number of computer numerical control (CNC) machines, and finally unloaded from the machining centre at the L/U station. Jobs in this machining centre are all carried by a material handling device - a rotary table. Once the operations of all jobs currently placed in the stations are finished, the rotary table rotates in a clockwise or counter-clockwise direction to move these jobs simultaneously to the next corresponding stations. Figure 1 shows a T-line machining centre with two CNC machines and a rotary table with three pallets. In this setting, while two jobs, say A and B, loaded in two pallets are being processed by CNC_2 and CNC_1, respectively, another job, say C, is concurrently being loaded onto the third pallet at the L/U station. After the processing operations on both machines and the loading operation are completed, the table rotates 120 degrees counter-clockwise so that job C is transported to CNC_1, and job B to CNC_2. Finished job A will be unloaded from the pallet at the L/U station and a new job, say D, will be loaded on the pallet.

Transporting jobs simultaneously is referred to as synchronous material movement in this paper. This study focuses on sequencing jobs in this particular type of machining centre with two CNC machines. Efficiency is one of the desirable characteristics of these machining centres, and minimising the makespan is equivalent to maximising the utilisation of the machining centre. Thus, the objective of this study is to find the job sequence that minimises the makespan. Huang (2008) has proven that the scheduling problem in a T-line machining centre even with a single machine is NP-hard in the strong sense. The problem with one machine is shown to be equivalent to the numerical matching problem with target sums which is known to be strongly NP-hard (Garey and Johnson, 1979). A similar argument can be applied to show NP-hardness of this problem with two machines by assuming processing times on the second machine to be zero for all jobs. In this study, we propose an exact dynamic programming (DP) algorithm for the two machine problem with the objective of minimising the makespan. Since the time complexity of the DP algorithm is exponential, it can only be used to solve small- and medium- size problems. For large-scale problems, two heuristic algorithms are also proposed to obtain a near-optimal solution in a short time. Computational results for these algorithms are also provided.

This paper is divided into five sections. Section 2 provides a literature review. Section 3 includes the statement of the problem, the proposed DP algorithm, and the analysis of its computational effort. In Section 4, a pair of two-phase heuristic algorithms is developed and results of numerical experiments for all algorithms are provided. Conclusions and directions for future research are stated in Section 5.

2 Literature review

Most of the literature regarding automated machining centres or robotic cells considers the scheduling of jobs and robot moves between machines. In these manufacturing cells, parts are usually loaded and unloaded in different locations and material movement between stations is asynchronous. Sethi et al. (1992) study the problem of sequencing jobs and robot moves in a robotic cell where a single robot is used to transport jobs between stations. The cell is a flow shop system where jobs pass sequentially through the input station, machine stations, and the output station. They show that only two possible optimal policies of robot moves exist for the two-machine robotic cell scheduling

problem with a single part type. For the problem with multiple part-types, a polynomial time algorithm is derived to minimise cycle time for a given fixed sequence of robot moves. Levner et al. (1995) propose a polynomial-time algorithm to obtain the minimum makespan for a two-machine robotic cell. In this robotic cell, there are two robots dedicated to load and unload jobs in each machine, and the loading and unloading times are job-dependent. There is also a transporting robot to move jobs from the first machine to the second machine. In addition, a job completed on the first machine should be transported to a storage buffer which is located in the range of the robot dedicated for the second machine. Logendran and Sriskandarajah (1996) develop analytical methods for determining an optimal sequence of jobs and robot moves with minimum cycle time in three different types of two-machine robotic cells: a robot-centred cell, a mobile robot cell, and an in-line robot cell. They consider scheduling problems with a single part-type and multiple part-types in these three cellular layouts. Given that n is the number of jobs, Hall et al. (1997) provide a $O(n^4)$ time algorithm to obtain an optimal part sequence and robot moves in a two-machine robotic cell with multiple part-types. Aneja and Kamoun (1999) formulate this problem as a travelling salesman problem with a special cost structure, and improve the complexity of the algorithm from $O(n^4)$ to $O(n \log n)$. Dawande et al. (2005) present a survey and summary of the recent developments regarding scheduling in robotic cells. They provide a classification scheme for scheduling problems of robotic cells based on the characteristics of the manufacturing cells such as robot devices, machine environments, and processing restrictions. They also discuss implementation issues and the use of optimal policies for different system settings.

Synchronous transportation of jobs between stations is a particular characteristic of the machining centre addressed in this study. Without considering the L/U station, if there are only two machines or stations in the machining centre with the mechanism of synchronous material movement, the problem is equivalent to a two-machine flow shop problem with blocking which can be solved in polynomial time by the Gilmore-Gomory (1964) algorithm. Hall and Sriskandarajah (1996) prove that a three-machine flow shop problem with blocking is strongly NP-complete. However, a three-machine flow shop with synchronous material movement is not reducible to the problem with blocking, because the constraint of blocking only restricts transfer of a job between two successive machines. In the problem with blocking, for example, a job completed on the second machine can be released to the third machine when it becomes idle. In the case of synchronous transfer, however, the job processed by the second machine can only be released to the downstream machine when both of the first and third machines finish their current operations. Soylu et al. (2007) consider a flow shop scheduling problem with synchronous transfer between stations. They develop a branch-and-bound algorithm with several lower and upper bounds to efficiently obtain the minimum makespan for a moderate-size problem. They indicate this type of manufacturing system with synchronous transfer is advantageous when set-ups for a transporter are timely or costly, or when buffer spaces are limited between stations or jobs are physically large. Huang and Hung (2010) applied a genetic algorithm combined with a local search to solve the problem with multiple machines. The local search was applied to the subsequence where the variance of the processing times of these jobs was the largest. However, in this paper, we propose an exact algorithm to obtain an optimal solution and derive some analytical results for the problem with two machines. Furthermore, we propose heuristic algorithms based on the insights from the analytical results.

3 DP algorithm

3.1 Problem definition and notation

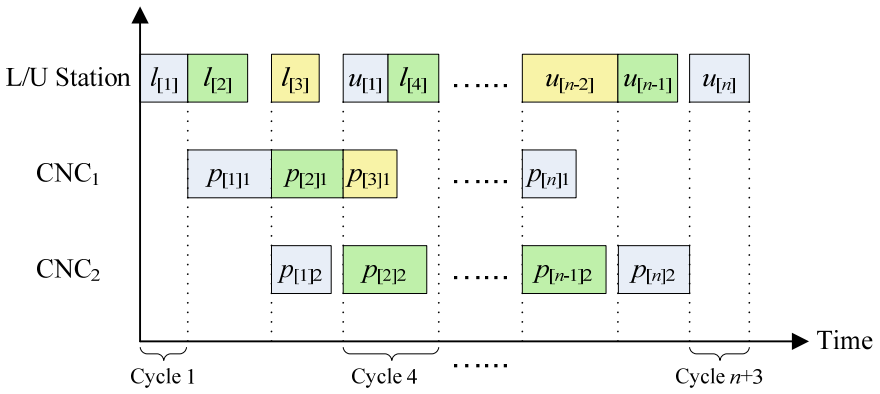
The T-line machining centre considered in this study consists of two CNC machines, denoted as CNC_1 and CNC_2, an L/U station, and a rotary table with three pallets. Jobs may require processing by both CNC machines or only one of them, and have to be loaded onto a pallet before processing. After being loaded onto a pallet, each job must pass through in the same sequence, first CNC_1 and subsequently CNC_2. Finally, the job will return back the L/U station to be unloaded from the machining centre. One pallet can only contain one job, and each CNC machine can only process one job at a time. Assume there are n jobs that have to be processed by the machining centre and all jobs are available at time zero. The time for each rotation of the rotary table is constant. For the makespan criterion, the optimal sequence is independent of the rotation time; thus, the rotation time can be neglected. The loading and unloading times for job j are l_j and u_j, and its processing times on CNC_1 and CNC_2 are p_{j1} and p_{j2}, respectively. The problem is to determine a job sequence that yields the minimum makespan.

Figure 2 illustrates a schedule of jobs at each station. Let C_i be the i^{th} cycle time representing the time period between rotations $i-1$ and i of the rotary table. The number of cycles in a schedule with n jobs is $n+3$. In the first three cycles, there are only loading operations performed at the L/U station. Similarly, only unloading operations are required in the last three cycles. From cycle 4 to cycle n, a job should be unloaded from the rotary table first and a new job will be loaded. Since the rotary table does not rotate until the completion of all operations performed at each station, a cycle time is equal to the largest operation time among the corresponding operations currently performed at each station. Thus, a cycle time can be represented as follows given a job sequence $job_{[1]}$, $job_{[2]}$,…, $job_{[n]}$ where the notation $job_{[i]}$ represents the job is sequenced in position i:

$$C_i = \max\left\{ p_{[i-1]1}, p_{[i-2]2}, l_{[i]} + u_{[i-3]} \right\}, i = 1,\ldots, n+3, \tag{1}$$

where $l_{[k]} = 0$, $p_{[k]1} = 0$, $p_{[k]2} = 0$, and $u_{[k]} = 0$ for $k \neq 1,\ldots, n$.

Figure 2 A schedule of jobs at each station in a two-CNC T-line machining centre (see online version for colours)



Therefore, the makespan of a sequence is the summation of all cycle times which is formulated as $\sum_{i=1}^{n+3} C_i$. Due to the systematic structure of the problem requiring a sequence of interrelated decisions, DP is an efficient approach to obtain an optimal solution (Dreyfus and Law, 1977). Thus, a DP algorithm is proposed in this study and a computational analysis for the algorithm is presented.

3.2 DP formulation

A forward DP procedure is formulated for the problem of finding the minimum makespan. The jobs are numbered from 1 to n. Let $N = \{1,2,\ldots, n\}$ be the set of jobs and S be a subset of N containing the jobs that have already been processed in the machining centre. Let g and h represent the jobs concurrently being processed on CNC_2 and CNC_1 respectively, and j be the job being loaded at the L/U station. Then, the DP formulation is as follows:

- *Optimal value function (OVF):* $f_i(S, g, h, j)$ = minimum completion time for processing jobs g and h on CNC_2 and CNC_1, respectively, unloading the last job in S and loading job j at the L/U station, given that the i jobs in S have already been completed.
- *Optimal policy function (OPF):* $p_i(S, g, h, j)$ = last job unloaded at the L/U station. Equivalently, this is also the last job added to set S.
- *Recurrence relation (RR):*

$$f_i(S,g,h,j) = \min_{k\in S}\{f_{i-1}(S\setminus\{k\},k,g,h) + \max\{p_{h1}, p_{g2}, l_j + u_k\}\};$$
$$i = 1,\ldots,n-3; \{g,h,j\} \subseteq N; S \subseteq N\setminus\{g,h,j\}, |S| = i.$$

- *Boundary condition (BC):*

$$f_0(\varnothing,g,h,j) = l_g + \max\{p_{g1}, l_h\} + \max\{p_{h1}, p_{g2}, l_j\}; \{g,h,j\} \subseteq N.$$

- *Answer (ANS):* $\min_{\{g,h\}\subseteq N}\{f_{n-2}(S,g,h,\varnothing)\}$,

  where

$$f_{n-2}(S,g,h,\varnothing) = \min_{k\in S}\{f_{n-3}(S\setminus\{k\},k,g,h) + \max\{p_{h1}, p_{g2}, u_k\}\} + \max\{p_{h2}, u_g\}$$
$$+ u_h; \{g,h\} \subseteq N; S = N\setminus\{g,h\}, |S| = n-2.$$

3.3 Analysis of computational effort

The computational effort of the DP algorithm is evaluated by the number of operations performed: 'addition' and 'comparison'. The numbers of operations required for each stage of the algorithm are summarised as shown in Table 1.

Table 1 Number of operations required for each stage

Stage	*Number of combinations*	*Additions*	*Comparisons*
BC ($i = 0$)	$n(n-1)(n-2)$	2	3
RR ($1 \le i \le n-3$)	$n(n-1)(n-2)C_i^{n-3}$	$2i$	$2i + (i-1)$
ANS f_{n-2} ($i = n-2$)	$n(n-1)$	n	$2(n-2) + (n-3) + 1$
Minimum makespan	1	0	$n(n-1) - 1$

In boundary condition, there are $n(n-1)(n-2)$ combinations for jobs g, h, and j, and each combination requires two additions and three comparisons to obtain the value for f_0. In recurrence relation, for each i, there are $n(n-1)(n-2)$ choices for jobs g, h, and j, and C_i^{n-3} combinations of jobs in set S. Each combination of (S, g, h, j) has i candidates in set S for k, and extra $i-1$ comparisons are required to obtain minimum values among these i candidates. In the answer formulation, there are $n(n-1)$ different (g, h) pairs for f_{n-2}, and each pair has $n-2$ candidates for k. Moreover, among these $n-2$ candidates $n-3$ comparisons are performed to acquire the minimum value for each f_{n-2}. Then, $n(n-1)-1$ comparisons are needed to obtain the minimum makespan among these f_{n-2}. Therefore, the total number of additions required is:

$$
\begin{aligned}
&= 2n(n-1)(n-2)\left(1+\sum_{i=1}^{n-3} i \cdot C_i^{n-3}\right) + n(n-1)(n) \\
&= 2n(n-1)(n-2)\left(1+(n-3)\sum_{j=0}^{n-4} C_j^{n-4}\right) + n^2(n-1) \\
&= 2n(n-1)(n-2)(n-3)2^{n-4} + n(n-1)(3n-4) \approx n(n-1)(n-2)(n-3)2^{n-3}.
\end{aligned}
$$

The total number of comparisons required is:

$$
\begin{aligned}
&= n(n-1)(n-2)\left(3\sum_{i=1}^{n-3} i \cdot C_i^{n-3} - \sum_{i=1}^{n-3} C_i^{n-3}\right) + 6n(n-1)(n-2) + n(n-1) - 1 \\
&= n(n-1)(n-2)(3(n-3)2^{n-4} - 2^{n-3} + 1) + 6n(n-1)(n-2) + n(n-1) - 1 \\
&= 3n(n-1)(n-2)(n-3)2^{n-4} + n(n-1)(n-2)(7 - 2^{n-3}) + n(n-1) - 1 \\
&\approx 3n(n-1)(n-2)(n-3)2^{n-4}.
\end{aligned}
$$

Thus, the computational effort for this DP algorithm is $O(n^4 2^{n-3})$. In addition, the computer memory required to store the calculated results of each state is also crucial for executing the DP algorithm. In order to compute the results of stage i+1, all states in stage i have to be stored and the number of states is $n(n-1)(n-2)C_i^{n-3}$, as shown in Table 1. Therefore, the computer space required to keep the results of these states could be another major restriction of the DP algorithm.

4 Two-phase heuristic algorithms

Because the problem is strongly NP-hard, heuristic algorithms are proposed to solve large-size problems efficiently. The proposed algorithms consist of two stages: the constructive stage and the improvement stage. In the constructive stage, two constructive

heuristics are developed. One forms an initial sequence by applying the Gilmore-Gomory algorithm to the problem while neglecting the loading and unloading times. The other forms an initial sequence by inserting a job in a position of a given sequence that yields the minimum makespan. In the improvement stage, the neighbourhood search algorithm with modified termination conditions is employed. Furthermore, a formulation to derive a lower bound (LB) value is also presented.

4.1 Constructive algorithms

When the loading and unloading times are smaller than the processing times, we only have to consider the processing times on machines CNC_1 and CNC_2. In this case, the problem can be regarded as a two-machine flow shop problem with blocking, which can be solved optimally by the Gilmore-Gomory algorithm. As a result, the sequence generated by the Gilmore-Gomory algorithm will be the initial seed for the improvement stage while neglecting the loading and unloading times. The makespan of the initial seed, including the loading and unloading times, will be calculated based on this sequence. This constructive algorithm is called CAGG (Constructive Algorithm Gilmore-Gomory).

Furthermore, a LB value can be derived based on the assumption of neglecting the loading and unloading times. In Figure 2, if the cycle time of cycle i(i = 2,…, n + 2) is identified by the processing time of CNC_1 or CNC_2, the minimum value of the summation of these cycles can be obtained by applying the Gilmore-Gomory algorithm to the problem which only considers the processing times on CNC_1 or CNC_2. This minimum value plus the cycle times of the first and last cycles, which are equivalent to the smallest loading and unloading times, will be a LB for the original problem.

Lemma 1. The value, $\min_{i=1,\ldots,n} l_i + MS_{GG} + \min_{i=1,\ldots,n} u_i$, is a LB for the makespan problem in a T-line machining centre with two CNC machines, where MS_{GG} is the optimal makespan obtained by the Gilmore-Gomory algorithm while neglecting loading and unloading times.

Proof: MS_{GG} is a LB to the makespan of cycles 2 to n + 2, and the values $\min_{i=1,\ldots,n} l_i$ and $\min_{i=1,\ldots,n} u_i$ are the LBs for C_1 and C_{n+3}, respectively. Thus, the proof is completed.□

Another constructive algorithm is based on the insertion heuristic. When one job has to be added to the current sequence, every unscheduled job will be inserted in each possible position of the current sequence and the combination of the job and the position with the minimum makespan is chosen. This constructive algorithm based on greedy insertion is called constructive algorithm greedy insertion (CAGI).

4.1.1 CAGI algorithm

N is a set containing all jobs; $N = \{1, 2, \ldots, n\}$. S is a set containing jobs which have been sequenced, and R is a set containing jobs which have not been sequenced. MS is a variable to record the current makespan.

Step 1 Let MS be a big number, $R = N$ and S = ∅.

Step 2 Schedule job k where $k \in \arg\min_{j=1,\ldots,n}\{l_j + p_{j1} + p_{j2} + u_j\}$. Let $S = \{k\}$ and $R = N\backslash\{k\}$.

Step 3 For each job j in R, insert it in each position i of S where $i = 1,\ldots, |S|+1$. Let the corresponding makespan be MS_{ij} where MS_{ij} can be obtained as follows:

$$\begin{aligned} MS_{ij} = MS &+ \max\left\{p_{[i-1]}, p_{[i-2]2}, u_{[i-3]} + l_j\right\} + \max\left\{p_{j2}, p_{[i-1]2}, u_{[i-2]} + l_{[i]}\right\} \\ &+ \max\left\{p_{[i]1}, p_{j2}, u_{[i-1]} + l_{[i+1]}\right\} + \max\left\{p_{[i+1]1}, p_{[i]2}, u_j + l_{[i+2]}\right\} \\ &- \max\left\{p_{[i-1]1}, p_{[i-2]2}, u_{[i-3]} + l_{[i]}\right\} - \max\left\{p_{[i]1}, p_{[i-1]2}, u_{[i-2]} + l_{[i+1]}\right\} \\ &- \max\left\{p_{[i+1]1}, p_{[i]2}, u_{[i-1]} + l_{[i+2]}\right\} - \max\left\{p_{[i+2]1}, p_{[i+1]2}, u_{[i]} + l_{[i+3]}\right\} \end{aligned}$$

where

$$\begin{aligned} l_{[|S|+1]} = l_{[|S|+2]} = l_{[|S|+3]} = p_{[0]1} = p_{[|S|+1]1} = p_{[|S|+2]1} = p_{[-1]2} = p_{[0]2} \\ = p_{[|S|+1]2} = u_{[-2]} = u_{[-1]} = u_{[0]} = 0. \end{aligned}$$

Step 3.1 if $MS < MS_{ij}$, then $MS = MS_{ij}$ and Let job j be f and position i be g.

Step 3.2 if $j = |R|$ and $i = |S|+1$, then insert job f in position g of sequence S, and $R = R \setminus \{f\}$

Step 4 If $R \neq \varnothing$, go to Step 3. Otherwise, go to Step 5.

Step 5 Output the job sequence, S and the makespan, MS.

Similar to Lemma 1, if the cycle times from cycles 2 to n+2 are determined by the loading and unloading times, the summation of the loading and unloading times will provide a LB to an optimal makespan as follows.

Lemma 2. The value, $\sum_{j=1}^{n}(l_j + u_j)$, is a LB for the makespan problem in a T-line machining centre with two CNC machines.

Proof: When neglecting the processing times in cycles 2 to n+2, each cycle length is determined by the loading and unloading times so that the summation of the loading and unloading times is a LB. □

In addition, a new LB which provides a tighter bound for any sequence can be derived from Lemma 1 and Lemma 2.

Theorem 1. The value, $\max\left\{\min_{i=1,\ldots,n} l_i + MS_{GG} + \min_{i=1,\ldots,n} u_i,\ \sum_{j=1}^{n}\left(l_j + u_j\right)\right\}$, is a LB for the makespan problem in a T-line machining centre with two CNC machines.

Proof: Derived directly from Lemmas 1 and 2. □

4.2 Modified neighbourhood search algorithm

After generating a seed, a better solution or sequence can be searched for improving the current makespan. Typically, neighbourhood solutions of the seed are generated and explored. Then, the sequence with the smallest makespan among these neighbourhood

sequences is selected as the seed for next iteration of the improvement stage. The procedure does not terminate until a further improved sequence cannot be found. The technique for the search algorithm is referred to as Neighbourhood Search. It is important to determine the method of generating neighbourhood solutions in the search algorithm, because the larger the number of candidate solutions explored, the better the improvements that can be obtained. One of mechanisms to generate neighbourhood sequences is known as adjacent pairwise interchange, which is based on switching pairs of adjacent jobs. In the proposed algorithm, not only the adjacent pairwise interchange is adopted, but also pairwise interchange of any two jobs is considered.

One of the weaknesses of the neighbourhood search algorithm is that the current solution may be trapped in a local optimum so that no better neighbour can be found with respect to the current seed. In order to escape from a local optimum, a mechanism to increase the diversification of the search region is incorporated into the neighbourhood search algorithm. If no more improvements can be found in the neighbourhood region of the current seed, a neighbourhood sequence with the identical makespan as the current seed is selected as the new seed (ties are broken arbitrarily). The pairwise interchange of jobs and the rule of escaping from a local optimum comprise the basic structure of the algorithm in the improvement stage. The modified neighbourhood search (MNHS) algorithm is explained in detail in the rest of this section.

4.2.1 MNHS algorithm

Let B denote the current best sequence and $MS(S)$ represent the makespan of sequence S. R is a set containing the sequences with identical makespan as the current seed. *Counter* is an index to record the number of random selections that have been performed. PARA is a parameter to determine the maximum number of random selections that can be executed.

Step 1 Let the sequence obtained from the constructed stage be the initial seed S and set *Counter* = 1 and $R = \varnothing$.

Step 2 Initialise $MS(B) = \infty$ and sequence S' obtained by adjacent pairwise interchange on S. $MS(S')$ is computed by using equation (A.1) from Appendix.

Step 2.2 If $MS(S') < MS(B)$, then set $B = S'$ and $MS(B) = MS(S')$.

Step 2.3 If $MS(S') = MS(S)$, then $R = R \cup \{S'\}$.

Step 3 If $MS(B) < MS(S)$, then set $S = B$, $R = \varnothing$, *Counter* = 1, and go to Step 2. Otherwise, go to Step 4.

Step 4 Initialise $MS(B) = \infty$ and generate sequence S' by swapping the positions of jobs i and j where $i = 1$ to $n - 2$ and $j = i + 2$ to n. Compute $MS(S')$ using equation (A.2) from Appendix.

Step 4.1 If $MS(S') < MS(B)$, then set $B = S'$ and $MS(B) = MS(S')$.

Step 4.2 If $MS(S') = MS(S)$, then $R = R \cup \{S'\}$.

Step 5 If $MS(B) < MS(S)$, then set $S = B$, $R = \varnothing$, *Counter* = 1, and go to Step 2.

Step 6 If *Counter* < PARA and $R \neq \varnothing$, then randomly select a sequence from R as new seed S, set $R = \varnothing$, *Counter* = *Counter* + 1, and go to Step 2. Otherwise go to Step 7.

Step 7 Output sequence S and makespan $MS(S)$.

Given an initial seed, a series of adjacent pairwise interchanges is performed to generate a list of new sequences. Through the interchange of jobs, the sequence with the smallest makespan becomes a new seed for the next iteration. However, if no better sequence is obtained, the general pairwise interchange is applied, which swaps any two jobs. The total number of possible sequences of a seed explored is $n(n-1)/2$.

If no improved sequence can be obtained after performing these two interchange schemes, remedial method is adopted to increase the diversification of the search algorithm which randomly selects a sequence from set R as a new seed. In addition, variable *Counter* records the number of the random selections that have been performed and the random selection is executed repeatedly until the counter reaches a predefined parameter (PARA) or set R is empty. In this condition, the whole improvement stage is terminated, and the current sequence and the corresponding makespan are reported.

4.3 Computational study

In order to evaluate the performance of the proposed heuristic algorithms, a series of experiments is conducted. The two constructive algorithms, CAGG and CAGI, are combined with the MNHS algorithm to form two two-phase algorithms. These heuristic algorithms are named CAGG_M and CAGI_M, respectively. These heuristic algorithms and the DP algorithm have been implemented in Borland C++ 5.5 to perform the computational experiments.

Three different scenarios are examined with respect to the loading, processing, and unloading times. Scenario I assumes the expected values of the summation of loading and unloading times of a job is equal to the expected values of its processing times. Scenario II assumes the expected values of processing times of a job are greater than the expected value of the summation of its loading and unloading times. The setting for Scenario III is opposite to the setting for Scenario II. All the loading, processing, and unloading times are randomly generated by using discrete uniform distributions. In addition, for each scenario, three problem sizes are considered; 10, 17 and 40 jobs for small-, medium-, and large-size problems, respectively. The number of jobs for medium-size problems is set to 17 because 17 is the maximum number of jobs that can be optimally scheduled by the proposed DP algorithm due to the memory constraint. In addition, parameter PARA in the MNHS algorithm is set to 10,000. The experimental setting and the rules to generate the testing data are summarised in Table 2.

Table 2 Experiments and data generating rules

	Small size	*Medium size*	*Large size*
Number of jobs (n)	10	17	40
Scenario I	$(l_j, p_{j1}, p_{j2}, u_j)$ = (U(1,7), U(1, 11), U(1, 11), U(1, 3))		
Scenario II	$(l_j, p_{j1}, p_{j2}, u_j)$ = (U(1,7), U(1, 15), U(1, 15), U(1, 3))		
Scenario III	$(l_j, p_{j1}, p_{j2}, u_j)$ = (U(1,10), U(1, 11), U(1, 11), U(1, 4))		

Note: *U denotes the discrete uniform distribution and all operation times are integer.

Ten runs have been executed for each scenario. All test problems except for the medium-size problems with the DP algorithm have been run on a Pentium 1.40 GHz PC with 1 GB RAM. The medium-size problems with the DP algorithm have been carried

out on an Intel Core 2 Duo 1.6GHz PC with 3 GB RAM. There are 90 instances tested for the two proposed heuristics. For small- and medium- size problems, optimal solutions have been obtained by the proposed DP algorithm. For large-size problems, however, only LB values derived from Theorem 1 can be used to compare the accuracy of the heuristic solutions. The average makespans and CPU times on ten runs for all instances tested are summarised in Table 3. Table 4 illustrates the average relative errors, which are the average percent deviations of the makespans obtained by these algorithms with respect to the optimal solutions, if available, or the LBs.

Table 3 Summary of average makespans and CPU times obtained by the DP and heuristics, and average LB

Scenario	*n*	*DP*		*CAGG_M*			*CAGI_M*			*LB*
		Optimum	*Time*	*S1*	*S2*	*Time*	*S1*	*S2*	*Time*	
I	10	81.2	0.24	89.3	82	0.16	83	82.2	0.16	77.5
	17	126.5	201.2	143.6	129.4	0.42	130.7	129.2	0.42	122.2
	40	–	–	309.2	271.1	1.98	279.8	270.5	1.98	254.7
II	10	99.8	0.20	106.4	101.4	0.15	101.8	100.9	0.17	95.7
	17	161.7	200.6	170.9	164.2	0.39	166.4	163.2	0.33	158.2
	40	–	–	378.9	355	1.89	361.5	354.2	1.94	344.6
III	10	84.8	0.19	95.3	86.6	0.20	86.1	85.4	0.15	81.7
	17	142	202.0	161.9	143.6	0.40	146.4	142.9	0.38	141
	40	–	–	374	325.4	1.74	331.9	324.2	1.69	321.7

Notes: S1: the constructive stage; S2: the improvement stage

Table 4 Summary of average relative errors from optimum and LB

Scenario	*n*	*RE from optimum (%)*				*RE from LB (%)*			
		CAGG_M		*CAGI_M*		*CAGG_M*		*CAGI_M*	
		S1	*S2*	*S1*	*S2*	*S1*	*S2*	*S1*	*S2*
I	10	10.14	1.05	2.27	1.25	15.49	5.93	7.24	6.16
	17	13.69	2.33	3.37	2.18	17.79	6.00	7.08	5.86
	40	–	–	–	–	21.50	6.47	9.93	6.25
II	10	6.66	1.63	2.07	1.15	11.30	6.04	6.50	5.53
	17	5.89	1.64	3.00	0.97	8.41	4.04	5.46	3.36
	40	–	–	–	–	10.06	3.07	4.93	2.82
III	10	12.41	2.22	1.56	0.72	16.89	6.30	5.60	4.71
	17	14.11	1.23	3.17	0.70	15.08	2.09	4.04	1.54
	40	–	–	–	–	16.40	1.21	3.25	0.82

The relative errors of the makespans obtained by the two proposed algorithms (CAGG_M and CAGI_M) with respect to the optimal solutions are 2.33% and 2.18% (n = 17 in

Scenario I) and with respect to the LBs are 6.47% and 6.25% (n = 40 in Scenario I). Moreover, optimal sequences can be found in most runs in Scenarios II and III, especially when the number of jobs is equal to 17. With respect to the constructive stage, the sequences formed by CAGI have much smaller makespan values than those obtained by CAGG. The MNHS algorithm significantly improves the makespan values given initial sequences. Regarding the computational effort, the CPU time is not a concern to solve large-size problems by the proposed algorithms. Computational results indicate that the two-phase algorithm (CAGI_M), which combines the insertion heuristic in the constructive phase and the MNHS algorithm in the improvement phase, is applicable to solve the makespan problem for a two-machine T-line machining centre.

Furthermore, the LB derived from Theorem 1 provides a good insight on the optimal makespan when the optimum is unavailable. When the processing times, on average, are larger than the loading and unloading times (Scenario II), the relative errors for the heuristic solutions computed with respect to the LBs decrease as the number of jobs increases. The similar trend can be also observed in Scenario III.

5 Conclusions

In this research, a new flow shop scheduling problem with the makespan criterion has been studied. In the problem, jobs are loaded and unloaded to the machining centre at the same station and are transported to next corresponding machines simultaneously by a rotary table. A DP algorithm has been formulated to solve small- and medium- problems optimally and two-phase heuristic algorithms have been proposed to solve large-scale problems. LB values for these problems are also provided. The computational study shows that the CAGI_M algorithm, which combines the greedy insertion and the MNHS algorithm, generates a sequence in two seconds within 6.25% on the average from a LB when the number of jobs is 40.

In a T-line machining centre with two CNC machines, heuristic algorithms have been finding near optimal solutions for the makespan problems. Therefore, finding optimality conditions or particular properties for special cases are promising research directions. For example, if the unloading times to be zero, then the problem becomes a three-machine flow shop problem with synchronous transfer which has been studied by Soylu et al. (2007). The complexity of this version of the problem can be further investigated. Moreover, a problem with constant loading and unloading times is also similar to the three-machine flow shop problem with synchronous transfer, but the first and last two cycles should be addressed with additional considerations.

Another special case in a two-machine T-line machining centre could assume that the processing times of one machine dominate the processing times of the other machine. The assumption means the minimum processing times of one machine is greater than or equal to the maximum processing times of the other machine. Due to the mechanism of synchronous material movement, only the maximum operation time constitutes the time period of each cycle. For example, if the processing times on machine 1 dominate the processing times on machine 2, then the operation on machine 2 can be always neglected except the second last cycle (cycle n+2) because its cycle time (C_{n+2}) is equal to $\max\{p_{[n]2}, u_{[n-1]}\}$. Therefore, this special case is similar to the problem with one machine.

References

Aneja, Y.P. and Kamoun, H. (1999) 'Scheduling of parts and robot activities in a two machine robotic cell', *Computers and Operations Research*, Vol. 26, No. 4, pp.297–312.

Cincinnati Milacron (1989) 'T-line machining center alternatives', *Manufacturing Engineering*, Vol. 103, No. 4, pp.14–15.

Dawande, M., Geismar, H.N., Sethi, S.P. and Sriskandarajah, C. (2005) 'Sequencing and scheduling in robotic cells: recent developments', *Journal of Scheduling*, Vol. 8, No. 5, pp.387–426.

Dreyfus, S.E. and Law, A.M. (1977) *The Art and Theory of Dynamic Programming*, Academic Press, New York.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability – A Guild to the Theory of NP-Completeness*, W.H. Freeman and Company, New York.

Gilmore, P.C. and Gomory, R.E. (1964) 'Sequencing a one state-variable machine: a solvable case of the traveling salesman problem', *Operation Research*, Vol. 12, No. 5, pp.655–679.

Hall, N.G. and Sriskandarajah, C. (1996) 'A survey of machine scheduling problems with blocking and no-wait in process', *Operations Research*, Vol. 44, No. 3, pp.510–525.

Hall, N.G., Kamoun, H. and Sriskandarajah, C. (1997) 'Scheduling in robotic cells: classification, two and three machine cells', *Operations Research*, Vol. 45, No. 3, pp.421–439.

Huang, K-L. (2008) *Flow Shop Scheduling with Synchronous and Asynchronous Transportation Times*, PhD Dissertation, Department of Industrial and Manufacturing Engineering, The Pennsylvania State University, University Park, PA.

Huang, K-L. and Hung, B-W. (2010) 'Hybrid genetic algorithms for flowshop scheduling with synchronous material movement', *Computers and Industrial Engineering (CIE), 2010 40th International Conference on*, July, DOI: 10.1109/ICCIE.2010.5668353.

Levner, E., Kogan, K. and Levin, I. (1995) 'Scheduling a two-machine robotic cell: a solvable case', *Annals of Operations Research*, Vol. 5, No. 1, pp.217–232.

Logendran, R. and Sriskandarajah, C. (1996) 'Sequencing of robot activities and parts in two-machine robotic cells', *International Journal of Production Research*, Vol. 34, No. 12, pp.3447–3463.

Sethi, S.P., Sriskandarajah, C., Sorger, G., Blazewicz, J. and Kubiak, W. (1992) 'Sequencing of parts and robot moves in a robotic cell', *International Journal of Flexible Manufacturing Systems*, Vol. 4, Nos. 3–4, pp.331–358.

Soylu, B., Kirca, Ö. and Azizoğlu, M. (2007) 'Flow shop-sequencing problem with synchronous transfers and makespan minimization', *International Journal of Production Research*, Vol. 45, No. 15, pp.3311–3331.

Appendix

The following equation calculates the makespan of sequence *S'* obtained in Step 2.1 of the MNHS algorithm:

$$\begin{aligned} MS(S') = MS(S) - \sum_{j=1}^{i+4} & \max\left\{p_{[j-1]1}, p_{[j-2]2}, u_{[j-3]} + l_{[j]}\right\} \\ & + \max\left\{p_{[i-1]1}, p_{[i-2]2}, u_{[i-3]} + l_{[i+1]}\right\} + \max\left\{p_{[i+1]1}, p_{[i-1]2}, u_{[i-2]} + l_{[i]}\right\} \\ & + \max\left\{p_{[i]1}, p_{[i+1]2}, u_{[i-1]} + l_{[i+2]}\right\} + \max\left\{p_{[i+2]1}, p_{[i]2}, u_{[i+1]} + l_{[i+3]}\right\} \\ & + \max\left\{p_{[i+3]1}, p_{[i+2]2}, u_{[i]} + l_{[i+4]}\right\}, \end{aligned} \tag{A.1}$$

where

$$l_{[n+1]} = l_{[n+2]} = l_{[n+3]} = p_{[0]1} = p_{[n+1]1} = p_{[n+2]1} = p_{[-1]2} = p_{[0]2} = p_{[n+1]2} = u_{[-2]} = u_{[-1]} = u_{[0]} = 0.$$

The following equation calculates the makespan of sequence S' obtained in Step 4.1.1 of the MNHS algorithm:

$$\begin{aligned} MS(S') = MS(S) - \sum_{k=i}^{i+3} \max\left\{p_{[k-1]1}, p_{[k-2]2}, u_{[k-3]} + l_{[k]}\right\} - Q \\ + \max\left\{p_{[i-1]1}, p_{[i-2]2}, u_{[i-3]} + l_{[j]}\right\} + \max\left\{p_{[j]1}, p_{[i-1]2}, u_{[i-2]} + l_{[i+1]}\right\} \\ + \max\left\{p_{[j+1]1}, p_{[i]2}, u_{[j-1]} + l_{[j+2]}\right\} \\ + \max\left\{p_{[j+2]1}, p_{[j+1]2}, u_{[i]} + l_{[j+3]}\right\} + V, \end{aligned} \tag{A.2}$$

where

$$l_{[n+1]} = l_{[n+2]} = l_{[n+3]} = p_{[0]1} = p_{[n+1]1} = p_{[n+2]1} = p_{[-1]2} = p_{[0]2} = p_{[n+1]2} = u_{[-2]} = u_{[-1]} = u_{[0]} = 0,$$

and Q and V are computed as follows:

If $j = i+2$,

$$Q = \sum_{k=j+2}^{j+3} \max\left\{p_{[k-1]1}, p_{[k-2]2}, u_{[k-3]} + l_{[k]}\right\}.$$

$$V = \max\left\{p_{[i+1]1}, p_{[j]2}, u_{[i-1]} + l_{[i]}\right\} + \max\left\{p_{[i]1}, p_{[i+1]2}, u_{[j]} + l_{[i+3]}\right\}.$$

Else if $j = i+3$,

$$Q = \sum_{k=j+1}^{j+3} \max\left\{p_{[k-1]1}, p_{[k-2]2}, u_{[k-3]} + l_{[k]}\right\}.$$

$$\begin{aligned} V = \max\left\{p_{[i+1]1}, p_{[j]2}, u_{[i-1]} + l_{[i+2]}\right\} + \max\left\{p_{[i+2]1}, p_{[i+1]2}, u_{[j]} + l_{[i]}\right\} \\ + \max\left\{p_{[i]1}, p_{[j-1]2}, u_{[j-2]} + l_{[j+1]}\right\}. \end{aligned}$$

Else,

$$Q = \sum_{k=j}^{j+3} \max\left\{p_{[k-1]1}, p_{[k-2]2}, u_{[k-3]} + l_{[k]}\right\}.$$

$$\begin{aligned} V = \max\left\{p_{[i+1]1}, p_{[j]2}, u_{[i-1]} + l_{[i+2]}\right\} + \max\left\{p_{[i+2]1}, p_{[i+1]2}, u_{[j]} + l_{[i+3]}\right\} \\ + \max\left\{p_{[j-1]1}, p_{[j-2]2}, u_{[j-3]} + l_{[i]}\right\} + \max\left\{p_{[i]1}, p_{[j-1]2}, u_{[j-2]} + l_{[j+1]}\right\}. \end{aligned}$$