Journal of the Society of Korea Industrial and Systems Engineering
Vol. 32, No. 3, pp.188－193, September 2009.

Fork-and-Join 시스템에서의 일정계획 문제

윤상흠* · 이익선**†

*영남대학교 경영학부
**동아대학교 경영학과

A Scheduling Problem in Fork-and-Join System

Sang Hum Yoon* · Ik Sun Lee**†

*School of Management, Yeungnam University
**School of Business, Dong-A University

본 논문은 조립과 분리시스템이 혼합된 Fork-and-Join 시스템에서의 일정계획문제를 고려하고 있다. 최초 단계에서는 구성품단위로 분리가 발생하고 두 번째 단계에서는 부품생산단계에서 각 부품 또는 구성품이 서로 다른 설비와 경로를 통해 독립적으로 생산된 후 최종 조립단계로 이동하게 되고, 그곳에서 조립공정을 통해 제품으로 완성된다. 본 논문에서는 이러한 Fork-and-Join 시스템에서 최종완료시간(makespan)을 최소화 할 수 있는 발견적 해법을 제안하고 이 해법의 최악오차한계(worst-case error bound)가 2라는 것을 증명한다. 또한, 제안된 문제의 효과적인 3가지 하한값(lower bound)을 제시하고 다양한 수치실험을 통해 제안된 발견적 해법의 결과와 하한값과의 비교를 통해 제안된 해법이 성능이 우수함을 증명한다.

Keywords : Fork-and-Join System, Scheduling, Worst-case Error bound

1. 서 론

This paper considers a scheduling problem for a Fork-and-Join (called as F&J) system. The F&J is a combined system in which both the two-stage assembly and the two-stage disassembly operations are performed. The two-stage assembly system consists of multiple fabrication machines in the first stage and a final assembly machine in the second stage. Each fabrication machine produces its own type of component independently of the other machines. The assembly machine can start its processing for a final product only when all the components of the product are available from the precedent fabrication machines. This shop model often appears in producing more larger volume of products than any product produced in serial flowshops. For example, the body and the chassis, which are the components of a fire engine, can be manufactured independently and then brought into the assembly machine. Production of the body and the chassis can take place in parallel, but the final assembly machine cannot start its processing until the body and the chassis are delivered to the assembly machine (referring to Lee et al. [10]). Sun et al. [13] have suggested another application of flexible manufacturing cell for machining various components and assembling them into many different kinds of products in small lot. Similarly, in computation work, two or more sub-programs (tasks) are independently processed first at their own parallel processors and then gathered at a

논문접수일 : 2009년 07월 30일　논문수정일 : 2009년 09월 01일　게재확정일 : 2009년 09월 01일
† 교신저자 lis1007@dau.ac.kr
※ 이 논문은 동아대학교 학술연구비 지원에 의하여 연구되었음.

main processor for final data-processing, where they either wait for all of their siblings to finish processing or are put together when processing is done on all the siblings [6]. See Potts et al. [12], Hariri and Potts [5], Yoon and Sung [14], Koulamas and Kyparisis[9], and Duda and Czachorski [1] for more industrial applications and the related scheduling researches in the two stage assembly system.

The two-stage disassembly system is the reversal of the assembly system, which begins with the assembly machine and ending with the associated multiple fabrication machines. The disassembly systems provide applications in industry such as splitting products into their constituent components in a non-destructive manner, which frequently appears in operation of waste handling and repair facilities [3]. When a mishap to a large product occurs, the product may be overhauled through the work sequence of a main disassembly station and its subsequent diagnosis/repair shops for the disassembled sub-parts. For example, the overhaul of aircraft involves a disassembly configuration [11]. Individual aircraft is disassembled into major components and these components are repaired simultaneously in separate phases. This aspect requires splitting and parallel operational phases to be introduced into the overall overhaul model. In the phases, even if the subsequent parallel shops are idle, they cannot start processing until arrival of a new aircraft completing its disassembly operation at the main splitting station. We can also see O'Shea et al. [11] and Gungor and Gupta [4] for reviews of various disassembly planning problems.

The F&J is concerned with a combined system in which both the assembly and disassembly operations are performed. Consider a multiple processor system in which all the sub-programs of a program are assigned separately to different processors. In the system, once a final processor receives end-signs from all of its precedent processors, it is allowed to execute its processing; for example, combining all the sub-programs into a final form. In manufacturing systems, such sub-program processings can be considered as disassembly processings concerned with several productions initiated simultaneously. The F&J is often called a Fork/Join network in queueing network theory.

The objective of this paper is to find the schedule which minimizes the maximum completion time of all jobs (or makespan) in the F&J. We first suggest a powerful heuristic, and then prove the worst-case error bound of the heuristic is 2. Also, we suggest three lower bounds for the makespan problem in F&J and then evaluate the performance of the heuristic through comparing with the lower bounds.

2. Problem Descriptions

An F&J has one assembly machine and one disassembly machine. All jobs first visit the disassembly machine and then pass through m-2 parallel machines and the assembly machine in sequence.

The following basic assumptions will be considered in the proposed problem;

(a) individual tasks are not preemptable,
(b) ready times of all jobs are zero,
(c) all processing times are known and deterministic, and
(d) only schedules having the same sequence on all machines(permutation schedules) are considered.

In scheduling theory, the assumption that jobs can be processed by at most one machine at the same time is often adapted. However, the assumption is not required in the proposed problem having partly-synchronous processing features.

It seems that permutation schedules are dominant for all the proposed problems, because the same sequence on parallel machines is profitable. Though it remains an open question for F&J, the assumption (d) is reasonable since only permutation schedules may be feasible in many manufacturing situations. Some examples of these situations include systems with conveyors for material transfer between machines without sophisticated control devices for altering the order of jobs, in which each machine operates in FIFO(first in, first out) rule.

For convenience, some notations are introduced as follows;

i, j : subscripts to denote specific jobs,
$[i]$, $[j]$: subscripts to indicate job positions,
k : subscript to denote a specific machine,
p_{ik} : processing time of job J_i on machine k,
p_k : set of processing times on machine k, $p_k = \{p_{1k}, p_{2k} \cdots, p_{nk}\}$,
S, S^* : a complete schedule and the optimal schedule, respectively, and
$C_{\max}(S)$: makespan of S in F&J.

Specifically, the proposed problem is NP-complete, which can also be verified by the following simple reasoning : A 4-F&J(with 2 parallel machines in the second stage) is the same as a 3-machine flowshop when the processing times on the intermittent parallel machines are the same, that is, $p_{i1} = p_{i2} = \cdots = p_{i(m-2)}$, for $i = 1, 2, \cdots, n$. It is well known that the

3-machine flowshop problem for makespan is NP-complete (referring to Garey et al. [2]), though the associated permutation sequences are dominant.

In F&J, the idle time may exist at the intermittent machines and the makespan is the sum of total idle time and total processing time on M_a so that the optimal schedule is obtained by minimizing the total idle time. That is,

$$C_{\max}(S) = \sum_{i=1}^{n} I_{[i]a} + T_a,$$

$$C_{\max}(S^*) = \min_{S\,IN\,\Omega}\left\{\sum_{i=1}^{n} I_{[i]a}\right\} + T_a,$$

where $I_{[i]k}$: the idle time immediately prior to $J_{[i]}$ at M_k,
Ω : set of all permutations, and
T_a : sum of all processing times at M_a.

And, the idle times on M_a can be expressed as follows;

$$I_{[i]a} = \max\left\{\max_{1\le k\le m-2}\left\{\sum_{j=1}^{i}(p_{[j]k} + I_{[j]k})\right\} - \sum_{j=1}^{i-1} p_{[j]a} - \sum_{j=1}^{i-1} I_{[j]a},\ 0\right\},$$

for $i = 2, \cdots, n$,
$I_{[1]a} = \max_{2\le k\le m-2}\{p_{[1]k} + I_{[1]k}\}$

3. Heuristic and Lower Bounds

In this section, a powerful heuristic procedure for the proposed problem is suggested. The following theorem shows that any heuristic procedure generating a permutation schedule guarantees the worst case error bound of 3 for the given problem.

Theorem 1 : Let S^* be an optimal schedule and S_H be any permutation schedule. Then, the following relation holds;

$$C_{\max}(S_H)/C_{\max}(S^*) \le 3.$$

Proof : Let's define

$$Y = \sum_{i=1}^{n} p_{id} + \sum_{i=1}^{n} p_{ia} + \max_{1\le k\le m-2}\left\{\sum_{i=1}^{n} p_{ik}\right\}$$

For any permutation schedule S, we can note that the completion time of any job is less than or equal to Y. Hence, we have

$$C_{\max}(S) \le Y \tag{1}$$

From the definition, we trivially have

$$C_{\max}(S^*) \le Y/3.$$

From (1), the proof is confirmed.

Now, we propose a heuristic algorithm using machine aggregation and Johnson's rule[8].

H1 Heurisitc

Step 1 : Compute $y_i = \sum_{k=1}^{m-2} p_{i,k}/(m-2)$ for $i = 1, 2, \cdots, n$.

Step 2 : Obtain two Johnson's sequences for two two-machine flowshop problem instances, $\{p_{i,d},\ y_i | 1 \le i \le n\}$ and $\{y_i,\ p_{i,a} | 1 \le i \le n\}$, respectively.

Step 3 : Select the better one between the obtained sequences.

Let S_H and $C_{\max}(S)$ denotes the schedule generated by the above H1 heuristic and its makespan, respectively. Then, we have the following theorem.

Theorem 2 : Let S^* be an optimal schedule. Then, the following relation holds;

$$C_{\max}(S_H)/C_{\max}(S^*) \le 2.$$

Proof : Let S_1 and S_2 be the schedules generated by applying Johnson's rule to the two-machine flowshop problem instances $\{p_{i,d},\ y_i | 1 \le i \le n\}$ and $\{y_i,\ p_{i,a} | 1 \le i \le n\}$, respectively, and let $C^*_{\max}(S_1)$ and $C^*_{\max}(S_2)$ be the optimal makespans of the mentioned two-machine flowshop problems, respectively. Then, it is clear that

$$C_{\max}(S_H) \le \min\{C^*_{\max}(S_1) + T_a,\ C^*_{\max}(S_2) + T_d\},$$

where $T_a = \sum_{i=1}^{n} p_{i,a}$ and $T_d = \sum_{i=1}^{n} p_{i,d}$.

From this, we have

$$2C_{\max}(S_H) \le C^*_{\max}(S_1) + C^*_{\max}(S_2) + T_a + T_d \quad (2)$$

Now, consider the feature of S^*. Noting the $C^*_{\max}(S_1)$ and $C^*_{\max}(S_2)$ are the makespans of two-machine problem, trivially we have

$$C_{\max}(S^*) \ge \max\{C^*_{\max}(S_1),\ C^*_{\max}(S_2)\}.$$

Thus, it holds that

$$2C_{\max}(S^*) \ge C^*_{\max}(S_1) + C^*_{\max}(S_2). \quad (3)$$

From (2) and (3), and using the relation $2C_{\max}(S^*) \ge T_a + T_d$, we obtain the relation $C_{\max}(S_H)/C_{\max}(S^*) \le 2$.

This completes the proof.

Now, we propose three lower bounds. The lower bounds can be used in some branch-and-bound algorithms and used to evaluate the effectiveness of heuristics for large-sized test instances.

For convenience of expressing the lower bounds, some additional notations are introduced as follows;

σ : the partial sequence of a sub set of jobs,
σ' : the set of jobs that are not contained in σ, and
$C_{\sigma k}$: the completion time of a partial sequence σ on M_k, $k = d,\ 1,\ 2,\ \cdots,\ m-2,\ a$.

It is easy to check that following four expressions can be lower bounds for all schedules that begin with a given partial sequence σ are respectively

$$LB_1 = \sum_{i=1}^{n} p_{i,d} + \min_{i\in\sigma'}\{\max_{1\le k\le m-2}\{p_{i,k}\} + p_{i,a}\}$$

$$LB_2 = \max_{1\le k\le m-2}\left\{C_{\sigma k} + \sum_{i\in\sigma'} p_{ik}\right\} + \min_{i\in\sigma'}\{p_{ia}\}$$

and

$$LB_3 = \min_{i\in\sigma'}\{p_{id} + p_{ia}\} + \max_{1\le k\le m-2}\left\{\sum_{i=1}^{n} p_{ik}\right\}$$

The final lower bound is

$$LB = \max\{LB_1,\ LB_2,\ LB_3,\ LB_4\} \quad (4)$$

4. Computational Experiments

This section evaluates the performance trend of the suggested H1 heuristic and the lower bound, LB, through numerical tests. Several uniform distributions are considered, as commonly considered in the literature. The processing times, p_{id}, p_{ia} and $p_{ik}(k=1,\ \cdots,\ m-2)$, are generated from $U[1,\ 10]$. For the comparison test, the number of machines (m) is chosen from the set {4, 8, 12}, and the number of jobs (n) is chosen from the set {4, 6, 8, 10, 20, 40, 60, 80}. For each number of machines and jobs, 20 problems are generated to gather some statistics. The experiments are conducted on a 2.4 Ghz Pentium IV with 512 MB memory.

The computational results are summarized as in <Table 1>, which includes the solution gaps of the H1 heuristic, where the solution gaps, Gap1(%) and Gap2(%) are represented by $((Sol_H - Sol_{Opt})/Sol_{Opt}) \times 100$ and $((Sol_H - LB)/Sol_{Opt}) \times 100$, where Sol_H denotes the feasible solution value generated by the H1 heuristic, and Sol_{Opt} denotes the optimal solution value found by the full enumeration procedure.

To evaluate the effectiveness of the lower bound procedure, LB, the solution gap is evaluated, where the solution gap, Gap3(%) is represented by $((Sol_H - LB)/LB) \times 100$.

Moreover, in the table, NO1 and NO2 denote the number of problems (out of 20) which finds the optimal solution value by the H1 heuristic and LB, respectively.

Note that the full enumeration procedure can find the optimal solution for the problem with at most 10 jobs, so that <Table 1>

<Table 1> Performance Test for Small-sized Problems

m	n	H1			LB	
		Gap1 aver.	Gap2 aver.	NO1	Gap3 aver.	NO2
4	4	5.15	18.18	8	12.49	3
	6	6.30	12.30	4	5.66	8
	8	8.26	10.32	1	1.90	10
	10	6.61	10.26	2	3.45	8
8	4	2.71	6.44	12	3.56	11
	6	8.18	11.31	2	2.89	8
	8	5.25	9.17	3	3.75	10
	10	7.27	8.86	2	1.52	12
12	4	4.47	9.66	5	4.98	9
	6	5.57	8.16	5	2.45	11
	8	6.43	11.10	2	4.42	8
	10	3.69	5.39	7	1.61	15

<Table 2> Performance Test for Large-sized Problems

m	n	H1	
		Gap2 aver.	Gap2 max
4	20	11.55	31.82
	40	10.00	21.89
	60	9.67	17.10
	80	9.87	15.49
	100	11.13	16.97
8	20	8.84	17.83
	40	6.10	13.90
	60	6.24	12.84
	80	5.68	9.51
	100	5.41	10.05
12	20	6.80	16.92
	40	6.41	14.46
	60	4.30	8.89
	80	4.71	7.85
	100	4.12	8.82

includes the problem instances with the job sizes between 4 and 10. From the table, it is noted that the H1 heuristic and LB are very effective. The H1 heuristic has the average gap value at most about 8.26% with very small time elapsed, and the LB procedure has at most 12.49%.

To review the performance trend of the H1 heuristic for the large job sizes between 20 and 100, <Table 2> is presented. The table shows that the performance of the heuristic has also the small average gap at most 11.55%, and moreover, the performance of the algorithm increases slightly as the number of the machines increases.

We consider a special situation where the parallel machines in the second stage are bottleneck. Thus, the processing times associated in the second stage are relatively larger than the processing times in other stages. The processing times, p_{id} and p_{ia} are generated from $U[1, 10]$, while the processing times, $p_{ik}(k=1, \cdots, m-2)$ are generated from $U[10, 30]$. It is observed from <Table 3> that as the performance of the Agg-heuristic gets better, compared with the computational results of <Table 1>.

Finally, consider another situation where the machines in the first and last stages are bottleneck. Thus, the processing times, $p_{ik}(k=1, \cdots, m-2)$ are generated from $U[1, 10]$, while the processing times, p_{id} and p_{ia} are generated from $U[10, 30]$. It is observed from <Table 4> that as the performance trend of the Agg-heuristic is similar, compared with the computational results of <Table 1>.

<Table 3> Performance Test for the situation with Bottleneck Second Stage

m	n	H1	
		Gap2 aver.	Gap2 max
4	20	0.36	1.25
	40	0.22	0.87
	60	0.20	0.60
	80	0.10	0.37
	100	0.08	0.24
8	20	0.47	1.42
	40	0.22	0.66
	60	0.11	0.43
	80	0.17	0.44
	100	0.11	0.28
12	20	0.53	1.67
	40	0.29	0.85
	60	0.19	0.51
	80	0.13	0.34
	100	0.10	0.27

<Table 4> Performance Test for the situation with Bottleneck first and final Stages

m	n	H1	
		Gap2 aver.	Gap2 max
4	20	5.52	26.54
	40	4.62	9.91
	60	6.16	23.77
	80	4.65	15.79
	100	2.87	9.89
8	20	6.00	15.23
	40	4.56	15.40
	60	4.81	13.20
	80	3.33	13.47
	100	2.91	10.63
12	20	7.47	21.17
	40	5.91	17.18
	60	5.97	14.17
	80	4.02	13.16
	100	3.32	11.04

5. Conclusion Remarks

This paper considered a makespan scheduling problem in the F&J system with non-serial flowshop configuration.

We suggested a heuristic using the well-known Johnson philosophy for the two-stage flowshop and evaluated its performance by worst-case analysis and numerical tests.

This is the first research for the F&J system in deterministic scheduling context, so that its result will provide underlined ideas for the further research in similar systems.

References

[1] Duda, A. and Czachorski, T.; "Performance Evaluation of Fork and Join Synchronization Primitives," *Acta Informatica*, 24 : 525-553, 1987.

[2] Garey, M. R. and Johnson D. S.; Computers and Intractability : A Guide to the Theory of NP-completeness, Freeman, 1979

[3] Gershwin, S. B.; Manufacturing Systems Engineering, Prentice-Hall, 1994.

[4] Gungor, A. and Gupta, S. M.; "Disassembly Sequence Planning for Products with Defective Parts in Product Recovery," *Computers and Industrial Engineering*, 35 : 161-164, 1998.

[5] Hariri, A. M. A. and Potts, C. N.; "A Branch and Bound Algorithm for the Two-stage Assembly Scheduling Problem," *European Journal of Operational Research*, 103 : 547-556, 1987.

[6] Heidelberger, P. and Trivedi K. S.; "Analytic Queueing Models for Programs with Internal Concurrency," *IEEE Transactions on Computers*, 32 : 73-98, 1983.

[7] Hodgson, T. J. and McDonald, G. W.; "Interactive Scheduling of a Generalized Flow Shop. Part 1 : Success through Evolutionary Development," *Interfaces*, 11 : 42-47, 1981.

[8] Johnson, S. M.; "Optimal Two-and Three-stage Production Schedules with Setup Times Included," *Naval Research Logistics Quarterly*, 1 : 61-68, 1954.

[9] Koulamas, C. and Kyparisis, G. J.; "The Three-stage Assembly flowshop Scheduling Problem," *Computers and Operations Research* 28 : 689-704, 2001.

[10] Lee, C.-Y., Cheng, T. C. E., and Lin, B. M. T.; "Minimizing the Makespan in the 3-machine Assembly-type Flowshop Scheduling Problem," *Management Science*, 3 : 616-625, 1993.

[11] O'Shea, B., Grewal S. S., and Kaebernick, H.; "State of the Art Literature Survey on Disassembly Planning," *Concurrent Engineering : Research and Applications*, 6 : 345-357, 1998.

[12] Potts, C. N.; Sevast'janov, S. V., Strusevich, V. A., Wassenhove, L. N. V., and Zwaneveld, C. M.; "The Two-stage Assembly Scheduling Problem : Complexity and Approximation," *Operations Research*, 43 : 346-355, 1995.

[13] Sun, X., Morizawa K., and Nagasawa, H.; "Powerful Heuristics to Minimize Makespan in Fixed, 3-machine, Assembly-type Flowshop Scheduling," *European Journal of Operational Research*, 146 : 499-517, 2003.

[14] Yoon, S. H., Lee I. S., and Sung, C. S.; "A Note on the Reversibility of the Two Stage Assembly Scheduling Problem," *International Journal of Management Science*, 13 : 25-34, 2007.