Engineering, 2014, 6, 877-885
Published Online December 2014 in SciRes. http://www.scirp.org/journal/eng
http://dx.doi.org/10.4236/eng.2014.613081

Scientific Research

A Branch-and-Bound Based Heuristic Algorithm for Minimizing Makespan in Machining-Assembly Flowshop Scheduling

Kazuko Morizawa

Graduate School of Engineering, Osaka Prefecture University, Osaka, Japan
Email: morizawa@eis.osakafu-u.ac.jp

Received 11 September 2014; 31 October 2014; accepted 23 November 2014

Abstract

This paper proposes a heuristic algorithm, called list-based squeezing branch and bound algorithm, for solving a machine-fixed, machining-assembly flowshop scheduling problem to minimize makespan. The machine-fixed, machining-assembly flowshop consists of some parallel two-machine flow lines at a machining stage and one robot at an assembly stage. Since an optimal schedule for this problem is not always a permutation schedule, the proposed algorithm first finds a promising permutation schedule, and then searches better non-permutation schedules near the promising permutation schedule in an enumerative manner by elaborating a branching procedure in a branch and bound algorithm. The results of numerical experiments show that the proposed algorithm can efficiently provide an optimal or a near-optimal schedule with high accuracy such as mean relative error being less than 0.2% and the maximum relative error being at most 3%.

Keywords

Scheduling, Heuristic, Branch and Bound Algorithm, Machining-Assembly Flowshop, Makespan

1. Introduction

Recently manufacturers face to more competitive situation, because of shorten product life cycles and diversification of products. Flexible Manufacturing Cell (FMC) has attracted a lot of attention as a production system to cope with a multi-product, small-lot production efficiently in such a situation. The FMC usually consists of two stages: A machining stage with some parallel machines (or flow lines) and an assembly stage with a few robots. Sun *et al*. [1] formulate the scheduling problem for minimizing makespan in an FMC as Machining-Assembly Flowshop Scheduling (shortly MAFS) problem to minimize makespan, and divide into two cases: A machine-

fixed case and a machine-unfixed case.

This paper deals with the machine-fixed MAFS problem with L (≥ 2) parallel, two-machine flow lines at the machining stage and one assembly robot at the assembly stage. Each of L component parts of any job is processed on a prespecified two-machine flow line at the machining stage, and these parts are assembled on an assembly robot at the assembly stage after all component parts have been completed. Although two-stage flow-shop scheduling problems with one machine at each stage to minimize makespan can be solved efficiently by Johnson's algorithm [2], the machine-fixed, MAFS problem is strongly NP-complete, even when the machining stage consists of two parallel machines [3] [4]. In case of MAFS with two-lines consisting of two machines at the machining stage, a branch and bound algorithm (shortly B & B) [5] and a heuristic method [6] have been proposed to solve a small-sized and a large-sized problem, respectively.

This paper proposes a kind of hybrid heuristic algorithms, incorporating a local search procedure into the squeezing B & B [7] [8]. The performance of the proposed method is compared with a branch and bound algorithm with a limited computation time of one hour. Numerical experiments solving one hundred instances generated randomly for each problem size are implemented to demonstrate that the proposed heuristic method can efficiently provide near-optimal schedules with high accuracy.

2. Scheduling Model

This paper deals with a machine-fixed MAFS model with the following conditions:

- A machining stage consists of L (≥ 2) parallel flow lines with two non-identical machines, named M_{lk}, $l = 1, 2, \cdots, L$, $k = 1, 2$, and an assembly stage consists of one robot M_2 (See **Figure 1**).
- Each of N jobs has L parts and these parts are processed on the pre-specified lines at the machining stage and assembled on a robot at the assembly stage.
- Any assembly operation for each job cannot be started until machining operations for all parts of each job have been completed.
- Machining time of l th parts of job J_i, p_{ilk}, $l = 1, 2, \cdots, L$, $k = 1, 2$, and assembly time, p_{i2}, $i = 1, 2, \cdots, N$, are all given constant.
- Setup time is independent of job sequence and included in each processing time.
- Transfer time between machines is negligible.
- All jobs are ready at time zero, and no job can be split or preempted.
- No machine can process more than one operation at a time, and all machines are always available during a scheduling period.

The scheduling criterion is to minimize makespan $F_{\max}$.

It has proved that the best permutation schedule is not always optimal to this scheduling problem [5]. But, fortunately, it can be shown that FCFS (First Come First Served) rule provides an optimal assembly schedule to minimize makespan under a set of given schedules for machining flow line s_l, $l = 1, 2, \cdots, L$. Therefore, it is sufficient to consider only one assembly schedule given by FCFS rule for each permutation/non-permutation schedule at the machining stage.

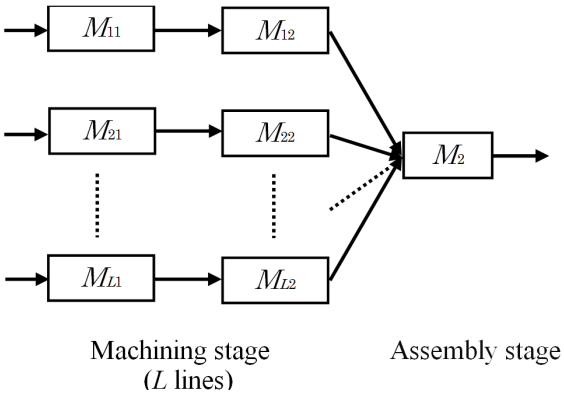


Figure 1. Scheduling model.

3. Heuristic Algorithm

3.1. Basic Concept

Since the MAFS problem treated in this paper is NP-complete, we propose an efficient heuristic method, called "List-Based Squeezing Branch and Bound Algorithm (LSQ)". The LSQ is a B & B-based local search algorithm elaborated for improving the efficiency of the squeezing B & B [7] [8] for the large-sized problems.

The squeezing B & B is a heuristic method which aims at obtaining a near-optimal schedule as close to the optimum as possible within a given computation time. In the squeezing B & B, parent nodes to be branched at branching level v are selected up to $NN(v)$ according to the minimum lower bound rule and are searched in parallel. $NN(v)$ is defined as $NN(v)=\min\{NL(v),NLB(v,\alpha)\}$, and $NN(0)=1$ (the root node is set as level 0). The notation $NLB(v,\alpha)$ stands for the number of nodes whose lower bounds are less than or equal to $(1+\alpha)LB_{\min}(v)$ for $v\geq 1$, where $LB_{\min}(v)$ is the minimum lower bound obtained at level v and α is a pre-specified parameter to control the size of $NLB(v,\alpha)$ so that $0\leq\alpha\leq 0.5$. Any node whose lower bound is larger than the value of $(1+\alpha)LB_{\min}$ is not selected as a parent node for next branching. Variety of the function $NL(v)$ specifies a squeezing pattern like $NL(v)=NL(0)$ (constant squeezing); $NL(0)-av$ (linear squeezing, $a>0$); $NL(0)-av^{b}$ (calm squeezing, $a>0$, $b>1$); $NL(0)-av^{1/b}$ (rapid squeezing, $a>0$, $b>1$).

After $NN(v)$ parent nodes are selected at branching level v, $(N-v)NN(v)$ child nodes are generated from $NN(v)$ parent nodes because the number of unscheduled jobs is just $N-v$ for each parent node (in permutation scheduling phase). From among these child nodes, $NN(v+1)$ nodes are selected as next parent nodes for further expansion of the search tree in the same way as the above.

The procedure is terminated when the branching process reaches the bottom level of the search tree, and then the best schedule is selected from among the schedules obtained at the bottom level as the solution by the squeezing B & B.

Since the squeezing B & B does not implement any backtracking, the time complexity of the squeezing B & B can be controlled by the $NL(v)$, e.g., it is at most $O(N^3)$ for branching if $NL(v)$ is proportional to N. To reduce the time complexity of the squeezing B & B, the LSQ selects some jobs from among the set of unscheduled jobs for generating child nodes from each parent node according to a "job-list". The number of jobs to be selected in this procedure is pre-specified as $X(<N-v)$. If the job sequence of the job-list is close to an optimal schedule, it can be expected that the restriction of unscheduled jobs to be branched by this method does not deteriorate the effectiveness of the proposed method and the time complexity of it is reduced to $O(N^2)$ for branching.

This branching procedure is called "list-based squeezing" and the B & B-based parallel search algorithm using the list-based squeezing is called list-based squeezing Branch and Bound algorithm (LSQ). In the same way as the squeezing B & B, the basic procedure of the LSQ is terminated when the branching process reaches the bottom level of the search tree, and then the best schedule obtained at the bottom level is selected as the solution. The quality of the solution can be improved by implementing the LSQ iteratively according to the new job-list which is renewed by the current best schedule with a better value of the performance measure than that of the current job-list.

Since the job-list in the LSQ is corresponding to an initial solution in a general local search procedure, the LSQ can be also considered as a kind of local search algorithms which searches neighborhood of an initial schedule in an enumerative manner according to the lower bound like a branch and bound algorithm and the size of neighborhood is restricted by the value of X. Therefore, the LSQ can be widely applied to any scheduling problems likewise a local search procedure.

For the MAFS problem of this paper, the LSQ is applied as a two-phase heuristic search algorithm. In the first phase, a promising permutation schedule is searched according to a job-list and then better non-permutation schedules are searched according to both some job-lists for non-permutation scheduling and the best permutation schedule obtained in the first phase. In both phases, the LSQ is implemented iteratively.

3.2. Job-List

A job-list used in the LSQ is an initial schedule for searching better schedules. In the LSQ, the job-list is obtained first by using any promising heuristic method and the neighborhood is searched in an enumerative man-

ner by employing a restricted branching procedure according to the job-list. The following four heuristic methods are proposed for obtaining a job-list for permutation scheduling to the MAFS problem.

1) Find a machining flow line l^* which satisfies $l^* \equiv \arg\left\{\max_{1\le l\le L}\sum_{i=1}^{N}\left(p_{il1}+p_{il2}\right)\right\}$ and construct the artificial two-machine flowshop problem with nominal processing times of $\left(\left(p_{il*1}+p_{il*2}\right)/2, p_{i2}\right)$ from the original MAFS problem. By applying Johnson's algorithm [2] to the artificial two-machine problem, an approximate permutation schedule for the original problem is obtained.

2) Construct L kinds of artificial three-machine flowshop problems with nominal processing times of $\left(p_{il1}, p_{il2}, p_{i2}\right)$, $l=1,2,\cdots,L$, from the original MAFS problem. By applying Rapid Access procedure (RA) [9] to each of the artificial three-machine flowshop problem, at most L kinds of approximate permutation schedules for the original problem are obtained.

3) Find a machining flow line l_i^* which satisfies $l_i^* \equiv \arg\left\{\max_{1\le l\le L}\left(p_{il1}+p_{il2}\right)\right\}$ for each J_i, $i=1,2,\cdots,N$. Construct an artificial three-machine flowshop problem with nominal processing times of $\left(p_{ili*1}, p_{ili*2}, p_{i2}\right)$ from the original MAFS problem. By applying the RA procedure to the artificial three-machine flowshop problem, an approximate permutation schedule for the original problem is obtained.

4) For each J_i, $i=1,2,\cdots,N$, generate at most 2L kinds of approximate permutation schedules by sequencing J_i at the first position and followed by $\overline{s_{lu3}}^*$, $u=1,2$, $l=1,2,\cdots,L$ ($\overline{s_{lu3}}^*$ is introduced for calculateing a lower bound $LB_1(s)$ and in this case it consists of (N-1) jobs except for J_i. The details are described in Subsection 3.4).

Select a schedule with the minimum makespan for the original MAFS problem from among the set of schedules generated by the above four heuristic methods and set the schedule as the job-list for permutation scheduling, denoted by $L1[1]$, $L1[2]$, $\cdots$, $L1[N]$.

The job-lists for non-permutation scheduling are obtained as follows:

1) Consider L parallel two-machine flow lines at the machining stage. By applying Johnson's algorithm to each line, L kinds of schedules s_l^{Johnson} are obtained for flow line l, $l=1,2,\cdots,L$, resulting in a job-list $L2[l][1]$, $L2[l][2]$, $\cdots$, $L2[l][N]$, $l=1,2,\cdots,L$.

2) Adopt the best permutation schedule obtained in the first phase to a job-list for non-permutation scheduling, resulting in a job-list $L2[L+1][1]$, $L2[L+1][2]$, $\cdots$, $L2[L+1][N]$.

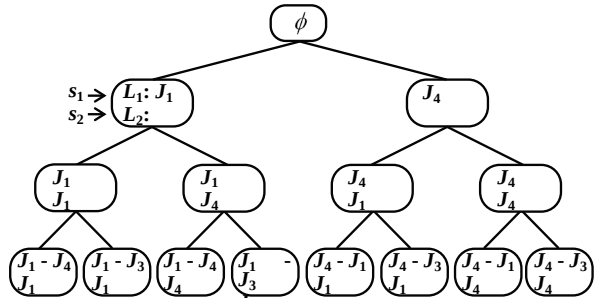
These two kinds of job-lists are used for selecting some unscheduled jobs to be branched in the non-permutation scheduling phase. Select first X unscheduled jobs according to each of these job-lists and generate at most $2X$ (at least X) child nodes by branching the selected jobs.

3.3. Branching Rules

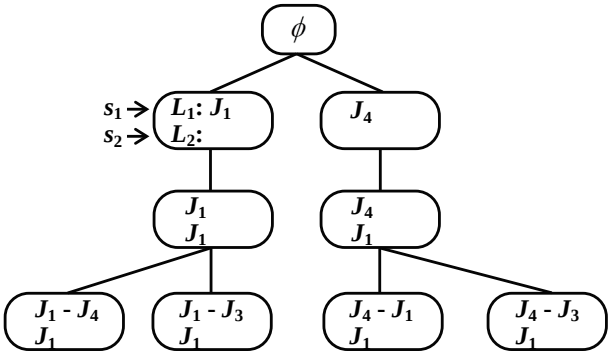
In the permutation scheduling phase of the proposed algorithm, the ordinary branching rule which generate nodes for the rth job in a schedule at branching level r, $r=1,2,\cdots,N$, is adopted. On the other hand, to search non-permutation schedules effectively, we adopt a branching rule for non-permutation scheduling proposed by Miyake *et al*. [5]. Since the jobs sequenced at the rth position on machining flow lines in a non-permutation schedule are not always the same, the branching rule generates nodes at each branching level by considering a schedule only for one of machining flow lines step by step. Concretely speaking, it generates nodes for the rth job in a schedule on machining flow line l at branching level $v=L(r-1)+l$, $v=1,2,\cdots,LN$, where $r=1,2,\cdots,N$, $l=1,2,\cdots,L$. We call this branching rule "all line search".

Furthermore, another branching rule is also proposed for more effective non-permutation scheduling. In this branching rule, find first a bottleneck machining flow line $l_{\max}$, where completion time of the last job is the latest among L machining flow lines in the current best schedule. Then fix the schedules on the machining flow lines except for $l_{\max}$ as the current best one and generate nodes for the rth job in a schedule on machining flow line $l_{\max}$ at branching level $v=L(r-1)+l_{\max}$, $r=1,2,\cdots,N$. We call this branching rule "bottleneck line search".

These two kinds of branching rules for non-permutation scheduling are illustrated in **Figure 2(a)** and **Figure 2(b)**, respectively.



(a)



(b)

Figure 2. Example of branching trees generated in non-permuation scheduling phase.(a) In case of all line search; (b) In case of bottleneck line search ($L=2$, $N=4$, $X=2$, $NN(v)=4$, $v=1,2,\cdots,LN$, $L2[1][i]=\{J_1,J_4,J_3,J_2\}$, $L2[2][i]=\{J_4,J_1,J_2,J_3\}$, $L2[3][i]=\{J_1,J_4,J_3,J_2\}$).

3.4. Lower Bound

Since the LSQ selects parent nodes to be branched according to the minimum lower bound rule, introducing the tight lower bound is important for getting a better performance. It is, however, very hard to define a tight lower bound directly for the MAFS problem, because a set of unscheduled jobs for each machining flow line is not always the same as these of the other lines in the non-permutation scheduling phase. Therefore, we adopt the following lower bound of a partial schedule $s=(s_1,s_2,\cdots,s_L)$ for this MAFS problem [5]. The lower bound is calculated by applying by a tight lower bound for $N/M/F/F_{\max}$ problem [7] to L kinds of the artificial three-machine flowshop problems with nominal processing times of (p_{il1},p_{il2},p_{i2}), $l=1,2,\cdots,L$.

$$LB(s)=\max\{LB_1(s),LB_2(s),LB_3(s)\} \tag{1}$$

where

$$LB_1(s)=\max_{1\le l\le L}\max_{1\le u<v\le 3}\left\{t_{lu}+Z_{uv}\left(\overline{s^*_{luv}}\right)-\sum_{m=u+1}^{v-1}X_m\left(\overline{s_l}\right)+\min_{J_i\in J\left(\overline{s_l}\right)}p_{il,v+1}\right\},$$

$$LB_2(s)=\max_{1\le l\le L}\max_{1\le u<3}\left\{t_{l,k(u)}+\min_{s_l}Y^{\beta}_{k(u)u3}\left(\overline{s_l}\right)\right\},$$

$$LB_3(s)=\max_{1\le l\le L}\left\{t_{l3}+X_3\left(\overline{s_l}\right)\right\}.$$

Note that $p_{il3}=p_{i2}$, $i=1,2,\cdots,N$, $l=1,2,\cdots,L$, and $p_{il,v+1}=0$ for $v=3$. Notations used in the above equations are as follows:

$\underline{J(s_l)}$: set of unscheduled jobs at machining flow line l, $l=1,2,\cdots,L$.

$\underline{s_L}$: a schedule consists of all jobs in $J(s_l)$, $l=1,2,\cdots,L$.

$s_l(j)$: a schedule consists of all jobs in $J(s_l)$ except for J_j, $l=1,2,\cdots,L$.

t_{lm} : completion time at machine M_{lm} under a partial schedule s (M_{l3} corresponds to M_2 in this scheme), $l = 1, 2, \cdots, L$.

$Z_{uv}\left(\overline{s_l}\right)$: makespan of $\overline{s_l}$ for the nominal two-machine flowshop problem with a pair of nominal processing times of ($\sum_{m=u}^{v-1} p_{ilm}$, $\sum_{m=u+1}^{v} p_{ilm}$).

$\overline{s_{luv}^*}$: a schedule consists of all jobs in $\overline{J\left(s_l\right)}$ that minimize $Z_{uv}\left(\overline{s_l}\right)$. $\overline{s_{luv}^*}$ is easily obtained by applying Johnson's algorithm to the artificial two-machine flowshop problem with nominal processing times ($\sum_{m=u}^{v-1} p_{ilm}$, $\sum_{m=u+1}^{v} p_{ilm}$) for all jobs in $\overline{J\left(s_l\right)}$.

$X_m\left(\overline{s_l}\right)$: total processing time of all jobs in $\overline{J\left(s_l\right)}$ on M_{lm} (M_{l3} stands for M_2).

$X_m\left(\overline{s_l}\right) \equiv \sum_{Ji \in \overline{J(s_l)}} p_{ilm}$.

$$\min_{\overline{s_l}} Y_{k(u)u3}^{\beta}\left(\overline{s_l}\right) \equiv \min\left\{\sum_{m=k(u)}^{u-1} p_{[\xi]_{u3}^* lm} + \min_{\overline{s_l}} Y_{u3}^{b}\left(\overline{s_l}\right), \min_{J_j \in \overline{J(s_l)}}\left\{\sum_{m=k(u)}^{u} p_{jlm} + \min_{\overline{s_l(j)}} Y_{u3}^{b}\left(\overline{s_l}(j)\right)\right\}\right\}$$

where, $p_{[\xi]_{u3}^* lm}$ denotes the processing time of job which is sequenced at the first position in $\overline{s_{lu3}^*}$.

$$Y_{u3}^{b}\left(\overline{s_l}\right) = Z_{u3}\left(\overline{s_l}\right) - \sum_{m=u+1}^{2} X_m\left(\overline{s_l}\right), \quad 1 \le u < 3 .$$

$$k(u) \equiv \arg\left\{\max_{1 \le k \le u}\left\{t_{lk} + \sum_{m=k}^{u-1} p_{[\xi]_{u3}^* lm}\right\}\right\}.$$

3.5. Algorithm

The basic algorithm of the LSQ with bottleneck line search for the non-permutation scheduling for this MAFS problem is presented as follows:

Step 1: Select a squeezing pattern and specify the values of α , X and $NL(v)$, $v = 0, 1, 2, \cdots, N$. Set $s^* = \phi$ and $F_{\max}^* = \infty$.

Step 2: Find a schedule with minimum makespan from among the schedules generated by using four heuristics described in 3.2. Set the job sequence of the schedule as the job-list for permutation scheduling $L1[i]$, $i = 1, 2, \cdots, N$.

Step 3: Set $v = 0$ and $NN(0) = 1$.

Step 4: Generate N child nodes from $NN(v)$ parent nodes by sequencing each unscheduled job for the parent node at the first position (Note that the current parent node is the root node). Go to Step 6.

Step 5: Select first X ($X = N - v$, if $X > N - v$) unscheduled jobs for each parent node according to the job-list and generate $X \cdot NN(v)$ child nodes from the $NN(L)$ parent nodes. Set $v = v + 1$.

Step 6: Calculate the lower bound for each child node by using Equation (1). If there exists nodes whose lower bounds are larger than $(1+\alpha) LB_{\min}(v)$, remove them from the search tree.

Step 7: If $v = N$, then select the incumbent best schedule from among the current $NN(N-1)$ nodes representing the corresponding $NN(N-1)$ schedules and go to Step 10.

Step 8: Determine the number of nodes to be selected, that is $NN(v)$.

Step 9: Select the $NN(v)$ nodes in non-decreasing order of lower bound from among the nodes at the current branching level *v* as next parent nodes and return to Step 5.

Step 10: Set the incumbent best schedule as s^p and the makespan of s^p as $F_{\max}^p$. If $F_{\max}^p < F_{\max}^*$, then go to Step 11. Otherwise, go to Step 12.

Step 11: Set $F_{\max}^* = F_{\max}^p$ and $s^* = s^p$. Renew the job-list $L1[i]$, $i = 1, 2, \cdots, N$, by the job sequence of the incumbent best schedule s^* . Return to Step 3.

Step 12: Set the job sequence of the schedule obtained by applying Johnson's algorithm for each machining flow line l as the job-list for non-permutation scheduling $L2[l][i]$, $l = 1, 2, \cdots, L$, and job sequence of the best permutation schedule as $L2[L+1][i]$, $i = 1, 2, \cdots, N$.

Step13: Find a machining flow line of which completion time of the last job is the latest among L machin-

ing flow lines in $s^* = (s_1^*, s_2^*, \cdots, s_L^*)$ and set the line number $l_{\max}$.

Step 14: Set $v = 0$, $l = 1$, $r = 1$ and $NN(0) = 1$.

Step 15: If $l \neq l_{\max}$, then go to Step 17.

Step 16: Select first X ($X = N - v$, if $X > N - v$) jobs which are unscheduled at machining flow line l for each parent node according to the job-list $L2[l][i]$. Generate $X \cdot NN(v)$ child nodes from each parent node. Set $v = v + 1$. Go to Step 18.

Step 17: Generate a child node for each parent node by sequencing the job sequenced at the rth position in s_l^* immediately after the partial schedule for machining flow line l of the parent node. Set $v = v + 1$.

Step 18: Calculate the lower bound for each child node. If there exists nodes whose lower bounds are larger than $(1+\alpha) LB_{\min}(v)$, remove them from the search tree.

Step 19: If $v = LN$, then select the best schedule as s^n from among the current $NN(LN)$ nodes representing the corresponding $NN(LN-1)$ schedules and go to Step 22. Otherwise, set $l = l + 1$. If $l > L$, then set $l = 0$ and $r = r + 1$.

Step 20: Determine the number of nodes to be selected $NN(v)$.

Step 21: Select the $NN(v)$ nodes in nondecreasing order of lower bound from among the nodes at the current branching level as next parent nodes and return to Step 15.

Step 22: Set the makespan of the schedule s^n as $F_{\max}^n$. If $F_{\max}^n < F_{\max}^*$, then go to Step 23. Otherwise, go to Step 24.

Step 23: Set $F_{\max}^* = F_{\max}^n$ and $s^* = s^n$. Renew the job-list $L2[l][i]$, $l = 1, 2, \cdots, L$, $i = 1, 2, \cdots, N$, by the job sequence of the incumbent best schedule s^*. Return to Step 14.

Step 24: The schedule s^* is the solution by the proposed LSQ.

In this algorithm, the Steps 1-11 present the permutation scheduling procedure and Steps 12-24 present the non-permutation scheduling procedure.

For the case that all lines search is adopted in the non-permutation scheduling phase, Steps 13, 15 and 17 are removed from the above algorithm and Step 16 is replaced by the following Step 16'.

Step 16': Select first X ($X = N - v$, if $X > N - v$) jobs which are unscheduled at machining flow line l for each parent node according to $L2[l][i]$ and $L2[L+1][i]$, $i = 1, 2, \cdots, N$. Generate at most $2X \cdot NN(v)$ (at least $X \cdot NN(v)$) child nodes from each parent node. Set $v = v + 1$. Go to Step 18.

4. Numerical Experiments

4.1. Experimental Conditions

To evaluate the performance of the proposed algorithm, numerical experiments are implemented under the following conditions.

One hundred instances are generated for each combination of $N(=10, 15, 20, 30)$ and $L(=2, 3, 5)$, and are solved through the proposed algorithm and the branch and bound algorithm with a limited computation time of one hour proposed by Miyake *et al.* [5]. Machining times and assembly time for each job are integers generated randomly from a uniform distribution with the range of [1, 100]. In the proposed algorithm, the constant squeezing pattern is adopted as $NL(v) = 2N$, $v = 1, 2, \cdots, LN$, according to the results of preliminary experiment and the value of X is specified as $X = 5$. The following four kinds of the proposed algorithm with different settings, called LSQ(a)-(d), are implemented to solve each instance and the best schedule obtained by these four kinds of LSQ is selected as a solution by the proposed method.

- LSQ(a): The LSQ with α = 0.00 and "all line search" for non-permutation scheduling;
- LSQ(b): The LSQ with α = 0.00 and "bottleneck line search" for non-permutation scheduling;
- LSQ(c): The LSQ with α = 0.05 and "all line search" for non-permutation scheduling;
- LSQ(d): The LSQ with α = 0.05 and "bottleneck line search" for non-permutation scheduling.

All algorithms are coded in C-language and run it on a personal computer with CPU of Phenom II X6 3.20 GHz.

4.2. Results

Results of numerical experiments are summarized in **Table 1** and **Tables 2-4**, where "*ta*(%)" denotes the total average relative error in makespan of the schedule obtained by each heuristic method compared with the optimal

Table 1. Experimental results of LSQ(a)-(d) and the proposed method.

(*N*, *L*)		LSQ(a)	LSQ(b)	LSQ(c)	LSQ(d)	Proposed LSQ
(10, 2)	*ta*	0.39	0.41	0.63	0.64	0.18
	na	1.43	1.72	1.76	1.57	1.20
	m	6.26	6.26	4.73	4.76	2.98
	p	73	76	64	59	85
(30, 5)	*ta*	0.02	0.03	0.12	0.04	0.01
	na	0.27	0.33	0.54	0.33	0.18
	m	0.55	0.80	2.15	0.82	0.30
	p	91	92	77	87	93

Table 2. Experimental results of the proposed method and the one-hour-truncated B & B (*L* = 2).

N	10		15		20		30	
Method	B & B	LSQ	B & B	LSQ	B & B	LSQ	B & B	LSQ
ta	0.00	0.18	0.10	0.08	0.15	0.04	0.40	0.01
na	0.00	1.20	1.07	0.53	0.78	0.47	1.34	0.31
m	0.00	2.98	3.57	1.52	2.91	1.09	4.34	0.43
p	100	85	91	84	81	92	70	96

Table 3. Experimental results of the proposed method and the one-hour-truncated B & B (*L* = 3).

N	10		15		20		30	
Method	B & B	LSQ	B & B	LSQ	B & B	LSQ	B & B	LSQ
ta	0.02	0.13	0.07	0.10	0.27	0.06	0.80	0.01
na	0.42	0.95	0.77	0.96	1.30	0.81	1.61	0.16
m	0.96	2.87	2.44	2.05	4.29	1.95	6.37	0.40
p	96	86	91	90	79	93	50	95

Table 4. Experimental results of the proposed method and the one-hour-truncated B & B (*L* = 5).

N	10		15		20		30	
Method	B & B	LSQ	B & B	LSQ	B & B	LSQ	B & B	LSQ
ta	0.03	0.16	0.08	0.11	0.22	0.04	0.86	0.01
na	0.80	0.68	0.69	1.08	1.03	0.37	1.65	0.18
m	1.33	1.91	1.79	1.74	4.29	1.06	3.66	0.30
p	96	77	88	90	79	88	48	93

(or best) schedule. The “best” schedule means the best of all schedules obtained by all heuristic algorithms and the one-hour-truncated branch and bound algorithm, and this term is used when the branch and bound algorithm cannot provide the “optimal” schedule within one hour. The notation “*na* (%)” stands for the average relative error calculated for the set of non-optimal (or non-best) schedules, “*m*(%)” denotes the maximum relative error

for each method and "*p*(%)" means the fraction of instances solved (or instances for which the "best" schedules are obtained) by each method.

Table 1 shows the results of the proposed method and LSQ(a)-(d) for $(N, L) = (10, 2)$ and (30, 5). As shown in **Table 1**, the LSQ(a) derives the best performance in terms of "*ta*" among the LSQ(a)-(d), though the value of "*m*" is higher than the others in case of $(N, L) = (10, 2)$. The performance of the proposed method, however, is superior to LSQ(a) in all terms of "*ta*", "*na*", "*m*" and "*p*". This fact indicates that the LSQ(a)-(d) do not work well individually but work well cooperatively.

Tables 2-4 show the experimental results of the proposed method and the one-hour-truncated branch and bound algorithm [5] for L = 2, 3, 5, respectively. In these tables, "LSQ" denotes the proposed method and "B & B" denotes the one-hour-truncated branch and bound algorithm, respectively. From **Tables 2-4**, we find that the performance of "B & B" deteriorates as the problem size increases, *i.e.*, the fraction of instances solved by B & B is 48% for $(N, L) = (30, 5)$ and the maximum relative error is 6.37% for $(N, L) = (30, 3)$. On the other hand, the proposed heuristic can steadily provide near-optimal schedules with high accuracy. Although the values of *p*, the fraction of instances solved by the proposed method, is from 77% to 96%, the maximum relative error is at most 2.98% and the total average relative errors are less than 0.2%. The average computation time to solve an instance by the proposed method is at most 40 seconds even for the case of $(N, L) = (30, 5)$.

5. Conclusion

In this paper, a branch-and-bound-based heuristic algorithm, called "list-based squeezing branch and bound algorithm (LSQ)" is proposed for solving a machine-fixed, machining-assembly flowshop (MAFS) scheduling problem with L parallel two-machine flow lines at the machining stage and one assembly robot at the assembly stage. Since an optimal schedule to minimize makespan for this MAFS problem is not always a permutation schedule, two-phase search is implemented by using the LSQ. The first phase provides a promising permutation schedule and the second phase searches better non-permutation schedules near the permutation schedule. Results of numerical experiments show that the proposed LSQ efficiently provides optimal or near-optimal schedules with total average relative error is less than 0.2% and the maximum relative error is at most 3%.

References

[1] Sun, X., Morizawa, K. and Nagasawa, H. (2003) Powerful Heuristics to Minimize Makespan in Fixed, 3-Machine, Assembly-Type Flowshop Scheduling. *European Journal of Operational Research*, **146**, 498-516. http://dx.doi.org/10.1016/S0377-2217(02)00245-X

[2] Johnson, S.M. (1956) An Optimal Two- and Three-Stage Production Scheduling with Setup Time Included. *Naval Research Logistics Quarterly*, **1**, 61-68. http://dx.doi.org/10.1002/nav.3800010110

[3] Lee, C.-Y., Cheng, T.C.E. and Lin, B.M.T. (1993) Minimizing the Makespan in the 3-Machine Assembly-Type Flowshop Scheduling Problem. *Management Science*, **39**, 616-625. http://dx.doi.org/10.1287/mnsc.39.5.616

[4] Potts, C.N., Sevast'janov, S.V., Strysevich, V.A., Van Wassenhove, L.N. and Zwaneveld, C.M. (1995) The Two-Stage Assembly Scheduling Problem: Complexity and Approximation. *Operations Research*, **43**, 346-355. http://dx.doi.org/10.1287/opre.43.2.346

[5] Miyake, Y., Morizawa, K. and Nagasawa, H. (2002) A Branch-and-Bound Algorithm for Minimizing Makespan in a Machine-Fixed, Machining-Assembly Flowshop with Parallel Flowshop Lines. *Journal of Japan Industrial Management Association*, **53**, 292-301.

[6] Nagasawa, H. and Morizawa, K. (2002) Heuristic Scheduling in Machining-Assembly Flowshop with Parallel Two-Machine Flow Lines at Machining Stage. *Journal of Japan Industrial Management Association*, **53**, 37-46.

[7] Morizawa, K., Sun, X. and Nagasawa, H. (1998) Squeezing Branch and Bound Algorithm and Its Application to an N/M/F/F Max Problem. *Proceedings of* 1998 *Japan USA Symposium on Flexible Automation*, Otsu, 12-15 July 1998, 913-916.

[8] Morizawa, K., Sun, X. and Nagasawa, H. (2003) Squeezing Branch and Bound Algorithm for the Machine-Fixed, Machining-Assembly Flowshop Scheduling Problem. *International Journal of Manufacturing Technology and Management*, **5**, 20-27. http://dx.doi.org/10.1504/IJMTM.2003.002526

[9] Dannenbring, G.D. (1977) An Evaluation of Flowshop Sequencing Heuristics. *Management Science*, **23**, 1174-1182. http://dx.doi.org/10.1287/mnsc.23.11.1174