AN INTEGRATED COLUMN GENERATION AND LAGRANGIAN RELAXATION FOR SOLVING FLOWSHOP PROBLEMS TO MINIMIZE THE TOTAL WEIGHTED TARDINESS

TATSUSHI NISHI, YUKINORI ISOYA AND MASAHIRO INUIGUCHI

Division of Mathematical Science for Social Systems
Graduate School of Engineering Science
Osaka University
1-3 Machikaneyama-cho, Toyonaka City Osaka 560-8531, Japan
{ nishi; inuiguchi }@sys.es.osaka-u.ac.jp; isoya@inulab.sys.es.osaka-u.ac.jp

Received September 2010; revised January 2011

Abstract. In this paper, we address a new integration of column generation and Lagrangian relaxation for solving flowshop scheduling problems to minimize the total weighted tardiness. In the proposed method, the initial columns are generated by using near-optimal dual variables for linear programming relaxation of Dantzig-Wolfe decomposition derived by the Lagrangian relaxation method. The column generation is executed just after the generation of base columns generated by near-optimal Lagrange multipliers. Computational results demonstrate that the integration of column generation and Lagrangian relaxation can drastically speed up the conventional column generation.

Keywords: Scheduling, Column generation, Lagrangian relaxation, Flowshop scheduling

1. Introduction. Scheduling has become an essential component in recent societies to meet versatile needs [15, 21, 22]. Column generation is an effective computing technique to obtain a tight lower bound for large scale combinatorial optimization problems. The application of the algorithm appears in wide variety of areas such as railway crew scheduling [17], vehicle routing problem [1, 10], ship scheduling [4], patients scheduling in hospital [19] and time tabling problems [27]. It is also beneficial to use the algorithm in the branch and price [6] for the exact algorithms. This is due to the fact that a tight lower bound can be obtained from Dantzig-Wolfe reformulation by column generation. Recently, column generation has been successfully applied to production scheduling problems: parallel machine scheduling to minimize the total weighted completion time [8, 31], parallel machine scheduling to minimize earliness and tardiness [9] and flowshop scheduling to minimize the total weighted completion time [16].

It has been well known that the convergence for column generation is extremely slow for large scale problems due to the degeneration in solving the restricted master linear programming problems [20]. This phenomenon is called the tailing-off effect in the column generation. Several improvements for the convergence of column generation have been studied. The oscillation of linear programming dual variables is controlled by the trust region method [20]. The dual cuts are used in stabilizing column generation [3]. The integration of column generation and Lagrangian relaxation (LR) is a promising way to reduce computational efforts for column generation. A combined column generation and Lagrangian relaxation is studied for a single machine scheduling problem with the common due date [30]. They report an improvement of the convergence of column generation by using the lower bound obtained by LR as the criterion of convergence [30]. The reduction
of computation time by combining column generation and LR is presented [12, 18]. Their idea is that the LR is applied to the master problem to approximate the optimal value of the dual variables or it is used to generate good columns, and add the columns that can be derived by subgradient optimization in the LR algorithm.

On the other hand, the lower bound for the problem can be obtained from LR by relaxing the machine capacity constraints for all machines to yield \( n \) single job subproblems with precedence constraints [7]. The optimal value of the continuous relaxation of the Dantzig-Wolfe reformulation and the optimal value of the Lagrangian dual are equal [14, 25]. However, in many cases, the subgradient optimization used in the LR cannot compute the optimal solution because it provides a near optimal solution for Lagrangian dual problems for large scale combinatorial optimization problems.

In this paper, we address a new integration of column generation and LR to improve the convergence of the conventional column generation for flowshop scheduling problems to minimize the total weighted tardiness. The problem is known to be an NP-hard combinatorial optimization problem. In the flowshop problem, there is a set of jobs that have to be processed on the multi-stage flowshop where each stage is composed of an identical machine. Each machine can process one operation at a time where a job consists of a set of operations for plural stages. These operations have to be processed sequentially satisfying the precedence constraints between the consecutive stages. Each operation has a fixed processing time where the preemption and splitting are not allowed. The problem to minimize the total weighted tardiness is known to be NP-hard in the ordinary sense even for single machine, and NP-hard in the strong sense. For flowshop problem to minimize the total weighted tardiness, exact algorithms and heuristic algorithms have been studied. The well-known heuristic algorithms are NEH algorithm [24], tabu search [5] and iterated greedy algorithm [28]. For the exact algorithms, a branch and bound on the makespan minimization [11, 23], and the branch and bound for two-machine flowshop to minimize total completion time [2] have been addressed. However, the current exact algorithms can treat the problems with the only limited problem size.

In the proposed method, the set partitioning formulation using Dantzig-Wolfe decomposition of the flowshop scheduling problem is decomposed into a set partitioning master problem and \( n \) single job subproblems. The linear programming relaxation of the set partitioning problem is solved by a column generation. The upper bound is generated by an iterated local search heuristic. The new idea of the integration of column generation and LR is that the promising base columns are generated by the near-optimal Lagrange multipliers by using LR dualizing machine capacity constraints beforehand from the relationship between Lagrange multipliers and linear programming dual variables for set partitioning formulation. The column generation is executed on the condition that initial base columns are near-optimal. The total number of replications and computation time for column generation is expected to be reduced compared with the conventional column generation.

The key idea is to generate the promising base columns by the solution of Lagrangian relaxation. It can significantly reduce the computation time of the column generation. If the solution derived by the Lagrangian relaxation is far from the near-optimal solution, the reduction of computation time is not significant because the non-promising base columns are increased. To make further improvement for the proposed method to resolve ill-conditions caused by the increase of the non-promising columns is to eliminate the unnecessary columns during the algorithm of the column generation. We compare the results with the conventional combined column generation and LR proposed by Degraeve et al. [12].
The paper consists of the following sections. In Section 2, the column generation for flowshop scheduling problems is explained. In Section 3, a new integrated column generation and LR is proposed. The algorithm for generating promising base columns is developed. Computational experiments are shown in Section 4. Section 5 concludes the paper and mentions the future works.

2. Column Generation for Flowshop Scheduling Problems. In this section, we describe a set partitioning formulation of flowshop scheduling problems to solve the problem by column generation.

2.1. Problem formulation. We consider a permutation flowshop scheduling problem to find a sequence of $N$ jobs on a single machine with $L$ stages to minimize the total weighted tardiness. Each job requires a set of $L$ operations where $N$ is the number of jobs, $L$ is the number of production stages, and $H$ is the total time horizon to be long enough to complete all of the operation of jobs. The release date for every job is zero. A job cannot start until its release date and preemption is not allowed. Each machine cannot execute two or more than two operations at a time. The tardiness of job $i$ is defined as $T_i = \max \{0, c_{i,L} - d_i\}$ where $c_{i,L}$ is the completion time of operation at stage $L$ and $d_i$ is due date for job $i$. The problem is to find a feasible schedule to minimize the total weighted tardiness.

The following notations are used for the original problem formulation.

**Decision variables:**
- $c_{i,l}$ completion time of the operation of job $i$ at stage $l$.

**Parameters:**
- $d_i$ due date of job $i$,
- $w_i$ weight of job $i$,
- $p_{i,l}$ processing time of the operation of job $i$ at stage $l$.

Let $\varphi(t)$ be a function where $\varphi(t) = 1$ if $t \geq 0$, and $\varphi(t) = 0$ if $t < 0$, the capacity constraint on each machine can be written by

$$\sum_{i=1}^{N} \{\varphi(t - c_{i,l} + p_{i,l} - 1) - \varphi(t - c_{i,l})\} \leq 1.$$  \hspace{1cm} (1)

We define $a_{i,l,t} = \varphi(t - c_{i,l} + p_{i,l} - 1) - \varphi(t - c_{i,l})$. The flowshop scheduling problem ($P$) is formulated as the following equations.

$$(P) \quad \min_{\{c_{i,l}\}} \sum_{i=1}^{N} w_i T_i$$  \hspace{1cm} (2)

s.t.  \hspace{0.5cm} $T_i = \max \{0, c_{i,L} - d_i\}$, \hspace{0.5cm} $i = 1, \ldots, N$, \hspace{1cm} (3)

$c_{i,l} \geq p_{i,l}$, \hspace{0.5cm} $i = 1, \ldots, N$, \hspace{0.5cm} $l = 1, \ldots, L$, \hspace{1cm} (4)

$c_{i,l-1} \leq c_{i,l} - p_{i,l}$, \hspace{0.5cm} $i = 1, \ldots, N$, \hspace{0.5cm} $l = 2, \ldots, L$, \hspace{1cm} (5)

$$\sum_{i=1}^{N} a_{i,l,t} \leq 1, \hspace{0.5cm} l = 1, \ldots, L, \hspace{0.5cm} t = 1, \ldots, H,$$ \hspace{1cm} (6)

$$a_{i,l,t} = \varphi(t - c_{i,l} + p_{i,l} - 1) - \varphi(t - c_{i,l}), \hspace{0.5cm} i = 1, \ldots, N,$$

\hspace{0.5cm} $l = 1, \ldots, L, \hspace{0.5cm} t = 1, \ldots, H.$ \hspace{1cm} (7)

The objective function of (2) is the sum of the total weighted tardiness. Constraints (3) define the tardiness for job $i$. Constraints (4) ensure the completion time constraint from the processing time requirement. Constraints (5) denote the technical precedence constraint ensuring that an operation cannot be started until its preceding operation is
completed. Constraints (6) and (7) define the capacity constraints on each machine at each stage. The overall objective is to minimize the total weighted tardiness subject to the constraints mentioned above. The problem is known to be NP-hard in the strong sense. Therefore, the exact algorithms will be intractable for large-sized problems.

2.2. Dantzig-Wolfe reformulation. The original problem can be reformulated as a set partitioning problem by Dantzig-Wolfe decomposition with an exponential number of columns. In order to decompose the problem into job-level subproblems, we define the job schedule. Each column represents a schedule for each job. An alternative approach is the Dantzig-Wolfe decomposition with a subproblem for each machine [13]. However, the subproblem is a single machine sequence problem which is still NP-hard. Therefore, we use the formulation with a subproblem for each job. Each operation in a job schedule satisfies the technical precedence constraint. The following variables are defined for the reformulation.

Sets:
Ω the set of all possible job schedules.

Parameters:
$c_{s,l}$ completion time of operation at stage $l$ for job schedule $s$,
$p_{s,l}$ processing time of operation at stage $l$ for job schedule $s$,
$w_s$ weight for job schedule $s$,
$d_s$ due date for job schedule $s$,
$T_s$ tardiness for job schedule $s$,
$X_{s,i}$ binary variable that takes the value of 1 if job schedule $s$ is related to job $i$ and 0 otherwise,
$a_{s,l,t}$ binary variable that takes the value of 1 if the operation of job schedule $s$ at stage $l$ is processed in time period $t$, and 0 otherwise.

Decision variables:
y$_s$ binary variable indicating schedule $s$ is adopted or not.
Let $y_s$ be a binary variable that takes 1 if schedule $s$ for each job is adopted, and 0 otherwise. $X_{s,i}$ is a binary variable that takes 1 if schedule $s$ for each job is related to job $i$, and 0 otherwise. $a_{s,l,t}$ is a binary variable that takes 1 if the operation of schedule $s$ at stage $l$ is processed on time period $t$, and 0 otherwise. Let $\Omega$ be the set of possible job schedules.

The flowshop scheduling problem ($P$) can be reformulated as a set partitioning problem ($SP$) when the set of job schedules $\Omega$ is given. The reformulation makes it possible to reduce the number of constraints, but it generates more decision variables.

\[
(SP) \quad \min_{\{y_s\}} \sum_{s \in \Omega} w_s T_s y_s \\
\text{s. t.} \quad \sum_{s \in \Omega} X_{s,i} y_s = 1, \quad i = 1, \ldots, N, \\
\sum_{s \in \Omega} a_{s,l,t} y_s \leq 1, \quad l = 1, \ldots, L, \quad t = 1, \ldots, H, \\
y_s \in \{0, 1\}, \quad \forall s \in \Omega.
\]

2.3. Column generation. The continuous relaxation of the Dantzig-Wolfe reformulation ($SP$) provides a tight lower bound for the original problem. When solving the continuous relaxation of ($SP$), column generation is used to deal with the large number of columns. In the column generation, starting with a restricted master problem ($LRSP$) with a small number of columns with $\bar{\Omega} \subset \Omega$, the other columns are generated and added to the columns of ($LRSP$) when they are required. This is executed by solving pricing
problem explained in Section 2.4. The restricted master problem \((LRSP)\) is formulated as:
\[
(LRSP) \quad \min_{\{y_s\}} \sum_{s \in \Omega} w_s T_s y_s
\]
\[\text{s. t.} \quad \sum_{s \in \Omega} X_{s,i} y_s = 1, \quad i = 1, \ldots, N, \quad (13)\]
\[
\sum_{s \in \Omega} a_{s,l,t} y_s \leq 1, \quad l = 1, \ldots, L, \quad t = 1, \ldots, H, \quad (14)\]
\[
y_s \geq 0, \quad \forall s \in \Omega. \quad (15)\]

A column represents a feasible schedule for each job. In the column generation, the master problem with the restricted number of columns \((LRSP)\) is solved. Let \(\pi^*, \lambda^*\) be the optimal dual variable for the constraint of (13) and (14), respectively. If there is a column \(s\) that makes the reduced cost
\[
R_s(\pi^*, \lambda^*) = w_s T_s \sum_{i=1}^{N} X_{s,i} \pi_i^* - \sum_{l=1}^{L} \sum_{t=1}^{H} a_{s,l,t} \lambda_{l,t}^*
\]
(16) negative, the column is added to the restricted column \(\Omega\). The value of objective function \(z_{LRSP}\) is decreased when \((LRSP)\) is solved repeatedly. If there is no column that makes the reduced cost negative, the current solution is optimal for \((LSP)\).

2.4. Pricing problem. The pricing problem is to derive a column that can make the reduced cost negative. The problem \((PS)\) can be reformulated as:
\[
(PS) \quad \min_{\{c_{s,l}\}} \left( w_s T_s - \sum_{i=1}^{N} X_{s,i} \pi_i^* - \sum_{l=1}^{L} \sum_{t=1}^{H} a_{s,l,t} \lambda_{l,t}^* \right) \quad (17)
\]
\[\text{s. t.} \quad T_s = \max\{0, c_{s,l} - d_s\}, \quad (18)\]
\[
c_{s,l} \geq p_{s,l}, \quad l = 1, \ldots, L, \quad (19)\]
\[
c_{s,l-1} \leq c_{s,l} - p_{s,l}, \quad l = 2, \ldots, L. \quad (20)\]

The pricing problem can be regarded as the job-level subproblem \((PS_i)\) that can be solved by dynamic programming [7]. The dynamic programming recursion is formulated as:
\[
f_{i,l}(x) = \begin{cases} 
\kappa_{i,l}(x) + \min_{0 \leq z \leq x-p_{i,l}} f_{i,l-1}(z) & (l = 1) \\
- \sum_{t=x-p_{i,l}+1}^{x} \lambda_{t,l}^* & (1 \leq l \leq L - 1) \\
w_i T_i \sum_{t=x-p_{i,l}+1}^{x} \lambda_{t,l}^* & (l = L).
\end{cases}
\]
(21)

where \(\kappa_{i,l}(x)\) is
\[
f_{i,l}(x) = \begin{cases} 
\kappa_{i,l}(x) + \min_{0 \leq z \leq x-p_{i,l}} f_{i,l-1}(z) & (l = 1) \\
- \sum_{t=x-p_{i,l}+1}^{x} \lambda_{t,l}^* & (1 \leq l \leq L - 1) \\
w_i T_i \sum_{t=x-p_{i,l}+1}^{x} \lambda_{t,l}^* & (l = L).
\end{cases}
\]

(21)

The dynamic programming recursion computes the optimal completion time for each job so that the sum of the weighted tardiness is minimized when the dual variable for \((LRSP)\) is obtained. The optimal completion time can be obtained by \(c_{i,l} = \arg \min_{0 \leq c_{i,l} \leq c_{i,l-1} - p_{i,l}} f_{i,l}\) recursively from stage \(L\) to stage 1. The computational complexity of solving \((PS_i)\) is \(O(HL)\).
2.5. **Algorithm of column generation.** The algorithm of the column generation consists of the following steps.

**Step 0: Generation of initial column**
The initial solution is constructed by an iterated greedy algorithm [28]. The derived schedule is set as the initial column.

**Step 1: Solving restricted master problem (LRSP)**
The linear programming problem is solved and then the dual solution is derived.

**Step 2: Solving pricing problem**
Each job-level subproblem \((P_{Si})\) \((i = 1, \ldots, N)\) is solved by dynamic programming using the dual solution of \((LRSP)\). If the solution that makes the reduced cost negative is obtained, the column is added to \((LRSP)\).

**Step 3: Evaluation of optimality**
If all of the reduced costs are non-negative, the current solution for \((LRSP)\) is the optimal solution for \((LSP)\) and go to Step 4. Otherwise return to Step 1.

**Step 4: Construction of feasible solution**
The solution for \((LSP)\) is not feasible for \((SP)\). Therefore, the construction of feasible solution is necessary. This procedure is explained in the next section.

2.6. **Construction of feasible solution.** The initial columns and feasible solutions are generated by iterated greedy (IG) local search algorithm [28]. The upper bound of the original problem is generated by the algorithm. The detail of the algorithm is shown as follows.

**Step 0:** Sort the sequence of jobs in descending order of the total processing time for each job.
**Step 1:** Select \(k\) jobs in the sequence. The sequence of jobs for the non-selected jobs is stored. \(m = 1\).
**Step 2:** A job is selected from the \(k\)th job in the order of the sequence. The job is inserted into the sequence of jobs in a position where the objective function is minimized.
**Step 3:** If the objective value for the newly generated sequence is better than that of the current sequence, the tentative solution is updated to the newly generated sequence.
**Step 4:** If \(m = m_{\text{MAX}}\), the algorithm is finished. Otherwise, \(m = m + 1\) and return to Step 1.

The parameter \(k\) for iterated greedy algorithm is set to 4 and \(m_{\text{MAX}} = 8000\) in this study.

3. **New Integration of Column Generation and Lagrangian Relaxation.** The convergence of the conventional column generation is slow due to the degeneracy of the simplex algorithm in solving the restricted master problem. This is called the tailing-off effect. To speed up the convergence of the conventional column generation, we propose a new integration of column generation and LR. The relation between the solution of LR and that of column generation is discussed. The new integration with the construction of base columns is proposed.

3.1. **Lagrangian relaxation.** If the machine capacity constraint is relaxed by Lagrange multiplier \(\{\mu_{i,l}\}\), the relaxed problem is formulated as:

\[
(LR) \quad \min_{\{c_{i,l}\}} \left\{ \sum_{i=1}^{N} w_i T_i - \sum_{l=1}^{H} \sum_{i=1}^{L} \left( 1 - \sum_{i=1}^{N} a_{i,l,t} \right) \mu_{t,l} \right\}
\]

s.t. \((3), (4), (5), (7)\) \hspace{1cm} (22)
The relaxed problem is reformulated as:

\[
(LR) \quad \min_{\{c_{i,l}\}} \left\{ \sum_{i=1}^{N} \left( w_i T_i + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{i,t,l} \mu_{t,l} \right) - \sum_{t=1}^{H} \sum_{l=1}^{L} \mu_{t,l} \right\}
\]

s.t. (3), (4), (5), (7) \hspace{1cm} (23)

The relaxed problem can be decomposed into individual job-level subproblem \((LR_i)\) that can be solved by dynamic programming. The dual problem can be formulated as:

\[
(LD1) \quad \max_{\{\mu_{t,l}\}} \left\{ \min_{c \in F} \sum_{i=1}^{N} \left( w_i T_i + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{i,t,l} \mu_{t,l} \right) - \sum_{t=1}^{H} \sum_{l=1}^{L} \mu_{t,l} \right\}
\]

\[
\mu_{t,l} \geq 0, \quad l = 1, \ldots, L; \quad t = 1, \ldots, H. \hspace{1cm} (24)
\]

where \(F = \{c_{i,l} | c_{i,l} \text{ satisfies } (4), (5)\}\). The Lagrangian dual is solved by the subgradient method. \(\gamma\) is subgradient of Lagrangian function, and \(\alpha\) is a constant.

\[
\gamma_{t,l} = \frac{\partial L(\mu)}{\partial \mu_{t,l}} = \sum_{i=1}^{N} a_{i,t,l} - 1, \hspace{1cm} (25)
\]

\[
\mu_{t,l} = \max\left\{0, \mu_{t,l} + \alpha \frac{UB - LB}{\sum_{l=1}^{H} \sum_{t=1}^{L} \gamma_{t,l}^2} \gamma_{t,l}\right\}. \hspace{1cm} (26)
\]

\(\alpha\) is the parameter for determining step size of subgradient optimization. In this study, \(\alpha_n\) at iteration \(n\) is updated by the following rule. The initial value of \(\alpha_1\) is set to 0.1 and updated by \(\alpha_{n+1} := \min(2\alpha_n, 2)\). If the lower bound is not updated 20 times, \(\alpha_{n+1} := \alpha_n \times 0.5\). The main advantage of the subgradient method is simple and extremely fast. However, there are some cases when the subgradient optimization is stopped after a fixed number of iterations without having the optimal value of \((LD1)\) in the computations. This is due to the non-convexity and discontinuity of the dual function for scheduling problems. To derive an optimal value of lower bound, column generation is required.

3.2. Relation between column generation and Lagrangian relaxation. In this section, we derive the relationship between Lagrange multipliers for Lagrangian relaxation and dual variables for column generation. It is known that the Lagrangian dual is equivalent to the dual problem for the continuous relaxation of set partitioning problem derived by Dantzig-Wolfe decomposition \([14, 25]\). Let \(k \in K\) be the number of iterations for subgradient optimization, and \(K\) be the set of iterations. The Lagrangian dual problem \((LD1)\) is reformulated as \((LD2)\).

\[
(LD2) \quad \max_{\{\eta, \mu\}} \left( \sum_{i=1}^{N} \eta_i - \sum_{t=1}^{H} \sum_{l=1}^{L} \mu_{t,l} \right) \hspace{1cm} (27)
\]

s.t. \(T_i^k = \max\{0, c_{i,L}^k - d_i\}, \quad \forall k \in K, \quad i = 1, \ldots, N,\)

\[
a_{i,t,l}^k = \varphi (t - c_{i,l}^k + p_{i,l} - 1) - \varphi (t - c_{i,t}^k), \quad \forall k \in K, \quad i = 1, \ldots, N, \quad l = 1, \ldots, L, \quad t = 1, \ldots, H, \hspace{1cm} (28)
\]

\[
\eta_i \leq w_i T_i^k + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{i,t,l}^k \mu_{t,l}, \quad \forall k \in K, \quad i = 1, \ldots, N, \hspace{1cm} (29)
\]

\[
\mu_{t,l} \geq 0, \quad l = 1, \ldots, L, \quad t = 1, \ldots, H. \hspace{1cm} (30)
\]
The dual problem \((DLD)\) for \((LD2)\) is described as:

\[
(DLD) \quad \min_{\{y_k, i\}} \sum_{k \in K} \sum_{i=1}^{N} w_i T_i^k y_{k,i} \tag{32}
\]

s.t. \[\sum_{k \in K} y_{k,i} = 1, \quad i = 1, \ldots, N, \] \tag{33}

\[
\sum_{k \in K} \sum_{i=1}^{N} a_{k,i}^l y_{k,i} \leq 1, \quad l = 1, \ldots, L, \quad t = 1, \ldots, H, \tag{34}
\]

\[y_{k,i} \geq 0, \quad \forall k \in K, \quad i = 1, \ldots, N. \tag{35}\]

The objective function of \((DLD)\) can be written as \[\min_{y_s} \sum_{s \in \Omega} w_s T_s y_s\] where \(\Omega\) is the set of feasible schedule for one job. \(\sum_{k \in K} y_{k,i} = 1, \quad i = 1, \ldots, N\) can be rewritten as:

\[\sum_{s \in \Omega} X_{s,i} y_s = 1, \quad i = 1, \ldots, N \tag{37}\]

\((34)\) can be written as:

\[
\sum_{k \in K} \sum_{i=1}^{N} a_{k,i}^l y_{k,i} \leq 1, \quad l = 1, \ldots, L, \quad t = 1, \ldots, H \tag{38}\]

\[\iff \sum_{s \in \Omega} a_{s,i}^l y_s \leq 1, \quad l = 1, \ldots, L, \quad t = 1, \ldots, H \tag{39}\]

From these equations, \((DLD)\) can be reformulated as \((LSP)\). Therefore, the problem \((LD1)\) is equivalent to \((DLSP)\). The relationship between Lagrange multiplier and dual variable can be obtained by setting

\[
\pi_i = \min_{c \in \mathcal{F}} L_i(\mu, c) \tag{40}
\]

\[
= \min_{c \in \mathcal{F}} \left( w_i T_i + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{i,t,l} \mu_{t,l} \right) \tag{41}
\]

\[
= \min_{k \in K} \left( w_i T_i^k + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{k,i}^l \mu_{t,l} \right) \tag{42}
\]

\[
\leq w_i T_i^k + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{k,i}^l \mu_{t,l}, \quad \forall k \in K, \quad l = 1, \ldots, N \tag{43}
\]

\[\iff \sum_{i=1}^{N} X_{s,i} \pi_i \leq w_s T_s + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{s,i}^l \mu_{t,l}, \quad \forall s \in \Omega \tag{44}\]

\((LD1)\) can be reformulated as:

\[
(LD3) \quad \max_{\{\pi, \mu\}} \left( \sum_{i=1}^{N} \pi_i - \sum_{t=1}^{H} \sum_{l=1}^{L} \mu_{t,l} \right) \tag{45}
\]
\[ \sum_{i=1}^{N} X_{s,i} \pi_i - \sum_{t=1}^{H} \sum_{l=1}^{L} a_{s,t,l} \mu_{t,l} \leq w_s T_s, \quad \forall s \in \Omega \]  
(46)

\[ \mu_{t,l} \geq 0, \quad t = 1, \ldots, H, \quad l = 1, \ldots, L. \]  
(47)

The relation between the Lagrangian relaxation and column generation is illustrated in Figure 1. The Lagrangian dual problem (LD) is equivalent to the dual of the continuous relaxation problem (LSP) of the set partitioning problem (SP) by Dantzig-Wolfe reformulation. The relationship between the Lagrange multipliers and dual variables is represented by using the following equations.

\[ \lambda_{t,l} = -\mu_{t,l}, \quad t = 1, \ldots, H, \quad l = 1, \ldots, L, \]  
(48)

\[ \pi_i = \min_{c \in \mathcal{F}} \left( w_i T_i + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{i,t,l} \mu_{t,l} \right), \quad l = 1, \ldots, N. \]  
(49)

3.3. Construction of candidates of base columns. In the column generation algorithm, columns are successively generated and added to the restricted master problem until the optimality condition is satisfied. At the convergence of column generation, an optimal solution for (LSP) can be derived. The derived columns are divided into base columns and non-base columns. If the column is a base column, the reduced cost is zero, otherwise the reduced cost is non-negative for a non-base column. The necessary condition for being a base column is

\[ R_i (\pi^*, \lambda^*) = w_i T_i - \sum_{l=1}^{L} \sum_{c_{i,l} = p_{i,l} + 1}^{c_{i,l}} \lambda_{i,l}^* - \pi_i^* = 0 \]  
(50)

where \( \pi_i^*, \lambda_{i,l}^* \) are the optimal dual solution for (LSP). This condition is not sufficient because there is non-base columns that make the reduced cost zero. On the other hand, all of reduced costs are non-negative at the reach of convergence for column generation. Under the condition, the following condition holds.

\[ \min_{\{c_{i,l}\} \in \mathcal{F}} R_i (\pi^*, \lambda^*) = 0, \quad i = 1, \ldots, N \]  
(51)

All of base columns can be enumerated when \( c_{i,l} \) satisfying (51) can be obtained.
3.4. **Acceleration of column generation.** The base columns are required to obtain an optimal lower bound for column generation. The number of iterations for column generation is only one when the optimal base columns are known. The number of iterations is expected to be reduced by enumerating near-optimal base columns before the column generation. To derive the base columns, near optimal dual solution constructed by Lagrange multipliers derived by Lagrangian relaxation is available. From the viewpoint, we propose a new integration of column generation and Lagrangian relaxation. Let $\tilde{\mu}_{i,t}^*$ be a near-optimal Lagrange multiplier. The near-optimal dual variable $\tilde{T}_i$ can be derived by

$$\tilde{T}_i = \min_{c \in F} \left( w_i T_i + \sum_{t=1}^{H} \sum_{l=1}^{L} a_{i,l,t} \tilde{\mu}_{i,t}^* \right), \quad i = 1, \ldots, N \quad (52)$$

By using the near-optimal dual variables derived by (52) and (53), the columns are generated to satisfy

$$c = \arg \min_{c \in F} R_i \left( \tilde{T}_i, \tilde{\lambda}_t \right), \quad i = 1, \ldots, N \quad (54)$$

In order to generate promising base columns, a variety of solutions with the same dual costs are required. Moreover, there is difference between optimal dual variables and the near-optimal ones derived by Lagrange multipliers. Therefore, a fixed number of base columns are created to satisfy (54).

3.5. **Algorithm to generate the promising base columns.** In order to generate the promising base columns satisfying (54), an efficient algorithm is required. If a number of initial base columns for column generation are generated with the information of the solution of Lagrangian dual, the computational expense to solve the restricted master problem (LRSP) is significantly increased. Therefore, we propose an algorithm that can generate a sufficient number of the initial base columns efficiently. The computational complexity is $O(NH^L)$ if a brute force search algorithm is executed. The following recursion is executed.

$$g_{i,l}(x) = \begin{cases} 
- \sum_{t=x-P_{i,l}+1}^{x} \tilde{\lambda}_{i,t} & (l = 1) \\
- \sum_{t=x-P_{i,l}+1}^{x} \tilde{\lambda}_{i,t} + \min_{0 \leq z \leq x-P_{i,l}} g_{i,l-1}(z) & (2 \leq l \leq L)
\end{cases} \quad (55)$$

The criteria function for adopting base column is defined as:

$$R_i \left( \tilde{\pi}_i^* st, \tilde{\lambda}_t^* \right) = w_i T_i - \sum_{t=c_{i,L}-P_{i,l-1}+1}^{c_{i,L}} \tilde{\lambda}_{i,t} + \min_{0 \leq z \leq c_{i,L}-P_{i,L}} g_{i,l-1}(x) - \tilde{\pi}_i^* \quad (56)$$

The function depends only on the completion time $c_{i,L}$ for job $i$ at the last stage. This makes it easier to compute base columns with the restricted number of columns. The third term of right-hand side of (56) can be computed by the recursion (55). If the following condition is satisfied, $c_{i,L}$ is fixed. Then the recursion is computed from stage $L - 1$ to stage 1.

$$c_{i,l} = \arg \min_{0 \leq z \leq c_{i,l+1}-P_{i,l+1}} g_{i,l}(x) \quad (57)$$

If the completion time for job is computed for all stages, the column is added to a set of base columns. The computational complexity is $O(H^2L)$. The detailed algorithm for the generation of base columns is described as follows.
Step 0: Set \( i = 1, x = 0 \).
Step 1: Compute \( c_{iL-1}, c_{iL-2}, \ldots, 1 \) recursively by the dynamic programming recursion
\[
c = \arg \min_{c_{iL} = x, c \in F} R_i \left( \tilde{\pi}^*, \tilde{\lambda}^* \right).
\]
Step 2: Add the column to the set of the candidate of base columns.
Step 3: \( x := x + 1 \). If \( h > H \), go to Step 5. Otherwise return to Step 1.
Step 4: \( i := i + 1 \). If \( i > N \), go to Step 6. Otherwise return to Step 1.
Step 5: Select \( K \) columns from the candidate of base columns in ascending order of
\[
R_i \left( \tilde{\pi}^*, \tilde{\lambda}^* \right)
\]
and set to the set of base columns. The number of base columns \( K \) is the tuning parameter for the proposed method.

Step 0 is the initialization of job number \( i \) and the completion time \( x \). Step 1 is prepared to solve the following optimization problem \((P_0)\) for job \( i \) where the completion time of the operation at the last stage is fixed to \( x \).

\[
(P_0) \quad \min_{\{c_{it}\}} \left( w_i T_i - \sum_{l=1}^{L} \sum_{t=c_{it}-p_{it}+1}^{c_{it}} \tilde{\lambda}^*_l \right) \quad (58)
\]
\[
\text{s.t. } c_{iL} = x, (4), (5). \quad (59)
\]

The base columns derived at Step 1 are added at Step 2. The completion time \( x \) and the job number are updated at Step 3 and Step 4. At the Steps 0 to 4 of the algorithm, a large number of base columns are generated. In order to reduce the number of base columns, \( K \) number of base columns are chosen from the set of the base columns by the ascending order of the reduced cost. In the proposed method, the candidates of base columns are generated from only one decision variable; the completion time of the operations at the last stage. If all decision variables are considered, the computational complexity of the algorithm becomes \( O(H^L) \). The feature of the proposed algorithm is that the candidate columns are generated from the completion time at the last stage. The other variables are determined by the dynamic programming recursions (55).

3.6. Algorithm of the integrated column generation and Lagrangian relaxation.

The proposed algorithm (CG-LR) consists of the following steps.

Step 0: Execute Lagrangian relaxation.
Step 1: Generation of near-optimal dual solution from the solution of Lagrangian relaxation.
Step 2: Generation of candidates of base columns.
Step 3: The initial feasible schedule is created by the iterated greedy algorithm.
Step 4: Execute column generation.

The algorithm consists of LR step and CG step. The LR step generates the promising candidate of base columns. The CG step executes the column generation algorithm to ensure the optimality of dual variables. The main characteristic of the proposed integration is that the Lagrangian relaxation is executed only one time before the CG step. Different from the combined column generation and Lagrangian relaxation proposed by Degraeve et al. [12], the LR step is not repeated in the immediate of column generation algorithm.

4. Computational Experiments. Computational experiments are conducted to show the effectiveness of the proposed method in this section.
4.1. Comparison of column generation (CG) and Lagrangian relaxation (LR). Ten cases of problem instances are created by the random numbers on uniform distribution on the interval shown in Table 1. \( P \) is the lower bound for makespan [29] that is given by

\[
P = \max \left\{ \max_{1 \leq i \leq L} \left( \sum_{i=1}^{N} p_{i,1} + \min_{l=1}^{i-1} \sum_{i'=1}^{L} p_{i',1} + \min_{l=1}^{L} \sum_{l'=1}^{l+1} p_{i,l'} \right), \max_{i} \sum_{l=1}^{L} p_{i,l} \right\} \tag{60}
\]

The time horizon \( H = \max_{i} d_{i} + (L - 1) \max_{i} p_{i,1} + N \max_{i} p_{i,1} \) is used [26]. The program is coded by C++ language of Visual C++ 2008 Express Edition. CPLEX10.1 (ILOG) is used to solve the linear programming problems of (LRSP). An Intel Pentium(R) IV 3.2GHz Processor with 1GB memory is used for computation.

Table 2 shows the comparison of performance between the column generation (CG) and Lagrangian relaxation (LR). UB, LB, DGAP and Time indicate upper bound, lower bound, duality gap and computation time, respectively. The quality of lower bound is measured by \( \text{DGAP} = \frac{UB - LB}{LB} \times 100 \). The duality gap for CG is better than that of LR for all cases. On the other hand, the computation time for CG is much larger than that of LR. The average CPU time for CG is approximately 10 times of that of LR for 3 stage flowshop with 50 jobs. This is because the number of columns and the number of dual variables are increased exponentially in the restricted master problem for CG with the increase of the size of problem. It takes much CPU time for solving the linear programming problem repeatedly. From these results, it is demonstrated that the quality of lower bound for CG is better than that of LR. This is due to the fact that LR method with subgradient algorithm cannot derive the optimal value of Lagrangian dual problem. However, CPU time for CG is significantly increased with the increase of problem size. Therefore, the reduction of CPU time is necessary for the conventional column generation for large scale problems. The computation time for CG-LR is reduced for almost all cases except case 1, 5 and 10. The reduction percentage is 25.6% in average. From the results, the proposed method (CG-LR) is effective than the conventional column generation without lowering the quality of solution. It is confirmed from the results of Table 2 that the lower bound for CG and CG-LR is better than that of LR because the optimal solution of the Lagrangian dual cannot be obtained by the subgradient algorithm for LR. The computation time for the proposed method is much shorter than the conventional CG while the lower bound for CG-LR is the same with CG.

<table>
<thead>
<tr>
<th>Table 1. Parameters for problem instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>due date ( (d_i) )</td>
</tr>
<tr>
<td>weight ( (w_i) )</td>
</tr>
<tr>
<td>processing time ( (p_i) )</td>
</tr>
</tbody>
</table>

4.2. Effects of the parameter \( K \) to the performance of CG-LR. In this section, we investigate the effects of the number of base columns generated in advance to the performance of the proposed method. The performance of CG-LR is evaluated when the parameter \( K \) is changed from 400 to 3200 for 50 job problems by increasing 200 for each case. Table 3 summarizes the effects of parameter \( K \) to the performance of CG-LR. In Table 3, basic rate indicates the rate of basic columns generated before the execution of column generation. It is given by \( \frac{\text{(number of base columns for the initial columns)}}{\text{(number of base columns at final condition)}} \times 100 \).
Table 2. Comparison of column generation and Lagrangian relaxation 
\((N = 50, L = 3)\)

<table>
<thead>
<tr>
<th>Case</th>
<th>Method</th>
<th>UB</th>
<th>LB</th>
<th>DGAP (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LR</td>
<td>1699</td>
<td>1521.5</td>
<td>11.67</td>
<td>40.47</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>1533.9</td>
<td>10.76</td>
<td>197.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>2499.2</td>
<td>13.76</td>
<td>40.59</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>LR</td>
<td>2843</td>
<td>2526.2</td>
<td>12.54</td>
<td>443.39</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>2564.2</td>
<td>12.08</td>
<td>440.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>6326</td>
<td>12.13</td>
<td>319.53</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>LR</td>
<td>7948</td>
<td>6790.2</td>
<td>17.05</td>
<td>39.86</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>6791.4</td>
<td>17.03</td>
<td>193.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>2163.9</td>
<td>17.61</td>
<td>40.59</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LR</td>
<td>2545</td>
<td>2175.1</td>
<td>17.01</td>
<td>261.13</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>2881.7</td>
<td>19.93</td>
<td>383.42</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>3456</td>
<td>20.05</td>
<td>262.66</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LR</td>
<td>3069</td>
<td>2748.1</td>
<td>11.68</td>
<td>40.27</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>2749.6</td>
<td>11.62</td>
<td>176.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>4229.7</td>
<td>13.18</td>
<td>144.00</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LR</td>
<td>4787</td>
<td>4237.4</td>
<td>12.97</td>
<td>41.09</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>4237.4</td>
<td>12.97</td>
<td>979.44</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>3584</td>
<td>12.44</td>
<td>564.19</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>LR</td>
<td>1257</td>
<td>1080.8</td>
<td>16.36</td>
<td>40.39</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>1137.8</td>
<td>10.48</td>
<td>152.88</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>3274.0</td>
<td>14.59</td>
<td>248.30</td>
<td></td>
</tr>
<tr>
<td>Ave.</td>
<td>LR</td>
<td>3751.4</td>
<td>3286.7</td>
<td>13.68</td>
<td>40.63</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>3286.7</td>
<td>13.68</td>
<td>422.87</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CG-LR</td>
<td>314.90</td>
<td>314.90</td>
<td>422.87</td>
<td></td>
</tr>
</tbody>
</table>

From the computational results, the number of iterations is reduced due to the increase of the number of base columns and the number of initial columns when the parameter \(K\) is slightly increased. However, the total computation time and the number of columns has the minimum when the parameter \(K = 1600\). The reduction of CPU time by integrating column generation and Lagrangian relaxation is not effective when the parameter \(K\) is small due to the fact that the number of base columns generated by using the solution derived by LR is sufficiently small. On the other hand, it requires too much CPU time when the parameter \(K\) is set too large due to the increase of computational efforts for solving linear programming problem for the linear programming solver. Because the computational efforts for solving \((LRSP)\) is almost the same when the number of jobs is fixed, the appropriate value of the parameter \(K\) can be obtained by increasing the value of parameter \(K\) until the total computation time has the minimum value. It requires preliminary experiments for each job. However, if the size of the master linear programming problem is almost the same, the appropriate parameter is almost the same. The
parameters derived for each problem size is shown in Table 4. Table 3 and Table 4 show that there is an optimal number of base columns $K$ that can reduce the computation time for CG-LR and it can be obtained by some preliminary experiments.

**Table 3. Effects of parameter $K$ to the performance of CG-LR**

<table>
<thead>
<tr>
<th>$K$</th>
<th>Time (s)</th>
<th>CG-Iteration</th>
<th>Iterations</th>
<th>Base rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>426.2</td>
<td>139.9</td>
<td>5162.3</td>
<td>44.05</td>
</tr>
<tr>
<td>400</td>
<td>380.2</td>
<td>131.7</td>
<td>4984.2</td>
<td>43.88</td>
</tr>
<tr>
<td>600</td>
<td>355.7</td>
<td>124.8</td>
<td>4905.4</td>
<td>43.80</td>
</tr>
<tr>
<td>800</td>
<td>322.7</td>
<td>117.3</td>
<td>4768.4</td>
<td>42.45</td>
</tr>
<tr>
<td>1000</td>
<td>311.2</td>
<td>109.2</td>
<td>4723.0</td>
<td>39.99</td>
</tr>
<tr>
<td>1200</td>
<td>308.71</td>
<td>104.1</td>
<td>4705.7</td>
<td>37.62</td>
</tr>
<tr>
<td>1400</td>
<td>315.6</td>
<td>103.8</td>
<td>4766.6</td>
<td>34.97</td>
</tr>
<tr>
<td>1600</td>
<td>305.6</td>
<td>94.9</td>
<td>4748.8</td>
<td>31.90</td>
</tr>
<tr>
<td>1800</td>
<td>322.4</td>
<td>92.7</td>
<td>4865.2</td>
<td>29.13</td>
</tr>
<tr>
<td>2000</td>
<td>323.9</td>
<td>91.2</td>
<td>4971.6</td>
<td>29.74</td>
</tr>
</tbody>
</table>

**Table 4. Parameter for the proposed method (CG-LR)**

<table>
<thead>
<tr>
<th>$N \times L$</th>
<th>Number of base columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 $\times$ 3</td>
<td>300</td>
</tr>
<tr>
<td>10 $\times$ 5</td>
<td>300</td>
</tr>
<tr>
<td>50 $\times$ 3</td>
<td>1800</td>
</tr>
<tr>
<td>50 $\times$ 5</td>
<td>3600</td>
</tr>
<tr>
<td>70 $\times$ 3</td>
<td>3600</td>
</tr>
<tr>
<td>70 $\times$ 5</td>
<td>6400</td>
</tr>
</tbody>
</table>

**4.3. Comparison of CG-LR and CG.** In this section, the performance of the integrated column generation and LR (CG-LR) and conventional column generation (CG) is evaluated for several problem instances. The due dates are generated by the random numbers on uniform distribution between $[P(1 - \beta - \gamma/2), P(1 - \beta + \gamma/2)]$. $\beta$ is the tardiness factor of jobs to determine the length of due dates, and $\gamma$ is the dispersion range of due dates. Five scenarios are generated as shown in Table 5. Computational results for average of ten instances for each size for each scenario are summarized in Table 6 when the number of jobs $N$ is set as 10, 50, 70 and the number of machines $L$ is set as 3 and 5.

Because the column generation can derive an optimal lower bound for the continuous relaxation of set partitioning formulation, the lower bound obtained by CG-LR is all the same with the original CG. Therefore, the performance is evaluated by the total computation time (Time), the number of iterations (CG-Iteration) and the number of columns generated (Number of columns). CPU time for CG-LR is the sum of CPU time of LR step (Step 0 of the algorithm in Section 3.6) and CG step (Step 1 to Step 4 of the algorithm in Section 3.6). For CG-LR method, the parameter is used as shown in Table 4.

From the results, the total CPU time for CG-LR is successfully reduced in scenario 2, 3, 4 and 5 except scenario 1. The average computation time for CG-LR is 87%, 64%, 48% and 41% of the original CG for scenario 2, 3, 4 and 5, respectively. The average number of iterations (CG-Iterations) is reduced for all scenarios. Especially, the CPU time reduction
is dominant for scenario 4 and 5 when the original CG takes much computational efforts with large number of columns. This is because the computing time for solving (LRSP) with large columns can be reduced for large scale instances requiring large number of iterations. The number of columns generated for CG-LR is significantly reduced to 68%, 55% for scenario 4, 5, respectively. The CPU time for CG-LR for scenario 1 is not better than the original CG for scenario 1 even though the number of iterations for CG-LR is reduced to 70% of the original CG. The original CG converges much faster than CG-LR for the instances with low tardiness factor ($\beta = 0.4$) and wide due date range ($\gamma = 1.2$). Because the CG-LR combines LR step and CG step, it requires at least 50 seconds for LR step and 50 seconds for CG step for 50 instances. From that reason, CG-LR is not effective for the instances when the original CG solves much faster with less computational effort without any tailing-off effects. In the column generation, the computing time for solving the linear programming is computationally expensive when the number of columns is increased. Therefore, the reduction of CPU time is much effective for the instances with a number of iterations required. On the other hand, the reduction is not effective for the cases when the conventional CG requires less number of iterations.

Table 5. Due date parameters for problem instances

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Tardiness factor $\beta$</th>
<th>Due date range $\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4</td>
<td>1.2</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

4.4. Comparison of CG-LR with the algorithm by Degraeve et al. [12]. The proposed method is compared with the conventional combined column generation and Lagrangian relaxation proposed by Degraeve et al. [12]. The outline of the algorithm of the combined column generation and Lagrangian relaxation by Degraeve et al. [12] is shown in Figure 2. In the algorithm, instead of continuing (LRP) reoptimization of the master problem, the inner loop of Lagrangian relaxation procedure is executed to update Lagrangian multipliers. This inner loop provides with new columns. After a fixed number of LR steps or if no new columns are added to (LRP), the algorithm returns to the CG procedure. The advantages of this procedure are that the lower bound derived by LR can be used as stopping criterion for the early termination of CG, and the new columns are generated by LR that may reduce the degeneracy of (LRP) with subgradient optimization that is less computational effort than the simplex method.

The problem instances are generated according to the parameters shown in Table 4. Computational experiments are executed to compare the performance of CG-LR and the algorithm by Degraeve et al. for $N = 10, 20, 50$ and $L = 3$. Computational results are summarized in Table 7. The parameter for the algorithm by Degraeve et al. is the number of inner loops, and the parameter for CG-LR is the parameter $K$. These parameters are optimally selected in the computational experiments.

From the computational results, the computation time for the algorithm by Degraeve et al. is faster than that of the proposed method for 10 job problems. However, CG-LR is much faster than the conventional method for 20 and 50 job instances. This is because the number of columns generated by the algorithm by Degraeve et al. is much larger than that of CG-LR. The convergence of column generation is faster when the number of columns are increased for small size problems because the computational expense to
Table 6. Performance evaluation of the proposed method

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Size (N x L)</th>
<th>Time (s)</th>
<th>CG-Iteration</th>
<th>Number of columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CG</td>
<td>CG-LR</td>
<td>CG-LR</td>
<td>CG</td>
</tr>
<tr>
<td>1</td>
<td>10 x 3</td>
<td>6.97</td>
<td>5.41</td>
<td>72.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 x 5</td>
<td>7.26</td>
<td>8.96</td>
<td>63.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 3</td>
<td>52.37</td>
<td>112.04</td>
<td>96.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 5</td>
<td>387.91</td>
<td>785.47</td>
<td>160.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 3</td>
<td>226.55</td>
<td>436.98</td>
<td>112.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 5</td>
<td>922.05</td>
<td>2688.15</td>
<td>174.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10 x 3</td>
<td>7.09</td>
<td>5.25</td>
<td>72.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 x 5</td>
<td>11.58</td>
<td>11.82</td>
<td>77.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 3</td>
<td>422.87</td>
<td>314.90</td>
<td>147.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 5</td>
<td>2339.4</td>
<td>2458.32</td>
<td>213.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 3</td>
<td>1494.97</td>
<td>1111.52</td>
<td>179.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 5</td>
<td>14291.26</td>
<td>11811.83</td>
<td>289.5</td>
</tr>
<tr>
<td>3</td>
<td>10 x 3</td>
<td>7.08</td>
<td>5.11</td>
<td>71.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 x 5</td>
<td>15.65</td>
<td>12.40</td>
<td>103.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 3</td>
<td>598.56</td>
<td>418.88</td>
<td>157.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 5</td>
<td>5171.86</td>
<td>3985.42</td>
<td>273.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 3</td>
<td>2014.53</td>
<td>1296.8</td>
<td>193.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 5</td>
<td>11151.97</td>
<td>9748.39</td>
<td>276.5</td>
</tr>
<tr>
<td>4</td>
<td>10 x 3</td>
<td>8.32</td>
<td>4.97</td>
<td>80.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 x 5</td>
<td>18.53</td>
<td>12.64</td>
<td>107.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 3</td>
<td>681.72</td>
<td>311.43</td>
<td>183.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 5</td>
<td>5329.64</td>
<td>2941.24</td>
<td>302.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 3</td>
<td>1970.93</td>
<td>947.19</td>
<td>198.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 5</td>
<td>15134.93</td>
<td>12204.6</td>
<td>324.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10 x 3</td>
<td>7.72</td>
<td>4.78</td>
<td>76.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 x 5</td>
<td>16.61</td>
<td>9.78</td>
<td>102.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 3</td>
<td>678.47</td>
<td>183.07</td>
<td>205.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 x 5</td>
<td>4692.99</td>
<td>1844.85</td>
<td>307.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 3</td>
<td>1856.41</td>
<td>334.28</td>
<td>227.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 x 5</td>
<td>13381.23</td>
<td>6298.35</td>
<td>344.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Combined column generation and Lagrangian relaxation by Degraeve et al. (2003)
solve linear programming for (LRSP) is not significant. However, the computing time to solve (LRSP) becomes extensive for large sized problems. For the algorithm by Degraeve et al., the restricted master linear programming can be solved easily when the number of jobs is 10. Moreover, the number of iterations is smaller than that of CG-LR. From the results, the algorithm by Degraeve et al. is effective only for small sized problem instances when a number of columns are added in an iteration of column generation algorithm.

For the proposed method (CG-LR), the number of columns added to the restricted master problem is smaller even though the number of iterations is larger than that of the conventional column generation. The proposed method is faster for large scale problems because the step of Lagrangian relaxation is not included in the loop of column generation procedure. The proposed method is effective for large-scaled problem instances by reducing the number of columns generated in the column generation procedure.

The computation time for the proposed CG-LR is much shorter than the conventional CG-LR proposed by Degraeve et al. for large scale problems because the number of columns generated by the conventional CG-LR is much larger than the proposed CG-LR.

**Table 7. Comparison of CG-LR and the algorithm by Degraeve et al. (2003)**

<table>
<thead>
<tr>
<th>Size (N x L)</th>
<th>Time (s)</th>
<th>CG-Iteration</th>
<th>Number of columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Degraeve et al.</td>
<td>CG-LR</td>
<td>Degraeve et al.</td>
</tr>
<tr>
<td>10 x 3</td>
<td>4.18</td>
<td>5.25</td>
<td>23.1</td>
</tr>
<tr>
<td>20 x 3</td>
<td>27.18</td>
<td>25.06</td>
<td>40.6</td>
</tr>
<tr>
<td>50 x 3</td>
<td>423.3</td>
<td>314.9</td>
<td>116.9</td>
</tr>
</tbody>
</table>

5. Conclusion. In this paper, an integrated column generation and Lagrangian relaxation for flowshop scheduling problems has been proposed. In the proposed method, Lagrangian relaxation has been executed before the execution of column generation. The derived near-optimal Lagrange multipliers have been used to derive the near-optimal dual solution for continuous relaxation of set partitioning formulation. The algorithm of generating base columns has been proposed. The base columns have been used as the initial columns for column generation. Computational results have demonstrated that the proposed method can reduce approximately 25% of CPU time for flowshop scheduling with 50 jobs and 3 stages compared with the conventional column generation. The performance of the proposed method is compared with that of the conventional combined column generation and Lagrangian relaxation proposed by Degraeve et al. The computational results have shown that the algorithm by Degraeve et al. is better for small scale problems and the proposed method is more efficient for large scale problems. The appropriate value of parameter $K$ is required for the proposed method. The efficiency of the integration of column generation and Lagrangian relaxation depends on the accuracy of Lagrange multiplies derived by Lagrangian relaxation. If the Lagrange multipliers are far from optimal solution, the computation time of column generation becomes much larger than the conventional column generation. In our future work, the improvement of the proposed method by eliminating unnecessary columns will be investigated. A method to eliminate unnecessary columns is studied by Akker et al. [30]. In their approach, the upper bound $UB$ and the lower bound $LB$ are computed by the novel algorithm during the execution of the column generation. The columns with the reduced cost that is larger than the current $UB - LB$ can be eliminated. Also, the applicability of the proposed methodology to other scheduling problems will be investigated as well.
REFERENCES


