Journal of the Korean Institute
of Industrial Engineers
Vol. 19, No.4, December., 1993.

가중치와 준비시간을 포함한 병렬처리의 일정계획에 관한 연구

Unrelated Parallel Processing Problems with Weighted Jobs and Setup Times in Single Stage

구제현* · 정종윤**

Jei-hyun Goo*, Jong-Yun Jung**

Abstract

An Unrelated Parallel Processing with Weighted jobs and Setup times scheduling prolem is studied. We consider a parallel processing in which a group of processors(machines) perform a single operation on jobs of a number of different job types. The processing time of each job depends on both the job and the machine, and each job has a weight. In addition each machine requires significant setup time between processing jobs of different job types. The performance measure is to minimize total weighted flow time in order to meet the job importance and to minimize in-process inventory. We present a 0−1 Mixed Integer Programming model as an optimizing algorithm. We also present a simple heuristic algorithm. Computational results for the optimal and the heuristic algorithm are reported and the results show that the simple heuristic is quite effective and efficient.

1. Introduction

Multiple-product, low to medium volume productions are currently more important than single-product, mass production. Even though mass production appears to dominate several industries, 75% of all parts that are manufactured in the industrialized nations are produced in volume of 50 or less [12]. FMCs are able to satisfy these modern industrial production requirements: A variety of parts is produced in low to medium volumes, with low production costs and a high level of quality.

Machine setup times are significant in the small volume manufacturing environments. Whenever there is a switch from processing

* Department of Industrial Engineering West Virginia University.

** Deparment of Industrial Engineering Chang Won National University.

a job of one type to a job of another type, a setup time is incurred. In many cases, jobs do not have equal importance or priority. One way to accommodate this characteristics is to assign a value or weighting factor to each job and to incorporate the weighting factors into the performance measure. Many problems studied in the machine scheduling literature do not include machine setup times, while some problems consider only setup costs. No work has been done on scheduling of unrelated parallel processing with weighted jobs and setup times.

Problem statement

In this study we consider a single stage manufacturing process with unrelated parallel processing, weighted jobs and multiple job types. In such a system, distinct machines are used to perform a single operation on a number of different job types. Machine setup is required if the job type is changed during the process. The processing time for a single job depends on both the job and the machine because of unrelated processing. The setup time for a job type also depends on both the job type and the machine. The performance measure is to minimize the total weighted flow time. The major assumptions made in this study are listed below.

- ○ A set of machines is specified.
- ○ A set of jobs is specified and the type of each job and processing time on each machine are known.
- ○ Each job has certain weight and the weight is known.
- ○ The individual job setup times are known and are included in the job processing times.
- ○ The setup times for each job type on each machine are known.
- ○ The setup times for different job types are sequence independent.
- ○ A machine can only work on one job at any time.
- ○ A job must be processed by one and only one machine.
- ○ Preemption is not allowed.
- ○ All jobs are available initially for processing.

The problem of minimizing makespan or mean weighted flow time for identical parallel processors without setups is known to be NP-complete [21]. The problem of minimizing total weighted flow time for Unrelated Parallel Processing with Weighted jobs and Setup times (UPPWS) is more complicated. No simple method is known to generate an optimal solution for NP-complete problems in a short time. Thus, complex optimization methods, such as branch and bound, or dynamic programming, or integer programming, may be resorted in order to obtain optimal solutions. Unfortunately, due to excessive CPU time and memory requirements, these methods can handle only small problems. Therefore, efficient heuristics that provide near optimal solutions are desirable.

The objectives of this study are 1) to develop a mathematical programming model for minimizing the total weighted flow time and 2) to develop a simple heuristic algorithm for the problem in order to generate near optimal solutions in short computational time.

2. Literature review

Single stage, Parallel processing

In this section, we will review the schedul-

ing algorithms that are related to the single stage, parallel processing. Parallel processors can be classified into three types depending on their speeds as listed below. Let $s_{i,j}$ be the processing speed of job j on machine i. Thus, if machine i processes job j, it requires a total amount of processing time equal to $a_j/s_{i,j}$. The quantity a_j is defined as the processing time when $s_{i,j}$ is qual to 1.

1) Identical processor: If all processors have equal job processing speeds, then processors are called identical. That is, $s_{i,j} = s_{k,j}$ for all machines and jobs.

2) Uniform processor: If the processors differ in their speeds, but the speed of each processor is constant and does not depend on the jobs, then they are called uniform. That is, $s_{i,j} = s_{i,n}$ for all machines and jobs.

3) Unrelated processor: If there is no particular relationship for processing speeds, then they are called unrelated.

While significant effort has been devoted to studying various aspects of machine scheduling (Baker [1], Coffman [3], Rinnooy Kan [17], French [8], Lavwler [13], Graham et al. [10], Graves [11], Blazewicz et al. [2], relatively little work exists on scheduling of parallel processing with setup times. Particularly, literature on scheduling of unrelated parallel processing with setup times is very scarce. We limit the review of previous work by discussing scheduling on the parallel processing with setup times.

Parallel processing with setup time.

Identical parallel processing :

The problem of scheduling parallel processors with sequence dependent setup times has been addressed by a number of researchers. Geoffrion and Graves [9] examined the problem of scheduling parallel production lines with changeover costs and formulated it as a quadratic assignment prblem. Parker et al.[15] used a vehicle-routing algorithm to solve the problem of minimizing total setup costs on parallel processors.

Tang and Wittrock [22] developed a two-phase heuristic procedure for minimizing makespan with identical parallel machines that require minor setups between part types of the same family, and major setups between part types of different families. This heuristic is based on the MULTIFIT algorithm [4] which does not consider setups. Wittrock [22] described an improved heuristic which is similar to the one described above. Tang [20] presented computational results from the two-phase heuristic, along with a tighter lower bound on makespan. Both procedures address the general case where setup times may be different for different families.

So [18] considered a similar case with differing major and minor setup times, but with machines having a fixed processing capacity. He developed and compared three heuristics with the capacity specified as a parameter that depends on the length of the scheduling horizon. The objective was to maximize some total reward function.

Rajgopal and Bidanda [16] examined the case where major setup times are identical for all families and minor setup times are identical for all part types, and no restriction on machine capacity. Two new heuristic procedures and one modified heuristic were proposed and tested with respect to makespan, average flowtime and time spent on setups.

Unrelated parallel processing :

Dietrich [6] proposed a two phase heuristic for the nonpreemptive scheduling problem. This heuristic first assigns jobs to machines so that some degree of balance is achieved, thus reducing makespan. Then the jobs assigned to each machine are scheduled to reduce setup times and minimize flow time. Dietrich and Escudero [7] also reported on preprocessing thechniques (basically primal cut generation) to improve an LP relaxation bound for the same problem. Unfortunately, the gap between the upper bound obtained form the heuristic and the lower bound obtained from the LP relaxation with preprocessing remained large. The same authors [5] proposed an alternative formulation which considers explicitly the splitting of a job type between several machines. They introduced several sets of additional variables and significantly improved the lower bound via a similar primal cut approach.

Lee [14] proposed a generic hybrid approximation procedure to generate improved lower bounds. His study is based on a dual approach where small decomposed Lagrangean dual problems are independently evaluated. Two Lagrangean duals were considered–a Lagrangean relaxation dual and a Lagrangean decomposition dual. The Lagrangean relaxation dual is first solved by a subgradient method to obtain a good initial lower bound for the second stage. Starting with the final multipliers returned from the first stage, Lee improved the lower bound by applying a dual ascent method to the Lagrangean decomposition dual.

3. Mathematical Programming Model

In this section, we formulate a quadratic mixed integer programming model for minimizing the total weighted flow time and then transform it into a mixed integer linear programming model.

We will use of the following notations and definitions.

Basic notations :

J, set of jobs $J=\{1, 2, \cdots, n\}$.

F, set of job types $F=\{1, 2, \cdots, t\}$.

M, set of machines $M=\{1, 2, \cdots, m\}$.

$F_m \subset F$, set of job types that can be processed by machine m.

t(j), type of job j, $t(j) \in F$ for all $j \in J$.

$J_m \subset J$, set of jobs that can be performed by machine m.

$J_t \subset J$, set of jobs for job type t.

$M_j \subset M$, set of machines that can process job j.

P_{jm}, processing time of job j on machine m for $j \in J_m$ and $m \in M$.

s_{tm}, setup time for job type t on machine m.

w_j, weight of job j for all $j \in J$.

Definition of variables :

JS^k_{jm}, job sequence variable

1 if job j is scheduled on machine m in position k.

0 otherwise

TS^k_{tm}, job type sequence variable

1 if a job of type t is scheduled on machine m in position k.

0 otherwise

CT^k_m, completion time variable

It indicates the completion time of the job in position k on machine m.

Z^k_{tm}, setup sequence variable

1 if setup of job type t is required on machine m in position k.

0 otherwise

Minimization of the total weighted flow time for the UPPWS

The objective is to minizize the total weighted flow time. This problem can be formulated as a quadratic mixed integer programming problem : The objective function is quadratic and constraints are linear.

$$\text{Min} \sum_{j=1}^{n} w_j \left(\sum_{m} \sum_{k=1}^{n} CT_m^k * JS_{jm}^k \right)$$

where $m \in M_j$ (1)

$$\text{s.t.} \sum_{m \in M_j} \sum_{k=1}^{n} JS_{jm}^k = 1 \quad \forall\, j \in J \tag{2}$$

$$\sum_{j=1}^{n} JS_{jm}^k \leq k = 1, \ldots, n ;$$

$\forall\, j \in J_m;\ \forall\, m \in M$ (3)

$$JS_{jm}^k \leq TS_{t(j)m}^k$$

$k=1,..n;\ \forall\, j \in J_m;\ \forall\, m \in M$ (4)

$$\sum_{t(j)=1}^{t} TS_{t(j)m}^k = 1$$

$k=1,..,n;\ \forall\, t(j) \in F_m;\ \forall\, m \in M$ (5)

$$TS_{t(j)m}^k - TS_{t(j)m}^{k-1} - Z_{t(j)m}^k \leq 0$$

$k=1,..n;\ \forall\, t(j) \in F_m;\ \forall\, m \in M$ (6)

$$\sum_{r=1}^{k} \sum_{j \in J_m} (p_{jm} * JS_{jm}^r) + \sum_{r=1}^{k} \sum_{t \in F} (s_{tm} * Z_{tm}^r) = CT_m^k$$

$k=1,.., n;\ \forall\, t \in F_m;\ \forall\, m \in M$ (7)

$$\sum_{j \in J_m} JS_{jm}^k \geq \sum_{j \in J_m} JS_{jm}^{K+1}$$

$\forall\, k=1, 2, ..., n-1;\ \forall\, m \in M$ (8)

$JS_{jm}^k \in \{0, 1\}$ $k=1, 2, .., n;$

$\forall\, j \in J_m;\ \forall\, m \in M$ (9)

$TS_{t(j)m}^k \geq 0 \quad k = 1, 2, .., n;$

$\forall\, t(j) \in F_m;\ \forall\, m \in M$ (10)

$Z_{t(j)m}^k \geq 0$ $k=1, 2,.., n;$

$\forall\, t(j) \in F_m\ \forall\, m \in M$ (11)

$CT_m^k \geq 0$

$k=1, 2,..,n;\ \forall\, m \in M$ (12)

Constraint (2) forces each job to be scheduled exactly one time on one of the machines. Constraint (3) forces at most one job to be assigned to a certain sequence on each machine. Constraint (4) forces the job type sequence variable to be 1 if the job sequence variable is 1. Constraint (5) forces at most one type sequence variable to be 1 to a certain sequence on each machine. Constraint (6) forces the setup sequence variable to be 1 if job type of the current sequence job is different from the previous sequence job. Constraint (7) calculates the completion time of each position on each machine. Constraint (8) forces jobs to be scheduled from the first position on each machine and forces jobs to be scheduled continuously. Constraint (9) forces the job sequence variables to take only the values of 0 or 1. Constraints (10), (11), and (12) force the job type sequence variables, the setup sequence variables, and the completion time variables to be nonnegative.

Transformation to a linear programming model

To transform the above quadratic integer programming model into a mixed integer linear programming model, we replace the product terms $CT_m^k * JS_{jm}^k$ in the objective function with new variables X_{jm}^k and introduce new constraints for these variables. Variables X_{jm}^k represents the completion time of job j on machine m if scheduled in position k, otherwise the variable can assume any value, which becomes zero because of the minimization of the objective function. The objective function becomes.

$$\text{Min} \sum_{j=1}^{n} w_j \left(\sum_{m} \sum_{K=1}^{n} X_{jm}^{k} \right) \text{ where } m \in M_j$$

The following constraints are added to the original formulation to force the value of X_{jm}^{k} to equal CT_m^k when JS_{jm}^{k} is equal to 1, otherwise it can have any nonnegative value. The constant u is an arbitrary large value.

$$X_{jm}^{k} \geq CT_m^k - u(1 - JS_{jm}^{k}) \quad \forall j, m, k \tag{13}$$

$$X_{jm}^{k} \geq 0 \quad k=1, 2, \ldots, n; \quad \forall j \in J_m; \forall m \in M \tag{14}$$

4. Simple Heuristic Algorithm

For a single machine with no setup times, weighted mean flow time is minimized by processing jobs in nondecreasing order of the ratio P_j/w_j. The ordering is referred to as weighted shortest processing time (WSPT) sequencing [1].

In this problem, we group jobs for each job type and apply the WSPT sequencing in order to obtain the optimal sequence within the group for each machine. After obtaining the optimal sequence for each group, we treat each group like a whole job. We calculate group processing time (including setup times) for each group on each machine and calculate total weight (w_t) for each group t. We calculate a group ratio (r_{tm}) for each group on each machine, which is defined as the group processing time divided by its total weight.

$$r_{tm} = (s_{tm} + \sum_{j \in J_t} P_j)/w_t \qquad \forall t \in F_m, m \in M$$

$$\text{where } w_t = \sum_{j \in J_t} w_j$$

It should be noted, that by treating groups (which at this stage are assumed to be indivisible) as big jobs with weights and processing times equal to the sum of weights and processing times respectively, of their individual jobs, the application of the WSPT rule will produce minimum weighted flow time for both groups and individual jobs. To show that the weighted flow times for individual group is minimized, let us consider a sequence S and a sequence S' of groups. Groups A and B are just interchanged in the sequences S and S'. The situation is depicted in Figure 1. Let C denote the completion time until the start of group A in S and group B in S'. C_A is the completion time of group A in S and C_B is the completion time of group B in S'. Let c_i indicate the completion time of each job i in group A and c_j indicate the completion time of each job j in group B. Also let w_i indicate the weight of each job i, WT(S) the weighted flow time of groups in S, WT(S') the weighted flow time of groups in S', WT_C the weighted flow time of groups up to C, and WT_E the weighted flow time of groups between D and E.

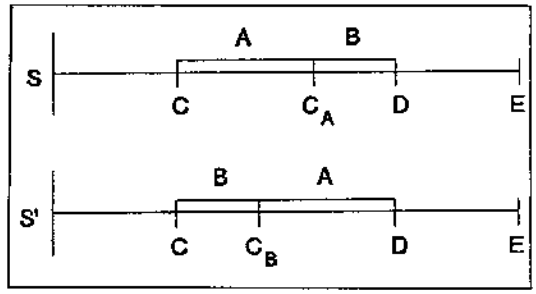

Fig 1. Interchange of contiguous groups

We obtain the weighted flow times of sequences S and S'.

$$WT(S) = WT_C + \sum_i w_i c_i + C\sum_i w_i + \sum_j w_j c_j + C_A \sum_j w_j + WT_E$$

$$WT(S') = WT_C + \sum_j w_j c_j + C\sum_j w_j + \sum_i w_i c_i + C_B \sum_i w_i + WT_E$$

Thus we have following equation.

$$WT(S)-WT(S')=C\sum_{i}w_i+C_A\sum_{j}w_j-C\sum_{j}w_j-C_B\sum_{i}w_i$$

$$=(C_A-C)\sum_{j}w_j-(C_B-C)\sum_{i}w_i$$

$$=(\text{Group pro. time for A})\sum_{j}w_j-(\text{Group pro. time for B})\sum_{i}w_i$$

From the derived equation, we can conclude that weighted flow time is minimized for the individual groups when scheduling groups in non-decreasing order of their ratios.

After obtaining the group ratios, we start to assign groups to machines. We first select the machine(s) which has (have) the smallest completion time. For the selected machine(s), we take the smallest group ratio and assign the group to the machine.
The proposed algorithm is summarized below.

Simple Heuristic Algorithm :

1. Group jobs for each job type.
2. Apply the WSPT sequencing for each group on each machine and obtain the optimal single machine sequence.
3. Calculate the group ratio for each group on each machine.
4. Select group(s) which can be processed on only one machine and assign it to the machine. For more than one group, assign them in non-decreasing order of their group ratios on the machine.
5. Select the machine(s) which has (have) the least completion time. For the selected machine, select the minimum group ratio. If several machines have the same minimum completion time, then select the combination of machine and group with minimum group ratio. Assign the associated group to the machine. At this time, the group must be placed on the machine such that the non-decreasing order of the group ratios on the machine is maintained.
6. Update the completion time for the machine and repeat Step 5 until all groups are scheduled.
7. Calculate the weighted flow time of each machine and the total weighted flow time.

5. Computational Experiments

In this section the results of experiments on the mathematical programming model and the huristic algorithm are reported. Optimal solutions are generated by the mathematical programming model and are used to measure performance of the heuristic algorithm.

To evaluate the effectiveness of the mathematical programming model and the heuristic algorithm, two sets of problems were generated :

Set 1) 2 machines, 2 job types, and 4 jobs,

Set 2) 2 machines, 4 job types, and 8 jobs.

In each problem set, every machine can process all job types. Weights of job j, w_j, are randomly selected from a discrete uniform distribution between 1 and 5. Processing times of job j on machine m, p_{jm}, are randomly selected from a discrete uniform distribution between 1 and 5. Setup times for job type t on machine m, s_{tm}, are randomly selected from a discrete uniform distribution between 1 and 10. In each set, 10 test problems were generated.

The MPSX/370 was used to solve the mathematical programming model on an IBM 3090 model 300E mainframe computer. FORTRAN computer codes for the heuristic

algorithm was written and tested on the same machine.

The heuristic algorithm is evaluated using two performance measures: soution quality and computation speed. The quality of a solution generated by the heuristic algorithm is measured in terms of its closeness to optimality. The solution quality is measured by the following quantity:

Quality measure $= 1 - \{V_{heur} - V_{opt}\}/V_{opt}$

where V_{heur} is the value of the solution obtained by the heuristic and V_{opt} is the value of an optimal solution obtaine using the mathematical programming model. The heuristic algorithm performs well when this measure approaches 1. The computation speed of the algorithm is measured by the amount of CPU time(reported in seconds) required to execute the algorithm.

The optimal solution and the solution of the heuristic algorithm were obtained for each test problem of the set 1 and the set 2. The results of the computations are presented for each test problem in Tables 1 and 2. Ineach of these tables, for each problem, "Math. Pro." and "Heuristic" refer to the mathematical programming model and the simple heuristic algorithm respectively. Table 3 shows the averages for each set and the "Number of Optimum" which refers to the number of times the optimal solution was obtained by the heuristic algorithm.

Table 1. Computational results for the set 1: Optimal algorithm versus heuristic algorithm

Problem number	CPU time(seconds)		Quality measure
	Math. Pro.	Heuristic	
1	1.98	0.46	1.000
2	4.66	0.47	0.839
3	2.95	0.46	1.000
4	4.47	0.45	1.000
5	3.18	0.45	1.000
6	4.21	0.47	1.000
7	4.67	0.47	1.000
8	3.65	0.45	1.000
9	4.06	0.46	0.819
10	3.73	0.46	0.886

Based on these results, the following observations can be made concerning the effectiveness and efficiency of the heuristic algorithm.

First, the heuristic algorithm generally apears to give very good estimates of the optimal solutions. This is indicated by the fact that the quality measures usually exceeded 85% in the individual test run and average quality measures by sets always exceeded 95% for the heuristic algorithm. For each set, the number of optimal solutions obtained with the heuristic algorithm was also good. It is worth noting that the quality of solutions obtained with the heuristic algorithm does not seem to decline as the size of problem increases.

Second, the computational efforts required

Table 2. Computational results for the set 2: Optimal algorithm versus heuristic algorithm

Problem number	CPU time(seconds)		Quality measure
	Math. Pro.	Heuristic	
1	18701.73	0.47	1.000
2	20508.28	0.46	0.992
3	10651.97	0.46	0.986
4	12220.97	0.47	1.000
5	7825.73	0.47	0.869
6	8193.23	0.46	0.848
7	11882.06	0.47	0.941
8	9260.18	0.47	1.000
9	3210.48	0.46	0.931
10	2094.51	0.47	1.000

Table 3. Averages of computational results for the sets 1 and 2: Optimal algorithm versus heuristic algorithm

Set	CPU time(seconds)		Quality measure	Number of Optimum
	Math. Pro.	Heuristic		
1	3.472	0.460	0.954	7
2	10451.910	0.466	0.957	4

by the heuristic algorithm seems to indicate that the algorithm provide very practical approach for the UPPWS. In contrast to the optimal algorithm for finding optimal solutions, for instance, the computational efforts required by the heuristic algorithm are very small. As shown in Table 3, the average running time of the optimal algorithm has increased from about four seconds for the first set to over 100,000 seconds for the slightly larger problems in the second set. In contrast, the average running times for the heuristic algorithm did not change much between the two sets.

6. Conclusion and Discussion

The minimization of total weighted flow time for the Unrelated Parallel Processing with Weighted jobs and Setup times problem has been studied. A mathematical programming model has been developed to generate an optimal solution. The computational results show that obtaining optimal solutions for larger problems would impose a heavy or even prohibitive computational burden. Therefore, efficient heuristic algorithms are needed to obtain good estimates of a global optimum. A simple heuristic algorithm has been developed. The computational results show that the heuristic is very effective and efficient in terms of the solution quality and the computation time.

The evaluation presented here is only for smaller problems because of the computational burden for obtaining optimal solutions for larger problems. To evaluate the performance of the heuristic for larger prob-

lems, it is desirable to develop lower bound methods in order to use lower bounds as benchmarks for comparisons with the heuristic solutions.

References

1. Baker, K. R., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.
2. Blazewicz, J., G. Finke, R. Haupt, and G. Schmidt, "New Trends in Machine Scheduling", *European Journal of Operational Research*, Vol. 46, pp. 303-317, 1988.
3. Coffman, E. G., Jr. (Ed.), *Computer and Job/Shop Scheduling Theory*, Wiley, New York, 1976.
4. Coffman, E. G. Jr., M. R. Garey, and D. S. Johnson "An application of bin-packing to multi-processor scheduling", *SIAM* Journal of computing, Vol. 7, No. 1, pp. 1-17, 1978.
5. Dietrich, B. L. and L. F. Escudero, "On Strengthening the Formulation of the Workload Allocation Problem for Parallel Unrelated Machines with Set-ups", *IBM Research Report*, T. J. Watson Research Center, Yorktown Heights, New York, 1989b.
6. Dietrich, B. L., "A Two Phase Heuristic for Scheduling on Parallel Unrelated Machines with Set-ups", *IBM Research Report*, T. J. Watson Research Center, Yorktown Heights, New York, 1988.
7. Dietrich, B. L. and L. F. Escudero, "On solving a 0-1 Model for workload Allocation on Parallel Unrelated Machines with Set-ups", IBM Research Report, T. J. Watson Reserarch Center, Yorktown Heights, New York, 1989a.
8. French, S., *Sequencing and Scheduling: an Introduction to the Mathematics of the Job-Shop, Horwood, Chichester*, 1982.
9. Geoffrion, A. M. and G. W. Graves, "Scheduling Parallel Production Lines with Changeover Costs: Practical Application of a Quadratic Assignment/LP Approach", *Operations Research*, Vol. 24, No. 4, pp. 595-610, 1976.
10. Graham, R. L., E. L. Lawler, J. K. Lenstra, and H. G. Rinnooy Kan, "Optimization and Approximation in Deterministic Sequencing and scheduling: A survey", Annals of Discrete Mathematics, Vol. 5, pp. 287-326, 1979.
11. Graves, S. C., "A review of Production Scheduling", *Operations Research*, Vol. 29, No. 4, pp. 646-675, 1981.
12. Greenwood, N. R., *Implementing Flexible Manufacturing Systems*, John Wiley & Sons, New York, 1988.
13. Lawler, E. L., "Recent Results in the Theory of Machine" Scheduling, in: A. Bachem, M. Grotschel and B. Korte (Eds.)", *Mathematical Programming: The State of Art-Bonn*, Springer, Berlin, pp. 202-234, 1982.
14. Lee, H. C., "Lagrangean Approximation Procedures for Certain Combinatorial Optimization Problems in Production/Manufacturing", *Ph. D. thesis*, Decision Sciences Dept., U. of Pennsylvania, Philadepia, PA., 1990.
15. Parker, R. G., R. H. Deane, and R. A. Holmes, "On the Use of a Vehicle Routing Algorithm for the Parallel Processor Problem with Sequence-Dependent Changeover Costs", *AIIE Transactions*, vol. 9, pp. 155-160, 1977.
16. Rajgopal, J. and B. Bidnada, "On scheduling parallel machines with two setup classes", *International Journal of Produc-*

tion Research, Vol. 29, No. 12, pp. 2443-2458, 1991.

17. Rinnooy Kan, A. H. G., *Machine Scheduling Problems: Classification, complexity and Computations*, Nijhoff, The hague, 1976.
18. So, K. C., "Some heuristics for scheduling jobs on parallel machines with setups", *Management Science*, Vol. 36, No. 4, pp. 4676-475, 1990.
19. Tang, C. S., "Scheduling Batches on Parallel Machines with Major and Minor Setups", *European Journal of Operations Research*, Vol. 46, pp. 28-37, 1990.
20. Tang, C. S. and R. J. Wittrock, "Parallel machine scheduling with major and minor setups", RC 11412, *IBM T. J.* Watson Research center, Yorktown Heihgts, NY., 1985.
21. Ullman, J. D., "Complexity of Scheduling Problems", *Computer and Job/Shop Scheduling Theory*, E. G. Coffman (Ed.), John Wiley, New York, 1976.
22. Wittrock, R. J., "Scheduling parallel machines with setups", *International Journal of Flexible Manufacturing systems*, Vol. 2, pp. 329-341. 1990.