A NEW LAGRANGIAN BOUND FOR MIN-SUM JOB-SHOP SCHEDULING

Shunji Tanaka

Institute for Liberal Arts and Sciences / Department of Electrical Engineering,
Kyoto University,
Kyotodaigaku-Katsura, Nishikyo-ku, Kyoto 615-8510, Japan
E-mail tanaka@kuee.kyoto-u.ac.jp

Ruslan Sadykov and Boris Detienne

INRIA team RealOpt and Mathematics Institute, University of Bordeaux,
33405 Talence, France
ruslan.sadykov@inria.fr, boris.detienne@math.u-bordeaux1.fr

Abstract

This study will propose a new Lagrangian bound for the min-sum job-shop scheduling problem. The proposed method is an integration of two types of Lagrangian relaxation methods known as job-level and machine-level decomposition methods. It will be shown by numerical experiments that the proposed method can improve the existing lower bounds significantly.

Keywords: Min-sum job-shop scheduling, Lagrangian relaxation, Job-level decomposition. Machine-level decomposition.

1. INTRODUCTION

The job-shop scheduling problem has been extensively studied for these decades as a challenging multi-machine scheduling problem. Most of the existing studies for job-shop scheduling treated the makespan minimization problem, and the constraint programming approach has successfully derived rich theoretical results (Carlier and Pinson, 1989; Brucker et al., 1994; Carlier and Pinson, 1994; Baptiste et al., 2001). In contrast, the mathematical programming approach seems promising for the problem with min-sum type objective functions such as total weighted completion time, total weighted tardiness, total weighted earliness-tardiness, and so on. Lagrangian relaxation is one of the key techniques in this approach, and it enables us to decompose the original problem into relatively easier subproblems. Chen et al. (1998) showed that relaxing machine capacity constraints in a time-indexed formulation of the problem yields job-level subproblems, which can be solved easily by dynamic programming or linear programming. Chen and Luh (2003) proposed an alternative method to relax precedence constraints among operations of a job. Because the resulting subproblems (referred to as machine-level subproblems) remain difficult to solve, they also relaxed constraints that define the earliness and tardiness of operations. On the other hand, Baptiste et al. (2008) solved the machine-level subproblems to optimality, and compared the two methods, the job-level and machine-level decomposition methods by numerical experiments. These three studies aimed at obtaining tight lower and upper bounds. With regard to an exact algorithm, Lancia et al. (2011) proposed a branch-and-price algorithm for the min-sum job-shop scheduling problem in which the pricing problem is the job-level subproblem. They also proposed a network flow formulation of the precedence constraints among operations. It is true that the algorithm is not so efficient for the problem with regular objectives as the branch-and-bound algorithms on a disjunctive graph (Singer and Pinedo, 1998; Brune et al., 2012). However, to the best of the authors' knowledge, there do not exist better exact algorithms for the problem with the nonregular earliness-tardiness objective.

In this paper, we will propose a new method to improve the Lagrangian lower bounds by the two relaxation methods. The proposed method is an integration of the two in the sense that the resulting Lagrangian relaxation is composed of both the types of subproblems. It is expected that this method enables us to improve the efficiency of exact algorithms for the min-sum job-shop scheduling problem based on the mathematical programming approach. The effectiveness of the proposed method will be demonstrated by numerical experiments.

2. THE MIN-SUM JOB-SHOP SCHEDULING PROBLEM

In this section we will describe the min-sum job-shop scheduling problem considered in this study. We will also introduce notations and definitions.

Let us suppose that a set of n jobs $\{\mathrm{J}_1, \ldots, \mathrm{J}_n\}$ should be processed on a set of m machines $\{\mathrm{M}_, \ldots, \mathrm{M}_m\}$. Each job J_i is given a release date $r_i \geq 0$, and is composed of n_i operations $\mathrm{O}_{i1}, \ldots, \mathrm{O}_{i,n_i}$ that should be processed in this order on $\mathrm{M}_{m_{i1}}, \ldots, \mathrm{M}_{m_{i,n_i}}$, respectively. Here, no reentrant is considered: m_{ik} satisfy $m_{ik} \neq m_{il}$ for any $k \neq l$. Operations $\mathrm{O}_{i1}, \ldots, \mathrm{O}_{i,n_i}$ of J_i cannot be processed simultaneously: O_{i2} cannot

be started before O_{i1} is finished, O_{i3} cannot be started before O_{i2} is finished, and so on. Each operation O_{ij} is given a processing time p_{ij} and a cost function $f_{ij}(t)$. Each machine can process at most one operation at a time, and preemption of the processing is forbidden. The completion time of O_{ij} is denoted by C_{ij}. The objective is to find a schedule that minimizes the sum of completion costs $\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}(C_{ij})$. Throughout this paper, we assume that the release dates r_i and the processing times p_{ij} are all integers. Accordingly, completion times C_{ij} are assumed to be all integers.

Here, we will summarize the notations and definitions used in this paper.

- n: the number of jobs,
- m: the number of machines,
- $\mathcal{N}$: the set of job indices ($\mathcal{N} = \{1, \ldots, n\}$),
- $\mathcal{M}$: the set of machine indices ($\mathcal{M} = \{1, \ldots, m\}$),
- J_i: the ith job ($i \in \mathcal{N}$),
- n_i: the number of operations in J_i ($i \in \mathcal{N}$),
- O_i: the set of operation indices of J_i ($O_i = \{1, \ldots, n_i\}$, $i \in \mathcal{N}$),
- O_{ij}: the jth operation of J_i ($i \in \mathcal{N}$, $j \in O_i$),
- M_k: the kth machine ($k \in \mathcal{M}$),
- r_i: the release date of J_i ($i \in \mathcal{N}$),
- p_{ij}: the processing time of O_{ij} ($i \in \mathcal{N}$, $j \in O_i$),
- m_{ij}: the index of the machine that processes O_{ij} ($i \in \mathcal{N}$, $j \in O_i$),
- $\mathcal{N}_k$: the set of the indices of the jobs that have an operation to be processed on M_k ($\mathcal{N}_k = \{i \in \mathcal{N} \mid m_{ij} = k, \exists j \in O_i\}$),
- o_{ik}: the index of the operation of J_i that should be processed on M_k ($m_{i,o_{ik}} = k$, $i \in \mathcal{N}_k$, $k \in \mathcal{M}$),
- $f_{ij}(t)$: the cost function of O_{ij} at t ($i \in \mathcal{N}$, $j \in O_i$),
- C_{ij}: the completion time of O_{ij} ($i \in \mathcal{N}$, $j \in O_i$),
- T: the length of the planning horizon,
- r_{ij}: the head of O_{ij} ($r_{ij} = r_i + \sum_{1 \le l \le j-1} p_{il}$, $i \in \mathcal{N}$, $j \in O_i$),
- q_{ij}: the tail of O_{ij} ($q_{ij} = \sum_{j \le l \le n_i} p_{il}$, $i \in \mathcal{N}$, $j \in O_i$),
- $\mathcal{T}_{ij}$: the set of possible starting times of O_{ij} ($\mathcal{T}_{ij} = [r_{ij}, T - q_{ij}] = \{r_{ij}, r_{ij}+1, \ldots, T - q_{ij}\}$, $i \in \mathcal{N}$, $j \in O_i$),
- $\mathcal{T}'_{ij}$: $\mathcal{T}'_{ij} = \mathcal{T}_{ij} \setminus \{T - q_{ij}\}$. ($i \in \mathcal{N}$, $j \in O_i$),
- $[S_k, T_k]$: the planning horizon on M_k ($S_k = \min_{i \in \mathcal{N}_k} r_{i,o_{ik}}$, $T_k = T - \min_{i \in \mathcal{N}_k}(q_{i,o_{ik}} - p_{i,o_{ik}})$),
- $\mathcal{T}_k$: the set of time instants in the planning horizon on M_k ($\mathcal{T}_k = [S_k, T_k - 1] = \{S_k, S_k + 1, \ldots, T_k - 1\}$).

3. TWO TIME-INDEXED FORMULATIONS FOR THE MIN-SUM JOB-SHOP SCHEDULING PROBLEM

In this section, we will introduce two types of time-indexed formulations for the min-sum job-shop scheduling problem. In the first formulation, precedence constraints among operations of a job are expressed simply by their starting times, whereas the second formulation proposed by Lancia et al. (2011) adopts a network flow formulation for them.

3.1. Time-Indexed Formulation Based on Starting Times (P_1)

Let us introduce the following binary decision variables x_{ijt} ($i \in \mathcal{N}$, $j \in O_i$, $t \in \mathcal{T}_{ij}$) that become 1 if and only if O_{ij} is started at t on $M_{m_{ij}}$. Then, our min-sum job-shop scheduling problem can be formulated as the following (P_1):

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} f_{ij}(t + p_{ij}) x_{ijt}, \tag{1}$$

$$\text{s.t.} \sum_{t \in \mathcal{T}_{ij}} x_{ijt} = 1, \qquad j \in O_i, i \in \mathcal{N}, \tag{2}$$

$$\sum_{\substack{i \in \mathcal{N}_k \\ s \in \mathcal{T}_{i,o_{ik}} \cap [t - p_{i,o_{ik}} + 1, t]}} x_{i,o_{ik},s} \le 1, \qquad t \in \mathcal{T}_k, k \in \mathcal{M}, \tag{3}$$

$$\sum_{t \in \mathcal{T}_{i,j+1}} t x_{i,j+1,t} \ge \sum_{t \in \mathcal{T}_{ij}} t x_{ijt} + p_{ij}, \quad j \in O_i \setminus \{n_i\}, i \in \mathcal{N}, \tag{4}$$

$$x_{ijt} \in \{0, 1\}, \qquad t \in \mathcal{T}_{ij}, j \in O_i, i \in \mathcal{N}. \tag{5}$$

In this formulation, constraints on job occurrences (2) require that every operation should be processed exactly once, and (3) are capacity constraints to ensure that more than one operation cannot be processed in $[t, t+1)$ on M_k. The precedence constraint between consecutive operations O_{ij} and $O_{i,j+1}$ of J_i is provided in terms of their starting times as in (4).

3.2. Time-Indexed Formulation Based on Network Flows (P_2)

Let us introduce additional continuous decision variables y_{ijt} ($i \in \mathcal{N}$, $j \in O_i$, $t \in \mathcal{T}'_{ij}$) that become 1 if and only if O_{ij} is not started until t on $M_{m_{ij}}$ although it is ready for processing. By these decision variables, the precedence constraints can be rewritten as in the following (P_2):

$$\min \ (1),$$

$$\text{s.t.} \ (3), (5),$$

$$x_{i,1,r_{i1}} + y_{i,1,r_{i1}} = 1, \qquad i \in \mathcal{N}, \tag{6}$$

$$x_{i1t} + y_{i1t} = y_{i,1,t-1}, \qquad t \in \mathcal{T}'_{i1} \setminus \{r_{i1}\}, i \in \mathcal{N}, \tag{7}$$

$$x_{i,1,T-q_{i1}} = y_{i,1,T-q_{i1}-1}, \quad i \in \mathcal{N}, \tag{8}$$

$$x_{i,j,r_{ij}} + y_{i,j,r_{ij}} = x_{i,j-1,r_{i,j-1}}, \qquad j \in O_i \setminus \{1\}, i \in \mathcal{N}, \tag{9}$$

$$x_{ijt} + y_{ijt} = x_{i,j-1,t-p_{i,j-1}} + y_{i,j,t-1}, \qquad t \in \mathcal{T}'_{ij} \setminus \{r_{ij}\}, j \in O_i \setminus \{1\}, \quad i \in \mathcal{N}, \tag{10}$$

$$x_{i,j,T-q_{ij}} = x_{i,j-1,T-q_{i,j-1}} + y_{i,j,T-q_{ij}-1},$$

$$j \in O_i \setminus \{1\},\ i \in \mathcal{N}, \quad (11)$$

$$0 \le y_{ijt} \le 1, \qquad t \in \mathcal{T}'_{ij},\ j \in O_i,\ i \in \mathcal{N}. \quad (12)$$

This formulation assumes one network for each job to express the corresponding precedence constraints. The network for J_i is composed of n_i layers, and the jth layer is composed of nodes v_{ijt} that represent discretized time instants t. A flow enters $v_{i,j+1,t+p_{ij}}$ from v_{ijt} if and only if O_{ij} is started at t, which is represented by x_{ijt}. Decision variables y_{ijt} represent flows from v_{ijt} to $v_{i,j,t+1}$. A unit flow enters the network only from $v_{i,1,r_{i1}}$, which is specified by (6). The flow exits only from the n_i-th layer. Since constraints (5) ensure the integrity of any flows from v_{ijt} to $v_{i,j+1,t+p_{ij}}$, y_{ijt} can take only binary variables although they are continuous. Thus, no integrity constraints are imposed on y_{ijt}. It should also be noted that constraints on job occurrences (2) are unnecessary in (P_2).

There is a slight modification from the formulation by Lancia et al. (2011) with regard to how the total flow is restricted. The total *outgoing* flow from the network is restricted to one in their formulation, whereas the total *incoming* flow is restricted to one in (P_2).

4. PROPOSED LAGRANGIAN RELAXATION APPROACH

One of the most intuitive ways to obtain a lower bound of the objective value of (P_1) or (P_2) is to solve the LP relaxation (LPR_1) of (P_1) or (LPR_2) of (P_2), where integrity constraints (5) are relaxed. However, it takes a considerable amount of computation time because the number of decision variables is large. An alternative way that is often employed in the literature is to apply the Lagrangian relaxation technique. As already explained in Introduction, there exist two Lagrangian relaxation methods. Chen et al. (1998) proposed a method to relax machine capacity constraints (3) in (P_1) or (P_2), which enables us to decompose the resulting Lagrangian relaxation into subproblems for individual jobs. Thus it is referred to as job-level decomposition. Another method was proposed by Chen and Luh (2003), and it relaxes precedence constraints (4) in (P_1). It is referred to as machine-level decomposition because subproblems for individual machines are derived. These subproblems are still difficult to solve to optimality, and hence Chen and Luh (2003) relaxed a different type of constraints as well to derive easier subproblems. On the other hand, Baptiste et al. (2008) solved them to optimality. They treated the job-shop scheduling problem to minimize total weighted earliness-tardiness, for which the machine-level subproblems are the single-machine earliness-tardiness problem. Therefore, they solved them by applying the branch-and-bound algorithm by Sourd and Kedad-Sidhoum (2003).

It is not difficult to prove that the job-level decomposition yields as tight a lower bound as (LPR_2) if the multipliers are adjusted to optimality. Although (LPR_1) cannot outperform (LPR_2) in terms of the tightness of the lower bound, the machine-level decomposition that starts from (P_1) can obtain a better lower bound than (LPR_1) by solving the subproblems to optimality. In this section, we will propose a new relaxation method that integrates these two. We will explain it by (P_1) for simplicity of explanation, but equivalent results are obtained even if we start from (P_2).

First, decision variables x_{ijt} are duplicated and (P_1) is rewritten as:

$$\begin{aligned}
&\min\ (1),\\
&\text{s.t. } (2), (3), (5),\\
&\quad x_{ijt} = x'_{ijt}, && t \in \mathcal{T}_{ij},\ j \in O_i,\ i \in \mathcal{N}. && (13)\\
&\quad \sum_{t \in \mathcal{T}_{ij}} x'_{ijt} = 1, && j \in O_i,\ i \in \mathcal{N}, && (14)\\
&\quad \sum_{t \in \mathcal{T}_{i,j+1}} t x'_{i,j+1,t} \ge \sum_{t \in \mathcal{T}_{ij}} t x'_{ijt} + p_{ij}, && j \in O_i \setminus \{n_i\},\ i \in \mathcal{N}, && (15)\\
&\quad x'_{ijt} \in \{0,1\}, && t \in \mathcal{T}_{ij},\ j \in O_i,\ i \in \mathcal{N}. && (16)
\end{aligned}$$

Please note that constraints on job occurrences (2) are duplicated at the same time, while precedence constraints (4) are expressed only by x'_{ijt}. Next, the violation of coupling constraints (13) is penalized by multipliers λ_{ijt} ($i \in \mathcal{N}$, $j \in O_i$, $t \in \mathcal{T}_{ij}$), which modifies (1) as follows:

$$\begin{aligned}
&\sum_{i \in \mathcal{N}} \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} f_{ij}(t + p_{ij}) x_{ijt} + \sum_{i \in \mathcal{N}} \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} \lambda_{ijt}(x'_{ijt} - x_{ijt})\\
&\qquad = \sum_{i \in \mathcal{N}} \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} \Lambda_{ijt} x_{ijt} + \sum_{i \in \mathcal{N}} \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} \lambda_{ijt} x'_{ijt}. \quad (17)
\end{aligned}$$

Here, Λ_{ijt} ($i \in \mathcal{N}$, $j \in O_i$, $t \in \mathcal{T}_{ij}$) are defined by

$$\Lambda_{ijt} = f_{ij}(t + p_{ij}) - \lambda_{ijt}. \quad (18)$$

From (17), we obtain the following relaxation (LR):

$$\begin{aligned}
&L(\boldsymbol{\lambda}) = \min \sum_{i \in \mathcal{N}} \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} \Lambda_{ijt} x_{ijt} + \sum_{i \in \mathcal{N}} \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} \lambda_{ijt} x'_{ijt}, \quad (19)\\
&\qquad \text{s.t. } (2), (3), (5), (14), (15), (16).
\end{aligned}$$

This relaxation can be decomposed into job-level subproblems $(\mathrm{LR}_i^{\mathrm{J}})$ ($i \in \mathcal{N}$) and machine-level subproblems $(\mathrm{LR}_k^{\mathrm{M}})$ ($k \in \mathcal{M}$). Job-level subproblems $(\mathrm{LR}_i^{\mathrm{J}})$ are defined by

$$\begin{aligned}
&L_i^{\mathrm{J}}(\boldsymbol{\lambda}) = \min \sum_{j \in O_i} \sum_{t \in \mathcal{T}_{ij}} \Lambda_{ijt} x_{ijt}, && && (20)\\
&\qquad \text{s.t. } \sum_{t \in \mathcal{T}_{ij}} x_{ijt} = 1, && j \in O_i, && (21)\\
&\qquad \sum_{t \in \mathcal{T}_{i,j+1}} t x_{i,j+1,t} \ge \sum_{t \in \mathcal{T}_{ij}} t x_{ijt} + p_{ij}, && && \\
&\qquad && j \in O_i \setminus \{n_i\}, && (22)\\
&\qquad x_{ijt} \in \{0,1\}, && t \in \mathcal{T}_{ij},\ j \in O_i. && (23)
\end{aligned}$$

Machine-level subproblems ($\mathrm{LR}_k^{\mathrm{M}}$) are defined by

$$L_k^{\mathrm{M}}(\boldsymbol{\lambda}) = \min \sum_{i\in\mathcal{N}_k}\sum_{t\in\mathcal{T}_{i,o_{ik}}} \lambda_{i,o_{ik},t} x'_{i,o_{ik},t}, \tag{24}$$

$$\text{s.t.} \sum_{t\in\mathcal{T}_{i,o_{ik}}} x'_{i,o_{ik},t} = 1, \qquad i\in\mathcal{N}_k, \tag{25}$$

$$\sum_{\substack{i\in\mathcal{N}_k \\ s\in\mathcal{T}_{i,o_{ik}}\cap[t-p_{i,o_{ik}}+1,t]}} x'_{i,o_{ik},s} \le 1, \ t\in\mathcal{T}_k, \tag{26}$$

$$x'_{i,o_{ik},t} \in \{0,1\}, \qquad t\in\mathcal{T}_{i,o_{ik}}, i\in\mathcal{N}_k, \tag{27}$$

and $L(\boldsymbol{\lambda})$ is given by

$$L(\boldsymbol{\lambda}) = \sum_{i\in\mathcal{N}} L_i^{\mathrm{J}}(\boldsymbol{\lambda}) + \sum_{k\in\mathcal{M}} L_k^{\mathrm{M}}(\boldsymbol{\lambda}). \tag{28}$$

Job-level subproblems ($\mathrm{LR}_i^{\mathrm{J}}$) are easy to solve by dynamic programming in $O(n_i T)$ time (Chen et al., 1998). Machine-level subproblems ($\mathrm{LR}_k^{\mathrm{M}}$) are the min-sum single-machine scheduling problem with general cost functions, which can be solved to optimality by applying the exact algorithm by Tanaka and Fujikuma (2012).

Relaxation (LR) integrates the two existing decomposition methods. Therefore, we can expect that $L(\boldsymbol{\lambda})$ by (LR) is at least as tight a lower bound as those by the two. Although the proofs are omitted here, this claim is true, and $L(\boldsymbol{\lambda})$ is not less than those by the job-level and machine-level decomposition methods, if λ_{ijt} are chosen as

$$\lambda_{ijt} = -\sum_{s\in[t,t+p_{ij}-1]} \lambda^{\mathrm{J}}_{m_{ij},t}, \tag{29}$$

and

$$\lambda_{ijt} = \begin{cases} f_{ij}(t+p_{ij}) + \lambda^{\mathrm{M}}_{ij} t, & j=1, \\ f_{ij}(t+p_{ij}) + (\lambda^{\mathrm{M}}_{ij} - \lambda^{\mathrm{M}}_{i,j-1})t, & j\in O_i\setminus\{1,n_i\}, \\ f_{ij}(t+p_{ij}) - \lambda^{\mathrm{M}}_{i,j-1} t, & j=n_i, \end{cases} \tag{30}$$

respectively. Here, $\lambda^{\mathrm{J}}_{kt} \ge 0$ ($k\in\mathcal{M}$, $t\in\mathcal{T}_k$) are the multipliers associated with capacity constraints (3) in the job-level decomposition, and $\lambda^{\mathrm{M}}_{ij} \ge 0$ ($i\in\mathcal{N}$, $j\in O_i\setminus\{n_i\}$) the multipliers associated with precedence constraints (4) in the machine-level decomposition.

5. UPPER BOUND OF THE LENGTH OF THE PLANNING HORIZON

Since the number of decision variables in time-indexed formulations depends on the length of the planning horizon, it is desirable to keep T as small as possible. However, too small T may exclude all optimal solutions. In this section, we will propose a simple method to calculate a tight bound of T that does not exclude at least one optimal solution.

The proposed method utilizes an upper bound UB of the objective value obtained by some heuristics. It is obvious that we can restrict our attention to those solutions whose objective values are less than or equal to UB, in order to find an optimal solution. Hence, we will search for such a T that the objective value becomes larger than UB if the last operation in a solution is completed after T .

First, let us assume that cost functions $f_{ij}(t)$ are regular: $f_{ij}(t)$ are nondecreasing functions of t. In this case, we only need to consider solutions without unforced idle time. Let us sequence the operations in the nondecreasing order of their completion times, and denote them by O_{i_1,j_1}, O_{i_2,j_2}, …, $\mathrm{O}_{i_N j_N}$, respectively, where N is the total number of the operations defined by $N=\sum_{i\in\mathcal{N}} n_i$. Suppose that the last operation $\mathrm{O}_{i_N j_N}$ is completed at T. Then, lower bounds of C_{i_l,j_l} ($1\le l<N$) are given by

$$C_{i_l,j_l} \ge T - \sum_{l+1\le k\le N} p_{i_k,j_k}, \tag{31}$$

because otherwise, O_{i_k,j_k} ($l+1\le k\le N$) can be completed earlier without increasing the objective value. From (31), we obtain a lower bound $\mathrm{LB}(T)$ of the objective value of this solution as follows:

$$\mathrm{LB}(T) = \sum_{1\le l\le N} f_{i_l,j_l}\left(\max\left(r_{i_l,j_l}+p_{i_l,j_l}, T - \sum_{l+1\le k\le N} p_{i_k,j_k}\right)\right). \tag{32}$$

$\mathrm{LB}(T)$ can be computed by solving the min-sum single-machine scheduling problem to schedule N operations O_{ij} ($i\in\mathcal{N}$, $j\in O_i$) without idle time, where the cost functions $f'_{ij}(t)$ are given by

$$f'_{ij}(t) = \begin{cases} f_{ij}(t+T-P), & t\ge P-T+r_{ij}+p_{ij}, \\ f_{ij}(r_{ij}+p_{ij}), & \text{otherwise.} \end{cases} \tag{33}$$

Here, P is the total processing time defined by

$$P = \sum_{i\in\mathcal{N}}\sum_{j\in O_i} p_{ij}. \tag{34}$$

If $\mathrm{LB}(T) > \mathrm{UB}$, we can safely say that the objective value of any solution in which not all the operations are completed until T is larger than UB. The single-machine scheduling problem can be solved by applying the exact algorithm by Tanaka et al. (2009), and the minimum T satisfying $\mathrm{LB}(T)\le\mathrm{UB}$ and $\mathrm{LB}(T+1)>\mathrm{UB}$ can be searched for by a bisection algorithm.

Next, let us consider the case when $f_{ij}(t)$ is nonregular. We assume that $f_{ij}(t)$ is regular when $t\ge\tau_{ij}\ge r_{ij}+p_{ij}$ even in this case, and define $\tau_{\max}$ by

$$\tau_{\max} = \max_{\substack{i\in\mathcal{N} \\ j\in O_i}} \tau_{ij}. \tag{35}$$

Since operations completed after $\tau_{\max}$ should be processed without idle time, the above procedure for regular cost functions is applicable if the cost functions in the single-machine problem are chosen as

$$f''_{ij}(t) = \begin{cases} f_{ij}(t+T-P), & t\ge P-T+\tau_{\max}, \\ \min_{t\le\tau_{\max}} f_{ij}(t), & \text{otherwise.} \end{cases} \tag{36}$$

6. NUMERICAL EXPERIMENTS

The proposed relaxation method was applied to the benchmark instances with 10 jobs of the JIT job-shop scheduling problem (Baptiste et al., 2008). In this problem, the number of operations in J_i is equal to the number of machines ($n_i = m$) for all $i \in \mathcal{N}$, and cost functions $f_{ij}(t)$ are specified by

$$f_{ij}(t) = \max\left(\alpha_{ij}(d_{ij} - t), \beta_{ij}(t - d_{ij})\right), \qquad (37)$$

where d_{ij} is the duedate of O_{ij}, and α_{ij} and β_{ij} are penalties for unit earliness and tardiness, respectively. We applied the conjugate subgradient algorithm (Wolfe, 1975; Sherali and Ulular, 1989) to adjust the multipliers for (LR). Because it is time-consuming to solve the machine-level subproblems to optimality, we adjusted the multipliers as follows:

(1) First, multipliers are adjusted for the Lagrangian relaxation by the job-level decomposition until a stopping criterion is satisfied.
(2) By setting the initial values as in (29), multipliers $\boldsymbol{\lambda}$ are adjusted for (LR), where machine-level subproblems ($\mathrm{LR}_i^{\mathrm{J}}$) are solved not to optimality but heuristically by the heuristics used for an initial upper bound in the exact algorithm by Tanaka and Fujikuma (2012), until a stopping criterion is satisfied.
(3) Multipliers $\boldsymbol{\lambda}$ are adjusted by solving ($\mathrm{LR}_i^{\mathrm{J}}$) to optimality by the exact algorithm (Tanaka and Fujikuma, 2012), until the decrease of the duality gap in 500 iterations becomes less than 0.1%.

As an upper bound UB necessary for determining the step size in the conjugate subgradient algorithm, the best result among those by metaheuristic algorithms (Monette et al., 2009; dos Santos et al., 2010; Wang and Li, 2014) was employed. At each iteration of the conjugate subgradient algorithm, individual subproblems ($\mathrm{LR}_i^{\mathrm{J}}$) and ($\mathrm{LR}_k^{\mathrm{M}}$) were solved in parallel by multi-threading. For comparison, two LP relaxations (LPR_1) and (LPR_2) were solved as well by Gurobi Optimizer 5.6.3. All the computation was performed on a desktop computer with an Intel Core i7-3970X CPU (3.5GHz, 6 cores) and 32GB RAM.

The results are summarized in Table 1. In this table, the column "instance" is for the instance type. I-X-Y-Z-U-V stands for the instance with $n =$X and $m =$Y. In addition, Z is the setting of the duedates d_{ij} ("tight" or "loose"), U is the setting of the earliness penalties α_{ij} ("equal" or "tard"), and V is the instance ID. Here, please note that "equal" does not mean that the earliness penalties are identical. In "equal", earliness penalties were chosen randomly in $[0.1, 0.3]$, whereas in "tard", they were in $[0.1, 1.0]$. The second column UB presents the best upper bound known so far. "Baptiste" represents the results by Baptiste et al. (2008), where "machine" and "job" are the lower bounds by the machine-level and job-level decomposition methods, respectively. (LPR_1) and (LPR_2) give the results of the two LP relaxations, respectively. The column "value" shows the optimal objective value, and "gap" the duality gap in percent computed by $(\mathrm{UB} - \mathrm{value})/\mathrm{UB}$. Finally, (LR) presents the results of the proposed method, and the column "time" is the total computation time.

From Table 1, we can see that (LPR_2) always yields a better lower bound than (LPR_1), and the difference of the two bounds becomes more significant as m becomes larger. The proposed method considerably improves the lower bound from that by (LPR_2), The lower bound is so tight that the duality gap was closed for 6 instances with $m = 2$, 2 instances with $m = 5$, and 2 instances with $m = 10$. It is also observed that the gap was relatively large for the instances with $m = 10$. The values in (Baptiste et al., 2008) do not seem to have reached the theoretical limit due to a time limit. As explained in the preceding section, "job" becomes equal to (LPR_2) if the multipliers are optimized. However, there still remains a gap for some instances. Moreover, "machine" is sometimes smaller than (LPR_1), although the former can be expected to exceed the latter. On the other hand, the lower bound by the proposed method, (LR), is always larger than (LPR_2). Although our method seems promising, we should also note that it took a large number of iterations and a long computation time due to poor convergence. For example, the total number of iterations for I-10-10-tight-equal-1 was over 60,000. Thus it is crucial to improve its convergence if we want to apply it as a part of exact algorithms.

7. CONCLUSION

In this study we proposed a new Lagrangian lower bound for the min-sum job-shop scheduling problem that integrates the job-level and machine-level decomposition methods. It was shown by numerical experiments for the instances of the earliness-tardiness job-shop scheduling problem that the proposed method is able to improve the previous lower bounds significantly. However, its convergence is still poor. Hence, future research directions will be to improve the convergence of the conjugate subgradient algorithm, to apply column generation instead of the Lagrangian relaxation technique, and so on.

References

Baptiste, P., C. Le Pape, W. Nuijten. (2001). *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems*, Kluwer Academic Publishers.

Baptiste, P., M. Flamini, and F. Sourd. (2008). Lagrangian bound for just-in-time job-shop scheduling. *Computers & Operations Research*, Vol. 35, pp. 906–915.

Brucker, P., B. Jurisch, and A. Krämer. (1994). The job-shop problem and immediate selection. *Annals of Operations Research*, Vol. 50, pp. 73–114.

Brune, R., G. Zäpfel, and M. Affenzeller. (2012). An exact approach for single machine subproblems in shifting bottleneck procedures for job shops with total weighted tardiness objective. *European Journal of Operational Research*, Vol. 218, No. 1, pp. 76–85.

Table 1 Lower Bounds by the Proposed Decomposition Method

Instance	UB	Baptiste et al. (2008)		(LPR_1)		(LPR_2)		Proposed (LR)		
		machine	job	value	gap	value	gap	value	gap	time (s)
I-10-2-tight-equal-1	461.96	433	434	430.009	6.92	460.037	0.42	461.960	0.00	8.25
I-10-2-tight-equal-2	448.32	418	357	431.008	3.86	436.327	2.68	448.320	0.00	43.74
I-10-5-tight-equal-1	689.11	536	660	527.803	23.41	662.302	3.89	688.674	0.06	2457.58
I-10-5-tight-equal-2	763.24	612	592	709.963	6.98	749.490	1.80	763.240	0.00	45.85
I-10-10-tight-equal-1	1277.44	812	1126	820.492	35.77	1130.020	11.54	1184.395	7.28	1906.62
I-10-10-tight-equal-2	1878.26	819	1535	765.859	59.22	1538.520	18.09	1659.251	11.66	9997.38
I-10-2-loose-equal-1	224.84	219	218	213.870	4.88	218.428	2.85	224.840	0.00	1.74
I-10-2-loose-equal-2	319.37	298	313	285.606	10.57	313.733	1.77	317.542	0.57	194.80
I-10-5-loose-equal-1	1740.08	1205	1263	1399.830	19.55	1627.050	6.50	1680.105	3.45	3551.86
I-10-5-loose-equal-2	967.73	780	878	730.402	24.52	882.336	8.82	945.206	2.33	3016.61
I-10-10-loose-equal-1	364.39	294	331	231.429	36.49	333.149	8.57	355.401	2.47	474.82
I-10-10-loose-equal-2	249.85	211	246	187.550	24.94	247.190	1.06	249.850	0.00	72.15
I-10-2-tight-tard-1	179.46	174	168	171.694	4.33	178.604	0.48	179.250	0.12	92.69
I-10-2-tight-tard-2	145.37	138	143	136.384	6.18	143.843	1.05	145.370	0.00	7.82
I-10-5-tight-tard-1	387.30	322	361	315.254	18.60	362.140	9.08	371.174	4.16	1134.36
I-10-5-tight-tard-2	632.91	461	420	504.153	20.34	567.812	10.29	610.905	3.48	1259.69
I-10-10-tight-tard-1	687.45	408	574	398.911	41.97	577.188	16.04	599.612	12.78	2536.37
I-10-10-tight-tard-2	779.30	469	666	433.908	44.32	667.751	14.31	717.610	7.92	7277.98
I-10-2-loose-tard-1	416.44	408	413	397.444	4.56	413.589	0.68	416.440	0.00	4.53
I-10-2-loose-tard-2	137.94	137	135	134.935	2.18	136.350	1.15	137.940	0.00	9.83
I-10-5-loose-tard-1	175.08	159	168	148.917	14.94	168.923	3.52	175.080	0.00	55.23
I-10-5-loose-tard-2	499.93	313	355	304.920	39.01	451.503	9.69	467.437	6.46	548.83
I-10-10-loose-tard-1	383.86	314	356	272.896	28.91	357.823	6.78	368.823	3.92	1675.99
I-10-10-loose-tard-2	144.94	119	138	103.772	28.40	138.638	4.35	144.940	0.00	41.01

Carlier, J. and E. Pinson. (1989). An algorithm for solving the job-shop problem. *Management Science*, Vol. 35, No. 2, pp. 164–176.

Carlier, J. and E. Pinson. (1994). Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, Vol. 78, pp. 146–161.

Chen, H., C. Chu, and J.-M. Proth. (1998). An improvement of the Lagrangean relaxation approach for job shop scheduling: A dynamic programming method. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 5, pp. 786–795.

Chen, H. and P.B. Luh. (2003). An alternative framework to Lagrangian relaxation approach for job shop scheduling. *European Journal of Operational Research*, Vol. 149, No. 3, pp. 499–512.

dos Santos, A.G., R.P. Araujo, and J.E.C. Arroyo. (2010). A combination of evolutionary algorithm, mathematical programming, and a new local search procedure for the just-in-time scheduling problem. *Lecture Notes in Computer Science*, Vol. 6073, pp. 10–24.

Lancia, G., F. Rinaldi, and P. Serafini. (2011). A time-indexed LP-based approach for min-sum job-shop problems. *Annals of Operations Research*, Vol. 186, pp. 175–198.

Monette, J.N., Y. Deville, and P. van Hentenryck. (2009). Just-in-time scheduling with constraint programming. In: *Proceedings of the 19th International Conference on Automation Planning and Scheduling (ICAPS'09)*, pp. 241–248.

Sherali, H.D. and O. Ulular. (1989). A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems. *Applied Mathematics and Optimization*, Vol. 20, pp. 193–221.

Singer, M. and M. Pinedo. (1998). A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE Transactions*, Vol. 30, No. 2, pp. 109–118.

Sourd, F. and S. Kedad-Sidhoum. (2003). The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, Vol. 6, pp. 533–549.

Tanaka, S., S. Fujikuma, and M. Araki. (2009). An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, Vol. 12, No. 6, pp. 575–593.

Tanaka, S. and S. Fujikuma. (2012). A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. *Journal of Scheduling*, Vol. 15, No. 3, pp. 347–361.

Wang, S. and Y. Li. (2014). Variable neighborhood search and mathematical programming for just-in-time job-shop scheduling problem. *Mathematical Problems in Engineering*, Vol. 2014, ID431325.

Wolfe. P. (1975). A method of conjugate subgradients for minimizing nondifferenciable functions. *Mathematical Programming Study*, Vol. 3, pp. 145–173.