A Column Generation Approach to Solve Proportionate Flexible Flow Shop for Common Due Date Scheduling

Der-Fang Shiau[1,2], Shu-Chen Cheng[3],Yueh-Min Huang[1] ,Pi-Chung Hsu[2]
[1]*Department of Engineering Science, National Cheng-Kung University, Tainan 701, Taiwan*
[2] *Department of Information Management, Fortune Institute of Technology, Kaohsiung County, Taiwan*
[3] *Department of Computer Science and Information Engineering, Southern Taiwan University of Technology*
Email:derfangs@center.fjtc.edu.tw, kittyc@mail.stut.edu.tw, huang@mail.ncku.edu.tw

Abstract-*This paper addresses a proportionate flexible flow shop for common due date scheduling problem. A proportionate flexible flow shop problem is generalization of the proportionate flow shop problem with multiple identical machines at any stage. The problem of minimizing total weighted deviations of job completion time from a common due date on a single machine is typical scheduling model in Just-In- Time production environment. In this paper, we propose a column generation approach which is based on some properties from V-shaped schedule on a single machine. This problem is actually a portioning problem, and a dynamic programming algorithm is proposed to find an early schedule and a tardy schedule with minimum reduced cost. The combination of column generation and linear programming demonstrate the capability of solving large scale problems. Computational result shows the effectiveness and the capability of solving problems with up to 40 jobs.*

Keywords: Proportionate flow shop, proportionate flexible flow shop, common due date, column generation, linear programming

1. Introduction

The just-in-time concept in many industrial manufacturing systems has become interest in machine scheduling problems. In the last four decades, many papers have been published in the scheduling area. It is desired to have jobs completed at time as close as possible to their respective due date. If a job is completed earlier than its due date, it has to be held as inventory and incurs an inventory cost. This is often the case when the products are physically large, then the buffer space in between two successive machines may have a limited capacity, causing *blocking*. In particular, common due date problems which are less complicated than the problems with distinct due date.

A Flexible flow shop is increasingly common in many manufacturing processes because of higher productivity of all jobs at one or more stages. A more general machine environment consists of a number of stages in series with a number of machines in parallel at each stage. A job has to be processed at each stage on only one of the machines. This machine environment is often referred to as a *flexible flow shop*, *multiprocessor flow shop* or *hybrid flow shop*. There are s stages in series $S=\{1,2,\ldots,s\}$; each stage $i \in S$ consists of m identical parallel machines $M=\{1,2,\ldots,m\}$ that have to process n jobs $N=\{1,2,\ldots,n\}$. One machine cannot be assigned to two jobs at a time, and each job can be processed by only one machine at each stage and preemption is not allowed. Each job $j \in N$ consists of a chain of s operations O_{ji} ($j \in N; i \in S$). As a flexible flow shop, which implies the operation O_{ji-1} is preceded by operation O_{ji}, that is, the execution of O_{ji} cannot start before the execution of O_{ji-1} has been finished. Operation O_{ji} has to process on machine $k \in M$ at stage $i \in S$, and requires an uninterrupted period of length p_{ji}. We further assume that the operation O_{ji} is processed by the kth machine at stage i, the operation O_{ji+1} must be processed by the kth machine at stage i+1, which implies that the algorithm is based on the list scheduling principle by assigning jobs to machines and job sequences for the first stage.

The problem of minimizing total weighted deviations of job completion times from a common due date is described as follows.

Let C_j ($j \in N$) denote the completion time of job j that has to complete on machine k at stage s($s \in S$) and should ideally be completed exactly on its due date d, which is common to the jobs on a single machine. We assume that this common due date is unrestricted large, that is, due dates that are large enough to not influence the assignment of the jobs completing before it. We define the earliness of the j-th job

$$E_j=\max\{0,d-C_j\}$$

And the tardiness of the j-th job

$$T_j=\max\{0,C_j-d\}$$

The objective is to focus on a schedule σ with minimum total weighted earliness and tardiness, obtaining

$$F(\sigma)=\sum_{j=1}^{n} [\alpha_j E_j+\beta_j T_j]$$

Where α_j and β_j are positive weights, denoted as $FFc \,||\sum_{j=1}^{n} [\alpha_j E_j+\beta_j T_j]$ for the flexible flow shop problem. We consider the special case of $FFc \,||\sum_{j=1}^{n} [\alpha_j E_j+\beta_j T_j]$, that is , the processing time of job j on each of the s stages is equal to P_j (i.e.,$P_{j1}=P_{j2}=\ldots=P_{js}=P_j$). In the literature, such a flow shop is referred to as a proportionate permutation flexible flow shop [1]. Minimizing the total weighted earliness and tardiness in a proportionate flexible flow shop is denoted by $FFc \,|P_{ji}=P_j|\sum_{j=1}^{n} [\alpha_j E_j+\beta_j T_j]$.

The problem of single machine scheduling with earliness and tardiness (E/T) penalties has been first introduced by Kanet [2]. The single machine case with symmetric E/T penalties is NP-hard [3]. Since then, several E/T scheduling problems have been presented and a reference can be found in [4]. However, most of the proposed models were focused on single machine scheduling problem; Emmons [5] and Hall [6] extended the E/T scheduling to parallel machines.

Minimizing the total weighted completion time in a two stages flow shop is already NP-hard [7]. Wittrock showed that the flexible flow shop scheduling problem is a NP-hard one [8]. Błażewicz, J., Pesch, E., Sterna, M., & Werner, F. [9] proposed a two-machine flow shop problem with weighted late work criterion and common due date. Shakhlevich [10] reported a $O(n^2)$ time algorithm to solve a proportionate flow shop with minimum weight completion time.

The flexible flow shop problem was first addressed by Salvador [11]. Brah [12] developed a branch and bound algorithm for the problem. However, these algorithms can only solve problems with a small size. Wang [13] and Xiao [14] also developed genetic algorithm for flexible flow shop to minimize the makespan. However, genetic algorithm is usually time–consuming for large scale problems and not suitable for fast scheduling.

The proportionate flexible flow shop for a common due date scheduling problem is performed two subproblems. The first one is to determine which jobs will be scheduled on each individual machine to share a common due date. The second one is to specify the order in which the jobs should be processed on each machine at any stage. Note that the processing order of the jobs is the same on each machine at any stage.

In this paper, we develop a decomposition approach applied to a Restricted Master Problem (RMP) with a set covering formulation in which the linear programming relaxation is solved efficiently by column generation. Our algorithm is based on the column generation approach independently proposed by Van Den Akker [15] for parallel machine scheduling problems. This approach has been successfully applied to many large scale parallel machine optimization problems. Chen and Powell [16] demonstrated that the column generation can solve very effectively with up to 100 jobs. To deal with the proportionate flexible flow shop scheduling problem, the framework of this approach is as follow. First, we formulate the problem as a set covering type formulation with an exponential number of variables (columns), n covering constraints ($n \in N$) and two side constraints, each of column which represents a partial schedule on a single machine, then solve the linear programming relaxation of this set covering formulation by a standard column generation [17]. If the solution of linear programming relaxation of the formulation is to be integral, then the optimal solution has been found for the problem $FFc \,|P_{ji}=P_j|\sum_{j=1}^{n} [\alpha_j E_j+\beta_j T_j]$, otherwise, we propose a branch and bound algorithm to identify an optimal solution.

2. Set Covering Type Formulation

In the following, we provide some optimality properties and formulate the problem as set covering type formulation.

2.1 Optimality Properties

Optimal solutions for the common due date scheduling problems on each single machine which are provided by the so called V-shaped schedules [4][18]:

(i) There is no idle time between jobs
(ii) The early jobs are scheduled in the non-decreasing order of the ratio α_j/P_j, that is, according to the Weighted Longest Processing Time first (WLPT) rule.
(iii) The tardy jobs are scheduled in the non-increasing order of the ratio β_j/P_j, that is, according to the Weighted Shortest Processing Time first (WSPT) rule.
(iv) One of the jobs completes exactly on time d.

In a given schedule for the problem, the corresponding single machine schedule on a given machine consists of two parts: *the early schedule*, consisting of the jobs completed before or on time d; *the tardy schedule*, consisting of the jobs completed after time d, and then combine them to form a single machine schedule. Due to the four properties, the value of common due date is irrelevant.

Property 1: In an optimal schedule for minimizing the total earliness and tardiness, the early jobs are scheduled according to LPT, and the late jobs are scheduled according to SPT [19].

In a proportionate flexible flow shop with s stages, it is clear that the late jobs are scheduled according to SPT rule, each job when completed at one stage does not have to wait for processing at next stage. Immediately it can start its processing at the following stage after completion at one stage. That is, the sum of the completion time is equal to the sum of the starting time at the first stage plus $\sum_{j=1}^{n} sp_j$.

2.2 Formulation

Let Ω be the set of all early partial schedules and tardy partial schedules, including the empty tardy schedule, and σ be any partial schedule ($\sigma \in \Omega$). For each job j ($j \in N$), let $a_{j\sigma}$ =1 if schedule $\sigma \in \Omega$ includes job j and 0 otherwise. $a_{n+1,\sigma}$ =1 only if σ is an early schedule, and $a_{n+2,\sigma}$ =1 only if σ is a tardy schedule. Let $C_j(\sigma)$ denote the completion time of job j on the machine k at the last stage s in σ. Let $C(\sigma)$ be the total cost of schedule $\sigma \in \Omega$. Note that the processing order of the jobs is the same on each machine at any stage, then

$$C_j(\sigma)=\sum_{i=1}^{j} p_i a_{i\sigma} +(s\text{-}1)\max\{p_1 a_{1\sigma},\ldots,p_j a_{j\sigma}\}$$

Hence, the cost $C(\sigma)$ of an early schedule σ is computed as

$$C(\sigma)=\sum_{j=1}^{n} \alpha_j E_j(\sigma) =$$
$$\sum_{j=1}^{n} \alpha_j a_{j\sigma}[d-(\sum_{i=1}^{j} p_i a_{i\sigma} +$$
$$(s-1)\max\{p_1 a_{1\sigma},\ldots p_j a_{j\sigma}\})]$$

It is clear that the cost of an early schedule is minimized by WLPT order and LPT order , that is , the early jobs are scheduled in the non-decreasing order of the α_j/P_j ratio and in order of non-increasing processing time (LPT).

For an optimal V-shaped schedule, the first job of a tardy schedule starts exactly at the due date. The cost $C(\sigma)$ of a tardy schedule σ is computed as

$$C(\sigma)=\sum_{j=1}^{n} \beta_j T_j(\sigma) =$$
$$\sum_{j=1}^{n} \beta_j a_{j\sigma} \sum_{i=1}^{j} p_i a_{i\sigma} +$$
$$(s-1)\sum_{j=1}^{n} \beta_j a_{j\sigma} \max\{p_1 a_{1\sigma},\ldots p_j a_{j\sigma}\}$$

The first term is exactly equal to the total weighted completion time of a tardy partial schedule σ on a single machine; this term is minimized by WSPT order. The second term is minimized by SPT rule, that is, the jobs are scheduled in order of non-decreasing processing time

For any early or tardy partial schedule $\sigma \in \Omega$, define 0-1 variables, y_σ=1 if schedule $\sigma \in \Omega$ is selected and 0 otherwise. Then the master problem can be formulated as the set covering problem.

$$Min \sum_{\sigma \in \Omega} C(\sigma) y_\sigma$$

subject to

$$\sum_{\sigma \in \Omega} a_{j\sigma} y_\sigma = 1, \quad \forall j \in N \qquad (1)$$

$$\sum_{\sigma \in \Omega} a_{n+1,\sigma} y_\sigma \le m \qquad (2)$$

$$\sum_{\sigma \in \Omega} a_{n+2,\sigma} y_\sigma \le m \qquad (3)$$

$$y_\sigma \in \{0,1\}, \quad \forall \sigma \in \Omega \qquad (4)$$

where constraint (1) means that each job is covered exact once. Constraint (2) ensures that there are at most m early partial schedules are selected. Constraint (3) ensures that there are at most m tardy partial schedules are selected. Note that each column in this formulation represents an early partial schedule or a tardy partial schedule and combine them to a single machine schedule and the set Ω contains an exponential number of schedules while the column generation is proceeded.

2.3 Column Generation Approach

In the column generation approach, the linear programming relaxation is obtained that the constraint (4) is relaxed to $0 \le y_\sigma \le 1$. This is because the column generation is a generalized linear programming for which an optimal solution of the relaxed problem is the lower bound of the integer optimal solution problem.

As the number of partial schedules on a machine, it is impossible to explicitly list all the columns when solving RMP. Instead, we use the column generation to generate necessary columns into RMP. To solve the restricted master problem, we apply the standard column generation in which the restricted master problem is a linear programming problem and can be solved efficiently. Each column represents an early schedule or a tardy schedule on one machine and is generated by solving a single machine subproblem. This procedure starts with a limited number of columns, that is, some initial set Ω of early and tardy schedules are needed to compute the initial dual variables. The initial solution has to be provided to the RMP and generate columns with the most negative reduced cost iteratively. To generate initial columns are becoming important, poorly selected initial columns lead the algorithm lost.

Single Machine Subproblems

The main idea behind column generation is that the occurrence of variables (columns) with negative reduced cost is not verified by enumerating all variables, but rather by solving an optimization problem. This optimization problem is called the *pricing problem* and is defined as the problem of finding the variable with minimum reduced cost to be added the restricted master problem. If neither an early

schedule, nor a tardy schedule with negative reduced cost exists, then the column generation procedure will be terminated and the problem for *FFc* $|P_{ji}=P_j| \sum_{j=1}^{n} [\alpha_j E_j + \beta_j T_j]$ is solved.

Let π_j denote the dual variable value corresponding to job j ($j \in N$) in constraint (1), and λ_1 and λ_2 denote the dual variable value corresponding to constraint (2) and (3). Then the reduced cost r_σ of any column $\sigma \in \Omega$ is given by:

$$r_\sigma = C(\sigma) - \sum_{j=1}^{n} a_{j\sigma}\pi_j - \lambda_1 - \lambda_2$$

We solve the pricing problem by finding the early schedule and tardy schedule with minimum reduced cost among all early and tardy schedules. To that end, we use two dynamic algorithms to find an early schedule with minimum reduced cost and a tardy schedule with minimum reduced cost.

In case of an early schedule, the vector $a_{n+1,\sigma}=1$ and $a_{n+2,\sigma}=0$, we essentially have to minimize $C(\sigma) - \sum_{j=1}^{n} a_{j\sigma}\pi_j$. Reindex the jobs in order of non-increasing α_j/P_j ratios, settling ties according to non-decreasing processing time, then for the earliness of any job j in the early schedule σ is computed as

$$E_j = d - C_j = \sum_{i=1}^{j-1} p_i a_{i\sigma}$$

Which implies that

$$r_\sigma = C(\sigma) - \sum_{j=1}^{n} a_{j\sigma}\pi_j - \lambda_1 =$$
$$\sum_{j=1}^{n} \alpha_j a_{j\sigma} \sum_{i=1}^{j-1} p_i a_{i\sigma} - \sum_{j=1}^{n} a_{j\sigma}\pi_j - \lambda_1$$

In case of a tardy schedule, the vector $a_{n+1,\sigma}=0$ and $a_{n+2,\sigma}=1$, then for the tardiness of any job j in the tardy schedule σ is computed as

$$T_j = C_j - d = \sum_{i=1}^{j} p_i a_{i\sigma} + (s-1)\max\{p_1 a_{1\sigma},\ldots,p_j a_{j\sigma}\}$$

Which implies that

$$r_\sigma = C(\sigma) - \sum_{j=1}^{n} a_{j\sigma}\pi_j - \lambda_2 =$$
$$\sum_{j=1}^{n} \beta_j a_{j\sigma} \sum_{i=1}^{j} p_i a_{i\sigma} +$$
$$(s-1)\sum_{j=1}^{n} \beta_j a_{j\sigma} \max\{p_1 a_{1\sigma},\ldots p_j a_{j\sigma}\} -$$
$$\sum_{j=1}^{n} a_{j\sigma}\pi_j - \lambda_2$$

The *pricing problem* is based on a dynamic programming that exploits the property that on each machine the jobs are sequenced in order of increasing indices. The approach solves a series of subproblems until it finds the solution of the problem. At each iteration, it determines the optimal solution for a subproblem. It finds a solution for the current subproblem by utilizing the dual variable values obtained earlier in the solution of the previous subproblems.

Dynamic Algorithm 1

To generate the early schedules with negative reduced cost, first reindex the jobs in order of non-increasing α_j/P_j ratios, settling ties according to nondecreasing processing time. Let $V(j, t)$ denote the minimum reduced cost in an early schedule in which the first job in the schedule starts at time $d- t$.

Initial condition:

$$V(j, t) = \begin{cases} -\lambda_1, & \text{if } j = 0 \text{ and } t = 0 \\ \infty, & \text{otherwise} \end{cases}$$

Recursive relation:

For $j=1,\ldots,n$, $t=0,\ldots,\sum_{i=1}^{j} P_i$

$$V(j, t) = min\{V(j\text{-}1, t),\ V(j\text{-}1, t\text{-}p_j) + \alpha_j(t\text{-}p_j) - \pi_j\} \quad (5)$$

The optimal value is computed:

$$\min_{0\le t\le P} V(n, t)$$

where $P = \sum_{j=1}^{n} P_j$. Then the early schedule with minimum reduced cost is solved by computing $\min_{0\le t\le P} V(n, t)$. The dynamic programming algorithm is based on the recursive relation that runs in $O(nP)$ time and space.

Dynamic Algorithm 2

To generate the tardy schedules with negative reduced cost in a similar fashion. Reindex the jobs in order of non-increasing β_j/P_j ratios, settling ties according to non-decreasing processing time. Let $\overline{V}(j, t)$ denote the minimum reduced cost in a tardy schedule in which the last job completes at time t.

$$\overline{V}(j, t) = \begin{cases} -\lambda_2, & \text{if } j = 0 \text{ and } t = 0 \\ \infty, & \text{otherwise} \end{cases}$$

Recursive relation:

For $j=1,\ldots,n$, $t=0,\ldots,\sum_{i=1}^{j} sP_i$

$$\overline{V}(j, t) = min\{\overline{V}(j\text{-}1, t), \overline{V}(j\text{-}1, t\text{-}sp_j) + \beta_j t - \pi_j\} \quad (6)$$

The optimal value is computed:

$$\min_{0\le t\le P} \overline{V}(n, t)$$

where $P = \sum_{j=1}^{n} sp_j$. Then the tardy schedule with minimum reduced cost is solved by computing $\min_{0\le t\le P} \overline{V}(n, t)$. Run both dynamic algorithms to determine the early schedule with minimum negative reduced cost and the tardy schedule with negative reduced cost. If both $V(n, t) \ge 0$ and $\overline{V}(n, t) \ge 0$, then the column generation procedure will be terminated and the problem has solved to optimality. If not, the new columns (the early schedules or the tardy schedules) are generated to be added into the restricted master problem. It is not necessary to have one column with the most negative reduced cost into the restricted master problem, if more than one columns with a negative reduced cost are available, then add multiple such columns to the restricted master problem. After the value of $V(n, t)$ or $\overline{V}(n, t)$ has been determined, the optimal sequence is obtained through

a simple backtracking procedure.

3. Branch and Bound Algorithm

A linear programming relaxation solved by column generation is not necessarily integral, so the branch and bound procedure now considers the LP relaxation of one of the subproblem and solves it. If the optimal solution is integral, in this case, each value of y_σ is either 1, or 0, then the branch of the tree does not have to be explored. If the optimal solution is not integral, then a fractional variable should be selected to branch on. For solving our problem, traditional branching on the y variable may create problems along a branch where a variable has been set to zero or one does not work in combination with column generation, that is, the branching $y_\sigma = 0$ means that this partial schedule is excluded, as pricing problem may generate this partial schedule (column) again when solving a single machine subproblem. Our branching strategy is based on the completion time of jobs appearing in a fraction solution instead of branching on the y's in the master problem.

Let y^* denote the optimal solution to the LP relaxation of the set covering formulation and Ω^* be the set of all early and tardy partial schedules $\sigma \in \Omega$ for which $y^*_\sigma > 0$. If y^* is integral, then y^* forms an optimal solution for $FFc\ |P_{ji}=P_j| \sum_{j=1}^{n} [\alpha_j E_j + \beta_j T_j]$., if not, the fractional schedules are obtained. Based on experiment observation, it occurred quite often for each job the completion time is equal in each partial schedule in Ω^* in which it occurs. That is, if the completion time of job j appears in one or more partial schedule $\sigma \in \Omega^*$ at time t, then the schedule is obtained by processing job j in the interval $[t - p_j, t]$. The following theorem illustrates this fact.

Theorem 1. If $C_j(\sigma) = C_j$ for each job j (j=1,…,n) and for each σ with $y^*_\sigma > 0$, then the schedule obtained by processing job j in the interval $[C_j - p_j, C_j]$ is feasible and has minimum cost. [15]

If the optimal solution to the linear programming relaxation does not form to be integral and does not satisfy the conditions of Theorem 1, then a branch and bound algorithm is required to find an optimal solution. The algorithm based on splitting the set of possible completion times. The following property is shown that if the optimal solution is fractional and does not satisfy the condition of Theorem 1, then there is at least one job j which satisfies

$$\sum_{\sigma \in \Omega^*} C_j(\sigma)\, y^*_\sigma > \min\{C_j(\sigma) \mid y^*_\sigma > 0\}$$

When apply branch and bound tree, it is necessary to know if there exists such a job j that does not satisfy the condition of Theorem 1. If any, we need to identify the job with the smallest index, then create two descendant nodes, one for the constraint that $C_j \le \min\{ C_j(\sigma) | y^*_\sigma > 0\}$ and another for $C_j \ge \min\{C_j(\sigma) | y^*_\sigma > 0\} + 1$. The first constraint specifies a deadline d_j at which job j must be completed, the second one specifies a release date $r_j = \min\{C_j(\sigma) | y^*_\sigma > 0\} + 1 - p_j$ before which job j cannot be started (In case of the tardy schedules ,$r_j = \min\{C_j(\sigma) | y^*_\sigma > 0\} + 1 - sp_j (s \in S)$).

To find an optimal solution, we have to generate columns after branching. This strategy can be easily incorporated into Algorithm 1 and 2. Simply, we have to replace equation (5) by

$$V(j, t) = \begin{cases} \min\{V(j-1,t), V(j-1, t-p_j) + \alpha_j (t - p_j) - \pi_j\}, \\ \qquad\qquad \text{if } r_j + p_j \le t \le d_j \\ V(j-1,t), \quad \text{otherwise} \end{cases}$$

and replace equation (6) by

$$\overline{V}(j, t) = \begin{cases} \min\{\overline{V}(j-1,t), \overline{V}(j-1, t-sp_j) + \beta_j t - \pi_j\}, \\ \qquad\qquad \text{if } r_j + sp_j \le t \le d_j \\ \overline{V}(j-1,t), \quad \text{otherwise} \end{cases}$$

4. Computational Experiments

In this section, we report the computation experiments for randomly generated test problem**.** Our algorithms involved are coded in C and tested on IBM server X series 232 with PIII processor. Linear programs involved in the column generation approach are solved by LINGO 8.0 .

To generate a test problem, there are five parameters to be determined: number of machines m at each stage, number of jobs n , processing time p_j for each job $j (j \in N)$ and earliness penalty weight α_j for each job $j (j \in N)$, tardiness penalty weight β_j for each job $j (j \in N)$

The five parameters are generated as follows:

Number of machines $M \in \{2,4,6\}$

Number of jobs $n \in \{20,30,40\}$

Processing time $p_j = [1, 30]$

Earliness penalty weight(α_j) = [1,100]

tardiness penalty weight(β_j) = [1,100]

Table 1 lists the computational result for problem $FFc\ |P_{ji}=P_j| \sum_{j=1}^{n} [\alpha_j E_j + \beta_j T_j]$ with processing time of jobs drawn from the intervals [1,30]**,** for each given combination of m and n, a total of 50 test problems are generated randomly and the header of columns are:

n	**:** Number of jobs
m	**:** Number of machines at each stage
IP-LP Gap	**:** The average gap in percentage between LP relaxation value and the

integer solution

WB : The number of problems solved at root node without any branching out of 50 problems.

ANN : Average number of B&B nodes explored for solving the problem.

CG : The average number of columns generated for solving the problem.

Table 1: Results for problem with processing time drawn form the distribution [1, 30]

n	*m*	IP-LP Gap	WB	ANN	CG
20	2	0.05%	30	3.8	445
30	2	0.16%	12	12.2	1811
40	2	0.2%	8	22.3	4436
20	4	0.15%	35	4.2	255
30	4	0.25%	16	5.5	825
40	4	0.15%	7	12.5	1532
20	6	0.01%	48	1.3	153
30	6	0.12%	32	6.1	896
40	6	0.08%	15	9.5	987

5. Conclusion

We have proposed an effective column generation approach for solving the class of proportionate flexible flow shop problem with a large common due date. Using this algorithm, we were able to solve problems with up to 40 jobs to optimality by solving the linear programming relaxation of a set covering formulation of the problem. From the computational results show that the integrality gap is extremely small, and also few nodes need to be explored in branch and bound tree, and many test problems are solved at the root node without branching.

An interesting topic for future research is the special case of proportionate flexible flow shop problem with the processing time $P_{ji}=p_j/s_{i,}$ where s_i is the speed of each machine at each stage and jobs have distinct due date.

References

[1] P.S. Ow, "Focused scheduling in proportionate flow shops**,** " *management Sci.* 31, 852-869, 1985

[2] J.J. Kanet, "Minimizing the average deviation of job completion times about a common due date, " *Naval Research Logistics Quarterly*, 28, 643-651, 1981

[3] M.R. Garey, R.E. Tarjan, G.T. Wilfong, "One-processor scheduling with symmetric earliness and tardiness penalties," *Mathematics of Operation Research*, 13, 330-348, 1988

[4] K.H. Baker and G.D. Scudder, "Sequencing with earliness and tardiness penalties: a review," O*peration Research 3*0, 22-36, 1990

[5] H. Emmons, "Scheduling to a common due date on parallel uniform processors," *Naval Research Logistics*, 34, 803-810, 1987

[6] N.G. Hall, "Single and Multiple-processor models for minimizing completion time variance, " *Naval Research Logistics Quarterly*, 33, 49-54, 1986

[7] Garey, Johnson and Sethi, "The complexity of flow shop and job shop scheduling**,**" *Mathematics of Operations Research***,** 117-129, 1976

[8] R.J. Wittrock, "An adaptable scheduling algorithms for flexible flow lines**,**" *Operations Research***, V**ol.33, No.4, 445-453, 1988

[9] J. Błażewicz, E. Pesch, M. Sterna and F. Werner, "The two-machine flow shop problem with weighted late work criterion and common due date," *European Journal of Operational Research,* 2004b

[10] N. Shakhlevich, H. Hoogeveen and M. Pinedo, "Minimizing total weighted completion time in a proportionate flow shop**,**" *Journal of Scheduling*, *J.Sched.* 1, 157-168, 1998

[11] M.S. Salvador, "A solution to a special class of flow shop scheduling problem**,** " *In Symposium of the Theory of Scheduling and Applications*, Springer Verlag, Berlin., 83-91, 1973

[12] S.A. Brah and J.L. Hunsucker, "Branch and Bound algorithm for a flow shop with multiple processors**,**" *Eur. J. Oper. Res*., Vol. 51, 88-99, 1991

[13] W.D. Xiao, P.F. Hao ,S. Zhang and X.H Xu , "Hybrid flow shop scheduling using genetic algorithms, " *IEEE Proceedings of the 3th World Congress on Intelligent Control and Automation*, Vol.1, 537-541, 2000

[14] L. Wang and D.W. Li, "A scheduling algorithm for flexible flow shop problem**,**" *IEEE Proceedings of the 4th World Congress on Intelligent Control and Automation*, Vol.4, 3106-3108, 2002

[15] J.M. ven den Akker, J.A. Hoogeveen and S.L. van de Velde, "Parallel machine scheduling by column generation," *Operations Research*, 47, 862-872, 1999

[16] Z.L. Chen and W.B. Powell, "Solving parallel machine scheduling problems by column generation**,**" *INFORMS Journal on Computing*, Vol. 11, No.1, 78-94, 1999

[17] Lasdon L.S., *Optimization theory for large systems*, MacMillan, New York, 1970

[18] M.A. Quaddus , "A generalized model of optimal due date assignment by linear programming," *Journal of the operation Research Society* 38, 353-359, 1987

[19] M. Pinedo, *Scheduling: theory, algorithm, and systems* **,** Prentice Hall, 2002