中間作業を伴う2機械ロボティクユニットの 性能保証のある近似スケジューリング*

軽野 義行[†]・木瀬 洋[†]・山本 健太[†]

Approximate Scheduling Algorithms with Performance Guarantee for a Two-Machine Robotic Unit with an Intermediate Station*

Yoshiyuki KARUNO[†], Hiroshi KISE[†] and Kenta YAMAMOTO[†]

In this paper we consider the scheduling problem of minimizing the maximum completion time (i.e., the makespan) for a two-machine robotic unit of flowshop type, in which each of n jobs is processed on the first machine and later on the second machine. There is an intermediate station between the two machines for intermediate operations such as washing, chip disposal, cooling, drying and/or quenching. If only permutation schedules are allowed (i.e., sequences of jobs on the two machines have to be the same), the problem can be solved in polynomial time, although non-permutation problem (i.e., the original problem) is NP-hard in the strong sense. It is already known that, if the optimal permutation schedule is used as an approximate solution for the non-permutation problem, it gives a maximum completion time within twice the optimal. In this paper, we present a different approximate algorithm which is based on a relaxation to the single-machine problem with delivery times, and show that its non-permutation schedule also gives a maximum completion time within twice the optimal. Moreover, we examine its approximation performance by means of numerical experiments, comparing with the optimal permutation scheduling.

1. Introduction

Many flexible manufacturing systems (FMS) have been installed around the world, aiming to achieve higher machine utilization and shorter manufacturing lead time for multi-product small-size production. Such automated systems consist of machining centers, transportation robots and/or automated guided vehicles (AGV), loading and unloading robots, automatic storage/retrieval stations (AS/RS) and so on, all of which are controlled by computers. In particular, small-scale manufacturing systems, which are often called *robotic units* or FMC's (Flexible Manufacturing Cells), are steadily increasing for the purpose of developing fully automated systems. In order to efficiently utilize such systems, many planning problems have to be solved, among them is the scheduling of jobs, which determines when and on what machine the jobs are processed and how these are transported

in the systems (e.g., see Blazewicz et al.[3]).

In this paper we consider the scheduling problem for a two-machine robotic unit of flowshop type, in which each of n jobs is processed on machine M_1 and later on machine M_2 . There is an intermediate station IS between M_1 and M_2 for intermediate operations such as washing, chip disposal, cooling, drying and/or quenching. Intermediate station IS can perform its operations for more than one job at a time, and it also plays a role of infinite capacity buffer for WIP's (Work-In-Processes). Many cutting machines require such intermediate operations before or after cutting operations. Actually, there exist many FMS's with intermediate stations in Japan[2].

The first analytical two-machine flowshop scheduling model was introduced by Johnson[8], and then a number of subsequent papers on different variants of the two-machine flowshop have appeared (e.g., see Mitten[13], Sule[15]). As stated in the above, transportation of the jobs between the machines in FMS's is usually performed by transportation robots and/or AGV's. Much of the scheduling literature, however, has rarely regarded the constraints induced by introducing such material handling devices to manufacturing environment. Only recently has the material

* 原稿受付 2001年2月7日

[†] 京都工芸繊維大学 工学学部 Kyoto Institute of Technology; Matsugasaki, Sakyo-ward, Kyoto-city, Kyoto 606-8585, JAPAN

Key Words: scheduling, robotic unit, two-machine flowshop, approximate algorithms, performance guarantee.

handling been incorporated explicitly in the scheduling problems (e.g., see Agnetis et al.[1], Hall et al.[7], Kise[10], Kise et al.[11], Levner et al.[12], Sethi et al.[14]).

The two-machine robotic unit to be discussed here employs four robots R_1 , R_2 , TR_1 and TR_2 ; Each job is first loaded to machine M_1 by loading robot R_1 . After being processed on M_1 , the job is unloaded from M_1 and transported to intermediate station IS by transportation robot TR_1 . Next, the job is served with intermediate operations by IS. Then, it is transported and loaded to machine M_2 by transportation robot TR_2 , and processed on M_2 . Finally, it is unloaded from M_2 by unloading robot R_2 . The loading, unloading and transportation times of these robots are assumed to be non-negligible.

The scheduling problem we consider here asks to find an optimal schedule that minimizes the maximum completion time (i.e., the makespan). In a general schedule, two sequences of jobs on M_1 and M_2 are allowed to be different. Such a schedule is called a *non-permutation* schedule. If only *permutation* schedules are allowed (i.e., the two sequences of jobs on M_1 and M_2 have to be the same), the problem is referred to as permutation version. Of course, since the jobs have to be served intermediate operations by IS, it is easy to show that in some cases the use of non-permutation schedules yields a shorter maximum completion time than that by the use of only permutation schedules. Unfortunately, the non-permutation problem (i.e., the original problem) is NP-hard in the strong sense, which can be proved straightforwardly from the result of Dell'Amico[5], while the permutation version of the problem can be solved in polynomial time[4,9].

A possibility of the optimal permutation schedule is to use it as an approximate solution for the non-permutation problem. From this view point, it is shown in Kise et al.[9] that the optimal permutation schedule gives a maximum completion time within twice the optimal. However, still this simplicity might be bought at the price of drastically inferior schedules, and the purpose of this paper is to study different approximate algorithms with performance guarantee which yield non-permutation schedules. Thus, in this paper, we present such an approximate algorithm, based on a relaxation of the non-permutation problem to the single-machine problem with delivery times. For the approximate algorithm, we show that its non-permutation schedule also gives a maximum completion time within twice the optimal, and then, provide numerical results for its performance, comparing with the optimal permutation scheduling.

This paper is organized as follows. Section 2. explains the dynamics of the two-machine robotic unit, and defines the mathematical notations of the scheduling problem. Section 3. proposes the approximate algorithm by being based on a relaxation of the problem to the single-machine problem with delivery times.

Section 4. gives different lower bounds for the non-permutation problem, and analyzes the approximation performance guarantee of the proposed algorithm. Section 5. provides numerical results for its performance, comparing with the optimal permutation scheduling. Finally, Section 6. states a conclusion.

2. Problem Description

The system consists of two machines M_1 and M_2 , an intermediate station IS, two transportation robots TR_1 and TR_2 , a loading robot R_1 , an unloading robot R_2 , an Input AS/RS and an Output AS/RS. The system layout is depicted in Fig. 1. In this section, we first explain the dynamics of the system, and then define the mathematical notations of the scheduling problem.

Fig. 1 Two-machine robotic unit with an intermediate station

2.1 Dynamics of the Robotic Unit

The followings are assumed for the system:

- A1) At time zero, there are n jobs available in Input AS/RS. They are to be processed first on machine M_1 , then on intermediate station IS, and finally on machine M_2 .
- A2) Each machine of M_1 and M_2 can perform at most one job at a time.
- A3) The capacity of Input AS/RS is sufficiently large for storing all jobs.
- A4) Loading robot R_1 loads a new job from Input AS/RS to machine M_1 immediately when M_1 becomes empty. The loading times of R_1 are considered to be non-negligible and job-dependent.
- A5) Machine M_1 starts processing a job immediately when its loading by loading robot R_1 has finished. No interruption is allowed for processing. The processing times on M_1 are considered to be non-negligible and job-dependent.
- A6) A job that has finished processing on machine M_1 is unloaded by transportation robot TR_1 if TR_1 is available; otherwise the job waits on M_1 until robot TR_1 comes back to M_1 . The unloading times of TR_1 are considered to be non-negligible and job-dependent. Robot TR_1 can handle at most one job at a time.

- A7) After unloading a job from machine M_1 , robot TR_1 transports it to intermediate station IS, drops it there, and returns back empty to M_1 . The transportation and the return times of TR_1 are considered to be non-negligible and job-independent.
- A8) Each job must stay on intermediate station IS for at least its own *operation time*. The operation times of jobs on IS are considered to be job-dependent.
- A9) The capacity of intermediate station IS is sufficiently large for processing and storing all jobs transported from machine M_1 by robot TR_1 , i.e., IS can perform its intermediate operations for more than one job at a time, and it also plays a role of infinite capacity buffer for the jobs.
- A10) Robot TR_2 transports a job from IS to machine M_2 , waits until M_2 becomes empty if necessary, loads the job to M_2 , and returns back empty to IS. The transportation and the return times of TR_2 are considered to be non-negligible and job-independent. The loading times of TR_2 are considered to be non-negligible and job-dependent. Transportation robot TR_2 can handle at most one job at a time.
- A11) Machine M_2 starts processing a job immediately when its loading by transportation robot TR_2 has finished. No interruption is allowed for processing. The processing times on M_2 are considered to be non-negligible and job-dependent.
- A12) Unloading robot R_2 unloads a job to Output AS/RS immediately when its processing on machine M_2 has finished. The unloading times of R_2 are considered to be non-negligible and job-dependent.
- A13) The capacity of Output AS/RS is sufficiently large for storing all jobs.

2.2 Notations

The followings are input parameters:

- n = the given number of jobs;
- J = $\{j \mid j=1,2,\dots,n\}$ is the set of n jobs;
For $j=1,2,\dots,n$:
- $l_1[j]$ = “loading time” for job j on machine M_1 , i.e., time required for robot R_1 to load job j into M_1 ;
- $p_1[j]$ = “processing time” for job j on machine M_1 ;
- $u_1[j]$ = “unloading time” for job j on machine M_1 , i.e., time required for robot TR_1 to unload job j from M_1 ;
- $l_2[j]$ = “loading time” for job j on machine M_2 , i.e., time required for robot TR_2 to load job j into M_2 ;
- $p_2[j]$ = “processing time” for job j on machine M_2 ;
- $u_2[j]$ = “unloading time” for job j on machine M_2 , i.e., time required for robot R_2 to unload job j from M_2 ;

- $w[j]$ = “operation time” for job j on intermediate station IS;
- t_1 = “transportation time” required for robot TR_1 to transport a job from M_1 to IS;
- t'_1 = “return time” required for robot TR_1 to return from IS to M_1 ;
- t_2 = “transportation time” required for robot TR_2 to transport a job from IS to M_2 ;
- t'_2 = “return time” required for robot TR_2 to return from M_2 to IS.

On the other hand, the followings are output parameters:

- $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$: a sequence of the jobs on machine M_1 , where $\lambda_k \in J$ stands for the k -th job to be processed on M_1 ;
- $\mu = (\mu_1, \mu_2, \dots, \mu_n)$: a sequence of the jobs on machine M_2 , where $\mu_k \in J$ stands for the k -th job to be processed on M_2 ;
- $\pi = (\lambda; \mu)$: a schedule of the jobs.

Note that the permutation version of the problem allows only permutation schedules $\pi = (\lambda; \mu)$ such that $\lambda = \mu$, i.e., $\lambda_k = \mu_k$ for all $k=1,2,\dots,n$. In the permutation version, we denote a permutation schedule by $\pi = (j_1, j_2, \dots, j_n)$, which means that $j_k = \lambda_k = \mu_k$ for all $k=1,2,\dots,n$.

A schedule $\pi = (\lambda; \mu)$ determines the following key time instants:

- $S_1[j]$: the time instant when R_1 starts loading job j on M_1 ;
- $F_1[j]$: the time instant when job j is released from M_1 by TR_1 ;
- $S_2[j]$: the time instant when TR_2 starts loading job j on M_2 ;
- $F_2[j]$: the time instant when job j is released from M_2 by R_2 , which also denotes the completion time of job j .
- $F_{max}(\pi)$: the maximum completion time attained by schedule π , i.e.,

$$F_{max}(\pi) = F_2[\mu_n]. \quad (1)$$

The objective is to find an optimal schedule $\pi = \pi^*$ that minimizes the maximum completion time $F_{max}(\pi)$.

As stated in the above, we consider two versions of the scheduling problem: That is, the permutation scheduling problem (PSP, for short) and the non-permutation scheduling problem (NSP).

The behavior of the robotic unit is illustrated in Fig. 2, which follows an example schedule $\pi = (\lambda; \mu) = ((1,2,3); (1,3,2))$ of three jobs.

3. The Proposed Approximate Algorithm

3.1 Basic Properties of the Problem

First, define the followings for each job $j \in J$:

$$W[j] = t_1 + w[j] + t_2, \quad (2)$$

$$a'[j] = l_1[j] + p_1[j] + u_1[j], \quad (3)$$

Fig. 2 Behavior of the two-machine robotic unit

$$b'[j] = l_2[j] + p_2[j] + u_2[j], \quad (4)$$

$$a[j] = \max\{l_1[j] + p_1[j], t_1 + t'_1\} + u_1[j], \quad (5)$$

$$b[j] = l_2[j] + \max\{p_2[j] + u_2[j], t'_2 + t_2\}, \quad (6)$$

$$A'[j] = a'[j] + W[j], \quad (7)$$

$$B'[j] = b'[j] + W[j], \quad (8)$$

$$A[j] = a[j] + W[j], \quad (9)$$

$$B[j] = b[j] + W[j]. \quad (10)$$

For problem PSP, Cheng and Kise[4] provided the following equation. For a given permutation schedule $\pi = (j_1, j_2, \dots, j_n)$, the maximum completion time $F_{max}(\pi)$ can be computed by:

$$F_{max}(\pi) = A'[j_1] + \max \left\{ \sum_{k=2}^n A[j_k], \right. \\ \left. \max_{2 \leq h \leq n-1} \left\{ \sum_{k=2}^h A[j_k] + \sum_{k=h}^{n-1} B[j_k] \right\}, \right. \\ \left. \sum_{k=1}^{n-1} B[j_k] \right\} + B'[j_n] - \sum_{k=1}^n W[j_k]. \quad (11)$$

By being based on this equation, Kise et al.[9] showed that problem PSP can be solved in $O(n^2)$ time. In this paper, we denote this $O(n^2)$ time algorithm by *OP*.

They also gave the following theorems:

[Theorem 1] [9] Problem NSP is NP-hard in the strong sense.

[Theorem 2] [9] For an instance of problem NSP, let π^* be an optimal schedule and let π_{OP} an optimal permutation schedule. Then,

$$F_{max}(\pi_{OP}) \leq 2 \cdot F_{max}(\pi^*) \quad (12)$$

holds, and this bound is asymptotically tight.

For problem NSP, we show that, given a sequence of jobs on machine M_1 , we can easily obtain an optimal sequence of the jobs on machine M_2 with respect to the sequence on machine M_1 . This property will be used later to present our approximate algorithm with a performance guarantee that yields a non-permutation schedule.

[Lemma 1] For an instance of problem NSP,

given a sequence $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ of the jobs on machine M_1 , an optimal sequence $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ on machine M_2 with respect to λ can be obtained in polynomial time.

(Proof) Note That $F_1[\lambda_k] = a'[\lambda_1] + \sum_{h=2}^k a[\lambda_h]$ is the time instant when job λ_k is released from machine M_1 by transportation robot TR_1 , $k = 1, 2, \dots, n$. Then, job λ_k will be ready to leave from intermediate station IS at time $F_1[\lambda_k] + W[\lambda_k]$. Since all the jobs μ_k except for the last job μ_n must spend $b[\mu_k]$ time units on machine M_2 (while the last job μ_n must spend $b'[\mu_n]$ time units on machine M_2), if the last job μ_n is fixed to some job, the optimal sequence μ of the jobs on machine M_2 can be obtained by sorting the remaining $n-1$ jobs in non-decreasing order of $F_1[j] + W[j]$, i.e.,

$$F_1[\mu_1] + W[\mu_1] \leq F_1[\mu_2] + W[\mu_2] \\ \leq \dots \leq F_1[\mu_{n-1}] + W[\mu_{n-1}] \quad (13)$$

(recall the minimization of the maximum completion time for the single-machine scheduling problem with release times[3]).

Therefore, by repeating the followings for all $k = 1, 2, \dots, n$, and then choosing the best sequence: fix the last job on machine M_2 to job λ_k , and sort the remaining $n-1$ jobs $j \in J \setminus \{\lambda_k\}$ in non-decreasing order of $F_1[j] + W[j]$, we can obtain an optimal sequence μ with respect to λ . $\square$

3.2 A Relaxation to the Single-Machine Problem with Delivery Times

In this subsection, we consider a relaxation of problem NSP to the single-machine scheduling problem with delivery times, and then obtain an approximate schedule for the NSP by solving the single-machine problem. The single-machine problem is solved to compute a sequence λ of jobs on machine M_1 , and an optimal sequence μ of the jobs on machine M_2 with respect to λ is obtained by using **Lemma 1**.

The single-machine problem to be incorporated in the proposed algorithm is as follows:

$$[1 | q[j] | C_{max}]$$

INPUT: A set of n jobs $J' = \{j \mid j = 1, 2, \dots, n\}$, processing times $p[j](>0)$ and delivery times $q[j](\geq 0)$ for

all $j \in J'$.

OUTPUT: An optimal processing sequence $\chi = (j_1, j_2, \dots, j_n)$ of the jobs on the single machine that minimizes the maximum completion time, i.e.,

$$C_{max}(\chi) = \max_{1 \leq h \leq n} \left\{ \sum_{k=1}^h p[j_k] + q[j_h] \right\}. \quad (14)$$

For an instance of problem $1 | q[j] | C_{max}$, it is well-known that an optimal sequence χ can be obtained by sorting the jobs in non-increasing order of their delivery times $q[j]$ (e.g., see Blazewicz et al.[3]).

Since more than one job can be performed at a time on intermediate station IS, the proposed algorithm will regard the time units of $W[j] + b'[j]$ for each job j as its delivery time $q[j]$ of the corresponding single-machine problem. In other words, the single-machine problem relaxes the restriction on machine M_2 which can perform at most one job at a time for the NSP. Moreover, in the relaxation, machine M_1 corresponds to the single machine, and hence the time units $a'[j]$ for the first job on machine M_1 and $a[j]$ for the remaining $n-1$ jobs will be regarded as the processing times $p[j]$ on the single machine. Thus, given a fixed λ_1 , we determine the sequence $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ on machine M_1 such that it satisfies

$$\begin{aligned} W[\lambda_2] + b'[\lambda_2] &\geq W[\lambda_3] + b'[\lambda_3] \\ &\geq \dots \geq W[\lambda_n] + b'[\lambda_n]. \end{aligned} \quad (15)$$

3.3 Approximate Algorithm A

In this subsection, we describe the detail of the proposed approximate algorithm, denoted by A:

[Algorithm A]

Step 1 (Initialization)

$J := \{1, 2, \dots, n\}$; $FMAX := \infty$; $j := 1$; Go to Step 2.

Step 2 (Ending test)

If $j > n$, then output the approximate schedule $\pi_A := \pi_{best}$; otherwise, go to Step 3.

Step 3 (Sequencing jobs on machine M_1)

- (i) $\lambda_1 := j$; $R := J \setminus \{j\}$;
- (ii) Find a permutation $\psi = (\psi(1), \psi(2), \dots, \psi(n-1))$ on R such that

$$\begin{aligned} W[\psi(1)] + b'[\psi(1)] &\geq W[\psi(2)] + b'[\psi(2)] \\ &\geq \dots \geq W[\psi(n-1)] + b'[\psi(n-1)]; \end{aligned}$$

- (iii) $\lambda_{i+1} := \psi(i)$ for all $i = 1, 2, \dots, n-1$; Go to Step 4.

Step 4 (Sequencing jobs on machine M_2)

- (i) Find a permutation $\sigma = (\sigma(1), \sigma(2), \dots, \sigma(n))$ on J by using λ such that

$$\begin{aligned} F_1[\sigma(1)] + W[\sigma(1)] &\leq F_1[\sigma(2)] + W[\sigma(2)] \\ &\leq \dots \leq F_1[\sigma(n)] + W[\sigma(n)]; \end{aligned}$$

- (ii) $k := n$; $FF := \infty$;
- (iii) If $k < 1$, then go to Step 5;

- (iv) Generate a permutation $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ on J by $\mu_i := \sigma(i)$ for $i = 1, 2, \dots, k-1$, $\mu_i := \sigma(i+1)$ for $i = k, k+1, \dots, n-1$ and $\mu_n := \sigma(k)$;
- (v) Compute the makespan $F_{max}((\lambda; \mu))$ for schedule $(\lambda; \mu)$;
- (vi) If $F_{max}((\lambda; \mu)) < FF$, then $FF := F_{max}((\lambda; \mu))$ and $\pi^{[j]} := (\lambda; \mu)$;
- (vii) $k := k-1$; Return to (iii) of Step 4.

Step 5 (Updating the best schedule)

- (i) $F_{max}(\pi^{[j]}) < FMAX$, then $FMAX := F_{max}(\pi^{[j]})$ and $\pi_{best} := \pi^{[j]}$;
- (ii) $j := j+1$; Return to Step 2.

In Step 3, the single-machine problem of $1 | q[j] | C_{max}$ is solved (see eq. (15)), and in Step 4, **Lemma 1** is applied (see eq. (13)).

The time complexity of the proposed algorithm A is $O(n^3)$, since algorithm A examines all possible combinations of the first job on machine M_1 and the last job on machine M_2 , and for each schedule corresponding to a combination it computes the maximum completion time (i.e., which requires $O(n^2 \times n)$).

4. Performance Guarantee of the Proposed Approximate Algorithm

4.1 Lower Bounds

The following lower bounds of the minimum maximum completion time $F_{max}(\pi^*)$ for the NSP are immediately obtained:

$$LB1 = \max \left\{ \sum_{j \in J} a'[j], \sum_{j \in J} b'[j] \right\}, \quad (16)$$

(see eqs. (3) and (4))

$$LB2 = \max \left\{ LB1, \max_{j \in J} \{a'[j] + W[j] + b'[j]\} \right\}. \quad (17)$$

(see eqs. (2), (3) and (4))

Now, let $\pi^* = (\lambda^*; \mu^*)$ be an optimal schedule for a problem instance of the NSP, where $\lambda^* = (\lambda_1^*, \lambda_2^*, \dots, \lambda_n^*)$ denotes its sequence of jobs on machine M_1 and $\mu^* = (\mu_1^*, \mu_2^*, \dots, \mu_n^*)$ its sequence of jobs on machine M_2 . Then, from similar ideas of eqs. (16) and (17), we obtain the following lower bounds:

$$LB1' = \sum_{j \in J \setminus \{\mu_n^*\}} b[j] + b'[\mu_n^*], \quad (18)$$

$$LB2' = \max \left\{ LB1', \{a'[\lambda_1^*] + W[\lambda_1^*] + b'[\lambda_1^*]\} \right\}. \quad (19)$$

Next, consider the following instance of problem $1 | q[j] | C_{max}$ (see Subsection 3.2) with $n-1$ jobs: the processing times are given by $p[j] = a[j]$ and the delivery times are by $q[j] = W[j] + b'[j]$ for all $j \in J \setminus \{\lambda_1^*\}$, and let C_{max}^* be the optimal maximum completion time for the instance of $1 | q[j] | C_{max}$. Then, by eq. (14), we have an additional lower bound immediately:

$$LB3' = \max\{LB2', \{a'[\lambda_1^*] + C_{max}^*\}\}, \quad (20)$$

since the single-machine problem relaxes the restriction on machine M_2 which can perform at most one job at a time for the NSP.

Note that, for an instance of the NSP and the approximate schedule π_A which solves the above instance of problem 1 $|q[j]| C_{max}$, intermediate station IS has completed its intermediate operations for all jobs at time $LB3'$, and hence transportation robot TR_2 can transport any job from IS to machine M_2 at this time (if no job has left from IS).

4.2 Approximation Ratio of the Proposed Algorithm

In this subsection, using lower bounds $LB1'$, $LB2'$ and $LB3'$, we show that the proposed algorithm A has an approximation performance guarantee:

[Theorem 3] For an instance of problem NSP, let π^* be an optimal schedule and let π_A the schedule obtained by algorithm A . Then,

$$F_{max}(\pi_A) \leq 2 \cdot F_{max}(\pi^*) \quad (21)$$

holds.

(Proof) Let $\pi^* = (\lambda^*; \mu^*)$ be an optimal schedule for any instance of the NSP, where $\lambda^* = (\lambda_1^*, \lambda_2^*, \dots, \lambda_n^*)$ denotes its sequence of jobs on machine M_1 and $\mu^* = (\mu_1^*, \mu_2^*, \dots, \mu_n^*)$ its sequence of jobs on machine M_2 . Since the proposed algorithm A examines all possible combinations of the first job on machine M_1 and the last job on machine M_2 , we can see that there is such a schedule among all schedules considered in algorithm A that the first job on machine M_1 is fixed to λ_1^* and the last job on machine M_2 is fixed to μ_n^* (possibly, $\lambda_1^* = \mu_n^*$). We denote the schedule by $\pi' = (\lambda'; \mu')$.

Now, consider time $LB3'$ for schedule π' . The proposed algorithm A solves 1 $|q[j]| C_{max}$ in which the first job on machine M_1 is fixed to λ_1^* , when it examines schedule π' . Since the maximum completion time of the single machine is C_{max}^* , intermediate station IS has completed all intermediate operations by time $LB3'$. Thus, transportation robot TR_2 can transport any job from IS to machine M_2 at time $LB3'$ (if no job has left from IS). Therefore,

$$\begin{aligned} F_{max}(\pi_A) &\leq F_{max}(\pi') \\ &\leq LB3' + \sum_{j \in J \setminus \{\mu_n^*\}} b[j] + b'[\mu_n^*] \\ &= LB3' + LB1' \\ &\leq 2 \cdot F_{max}(\pi^*). \end{aligned}$$

□

However, it is an open question whether the bound of 2 is tight or not.

5. Numerical Experiments

In this section, we compare the proposed approximate algorithm A with the optimal permutation sched-

uling algorithm OP by means of numerical experiments. We also give the results of the improved optimal permutation schedules, denoted by $I-OP$ (i.e., in which the sequence of jobs on machine M_2 is optimized by applying **Lemma 1**, with respect to the sequence of jobs on machine M_1 obtained by algorithm OP), and of the well-known FCFS (First-Come-First-Served) schedules (any FCFS schedule is a permutation schedule). Note that each algorithm of A , $I-OP$ and OP gives a maximum completion time within twice the optimal (see **Theorem 2** and **Theorem 3**).

We introduce an additional input parameter for the numerical experiments: Let $n' (\leq n)$ be the number of jobs that require intermediate operations (and hence, the operation times $w[j]$ on intermediate station IS are zero for the remaining $n - n'$ jobs), then we define

$$\alpha = \frac{n'}{n} \times 100 [\%], \quad (22)$$

which is referred to as *intermediate operation rate*.

The test problem instances are randomly generated as follows:

- The total number of jobs n : 50 (fixed).
- The intermediate operation rate α : 5, 10, 20, 50 and 100 [%].
- The loading and unloading times l_1 , u_1 , l_2 and u_2 : all zero (fixed).
- The processing times $p_1[j]$ and $p_2[j]$: uniformly random in [13, 187], and hence the mean value of these processing times $p_{mean} = 100$.
- The operation times $w[j]$: uniformly random in $[(1 - \sqrt{3} \cdot CV) \times w_{mean}, (1 + \sqrt{3} \cdot CV) \times w_{mean}]$, where CV is the coefficient of variation and w_{mean} is the mean value for $w[j]$. The coefficient of variation CV is fixed to 0.5, and the mean value w_{mean} is changed from $0.1n \times p_{mean}$ to $n \times p_{mean}$.
- The transportation and the return times $t1$, $t2$, $t1'$ and $t2'$: all zero (fixed).

Now, let γ be the ratio of the mean operation time to the total processing time on a machine, called *operation time ratio*, i.e.,

$$\gamma = \frac{w_{mean}}{n \times p_{mean}}, \quad (23)$$

then this parameter is to be examined from 0.1 to 1. Moreover, we use the following equation to compute the relative deviation of an approximate algorithm $H \in \{A, I-OP, FCFS\}$ from the maximum completion time of the optimal permutation schedule:

$$\begin{aligned} R.D.(H) &= \frac{F_{max}(\pi_H) - F_{max}(\pi_{OP})}{F_{max}(\pi_{OP})} \times 100 [\%] \end{aligned} \quad (24)$$

The program is written in C. It is compiled by Microsoft Visual C++, and run on a personal computer GATEWAY2000/P5-133. The CPU times of the pro-

posed algorithm A and the optimal permutation algorithm OP are no more than 30[msec] and 10[msec], respectively, for a problem instance with fifty jobs tested.

All of Tables 1, 2, 3, 4 and 5 provide the relative deviations of A , $I-OP$ and FCFS from the optimal permutation schedules OP , computed by eq. (24), with the intermediate operation rate of $\alpha = 5, 10, 20, 50$ and 100[%], respectively. Each of the data is the mean value for 50 problem instances tested.

In Table 1, the proposed algorithm A yields better schedules than the optimal permutation algorithm OP for the operation time ratio γ no less than 0.3, and better than the improved optimal permutation

Table 1 Performance of algorithms (1)
Relative Deviation [%]

γ	A	$I-OP$	FCFS
0.1	3.4 (0)	0.0 (0)	11.2 (0)
0.2	2.1 (13)	-1.2 (23)	21.3 (0)
0.3	-6.4 (39)	-9.2 (43)	21.7 (0)
0.4	-15.0 (47)	-16.8 (48)	20.1 (0)
0.5	-22.4 (49)	-20.3 (49)	18.4 (0)
0.6	-28.5 (49)	-21.0 (49)	17.0 (0)
0.7	-32.5 (49)	-19.9 (49)	15.9 (0)
0.8	-33.8 (50)	-19.1 (50)	14.9 (0)
0.9	-33.7 (50)	-18.1 (50)	13.9 (0)
1.0	-33.1 (50)	-17.1 (50)	13.1 (0)

$n = 50, p_{mean} = 100, \alpha = 5[\%]$

$w_{mean} = \gamma n \times p_{mean}, CV = 0.5$

(): The number of problem instances for which each algorithm found better schedules than the optimal permutation schedules (the total number of instances is 50)

Table 2 Performance of algorithms (2)
Relative Deviation [%]

γ	A	$I-OP$	FCFS
0.1	9.0 (0)	0.0 (0)	13.5 (0)
0.2	6.2 (2)	-3.1 (37)	22.6 (0)
0.3	-5.5 (42)	-13.5 (49)	20.8 (0)
0.4	-15.4 (48)	-19.3 (50)	18.6 (0)
0.5	-23.5 (50)	-20.1 (50)	16.8 (0)
0.6	-30.1 (50)	-19.3 (50)	15.3 (0)
0.7	-34.3 (50)	-18.1 (50)	14.1 (0)
0.8	-34.5 (50)	-17.0 (50)	13.0 (0)
0.9	-33.6 (50)	-15.9 (50)	12.1 (0)
1.0	-32.1 (50)	-15.0 (50)	11.4 (0)

$n = 50, p_{mean} = 100, \alpha = 10[\%]$

$w_{mean} = \gamma n \times p_{mean}, CV = 0.5$

(): The number of problem instances for which each algorithm found better schedules than the optimal permutation schedules (the total number of instances is 50)

Table 3 Performance of algorithms (3)
Relative Deviation [%]

γ	A	$I-OP$	FCFS
0.1	13.4 (0)	0.0 (0)	15.3 (0)
0.2	12.0 (0)	-4.3 (42)	23.8 (0)
0.3	-0.8 (29)	-14.9 (50)	21.3 (0)
0.4	-11.7 (48)	-20.3 (50)	18.9 (0)
0.5	-20.5 (50)	-18.6 (50)	16.9 (0)
0.6	-27.7 (50)	-17.6 (49)	15.3 (0)
0.7	-33.0 (50)	-16.8 (50)	14.0 (0)
0.8	-34.1 (50)	-15.8 (50)	12.9 (0)
0.9	-32.7 (50)	-14.8 (50)	12.0 (0)
1.0	-30.9 (50)	-14.9 (50)	11.2 (0)

$n = 50, p_{mean} = 100, \alpha = 20[\%]$

$w_{mean} = \gamma n \times p_{mean}, CV = 0.5$

(): The number of problem instances for which each algorithm found better schedules than the optimal permutation schedules (the total number of instances is 50)

Table 4 Performance of algorithms (4)
Relative Deviation [%]

γ	A	$I-OP$	FCFS
0.1	16.0 (0)	0.0 (0)	16.5 (0)
0.2	22.0 (0)	-3.1 (43)	23.5 (0)
0.3	12.8 (0)	-10.7 (50)	20.6 (0)
0.4	0.8 (24)	-17.6 (50)	18.1 (0)
0.5	-9.4 (49)	-17.9 (50)	16.2 (0)
0.6	-17.8 (50)	-16.9 (50)	14.6 (0)
0.7	-24.7 (50)	-15.4 (49)	13.3 (0)
0.8	-30.2 (50)	-14.5 (50)	12.2 (0)
0.9	-31.3 (50)	-13.6 (50)	11.3 (0)
1.0	-29.8 (50)	-12.9 (50)	10.5 (0)

$n = 50, p_{mean} = 100, \alpha = 50[\%]$

$w_{mean} = \gamma n \times p_{mean}, CV = 0.5$

(): The number of problem instances for which each algorithm found better schedules than the optimal permutation schedules (the total number of instances is 50)

algorithm $I-OP$ for the operation time ratio γ no less than 0.5. In Tables 2, 3 and 4, algorithm A yields better schedules than the optimal permutation algorithm OP for the operation time ratio γ no less than 0.3, 0.3 and 0.5, respectively, and better than the improved optimal permutation algorithm $I-OP$ for the operation time ratio γ no less than 0.5, 0.5 and 0.6, respectively. However, in Table 5, algorithm A yields no better schedules than the optimal permutation algorithm OP for the operation time ratio γ less than 0.8, and no better than the improved optimal permutation algorithm $I-OP$ for the operation time ratio γ less than 1.0.

From the results of algorithms A and $I-OP$ in

Table 5 Performance of algorithms (5)
Relative Deviation [%]

γ	A	I-OP	FCFS
0.1	11.5 (0)	-0.0 (3)	14.5 (0)
0.2	20.7 (0)	-0.4 (22)	21.3 (0)
0.3	19.9 (0)	-0.7 (30)	20.8 (0)
0.4	17.5 (0)	-2.2 (44)	18.8 (0)
0.5	14.4 (0)	-4.8 (48)	16.9 (0)
0.6	11.1 (0)	-8.3 (50)	15.4 (0)
0.7	5.6 (3)	-11.0 (50)	14.0 (0)
0.8	-0.9 (32)	-12.3 (50)	12.9 (0)
0.9	-6.8 (47)	-12.2 (50)	12.0 (0)
1.0	-11.9 (50)	-11.9 (49)	11.2 (0)

$n = 50$, $p_{mean} = 100$, $\alpha = 100$ [%]

$w_{mean} = \gamma n \times p_{mean}$, $CV = 0.5$

(): The number of problem instances for which each algorithm found better schedules than the optimal permutation schedules (the total number of instances is 50)

all the tables, we can mention that we had better consider the non-permutation optimization for maximizing the system efficiency (i.e., for minimizing the maximum completion time) when the operation times on the intermediate station are relatively large to the total processing time on the machines, and when the number of jobs with non-zero operation times is relatively small to the total number of jobs to be processed.

6. Conclusion

In this paper, we considered the scheduling problem for a two-machine robotic unit with an intermediate station. The non-permutation problem (i.e., the original problem) is NP-hard, while the permutation version of the problem can be solved in polynomial time. It is already known that, if the optimal permutation schedule is used as an approximate solution for the non-permutation problem, it gives a maximum completion time within twice the optimal. However, still this simplicity is bought at the price of drastically inferior schedules in some cases. In this paper, we proposed a different polynomial time approximate algorithm with a performance guarantee, by using a relaxation of the non-permutation problem to the single-machine problem with delivery times, such that its non-permutation schedule also gives a maximum completion time within twice the optimal. This result can be viewed as an extension of Dell'Amico's[5], since the scheduling problem is generalized from the two-machine flowshop with time lags by allowing the loading, unloading, transportation and return times of robots to be non-negligible. We then provided the numerical results for the proposed algorithm's performance, which showed that it obtained better schedules than the optimal permutation schedules when the

operation times on the intermediate station are relatively large to the total processing time on the machines, and when the number of jobs with non-zero operation times is relatively small to the total number of jobs to be processed.

Acknowledgements

This research was partially supported by a Scientific Grant-in-Aid from the Ministry of Education, Science, Sports and Culture of Japan.

References

- [1] A. Agnetis, D. Pacciarelli and F. Rossi: Lot scheduling in a two-machine cell with swapping devices; *IIE Transactions*, Vol. 28, pp. 911-917 (1996)
- [2] Association of Mechanical Technology of Japan (ed.): *A Collection of FMS's in Japan*, Machinist Publisher, in Japanese (1982)
- [3] J. Blazewicz, K. Ecker, G. Schmidt and J. Weglarz: *Scheduling in Computer and Manufacturing Systems*, Springer-Verlag (1993)
- [4] J. -L. Cheng and H. Kise: Optimal scheduling for automated two-machine manufacturing systems with intermediate operations; *Transactions of the Institute of System, Control and Information Engineers*, Vol. 8, No. 9, pp. 491-497, in Japanese (1995)
- [5] M. Dell'Amico: Shop problems with two machines and time lags; *Operations Research*, Vol. 44, No. 5, pp. 777-787 (1996)
- [6] M. Garey and D. Johnson: *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Company (1979)
- [7] N. G. Hall, H. Kamoun and C. Sriskandarajah: Scheduling in robotic cells: classification, two and three machine cells; *Operations Research*, Vol. 45, No. 3, pp. 421-439 (1997)
- [8] S. M. Johnson: Optimal two- and three-stage production schedules with setup times included; *Naval Research Logistics Quarterly*, Vol. 1, No. 1, pp. 61-68 (1954)
- [9] H. Kise, Y. Karuno and J. -L. Cheng: Optimal permutation and non-permutation scheduling for a two-machine robotic unit with an intermediate station; *Proceedings of the IASTED International Conference, Robotics and Applications 2000*, pp. 241-247 (2000)
- [10] H. Kise: On an automated two-machine flowshop scheduling problem with infinite buffer; *Journal of the Operations Research Society of Japan*, Vol. 34, No. 3, pp. 354-361 (1991)
- [11] H. Kise, T. Shioyama and T. Ibaraki: Automated two-machine flowshop scheduling: a solvable case; *IIE Transactions*, Vol. 23, No. 1, pp. 10-16 (1991)
- [12] E. Levner, K. Kogan and I. Levin: Scheduling a two-machine robotic cell: a solvable case; *Annals of Operations Research*, Vol. 57, pp. 217-232 (1995)
- [13] L. G. Mitten: Sequencing n jobs on two machines with arbitrary time lags; *Management Science*, Vol. 5, pp. 293-297 (1959)
- [14] S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz and W. Kubiak: Sequencing of parts and robot moves in a robotic cell; *The Inter-*

national Journal of Flexible Manufacturing Systems,
Vol. 4, pp. 331–358 (1992)

- [15] D. R. Sule: Sequencing n jobs on two machines with setup, processing and removal times separated; *Naval Research Logistics Quarterly*, Vol. 29, No. 3, pp. 517–519 (1982)