MISTA 2009

Lagrangian Relaxation and Cut Generation for Sequence Dependent Setup Time Flowshop Scheduling Problems

Tatsushi Nishi* · **Yuichiro Hiranaka**

Abstract Lagrangian relaxation technique is successfully applied to solve sequence dependent setup time flowshop problem to minimize the total weighted tardiness. The relaxed problem can be decomposed into each job-level subproblem that can be effectively solved by dynamic programming. Two types of the additional constraints for the violation of sequence dependent setup time constraints are imposed to the relaxed problem to strengthen the lower bound. The decomposed subproblem with the additional constraints is also effectively solved by the novel dynamic programming. Computational results show that the lower bound derived by the proposed method is extremely better than that of branch and bound algorithm.

1 Introduction

The sequence dependent setup time flowshop scheduling problem to minimize total weighted tardiness is known to be NP-hard combinatorial optimization problem. Many industrial scheduling problems can be modeled as the sequence dependent setup time (SDST) flowshop scheduling problems. The problem treated in this study is SDST flowshop where there is a set of jobs to be processed on multi-stage flowshop where each stage is composed of single machine. A job consists of a set of operations which have to be processed sequentially for plural stages. Each operation has a fixed processing time where preemption and splitting are not allowed. The sequence of operations is the same for all stages. Set up time is incurred before the processing an operation depending on the operation processed just before. The objective function to be minimized is the total weighted tardiness.

Exact algorithms and heuristic algorithms have been studied for solving SDST flowshop problems. A well-known heuristic is NEH algorithm for m-machine, n-job

Corresponding author to be addressed
*Tatsushi Nishi
Osaka University
E-mail: nishi@sys.es.osaka-u.ac.jp

Yuichiro Hiranaka
Osaka University
E-mail: hiranaka@inulab.sys.es.osaka-u.ac.jp

flowshop proposed by Nawaz et al. (1983). The iterated greedy search algorithm based on the use of NEH heuristic by Ruiz and Stützle (2008), randomized search combined with tabu search by Eren and Grüner (2006), and genetic algorithm of Ruiz and Stützle (2006) have been reported. For exact algorithms, branch and bound with dominance elimination criterion by Ríos-Mercado and Bard (1999), a branch and cut algorithm of Stecco et al. (2008) have been reported for the minimization of makespan. These exact algorithms can solve the problems optimally only with the limited size of problems, e.g within 10 jobs, and few machines for SDST flowshop. Heuristic algorithms can handle large-sized problems. However, the solution for those algorithms often can trapped into bad local optimum, and heuristic algorithms cannot evaluate the optimality of solutions. Therefore it is required to derive good bounds with reasonable time.

Lagrangian relaxation (LR) is used to obtain a good lower bound that can successfully applied to derive a near optimal solution for single machine and parallel machine scheduling problems by Luh et al. (1990), a practical jobshop scheduling problems of Hoitomt et al. (1993). In the LR, the machine capacity constraints are relaxed by Lagrange multipliers to decompose the relaxed problem into job-level subproblem that can effectively solved by dynamic programming (Chen et al. 1995). Subgradient optimization is used to solve Lagrangian dual problem. A critical issue to be solved for standard LR technique for scheduling is slow convergence of lower bound computation due to the existence of duality gaps. To strengthen the lower bound, Lagrangian relaxation and cut generation has been developed for flowshop problems to minimize the weighted tardiness by Nishi et al. (2007). Cuts for the capacity constraint violation are generated and imposed to the relaxed problem to strengthen the lower bound. This paper concentrates on the Lagrangian relaxation combined with cut generation for SDST flowshop to minimize the total weighted tardiness. The first step of the algorithm is to decompose the original problem into individual job-level subproblem as addressed in Nishi et al. (2007). The main difference between the study and Nishi et al. (2007) is that the sequence dependent setup constraints are taken into account in this work. If the problem is decomposed into job-level subproblem, the setup constraints cannot be easily handled. To consider the setup time constraints for the SDST flowshop, new types of additional constraints are developed and imposed on the decomposed subproblem to improve the bound.

In this paper we propose Lagrangian relaxation and cut generation for SDST flowshop with total weighted tardiness (SDST-WT). The original problem is decomposed into job-level subproblems by relaxing machine capacity constraints and setup time constraints. Valid inequalities are generated to improve the lower bound. The contribution of the paper is stated as follows. We show that the SDST-WT flowshop problem is decomposable into independent job-level subproblems when capacity constraints are relaxed by Lagrange multipliers and some of setup time constraints are eliminated. Two types of cuts are created to strengthen the lower bound derived by solving relaxed problem. Computational experiments show that the proposed method can derive relatively small duality gap for SDST flowshop with up to 50 job and 3 machines within 2,000 seconds.

2 Problem definition and formulation

The problem treated in this study is permutation flowshop scheduling problem with N jobs and L stages where each stage has single machine to minimize the total weighted tardiness. Sequence dependent setup time is incurred before the processing of the operation when different job is processed successively after an operation of job. The decision variables and constants for the problem are as follows.

Decision variables:

$c_{i,l}$	: completion time for the operation of job i at stage l
$s_{i,l}$	: setup time for the operation of job i at stage l
$m_{i,l}$	: machine allocated for job i at stage l
$\delta_{j,i}$	: 0-1 binary variable which takes the value 1 if the operation of job i is processed immediately after the operation of job j

Parameters:

d_i	: due date of job i
w_i	: weight of job i
$p_{i,l}$	: processing time for the operation of job i at stage l
H	: time horizon
N	: number of jobs
L	: total number of stages
M	: total number of machines
$S_{j,i,l}$	: setup time incurred for the operation of job i immediately after the operation of job j at stage l
$\mathcal{M}_{i,l}$	: set of machines which can process the operation of job i at stage l.

Let job 0 be a dummy job representing the job before the first job is processed. It is assumed that $\delta_{j,i} = 0$ if $j = i$, $c_{0,l} = s_{0,l} = m_{0,l} = \delta_{j,0} = 0$, $d_0 = w_0 = p_{0,l} = 0$. To decompose the problem into job-level subproblem, we define the decision variable for the completion time for the operation of job j at stage l: $c_{j,l}$, and the decision variable for the selection of setup time for job j: $\delta_{j,i} \in \{0, 1\}$ which takes the value 1 if the operation for job i selects the setup time before the operation of job i just after the operation of job j and zero otherwise. Let $\varphi(\tau)$ be the function where $\varphi(\tau)$ takes the value $\varphi(\tau) = 1$ if $\tau \geq 0$, and $\varphi(\tau) = 0$ otherwise, ζ_u be the pair of job i and stage l where the available machine which can process the operation of job i at stage l. These functions can be written as:

$$\varphi(\tau) = \begin{cases} 1, & \text{if } \tau \geq 0 \\ 0, & \text{otherwise,} \end{cases}$$

$$\zeta_u = \{(i, l) \mid m_{i,l} = u\}.$$

Using the functions, the SDST flowshop scheduling problem with the objective of the minimization of total weighted tardiness (P) can be formulated as the following equations.

$$(P) \qquad \min \sum_{i=1}^{N} w_i T_i \tag{1}$$

$$\text{s. t.} \quad T_i = \max\{0,\ c_{i,L} - d_i\}, \quad i = 1, ..., N, \tag{2}$$

$$c_{i,l} \geq s_{i,l} + p_{i,l}, \quad i = 1, ..., N, \quad l = 1, ..., L, \tag{3}$$

$$c_{i,l-1} \leq c_{i,l} - p_{i,l}, \quad i = 1, ..., N, \quad l = 2, ..., L, \tag{4}$$

$$\sum_{(i,l) \in \zeta_u} \{\varphi(\tau - c_{i,l} + p_{i,l} + s_{i,l} - 1) - \varphi(\tau - c_{i,l} - 1)\} \leq 1,$$

$$\tau = 1, ..., H, \quad u = 1, ..., M, \tag{5}$$

$$s_{i,l} = \sum_{j=0}^{N} S_{j,i,l}\delta_{j,i}, \quad i = 1, ..., N, \quad l = 1, ..., L, \tag{6}$$

$$\sum_{j=0}^{N} \delta_{j,i} = 1, \quad i = 1, ..., N, \tag{7}$$

$$\sum_{i=1}^{N} \delta_{j,i} \le 1, \quad j = 0, ..., N, \tag{8}$$

$$B(1 - \delta_{j,i}) + c_{i,l} - p_{i,l} - s_{i,l} \ge c_{j,l}, \quad j = 0, ..., N, \quad i = 1, ..., N, \quad l = 1, ..., L. \tag{9}$$

B is a sufficiently large constant. (1) is the objective function of the sum of the total weighted tardiness. (2) defines tardiness penalty. (3) restricts the completion time for each job at each stage. (4) represents the technical precedence constraints for each job at each stage. (5) is the machine capacity constraints ensuring that each machine can process only one operation at the same time during the processing. (6) expresses the setup time constraints for each job at each stage. (7) ensures that each job can select only one pre-setup time. (8) represents that the number of jobs selected for the setup time just after the operation of job j is equal or less than 1. (9) specifies the condition that the completion time for each job is consistent with the selection of setup time for each job.

3 Lagrangian relaxation and dynamic programming

3.1 Lagrangian relaxation

The problem (P) is NP-hard problem in strong sense. If the constraints (5), (8) and (9) are relaxed, the original problem (P) can be decomposed into each job-level sub-problem that can effectively solved in polynomial time. The relaxed problem obtained from the Lagrangian relaxation of (5) with non-negative Lagrange multiplier $\lambda_{\tau,m_{i,l}}$, the Lagrangian relaxation of (8) with non-negative Lagrange multiplier μ_j, and the elimination of constraints (9) is formulated as follows.

$$(RP) \quad \min L(\lambda, \mu) \tag{10}$$

s. t.

$$L(\lambda, \mu) = \sum_{i=1}^{N} \left\{ w_i T_i + \sum_{l=1}^{L} \sum_{\tau=c_{i,l}-p_{i,l}-s_{i,l}+1}^{c_{i,l}} \lambda_{\tau,m_{i,l}} + \sum_{j=0}^{N} \mu_j \delta_{j,i} \right\} - \sum_{u=1}^{M} \sum_{\tau=1}^{H} \lambda_{\tau,u} - \sum_{j=0}^{N} \mu_j \tag{11}$$

$$\lambda_{\tau,u} \ge 0, \quad \tau = 1, \ldots, H, \quad u = 1, \ldots, M, \tag{12}$$

$$\mu_j \ge 0, \quad j = 0, \ldots, N, \tag{13}$$

$$(2),\ (3),\ (4),\ (6),\ (7).$$

The decomposed subproblem (SP_i) for each job i is described as follows.

$$(SP_i) \quad \min L_i(\lambda, \mu) \tag{14}$$

$$L_i(\lambda, \mu) = \min \left\{ w_i T_i + \sum_{l=1}^{L} \sum_{\tau=c_{i,l}-p_{i,l}-s_{i,l}+1}^{c_{i,l}} \lambda_{\tau, m_{i,l}} + \sum_{j=0}^{N} \mu_j \delta_{j,i} \right\} \tag{15}$$

$$T_i = \max\{0,\ c_{i,L} - d_i\}, \tag{16}$$

$$c_{i,l} \geq s_{i,l} + p_{i,l},\ l = 1, \ldots, L, \tag{17}$$

$$s_{i,l} = \sum_{j=0}^{N} S_{j,i,l} \delta_{j,i}, l = 1, \ldots, L, \tag{18}$$

$$\sum_{j=0}^{N} \delta_{j,i} = 1. \tag{19}$$

3.2 Dynamic programming for subproblem with setup time selection

The subproblem (SP_i) can be solved optimally by the dynamic programming recursion extending the idea of algorithm provided by Chen et al. (1998). Let $q_{i,l}(u, t, \delta_{j,i})$ denote the criterion value for the completion of the operation of job i on machine u at time t. Let $\mathcal{F}$ be the set of time periods satisfying $\mathcal{F} = \{t \mid t = 0, \ldots, H\}$.

$$q_{i,l}(u, t, \delta_{j,i}) =$$
$$\begin{cases} w_i T_i + \displaystyle\sum_{x=t-p_{i,l}-s_{i,l}+1}^{t} \lambda_{x,u} + \sum_{j=0}^{N} \mu_j \delta_{j,i}, & \text{if } l = L, \\ \displaystyle\sum_{x=t-p_{i,l}-s_{i,l}+1}^{t} \lambda_{x,u} + \sum_{j=0}^{N} \mu_j \delta_{j,i}, & \text{otherwise,} \end{cases}$$
$$j = 0, ..., N,\ l = 1, ..., L,\ u \in \mathcal{M}_{i,l},\ t \in \mathcal{F}.$$

where $\mathcal{M}_{i,l}$ is the set of machines which can process the operation of job i at stage l. The Property 1 and Property 2 hold.

[Property 1.] The solution of (SP_i) with the fixed completion time as the earliest completion time $\mathcal{T}(i, l)$ for the operation of job i at stage l $(l = 1, \ldots, L)$ satisfying

$$\mathcal{T}(i, l) = \begin{cases} p_{i,l} + S_{0,i,l}, & \text{if } l = 1 \\ \max\{\mathcal{T}(i, l-1) + p_{i,l},\ p_{i,l} + S_{0,i,l}\}, & \text{otherwise.} \end{cases}$$

is optimal when $\delta_{0,i} = 1$ is the optimal solution for SP_i.

[Proof of Property 1.] It is clear that the objective function of the total weighted tardiness is regular and $\mathcal{T}(i, l)$ is the earliest completion time for job i. Therefore Property 1 holds.

[Property 2.] The following inequality is a valid inequality for (P):

$$B(1 - \delta_{j,i}) + c_{i,l} \geq \mathcal{H}(i, j, l),$$
$$i,\ j = 1, \ldots, N,\ i \neq j,\ l = 1, \ldots, L. \tag{20}$$

where

$$\mathcal{H}(i,j,l) = \begin{cases} \mathcal{T}'(j,l) + p_{i,l} + S_{j,i,l}, & \text{if } l = 1, \\ \max\{\mathcal{H}(i,j,l-1) + p_{i,l}, \mathcal{T}'(j,l) + p_{i,l} + S_{j,i,l}\}, & \text{otherwise.} \end{cases}$$

$$\mathcal{T}'(i,l) = \begin{cases} p_{i,l} + \min_{k=0,1,\dots,N} S_{k,i,l}, & \text{if } l = 1, \\ \max\{\mathcal{T}'(i,l-1) + p_{i,l},\ p_{i,l} + \min_{k=0,1,\dots,N} S_{k,i,l}\}, & \text{otherwise.} \end{cases}$$

[Proof of Property 2.] $\mathcal{H}(i,j,l)$ is the earliest completion time for job i when the operation for job i is processed after the operation of job i at stage l for $i,j = 1,...,N$. (20) is obtained by modifying $c_{j,l}$ into $\min c_{j,l}$ in (9). This reveals that (20) is the relaxation of constraint (9). (20) is a valid inequality for (P). Therefore Property 2 holds.

In order to utilize the properties for dynamic programming recursion, the function $h_{i,l}$ is defined. $h_{i,l}(u,t,\{\delta_{j,i}\})$ is the criterion value for the completion of operation at time t for job i on machine u at stage l when $\delta_{j,i}$ is given. $h_{i,l}(u,t,\{\delta_{j,i}\})$ is obtained by fixing earliest completion time for job i if $\delta_{0,i} = 1$, and it is obtained satisfying (20) when $\delta_{j,i} = 1,\ \ j \neq 0$. The dynamic programming recursion for solving subproblem (SP_i) is described as follows.

$$h_{i,l}(u,t,\{\delta_{j,i}\}) = \begin{cases} +\infty, & \text{if } (\delta_{0,i} = 1) \wedge (t \neq \mathcal{T}(i,l)), \\ +\infty, & \text{if } (j \neq 0) \wedge (\delta_{j,i} = 1) \wedge (t < \mathcal{H}(i,j,l)), \\ +\infty, & \text{if } t < H - \sum_{k=l+1}^{L} p_{i,k}, \\ q_{i,l}(u,t,\{\delta_{j,i}\}), & \text{otherwise,} \end{cases}$$
$$j = 0,...,N,\ l = 1,\dots,L,\ u \in \mathcal{M}_{i,l},\ t \in \mathcal{F}. \tag{21}$$

$$f_{i,l}(u,t,\{\delta_{j,i}\}) = \begin{cases} h_{i,l}(u,t,\{\delta_{j,i}\}), & \text{if } l = 1, \\ h_{i,l}(u,t,\{\delta_{j,i}\}) + \min_{v \in \mathcal{M}_{i,l+1}} g_{i,l+1}(v, t + p_{i,l+1}, \{\delta_{j,i}\}), & \text{otherwise,} \end{cases}$$
$$j = 0,\dots,N, l = 1,\dots,L, u \in \mathcal{M}_{i,l},\ t \in \mathcal{F}. \tag{22}$$

$$g_{i,l}(u,t,\{\delta_{j,i}\}) = \begin{cases} +\infty, & \text{if } t < p_{i,l}, \\ \min_{t \le x \le H} f_{i,l}(u,x,\{\delta_{j,i}\}), & \text{otherwise.} \end{cases} \tag{23}$$

$f_{i,l}(u,t,\delta_{j,i})$ is the optimal criterion value of the state $(u,t,\{\delta_{j,i}\})$ for the operation of job i on machine u at stage l. The optimal state $f(u^{l*}, t^{l*}, \{\delta_{j,i}^{l*}\})$ at stage l for job i is obtained by the forward recursion.

$$(u^{1*}, t^{1*}, \{\delta_{j,i}^{1*}\}) = \arg \min_{0 \le t \le H, u \in \mathcal{M}_{i,1}, \{\delta_{j,i} | \forall j\}} f_{i,1}(u,t,\{\delta_{j,i}\}) \tag{24}$$

$$(u^{l*}, t^{l*}, \{\delta_{j,i}^{l*}\}) = \arg \min_{t^{(l-1)*} + p_{i,l} \le t \le H,\ u \in \mathcal{M}_{i,L},\ \{\delta_{j,i}\} = \{\delta_{j,i}^{(l-1)*}\}} f_{i,l}(u,t,\{\delta_{j,i}\}). \tag{25}$$

The recursion is repeated until all of stages from stage 1 to stage L for each job i. Since the number of all combinations for $\{\delta_{j,i}\}$ is N for each job j due to $\sum_{j=0}^{N} \delta_{j,i} = 1$ and $\delta_{j,i} \in \{0,1\}$, the computational complexity for solving subproblem (SP_i) is $O(NMH)$.

The Lagangian dual problem (DP) is formulated as

$$(DP)\ \max Q(\lambda,\mu) \tag{26}$$

$$\text{s. t. } Q(\lambda,\mu)=\sum_{i=1}^{N}\min L_i(\lambda,\mu)-\sum_{u=1}^{M}\sum_{\tau=1}^{H}\lambda_{\tau,u}-\sum_{j=0}^{N}\mu_j, \tag{27}$$

$$(2),(3),(4),(6),(7),(12),(13).$$

The subgradient optimization is used to derive near-optimal Lagrange multipliers. α is the step size parameter.

$$\lambda_{\tau,u}=\max\{0,\ \lambda_{\tau,u}+\alpha\frac{UB-LB}{\sum_{u=1}^{M}\sum_{\tau=1}^{H}h_{\tau,u}^2}h_{\tau,u}\}, \tag{28}$$

$$\mu_j=\max\{0,\ \mu_j+\alpha\frac{UB-LB}{\sum_{j=0}^{N}h_j'^2}h_j'\}, \tag{29}$$

where $h_{\tau,u}=\sum_{(i,l)\in\zeta_u}\{\varphi(\tau-c_{i,l}^*+p_{i,l}+s_{i,l}^*-1)-\varphi(\tau-c_{i,l}^*-1)\}-1$, $h_j'=\sum_{u=1}^{N}\delta_{j,i}^*-1$. $\{c_{i,l}^*\}$, $\{s_{i,l}^*\}$, $\{\delta_{j,i}^*\}$ is the solution derived by solving subproblem SP_i.

4 Cut generation

4.1 Cuts for sequence dependent setup time constraint violation

It is extremely difficult to obtain a good lower bound for solving Lagrangian dual (DP) because the consistency violation constraints (9) for the selection of setup time with the precedence relationship of completion times are eliminated for the problem (RP) as explained in section 3. Cuts are added to the (RP) to strengthen the lower bound. Let $\mathcal{N}$ be the set of jobs except dummy job 0. The cuts for the violation of constraints are as follows:

$$\begin{aligned}&\exists(i,j)\quad\text{such that for } i,\ j\in\mathcal{N}\\&(\delta_{j,i}+\delta_{i,j}\le 1)\wedge(\delta_{k,i}+\delta_{k,j}\le 1,\ k=0,\dots,N)\end{aligned} \tag{30}$$

$$\begin{aligned}&\exists(i,j)\quad\text{such that for } i,\ j\in\mathcal{N}\\&(\delta_{j,i}+\delta_{i,j}\le 1)\wedge(c_{i,1}-p_{i,1}-s_{i,1}=c_{j,1},\ \text{if } \delta_{j,i}=1)\\&\wedge(c_{j,1}-p_{j,1}-s_{j,1}=c_{i,1},\ \text{if } \delta_{i,j}=1)\end{aligned} \tag{31}$$

The main idea for the cut generation is states as follows. (30) represents the additional constraint that can eliminate the inconsistency for the selection of setup time between the solution of subproblem for job i and job j. (31) is the additional constraint that can reduce the capacity constraint violation from the selection of setup time due to the characteristic of zero idle time for the first stage in the SDST flowshop problem.

More specifically, (30) ensures that the solution with $\delta_{j,i}=\delta_{i,j}=1$ is not satisfied and the setup time cannot be selected two or more than two jobs for job k at a time, that is clearly a valid cut because all feasible solution for the original problem satisfy (30). (31) represents the constraint that the completion for the operation of job i (j) is equal to the starting time for the operation of job j (i) at stage 1 if job i (j) selects the setup time for job j (i). For flowshop problems, idle time at stage 1 is always zero for regular objective function. (31) restricts that the completion time for job j is equal to the starting time for job i minus setup time for job i if the setup time for the operation of job i immediately after the operation of job j is selected at stage 1, and vice visa. The lower bound can be strengthened if the cuts (30), (31) are added to (RP) because the feasible region for (RP) is reduced.

4.2 Cut generation method

Both of (30) and (31) are related to a pair of job i and job j. The cut (30) and (31) are generated at the same time for each pair of two jobs. To generate the cuts, an appropriate pair of two jobs (i, j) must be selected. In this study, the following algorithm is executed for the cut generation. The main idea is that the pair of cut is selected so that the expected difference of setup time $\Delta S_{j,i} = |S_{j,i,1} - \min_{k \in \mathcal{N} \setminus \{j\}} S_{k,i,1}|$ is maximized. This is an empirical rule derived from our preliminary experiments. The optimality of solution derived by the LR highly depends on the selection of setup time for each job. Therefore it is expected to expedite search by the selection of large expected difference of setup time.

[Cut generation algorithm]
Step 1. $\mathcal{C} = \phi$.
Step 2. Fix the solution for an optimal solution to provide the best lower bound obtained by solving each job-level subproblem.
Step 3. Enumerate a pair of jobs (i, j) $(i,\ j \in \mathcal{N})$ that satisfies $\delta_{j,i} = 1 \wedge c_{i,1} - p_{i,1} - s_{j,1} \neq c_{j,1}$. Let $\mathcal{E}$ denote the set of the pairs (i, j) $(i,\ j \in \mathcal{N})$.
Step 4. For each element in $\mathcal{E}$, calculate $\Delta S_{j,i} = |S_{j,i,1} - \min_{\{k \in (\mathcal{N} \setminus \{j\})\}} S_{k,i,1}|$. If $\delta_{i,j} = 1$, then $\Delta S_{j,i} = \Delta S_{j,i} + \Delta S_{i,j}$.
Step 5. Select a pair of (i, j) which provides the largest $\Delta S_{j,i}$ from $\mathcal{E}$. The cut for the pair of (i, j) is generated and go to Step 6. If $\mathcal{E} = \phi$, all of the pairs in $\mathcal{N} \setminus \mathcal{C}$ are set to $\mathcal{E}'$ and go to Step 7.
Step 6. $\mathcal{C} = \mathcal{C} \cup \{i, j\}$. Delete the pair of jobs which includes at least one of job i and job j from $\mathcal{E}$ and return to Step 5.
Step 7. Select a pair of (i, j) randomly from $\mathcal{E}'$. The cut for the selected pair is generated. If $\mathcal{E}' = \phi$, the algorithm is finished.
Step 8. Delete all of the pair of jobs which includes at least one of job i and job j from $\mathcal{E}'$ and return to Step 7.

Both of cuts (30) and (31) for a pair of two jobs are generated simultaneously. The number of generated cuts for (30) and (31) is restricted so that two or more than two cuts for each pair of job (i, j) is not generated to reduce the total computing time. However, it is ensured that at least a cut for (30) and (31) is generated for every job.

Example. Consider an example of SDST flowshop with 3 jobs and 2 stages. Each stage has single machine. Fig. 1(a) shows a tentative result obtained by solving (DP). Assume the solution $\delta_{2,1} = 1,\ \delta_{1,2} = 1,\ \delta_{1,3} = 1$ when the setup time for job 1 is $S_{0,1,1} = S_{0,1,2} = 5$, $S_{2,1,1} = S_{2,2,2} = 2$, $S_{3,1,1} = S_{3,1,2} = 4$, the setup time for job 2 is $S_{0,2,1} = S_{0,2,2} = 2$, $S_{1,2,1} = S_{1,2,2} = 1$, $S_{3,2,1} = S_{3,2,2} = 5$, and the setup time for job 3 is $S_{0,3,1} = S_{0,3,2} = 7$, $S_{1,3,2} = S_{1,3,2} = 3$, $S_{2,3,1} = S_{2,3,2} = 4$. The candidate of cuts is $\epsilon = \{(1,2),\ (2,1),\ (1,3)\}$. For each candidate, $\Delta S_{j,i}$ is computed. $\Delta S_{2,1} = \Delta S_{2,1} + \Delta S_{1,2} = |S_{2,1,1} - S_{3,1,1}| + |S_{1,2,1} - S_{0,2,1}| = 3,\ \Delta S_{1,3} = |S_{1,3,1} - S_{2,3,1}| = 1.$

(1,2) is selected because $\Delta S_{2,1}$ is the maximum. $\mathcal{C} = \{1,2\}$ and $\epsilon = \phi$ because the element including 1 or 2 is excluded from ϵ. $\epsilon' = \mathcal{N}\backslash\mathcal{C} = \phi$ then the algorithm is finished. An example of results with cut (1,2) is shown in Fig. 1(b).

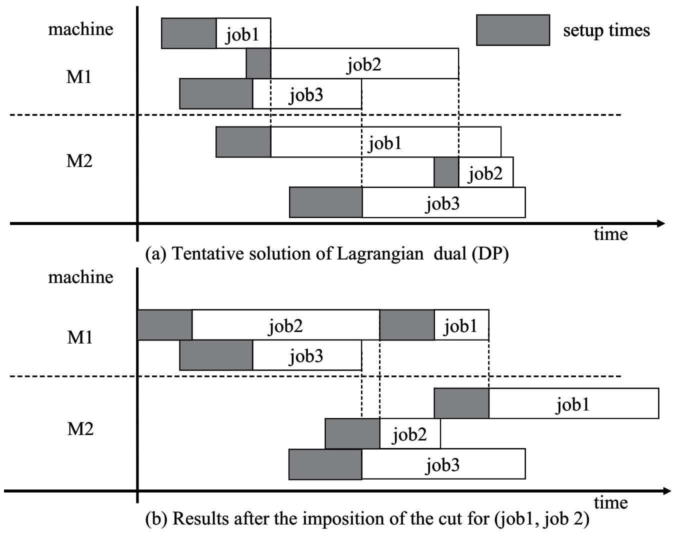

Fig. 1 Example of cut generation for SDST flowshop

4.3 Dynamic programming with cut generation

The dynamic programming recursion for the relaxed problem with the additional constraints on any two jobs is explained in this section. To solve the subproblem with cut on the operations for two jobs $i, j \in \mathcal{C}$, the recursion for the operation of job i at stage l is as follows.

$$
\begin{aligned}
&f'_{i,l}(u,t,\{\delta_{k,i}\}) = \\
&\begin{cases}
f_{i,l}(u,t,\{\delta_{k,i}\}) + \min\limits_{t' \in \mathcal{F}, \{\delta_{k',j} | \forall k',\ k' \neq k\}} f_{j,l}(u,t',\{\delta_{k',j}\}), \\
\qquad \text{if } (l=1) \wedge (\delta_{j,i} = \delta_{i,j} = 0), \\
f_{i,l}(u,t,\{\delta_{k,i}\}) + \min\limits_{\{\delta_{k',j}\ |\ \forall k', k' \neq k\}} f_{j,l}(u, t - p_{i,l} - s_{i,l}, \{\delta_{k',j}\}), \\
\qquad \text{if } (l=1) \wedge (\delta_{j,i} = 1) \wedge (\delta_{i,j} = 0), \\
f_{i,l}(u,t,\{\delta_{k,i}\}) + f_{j,l}(u, t + p_{j,l} + s_{j,l}, \{\delta_{k',j}\}), \\
\qquad \text{if } (l=1) \wedge (\delta_{j,i} = 0) \wedge (\delta_{i,j} = 1), \\
f_{i,l}(u,t,\{\delta_{k,i}\}), \quad \text{if } l \neq 1 \\
+\infty, \qquad \text{otherwise},
\end{cases} \\
&k, k' = 0, ..., N,\ l = 1, ..., L,\ u \in \mathcal{M}_{i,l},\ t \in \mathcal{F}. \qquad (32)
\end{aligned}
$$

$f'_{i,l}(u,t,\{\delta_{k,i}\})$ is the optimal criteria value for job i and job j which is the sum of the optimal cost for the state $(u,t,\{\delta_{k,i}\})$ of job i and the optimal cost for the state

$(u, t, \{\delta_{k,i}\})$ of job j under the constraint between the operation of job i and job j at stage l. There are three cases if the stage $l = 1$. The first recursion implies that the optimal cost is calculated by selecting the minimum cost if $(\delta_{j,i} = 0) \wedge (\delta_{i,j} = 0)$. The second recursion implies that job i selects the setup time that requires immediately after the processing of the operation of job j. The third recursion implies that job j selects the setup time that requires immediately after the processing of the operation of job i. The cut (30) is realized by setting $f'_{i,l}(u, t, \{\delta_{k,i}\}) = \infty$ if $(\delta_{j,i} = 1) \wedge (\delta_{i,j} = 1)$ or $(\delta_{k,i} = 1) \wedge (\delta_{k,j} = 1)$. The cut (31) is realized by the second and third recursions in (32). The optimal state with the constraints can be obtained by forward recursion by (24) and (25) in the same manner.

The optimal criteria value for the operation of job j at stage l is calculated as follows.

$$
\begin{aligned}
& f'_{j,l}(u, t, \{\delta_{k,j}\}) = \\
& \begin{cases}
f_{j,l}(u, t, \{\delta_{k,j}\}) + \min\limits_{t' \in \mathcal{F}, \{\delta_{k',i} | \forall k', k' \neq k\}} f_{i,l}(u, t', \{\delta_{k',i}\}), \\
\qquad\qquad \text{if } (l = 1) \wedge (\delta_{i,j} = \delta_{j,i} = 0), \\
f_{j,l}(u, t, \{\delta_{k,j}\}) + \min\limits_{\{\delta_{k',i} | \forall k', k' \neq k\}} f_{i,l}(u, t - p_{j,l} - s_{j,l}, \{\delta_{k',i}\}), \\
\qquad\qquad \text{if } (l = 1) \wedge (\delta_{i,j} = 1) \wedge (\delta_{j,i} = 0), \\
f_{j,l}(u, t, \{\delta_{k,j}\}) + f_{i,l}(u, t + p_{i,l} + s_{i,l}, \{\delta_{k',i}\}), \\
\qquad\qquad \text{if } (l = 1) \wedge (\delta_{i,j} = 0) \wedge (\delta_{j,i} = 1), \\
f_{j,l}(u, t, \{\delta_{k,j}\}), \quad \text{if } l \neq 1, \\
+\infty, \qquad \text{otherwise},
\end{cases} \\
& k, k' = 0, ..., N, \ l = 1, ..., L, \ u \in \mathcal{M}_{j,l}, \ t \in \mathcal{F}.
\end{aligned}
\tag{33}
$$

The recursion for the operation of job j at stage l can be computed in the same way as (32). The computing time for solving subproblem for job i and job j with additional constraints is $O(N^2MH)$. It is possible to construct the same type of cuts for any n jobs. The computing time is $O(N^nMH)$. The cuts up to any 3 jobs, any 4 jobs are applied for our numerical experiments.

5 Overall algorithm of Lagrangian relaxation and cut generation

The proposed algorithm is explained in this section. To explain the algorithm, the ordinary Lagrangian relaxation (LR) based on job-based decomposition is explained as follows.

[LR algorithm]

Step 1. Set the Lagrange multipliers (e.g $\lambda_{\tau,u} = 0$, $\mu_j = 0$).

Step 2. Solve the subproblem (SP_i) by the dynamic programming recursion of (22) and compute a lower bound LB.

Step 3. Construct a feasible solution by NEH algorithm combined with local search and compute an upper bound UB.

Step 4. If GAP=$UB - LB$ is less than 1 or the lower bound is not updated for a fixed time, the algorithm is finished.

Step 5. Update the Lagrange multipliers by (28) and (29). Then return to Step 2.

The proposed algorithm consists of the following steps. There are two types of Lagrange multipliers μ_j and $\lambda_{\tau,u}$. To solve the Lagrangian dual problem efficiently, Lagrange multiplier is independently optimized. The overall algorithm of the proposed method consists of the following steps.

[Proposed algorithm]

Step 1. Construction of a feasible solution by [**NEH algorithm combined with local search**]. Set an initial Lagrange multipliers.

Step 2. Execute [**LR algorithm**] with a fixed μ_j as the solution with the best lower bound. Only $\lambda_{\tau,u}$ is optimized by the subgradient method. The dynamic programming recursions of (32), (33) are used to solve the subproblem with cuts in the LR algorithm.

Step 3. Execute [**LR algorithm**] with fixed $\lambda_{\tau,u}$ as the solution with the best lower bound. Only μ_j is optimized by the subgradient method. The dynamic programming recursions of (32), (33) are used to solve the subproblem with cuts in the LR algorithm.

Step 4. If GAP=$UB - LB$ is less than 1, or the lower bound has not been updated for predetermined number of times, the algorithm is finished. Otherwise return to Step 2.

Step 5. Set the Lagrange multipliers with the best lower bound.

Step 6. Execute [**Cut generation algorithm**].

Step 7. Execute [**LR algorithm**] with fixed μ_j as the solution with the best lower bound. Only $\lambda_{\tau,u}$ is optimized by the subgradient method. The dynamic programming recursions of (32), (33) are used to solve the subproblem with cuts in the LR algorithm.

Step 8. Execute [**LR algorithm**] with fixed $\lambda_{\tau,u}$ as the solution with the best lower bound. Only μ_j is optimized by the subgradient method. The dynamic programming recursions of (32), (33) are used to solve the subproblem with cuts in the LR algorithm.

Step 9. If GAP=$UB - LB$ is less than 1, or the lower bound has not been updated for predetermined number of times, the algorithm is finished. Otherwise return to Step 7.

In the proposed algorithm, NEH algorithm combined with local search[1] is used to derive an upper bound only in the first step. The algorithm to derive an upper bound consists of the following steps.

[NEH algorithm combined with local search]

Step 1. Let $\mathcal{S}'$ be the sequence of jobs are sorted according to the earliest starting time in ascending order.

Step 2. $k := 1,\ \mathcal{S} = \mathcal{S}',\ UB = +\infty$.

Step 3. Select 5 jobs from $\mathcal{S}$. $\mathcal{R}$ is the set of the selected 5 jobs.

Step 4. $k' := 1,\ r' := 1,\ UB' := +\infty$.

Step 5. Select a job j randomly from $\mathcal{R}$. The selected job is inserted into the k' th position in $\mathcal{S}'$. A feasible schedule is obtained by dispatching the set of jobs $\mathcal{S}'$ in forward. Based on the schedule, an upper bound UB'' is calculated.

Step 6. If $UB'' \leq UB'$, then $UB' = UB''$ and $r' = k'$. Delete job j from $\mathcal{S}'$, $k' := k' + 1$. If $k' \leq |\mathcal{S}'| + 1$ then return to Step 5.

Step 7. If $k' > |\mathcal{S}'| + 1$, then job j is inserted in the r'th position in $\mathcal{S}'$. $\mathcal{R} = \mathcal{R} \backslash j$ and return to Step 4. If $\mathcal{R} = \phi$ and $UB' \leq UB$, then $UB = UB'$, $\mathcal{S} = \mathcal{S}'$, $k := k + 1$. If k is more than predetermined value, the algorithm is finished. Otherwise return to Step 3.

6 Computational experiments

Computational experiments are executed for the SDST flowshop with total weighted tardiness (SDST-WT) and the SDST flowshop with total weighted flowtime (SDST-WFT) problems. The flowshop with total weighted flowtime can be realized easily by setting the due date for all jobs zero. The parameters for the problem instances are shown in Table 1. The number of jobs N is selected from $N = \{5,\ 10,\ 20,\ 30, 40, 50\}$, the number of stages $L = 3$. Ten instance problems are created for each case. Setup time $S_{j,i,l}$ for each instance is selected from uniform distribution on [1, 9] on the condition that $S_{j,i,1} = S_{j,i,2} = \cdots = S_{j,i,L}$. The processing time is generated from uniform distribution on [1, 20]. The convergence condition for [**LR algorithm**] is that LB and UB is less than 1.00 or the lower bound has not been updated 50 times, and the convergence condition for [**Proposed algorithm**] at Step 4 and Step 9 is that the difference between LB and UB is less than 1.00 or the lower bound has not been updated 2 times. The condition The Intel Pentium D 3.4GHz with 1GB memory is used for computation.

The effects of cut generation are investigated for SDST-WT problems. The lower bound derived by the proposed method with no cuts (LB with no cut), the proposed method with cut (30) (LB with cut1), the proposed method with cut (31) (LB with cut2), the proposed method with cut (30) and (31) (LB with cut1+cut2) are summarized in Table 2. The results of computation time are summarized in Table 3.The results demonstrate that LB with cut2 is more better than LB with cut1 for all problems and cut1+cut2 is more effective because the total CPU time for cut1+cut2 is almost the same as that of cut2. The performance of cut1 is not effective for large-sized problems. This is because cut1 is very weak for large-sized problems when the feasible region for relaxed problem is too large. From these results, it is demonstrated that LR with cut1+cut2 is more effective than LR with no cuts without significantly increasing the total computation time.

The performance of duality gap for the proposed method with two types of cuts is demonstrated for SDST-WT and SDST-WFT problems. Average computational results of ten cases are summarized in Tables 4, 5. The performance of average duality gap for the proposed method for SDST-WT, SDST-WFT is within 14% for all problems. Especially for SDST-WFT, duality gap is within 12% in worst cases although the total computation time is within 2000 second for all cases. The results demonstrate that the proposed method can derive near-optimal solution for SDST-WT, SDST-WFT problems.

In order to compare the performance between the other methodologies, the performance of lower bound for three types of methods are compared for the proposed method (LRCUT), branch and bound for solving integer programming problem by CPLEX10.1 (CPLEX) with maximum 3,600 seconds of computation time, and branch and bound method (B&B) with maximum 3,600 seconds of computation time. For B&B, the branching is executed by enumerating the sequence of operations and calculating the starting time of operations in forward. The lower bound for each node is computed by

$$LB_{B\&B} = \sum_{i \in N_{fix}} w_i \max\{0,\ c_{i,L}^{fix} - d_i\} + \sum_{i \in N_{free}} w_i \max\{0,\ c_{i,L}^{free} - d_i\} \qquad (34)$$

where N_{fix} is the fixed job sequence, and N_{free} is the unfixed job sequence in the branching tree. $c_{i,L}^{fix}$ is the earliest completion time at the last stage L considering

Table 1 Parameters for the instance problems

Number of jobs: N	5, 10, 20, 30, 40, 50
Number of stages: L	3
Due date: d_i for SDST-WT	[0, $2N$]
Due date: d_i for SDST-WFT	0
Weight: w_i	[1, 10]
Processing time: $p_{i,l}$	[1, 20]
Setup time: $S_{j,i,l}$	[1, 9]

Table 2 Effects of cut generation to the performance of lower bound

N	LB without cut	LB with cut1	LB with cut2	LB with cut1+cut2
10	3857.8	3870.4	3901.4	3901.9
20	9060.4	9067.2	9193.0	9193.6
30	21280.1	21285.3	21387.5	21385.8
40	31138	31138	31174.2	31173.7
50	45984	45984	46015.5	46015.5

Table 3 Effects of cut generation to the total computation time [sec.]

N	Time without cut	Time with cut1	Time with cut2	Time with cut1+cut2
10	9.95	12.54	12.69	12.60
20	88.49	100.3	109.2	109.2
30	296.21	327.69	338.9	339.1
40	633.14	633.14	648.1	644.3
50	1164.24	1286.47	1301.4	1301.5

precedence constraints and setup time in the fixed sequence N_{fix}, $c_{i,L}^{free}$ is the earliest completion time at the last stage L for the job in the last sequence in N_{free}. If the computation time for CPLEX and B&B is larger than 3600 sec., the NEH algorithm combined with local search is executed to derive a feasible solution.

If the number of jobs N is 5 or 10, the lower bound for CPLEX or B&B is better than that of the proposed method. However, if the number of jobs is increased, the lower bound derived by CPLEX or B&B is extremely smaller than that of the proposed method. This is because computational complexity is significantly increased if the number of jobs is increased. It is extremely difficult for CPLEX or B&B to obtain a good lower bound in realistic computation time. The proposed method can generate good lower bound even for 50 job problem in reasonable computation time. It is demonstrated that the lower bound of the proposed method is better than that of the conventional methods.

7 Concluding remarks

In this paper, we have proposed a Lagrangian relaxation and cut generation for sequence dependent setup time flowshop problem with the total weighted tardiness. The original problem has been decomposed into job-level subproblems by the Lagrangian relaxation of capacity constraints and the removal of sequence dependent setup time constraints. The additional constraints on the relaxed problem have been imposed to strengthen the lower bound. Computational experiments have demonstrated the effectiveness of the lower bound of the proposed method compared with that of the CPLEX

Table 4 Computational results for SDST-WT

N		UB	LB	Time(s)	DGAP(%)	$\frac{UB_{LRCUT}}{UB_{CPLEX}}$[%]	$\frac{UB_{LRCUT}}{UB_{CPLEX}}$[%]
5	LRCUT	1279.6	1256.2	2.1	2.09	98.17	98.17
5	CPLEX	1279.6	1279.6	0.33	0		
5	B&B	1279.6	1279.6	0.17	0		
10	LRCUT	4243.6	3930.1	18.68	7.87	93.92	92.61
10	CPLEX	4243.6	4184.7	1111.4	1.11		
10	B&B	4243.6	4243.6	7.11	0		
20	LRCUT	10491.3	9244.3	154.33	13.49	287.08	136.42
20	CPLEX	-	3220.1	3600	-		
20	B&B	10477.7	6776.5	3600	54.53		
30	LRCUT	24251.2	21456.1	480.9	13.1	575.49	256.77
30	CPLEX	-	3728.3	3600	-		
30	B&B	24200.6	8356.2	3600	190.32		
40	LRCUT	35196.1	31203.3	908.02	12.97	878.67	486.30
40	CPLEX	-	3551.2	3600	-		
40	B&B	35202.8	6416.5	3600	472.58		
50	LRCUT	51984.6	46016.7	1782.97	13.19	1530.27	1048.03
50	CPLEX	-	3007.1	3600	-		
50	B&B	51913.4	4390.8	3600	1138.81		

Table 5 Computational results for SDST-WFT

N		UB	LB	Time(s)	DGAP(%)	$\frac{UB_{LRCUT}}{UB_{CPLEX}}$[%]	$\frac{UB_{LRCUT}}{UB_{CPLEX}}$[%]
5	LRCUT	1583.1	1553.4	1.41	1.9	98.12	98.12
5	CPLEX	1583.1	1583.1	0.34	0		
5	B&B	1583.1	1583.1	0.18	0		
10	LRCUT	4719.5	4429.1	19.38	6.43	95.23	93.85
10	CPLEX	4727.9	4651.1	1149.1	1.25		
10	B&B	4719.5	4719.5	7.19	0		
20	LRCUT	13403.9	12109.3	163.67	10.68	212.35	129.67
20	CPLEX	-	5702.6	3600	-		
20	B&B	13439.6	9338.7	3600	43.2		
30	LRCUT	26641.8	23951.6	435.89	11.37	314.67	189.06
30	CPLEX	-	7611.6	3600	-		
30	B&B	26445.5	12668.8	3600	108.59		
40	LRCUT	44060.8	39501.3	984.01	11.73	400.73	269.76
40	CPLEX	-	9857.4	3600	-		
40	B&B	44120.2	14642.9	3600	202.13		
50	LRCUT	69755.6	63190.2	1803.24	10.48	510.44	393.88
50	CPLEX	-	12379.6	3600	-		
50	B&B	68941.6	16042.9	3600	330.41		

and the standard branch and bound algorithm. Future work is to eliminate duality gap for large-sized problems by the consideration of cuts with reasonable computation time.

References

1. R. Ruiz, T. Stützle, “An iterated greedy heuristic for the sequence dependent setup times flowshop scheduling problem with makespan and weighted tardiness objectives”, *European Journal of Operational Research*, vol. 187, pp. 1143–1159, 2008.
2. T. Eren, E. Güner, “A bicriteria flowshop scheduling problem with setup times”, *Applied Mathematics and Computation*, vol. 183, pp. 1292–1300, 2006.
3. R. Ruiz, C. Maroto, J. Alcaraz, “Solving the flowshop scheduling problem with sequence dependent setup times using advances metaheuristics”, *European Journal of Operational Research*, vol. 165, pp. 34–54, 2005.

4. M. Nawaz, E.E. Enscore Jr., I. Ham, "A heuristic algorithm for the m-Machine n-Job Flow-shop Sequencing Problem", *OMEGA*, vol. 11, no. 1, pp. 91–95, 1983.
5. X. Luo, F. Chu, "A branch and bound algorithm of the single machine schedule with sequence dependent setup times for minimizing total tardiness", *Applied Mathematics and Computation*, vol. 183, pp. 575–588, 2006.
6. P.L. Rocha, M.G. Ravetti, G.R. Mateus, P.M. Pardalos, "Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times", *Computers and Operations Research*, vol. 35, pp. 1250–1264, 2008.
7. T. Ibaraki and Y. Nakamura, "A dynamic programming method for single machine scheduling", *European Journal of Operational Research*, vol. 38, no. 7, pp. 1066–1079, 1993
8. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job-shop scheduling problems,", *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 1-13, 1993
9. P.B. Luh, D.J. Hoitomt, E. Max, and K.R. Pattipati, "Scheduling generation and reconfiguration for parallel machines," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 687–696, 1990.
10. T. Nishi, Y. Hiranaka, M. Inuiguchi, "A successive Lagrangian relaxation method for solving flowshop scheduling problems with total weighted tardiness," in *Proc. 3rd Annual Conference on Automation Science and Engineering*, pp. 875–880, 2007.
11. H. Chen, C. Chu and J.M. Proth, "A more efficient Lagrangian relaxation approach to job-shop scheduling problems," *IEEE Int. Conf. Robotics and Automation*, pp. 495–501, 1995.
12. H. Chen, C. Chu and J.M. Proth, "An improvement of the Lagrangian relaxation approach for job shop scheduling: A dynamic programming method," *IEEE Trans. Robot. Automat*, vol. 13, no. 5, pp. 786–795, 1998.
13. G. Stecco, J.F. Cordeau, E. Moretti, "A branch-and-cut algorithm for a production scheduling problem with sequence-dependent and time-dependent setup times," *Computers and Operations Research*, vol. 35, pp. 2635–2655, 2008.
14. R.Z. Ríos-Mercado, J.F. Bard, "A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times," *IIE Transactions*, vol. 31, pp. 721–731, 1999.