◈ Research Paper

Scheduling for a Two-Machine, M-Parallel Flow Shop to Minimize Makespan

Lee, Dong Hoon*
Lee, Byung Gun**
Joo, Cheol Min**
Lee, Woon Sik***

Abstract

This paper considers the problem of two-machine, M-parallel flow shop scheduling to minimize makespan, and proposes a series of heuristic algorithms and a branch and bound algorithm. Two processing times of each job at two machines on each line are identical on any line. Since each flow-shop line consists of two machines, Johnson's sequence is optimal for each flow-shop line. Heuristic algorithms are developed in this paper by combining a "list scheduling" method and a "local search with global evaluation" method. Numerical experiments show that the proposed heuristics can efficiently give optimal or near-optimal schedules with high accuracy.

Keywords: parallel flow shop; makespan; heuristic algorithm; branch and bound algorithm

1. Introduction

Flow shop scheduling to minimize makespan is one of traditional scheduling problems and is recently extended to some modified versions such as (i) flow shop scheduling with parallel machines at each stage [1][2][5], (ii) parallel flow shop scheduling [4][7], and (iii) machining-assembly flow shop scheduling[3][6]. The first version of flow shop scheduling is sometimes called the"hybrid flow shop scheduling" [2]. This type of flow shop is employed to increase the capacity of a bottleneck stage by installing identical machines in parallel at this stage. If it is required to increase the capacity of the whole flow shop line, the second type of flow shop may be introduced to double its capacity.

Sundararaghavan et al. [7] proposed a heuristic algorithm for providing near optimal schedules to a parallel non-identical flow shop scheduling problem, which can be easily transformed into the above identical case. Although they concluded that the performance of their heuristic was good, their numerical experiments were so restricted. For instance, the coefficient of deviation of processing times is so small that their conclusion can not be extended to general cases. Therefore, we proposed a more effective heuristic algorithm to the parallel identical flow shop scheduling problem by combining a "list scheduling" method and a "local search with global evaluation" method [4].

* Department of Industrial Engineering, Osaka Prefecture University, Japan
** Department of Industrial Engineering, Dongseo University
*** Department of Industrial Engineering, Pukyong National University

2. Scheduling Model

We consider parallel identical flow-shop lines with the following conditions:

(1) Each job can be processed by any one of L flow-shop lines consisting of two machines $M_{\ell j}$, j=1,2, ℓ =1~L

(2) Processing times of job J_i at machines $M_{\ell j}$, j=1,2, denoted by p_{ij}, j=1,2, are identical on any flow-shop line, that is, independent of ℓ.

The criterion is to minimize makespan $F_{\max}$, the smallest completion time among L flow-shop lines.

Notations used in this paper are given below.

s_ℓ : partial schedule on line ℓ; $s \equiv (s_1, s_2, \cdots, s_L)$

$J(s)$: a set of jobs included in schedule s

$s_{\ell[i]}$: job number sequenced at position i in s_ℓ, $i=1\sim|J(s_\ell)|$

$t_{\ell j}(s_\ell)$: completion time at machine $M_{\ell j}$ under s_ℓ

$M(s)$: makespan for partial schedule s ($F_{\max}(s)= M(s)$, if s is a whole schedule.)

$\bar{s}$: any schedule on line ℓ for remaining jobs; $\bar{s} \equiv (\bar{s}_1, \bar{s}_2, \cdots, \bar{s}_L)$

3. Branch and Bound Algorithm

Since Johnson's rule provides an optimal schedule to minimize makespan in each two-machine flow-shop line once job assignment has been finished, it suffices to search all combinations of job assignment among flow-shop lines. Systematic enumeration of the whole combinations of job assignment can be implemented by modifying the branching method provided by Brah and Hunsucker [1].

A sample branching tree for the case of $(N, L)= (4, 2)$ is shown in Figure 1, where a square denotes job assignment to the next new flow-shop line and a circle denotes job assignment to the current line. The number in any circle or any square stands for the job number assigned to the corresponding line. The branching rules are as follows:

(1) The source node is a square node with the first job J_1; (2) Child nodes include only one square node including job with the smallest job number in the remaining jobs; (3) When the Lth square node is generated on any path, all the remaining jobs are assigned to the current line and branching along this path is terminated; (4) Any path includes just L square nodes, that is, if the number of remaining jobs is equal to the number of new lines, all remaining jobs are assigned to the new lines one by one and the path is closed; (5) Any remaining job with job number larger than the job number in the parent node can generate a new circle node, as long as no condition in rule (3) or(4) is not satisfied.

Consider a combination of a partial schedule s and any schedule $\bar{s}$ including all remaining jobs, denoted by $s \| \bar{s}$. Since Johnson's sequence is optimal for each line under a given job assignment, any set of jobs assigned to each line, $J(s_\ell \| \bar{s}_\ell)$, should be resequenced according to Johnson's rule, resulting in the schedule denoted by $[J(s_\ell \| \bar{s}_\ell)]^*$, $\ell = 1\sim L$, and the following relationship holds: $F_{\max}([J(s\|\bar{s})]^*) \leq F_{\max}(s\|\bar{s})$. However, if all jobs are renumbered according to

Johnson's order in advance and if a branching method follows the above rules, we get $[J(s_\ell \| \bar{s}_\ell)]^* = s_\ell \| \bar{s}_\ell$, $\ell = 1 \sim L$, and $F_{\max}([J(s\|\bar{s})]^*) = F_{\max}(s\|\bar{s})$.

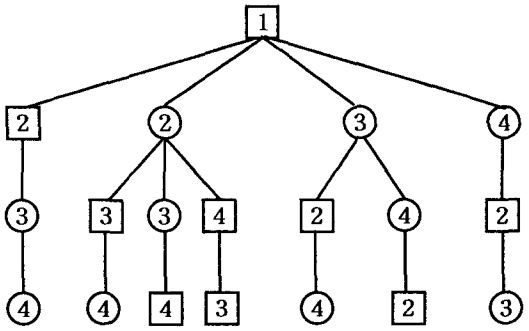


Fig 1. A sample branching tree: $(N, L) = (4, 2)$

Therefore, the following relationships hold:

$$\min_{\bar{s}} F_{\max}([J(s\|\bar{s})]^*) = \min_{\bar{s}} F_{\max}(s\|\bar{s})$$

$$= \min_{\bar{s}} \max_{1\le \ell \le L} \max\left\{ t_{\ell 1}(s_\ell) + M(\bar{s}_\ell),\ t_{\ell 2}(s_\ell) + \sum_{i\in J(\bar{s}_\ell)} p_{i2} \right\}$$

$$\ge \min_{\bar{s}} \max\left\{ \frac{1}{L}\sum_{\ell=1}^{L}[t_{\ell 1}(s_\ell) + M(\bar{s}_\ell)],\ \frac{1}{L}\sum_{\ell=1}^{L}[t_{\ell 2}(s_\ell) + \sum_{i\in J(\bar{s}_\ell)} p_{i2}] \right\}$$

$$\ge \max\left\{ \frac{1}{L}\left[\sum_{\ell=1}^{L} t_{\ell 1}(s_\ell) + M([J(\bar{s})]^*) \right],\ \frac{1}{L}\left[\sum_{\ell=1}^{L} t_{\ell 2}(s_\ell) + \sum_{i\in J(\bar{s})} p_{i2} \right] \right\}$$

Assume that a job has been just assigned on line L_0 with partial schedule s_{L_0} at the current branching node. Then, at this point, $L_0 - 1$ partial schedules for lines 1 through $L_0 - 1$ have been determined and there are $L - L_0$ lines remains empty with unscheduled jobs $J(\bar{s})$. Using this fact and the above relationship, we get the lower bound for a node with s as follows:

$$LB(s) \equiv \max\ \{ \max_{1\le \ell \le L} M(s_\ell),$$

$$\frac{1}{L-L_0+1}\ [t_{L_0 1}(s_\ell) + M([J(\bar{s})]^*)],$$

$$\frac{1}{L-L_0+1}\ [t_{L_0 2}(s_\ell) + \sum_{i\in J(\bar{s})} p_{i2}])$$

4. Heuristic Algorithms

Sundararaghavan et al. [7] suggested that the heuristic using a list based on a nonincreasing order of total processing time is superior to the others and its maximum deviation from the exact solution is only 3% for N=15. However, in their numerical experiments, processing times were generated from the uniform distribution on [5,35], decreasing the relative deviation compared to a case generated form a uniform distribution with a smaller lower limit and a wider range such as [1,100] used in this paper. Therefore, we proposes new heuristics by (1) constructing some better lists of

jobs, (2) assigning jobs to a flow-shop line with the smallest makespan calculated on the assumption that the incumbent job be assigned to each line, (3) making local search with some modifications of the current list, (4) keeping the best three lists for further search, and (5) performing all partial schedules on the assumption that all remaining jobs be scheduled through the smallest-makespan rule given in (2).

For the two-machine flow shop scheduling, we can illustrate many kinds of possible lists as shown in Figure 2, where "A"and "B" denote the set of jobs with $p_{i1} \leq p_{i2}$ and with $p_{i1} > p_{i2}$, respectively. The set $A \cup B$ stands for the whole set of jobs. the partial list $LS_A^{(1)}$ denotes a list of jobs in A in nondecreasing order of p_{i1}, and $\overline{LS}_A^{(1)}$ means its inverse list. Using these partial lists, we can express Johnson's order as $LS_A^{(1)} \circ LS_B^{(2)}$. In a similar way, we can construct a list $LS_{A\cup B}^{(1+2)}$ (or $\overline{LS}_{A\cup B}^{(1+2)}$) by sequencing jobs in nondecreasing (of nonincreasing) order of $p_{i1}+p_{i2}$, which is equivalent to the lists constructed by Sundararaghavan et al.[5]. Using a min-max type sequence, we can also construct a list of all jobs in nondecreasing order of $\min(p_{i1}, p_{i2})$ (or $\max(p_{i1}, p_{i2})$), denoted by $LS_{A\cup B}^{\min}$(or $LS_{A\cup B}^{\max}$), respectively.

Although many possible lists can be constructed by combining and modifying these lists, we employ the following promising lists based on Johnson's order.

LSA: Johnson's order ($LS_A^{(1)} \circ \overline{LS}_B^{(2)}$)

LSB: Inverse Johnson's order ($LS_B^{(2)} \circ \overline{LS}_A^{(1)}$)

LSC: Sequence job first in order, starting from job J_k with $p_{k1}+p_{k2} = \max_{i \in A\cup B}(p_{i1}+p_{i2})$, as shown in Figure 2, and then sequence remaining jobs in *LSA* order

LSD: Sequence jobs first in *LSA* order, starting from the above job J_k and then sequence remaining jobs in *LSB* order

LSE: Sequence all jobs in nondecreasing order of $\min(p_{i1}, p_{i2})$ ($LS_{A\cup B}^{\min}$)

LSF: Sequence all jobs in nonincreasing order of $Max(p_{i1}+p_{i2})$ ($\overline{LS}_{A\cup B}^{(1+2)}$)

LSG: Sequence jobs in $LS_A^{(1)} \circ \overline{LS}_B^{(1)}$ order

LSH: Sequence jobs in $LS_B^{(2)} \circ \overline{LS}_A^{(2)}$ order

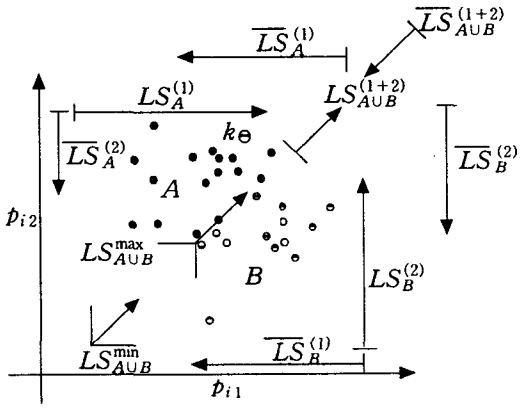
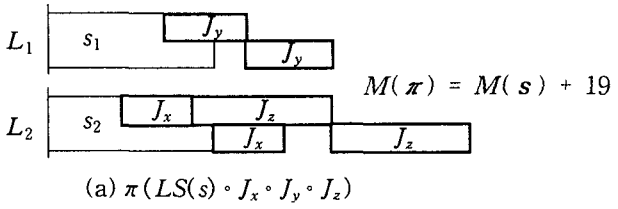


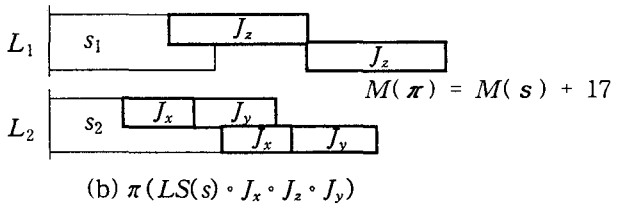
Fig 2. Possible lists of jobs

Let $\pi(PL) \equiv (s_1, s_2, \cdots, s_L)$ be a schedule vector generated by assigning next job to a line with the smallest makespan calculated on the assumption that each incumbent job is assigned to each line in Johnson's order, one by one according to partial list PL. For example, in Figure 3(a) where the partial list is given by $PL = LS(s) \circ J_x \circ J_y \circ J_z$ and $(M(s_{\ell 1}), M(s_{\ell 2}))=(7,10)$ for $\ell = 1$; (4,11) for $\ell = 2$, and $(p_{i1}, p_{i2})=(5,6)$ for $i = x$; (6,7) for $i = y$; (11,10) for i=z, job J_x is assigned to line L_2 because the resulting makespan becomes smaller than in case of assigning it to line L_1. In a similar manner, jobs J_y and J_z are assigned to lines L_1 and L_2, respectively.

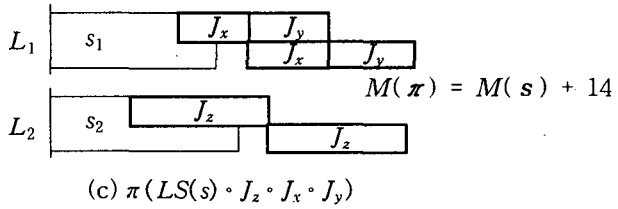
If the last two jobs in this list are exchanged with each other, a different schedule is obtained in the same manner as the above, reducing the makespan as shown in Figure 3(b). If the last job is inserted before the third last job in the original list, the makespan decreases further as shown in Figure 3(c). This type of reduction will occur when a job with a large total processing time follows two adjacent jobs with small total processing times in a list. Therefore, we implement a local search by checking these job-exchanges in the current list.



(a) $\pi(LS(s) \circ J_x \circ J_y \circ J_z)$



(b) $\pi(LS(s) \circ J_x \circ J_z \circ J_y)$



(c) $\pi(LS(s) \circ J_z \circ J_x \circ J_y)$

Fig 3. Basic concept of local search

Local search will generate a lot of candidate partial schedules but only a few partial schedules are selected for further search. However, some partial schedules may include a part of an optimal schedule. Therefore, we perform each partial schedule by assigning remaining jobs in the following manner.

Global Evaluation I. For a given list LS, a partial list PL and a partial schedule $\pi(PL)$, using the assignment rule employed in constructing $\pi(PL)$, we assign all remaining jobs to flow shop lines successively according to the remaining list $\overline{PL}(\equiv LS\backslash PL)$, and evaluate the tentative value of makespan $M(\pi(PL\circ\overline{PL}))$.

Global Evaluation II. For a given list LS, PL and $\pi(PL)$, lettering job $J_{[i]}$ be the next job following PL in LS, we assign $J_{[i]}$ to line L_l, and then we assign all remaining jobs to lines successively according to $\overline{PL}\backslash\{J_{[i]}\}$, using the assignment rule employed in constructing $\pi(PL)$. We denote tentative value of makespan $M(\pi(PL\circ\overline{PL})^{(l)})$ in this case. We repeat this evaluation procedure for $l=1\sim L$.

Global Evaluation III. Global evaluation III is the same as Global evaluation II except for replacing $J_{[i]}$ with $J_{\bar{k}}$, $\bar{k}=aug\{\max_{i\in\overline{PL}}(p_{i1}+p_{i2})\}$, and denoting the tentative value of makespan $M(\pi(PL\circ\overline{PL})^{(\ell)})$.

Combining the local search method and the global evaluation technique, we propose the following algorithm, called "Local Search with Global Evaluation(LSGE)" and propose its algorithm as follows:

〈 ***Local Search with Global Evaluation*** 〉

Step 1. Construct list $LS\equiv J_{[1]}\circ J_{[2]}\circ\cdots\circ J_{[N]}$.(Assume $N > L$)

Step 2. Set $i=L$, $PL=J_{[1]}\circ J_{[2]}\circ\cdots\circ J_{[L]}$, and $\pi(PL)=(J_{[1]}\circ J_{[2]}\circ\cdots\circ J_{[L]})$

Step 3. Calculate $M(\pi(PL\circ\overline{PL}))$ and set $\tilde{\pi}=\pi(PL\circ\overline{PL})$ and $UB=M(\tilde{\pi})$, where $\overline{PL}=LS\backslash PL$ satisfying $PL\circ\overline{PL}=LS$.

Step 4. Generate $\pi(PL\circ\overline{PL})^{(\ell)}$ and calculate $M(\pi(PL\circ\overline{PL})^{(\ell)})$ for $\ell=1\sim L$, where $\pi(PL\circ\overline{PL})^{(\ell)}$ denotes a schedule obtained by assigning job $J_{[i+1]}$ to line L_l after sequencing all jobs in PL and by continuing to sequence remaining jobs in $\overline{PL}\backslash\{J_{[i+1]}\}$.

Step 5. If $M(\pi(PL\circ\overline{PL})^{(\ell)}) < UB$, then set $\tilde{\pi}=\pi(PL\circ\overline{PL})^{(\ell)}$ and $UB=M(\tilde{\pi})$. Repeat this step for $\ell=1\sim L$.

Step 6. Generate $\overline{\pi}(PL\circ\overline{PL})^{(\ell)}$ and calculate $M(\overline{\pi}(PL\circ\overline{PL})^{(\ell)})$ for $\ell=1\sim L$, where $\overline{\pi}(PL\circ\overline{PL})^{(\ell)}$ denotes a schedule obtained by assigning jobs $J_{\bar{k}}$, $\bar{k}=aug\{\max_{i\in\overline{PL}}(p_{i1}+p_{i2})\}$ to line L_ℓ after sequencing all jobs in PL and by continuing to sequence jobs in $\overline{PL}\backslash\{J_{\bar{k}}\}$.

Step 7. If $M(\overline{\pi}(\mathrm{PL}\circ\overline{PL})^{(\ell)}) < UB$, then set $\tilde{\pi}=\overline{\pi}(\mathrm{PL}\circ\overline{PL})^{(\ell)}$ and $UB=M(\tilde{\pi})$.

Repeat this step for $\ell=1\sim L$.

Step 8. Set $i=i+1$ and $PL=PL\circ J_{[i]}$.

Step 9. Generate $\pi(PL)$ and calculate $M(\pi(PL))$.

Step 10. Construct $PL^{(1)}$ by exchanging the last two jobs in PL, and $PL^{(2)}$ by inserting the last job before the third last job in PL.

Step 11. Generate $\pi(PL^{(k)})$, $k=1,2$ and calculate $M(\pi(PL^{(k)}))$, $M(\pi(PL^{(k)} \circ \overline{PL}))$, $k=1,2$

Step 12. If $M(\pi(PL^{(k)} \circ \overline{PL})) < UB$, then set $\tilde{\pi} = \pi(PL^{(k)} \circ \overline{PL})$ and $UB=M(\tilde{\pi})$. Repeat this step for $k=1,2$.

Step 13. Rename $PL, PL^{(1)}, PL^{(2)}$ as PL_j, $j=1\sim3$, so that $M(\pi(PL_1)) \leq M(\pi(PL_2)) \leq M(\pi(PL_3))$.

Step 14. If $i=N$, then go Step 28.

Step 15. Generate $\pi(PL_j \circ \overline{PL_j})^{(\ell)}$ in the same way as in Step 4 and calculate $M(\pi(PL_j \circ \overline{PL_j})^{(\ell)})$ for $\ell=1\sim L$ and $j=1\sim3$.

Step 16. If $M(\pi(PL_j \circ \overline{PL_j}))^{(\ell)} < UB$, then set $\tilde{\pi} = \pi(PL_j \circ \overline{PL_j})^{(\ell)}$ and $UB=M(\tilde{\pi})$. Repeat this step for $\ell=1\sim L$ and $j=1\sim3$.

Step 17. Generate $\overline{\pi}(PL_j \circ \overline{PL_j})^{(\ell)}$ in the same way as in Step 6 and calculate $M(\overline{\pi}(PL_j \circ \overline{PL_j})^{(\ell)})$ for $\ell=1\sim L$ and $j=1\sim3$.

Step 18. If $M(\overline{\pi}(PL_j \circ \overline{PL_j})^{(\ell)}) < UB$, then set $\tilde{\pi} = \overline{\pi}(PL_j \circ \overline{PL_j})^{(\ell)}$ and $UB = M(\tilde{\pi})$. Repeat this step for $\ell=1\sim L$ and $j=1\sim3$.

Step 19. Set $i=i+1$ and $PL_j = PL_j \circ J_{[i]}$, $j=1\sim3$.

Step 20. Generate $\pi(PL_j)$ and calculate $M(\pi(PL_j))$, $j=1\sim3$.

Step 21. Construct $PL_j^{(k)}$, $k=1,2$ for PL_j, $j=1\sim3$, in the same manner as in Step 10.

Step 22. Generate $\pi(PL_j^{(k)})$, $k=1,2$ and calculate $M(\pi(PL_j^{(k)}))$, $M(\pi(PL_j^{(k)} \circ \overline{PL_j}))$, $k=1,2$, $j=1\sim3$.

Step 23. If $M(\pi(PL_j^{(k)} \circ \overline{PL_j})) < UB$, then set $\tilde{\pi} = \pi(PL_j^{(k)} \circ \overline{PL_j})$ and $UB=M(\tilde{\pi})$. Repeat this step for $k=1,2$, $j=1\sim3$.

Step 24. From the lists, PL_j and $PL_j^{(k)}$, $k=1,2$, $j=1\sim3$, select the three values of $M(\pi(PL_*^*))$, removing a list so that no same schedule are included. (If such lists exist, break ties arbitrarily.) Rename these three lists PL_j, $j=1\sim3$, so that $M(\pi(PL_1)) \leq M(\pi(PL_2)) \leq M(\pi(PL_3))$.

Step 25. Calculate an indicator LB:

$LB = \min_{1\leq j\leq 3} \max \{M(\pi(PL'_j)), [\Sigma_{\ell=1}^{L} M(s_\ell(\pi(PL'_j))) + \Sigma_{j\in\overline{PL'_j}} \tilde{p}_j] / L\}$,

where PL'_j is a list obtained by removing the last two jobs from PL_j and $\overline{PL'_j} = LS \setminus PL'_j$, and $\tilde{p}_j = p_{j2}$ if $LS = LSA$; p_{j1} if $LS = LSB$; $\min(p_{j1}, p_{j2})$ otherwise.

Step 26. If $LB \geq UB$, then go to Step 29.

Step 27. Return to Step 14.

Step 28. If $M(\pi(PL_1)) < UB$, then set $\tilde{\pi} = \pi(PL_1)$ and $M(\tilde{\pi}) = M(\pi(PL_1))$.

Step 29. Stop: $\tilde{\pi}$ is the solution for list *LS*.

This LSGE algorithm is implemented for the above lists *LSA*~*LSH* and the best solution among them is selected as the final solution of the heuristics.

5. Numerical Experiments

All algorithms are codes in C language and are implemented with Pentium II Processor 450MHz. Processing times are generated randomly from the uniform distribution on [1,100]. 1000 instances are L=2, J=20. 100 instances are solved for L=3,5, J=20 and L=2,3,5, J=25,30. The results are shown in Table 1, where "*SKV*" means the solution obtained by the heuristic algorithm proposed by Sundararaghavan et al.[7]., "ta" denotes mean relative error in total, "na" denotes mean relative error in nonoptimal instances, "m" denotes the maximum relative error, "*p*%" denotes the fraction of optimal instances (%) and the number in [ ] denotes the fraction of optimal solutions obtained by the branch and bound algorithm within one hour.

For relatively small-sized problems, especially when N=10 (for L=2~5) or L=2 (for N=10~30), the proposed heuristic algorithm provides optimal schedules efficiently to almost all instances. The branch and bound(*B&B*) can one-hour are coincident with those obtained by Heuristic, meaning that those solutions are optimal or very close to optimal solutions.

For relatively large-sized problems, when N=15~30 and L=3~5, the Heuristic can also provide optimal or best solutions to more than half the instances, although the performance of the one-hour limited *B&B* deteriorates rapidly as the problem size increases. The performance of the *SKV* remains worse than of the Heuristic.

Table 1. The results of numerical experiments

N	Heuristic	*B&B*	*SKV*	Heuristic	*B&B*	*SKV*	Heuristic	*B&B*	*SKV*
10	L 2			L 3			L 5		
ta	0.0000	0.0000	0.0277	0.0002	0.0000	0.0370	0.0000	0.0000	0.0129
na	0.0051	0.0000	0.0307	0.0097	0.0000	0.0425	0.0000	0.0000	0.0419
m	0.0091	0.0000	0.1525	0.0207	0.0000	0.1905	0.0000	0.0000	0.2593
p	99.3	100.0[100]	9.8	98.1	100.0[100]	12.9	100.0	100.0[100]	69.1
15	L=2			L 3			L=5		
ta	0.0001	0.0000	0.0277	0.0015	0.0000	0.0384	0.0025	0.0000	0.0445
na	0.0026	0.0000	0.0307	0.0044	0.0000	0.0388	0.0097	0.0000	0.0484
m	0.0056	0.0000	0.1020	0.0156	0.0000	0.1155	0.0380	0.0000	0.1861
p	97.8	100.0[100]	4.8	66.4	100.0[100]	0.9	74.7	100.0[100]	8.1
20	L 2			L 3			L 5		
ta	0.0000	0.0000	0.0126	0.0013	0.0000	0.0289	0.0055	0.0169	0.0502
na	0.0019	0.0000	0.0136	0.0038	0.0000	0.0292	0.0107	0.0529	0.0507
m	0.0022	0.0000	0.0664	0.0122	0.0000	0.0739	0.0325	0.1400	0.1122
p	99.1	100.0[100]	7.1	66.0	100.0[100]	1.0	48.0	68.0[31.0]	1.0
25	L=2			L 3			L=5		
ta	0.0000	0.0000	0.0089	0.0002	0.1232	0.0277	0.0000	0.2719	0.0373
na	0.0017	0.0000	0.0095	0.0029	0.2124	0.0307	0.0000	0.2719	0.0373
m	0.0018	0.0000	0.0333	0.0047	0.3734	0.1525	0.0000	0.5379	0.1124
p	98.0	100.0[100]	7.0	92.0	42.0[0.0]	2.0	100.0	0.0[0.0]	0.0
30	L=2			L 3			L=5		
ta	0.0000	0.0000	0.0089	0.0000	0.4121	0.0163	0.0000	0.5592	0.0299
na	0.0000	0.0000	0.0091	0.0000	0.4121	0.0164	0.0000	0.5592	0.0299
m	0.0000	0.0000	0.0286	0.0000	0.4743	0.0462	0.0000	1.0809	0.0694
p	100.0	100.0[100]	3.0	100.0	0.0[0.0]	1.0	100.0	0.0[0.0]	0.0

6. Conclusion

A "Local Search with Global Evaluation(LSGE)" heuristic algorithm and a branch and bound method were proposed in parallel identical flow shop scheduling. Numerical experiments showed that the proposed heuristic can provide near-optimal solutions efficiently with high accuracy.

References

1. Brah, S.A. and J.L. Hunsucker, "Branch and Bound Algorithm for the Flow Shop with Multiple Processors", *Eur J Opl Res*, Vol.51, pp.88-99, 1991.

2. Gupta, J.N.D. and E.A. Tunc, "Schedules for a Two-stage Hybrid Flowshop with Parallel Machines at the Second Stage", Int *J Prod Res*, Vol.29, No.7, pp.1489-1502, 1991.

3. Hariri, A.M.A. and C.N. Potts, "A Branch and Bound Algorithm for the Two-Stage Assembly Scheduling Problem", *Eur J Opl Res*, Vol.103, pp.547-556, 1997.

4. Lee, D.,K. Morizawa and H. Nagasawa, "Two-machine Parallel Flow Shop Scheduling to Minimize makespan", *Proceedings of the 2nd APIEMS*, Kanazwa, Japan, pp. 217-220, 1999.

5. Nowicki, E. and C. Smutnicki, "The Flow Shop with Parallel Machines: A Tabu Search Approach", *Eur J Opl Res*, Vol.106, pp.226-253, 1998.

6. Sun, X., A. Nishiura, K. Morizawa and H. Nagasawa, "Machine-Fixed, Machining-Assembly Flow Shop Scheduling", *Japan Industrial Managament Association*, Vol.50,

No.5, pp283-289, 1999.

7. Sundararaghavan, P.S., A.S. Kunnathur and I. Viswanathan, "Minimizing Makespan in Parallel Flowshops", *J Opl Res Soc*, Vol.48, No.8, pp.834-842, 1997.