Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

European Journal of Operational Research 149 (2003) 499–512

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Discrete Optimization

# An alternative framework to Lagrangian relaxation approach for job shop scheduling

Haoxun Chen *, Peter B. Luh

*Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269, USA*

Received 9 May 2000; accepted 8 May 2002

**Abstract**

A new Lagrangian relaxation (LR) approach is developed for job shop scheduling problems. In the approach, operation precedence constraints rather than machine capacity constraints are relaxed. The relaxed problem is decomposed into single or parallel machine scheduling subproblems. These subproblems, which are NP-complete in general, are approximately solved by using fast heuristic algorithms. The dual problem is solved by using a recently developed "surrogate subgradient method" that allows approximate optimization of the subproblems. Since the algorithms for subproblems do not depend on the time horizon of the scheduling problems and are very fast, our new LR approach is efficient, particularly for large problems with long time horizons. For these problems, the machine decomposition-based LR approach requires much less memory and computation time as compared to a part decomposition-based approach as demonstrated by numerical testing.

*Keywords:* Scheduling; Job shops; Lagrangian relaxation; Manufacturing systems

## 1. Introduction

Scheduling is to allocate finite resources over time to accomplish a given set of tasks. High quality schedules can improve delivery performance, reduce inventory costs, and are very important to manufacturers in today's time-based competition. To obtain a high quality schedule within an acceptable computation time, however, is extremely difficult because of the NP-hard nature of the problem and the large sizes of practical applications [12].

Scheduling methods in the literature can be classified into optimization methods, approximate and heuristics methods, and dispatching rules. Optimization methods include branch and bound and dynamic programming (DP) (e.g., [3] and [17]). Although they can obtain optimal schedules, they are computationally intensive in general. By contrast, approximate and heuristics methods make a tradeoff between solution quality and computation time. These methods include simulated annealing [11], genetic algorithms [7,8,13],

* Corresponding author. Current Address: Laboratoire d'Optimisation des Systèmes Industriels, Université de Technologie de Troyes, 12 Rue Marie Curie, BP 20, 10010 Troyes Cedex, France. Tel.: +33-325-715642; fax: +33-325-715649.

*E-mail addresses:* hxchen@engr.uconn.edu, chen@utt.fr (H. Chen), luh@engr.uconn.edu (P.B. Luh).

tabu search [6,14], constraint programming [15], shifting bottleneck procedure [1], and Lagrangian relaxation [4,10,19]. Dispatching rules are too simple to be effective in most cases [2,16].

Lagrangian relaxation (LR) has recently emerged as a practical approach for complex scheduling problems [4,10,19]. It is a decomposition and coordination approach which can obtain near optimal schedules with quantifiable quality in a reasonable computation time for practical scheduling problems. In this approach, machine capacity constraints are first relaxed by using Lagrange multipliers. The relaxed problem can be decomposed into a set of part subproblems that are solved by using DP. The multipliers are then iteratively adjusted at the "high level" based on the degree of constraint violation. At the termination of such iterations, a simple heuristic is applied to adjust subproblem solutions to obtain a feasible schedule satisfying all constraints.

This approach is quite efficient in general. However, as the computation complexity of the DP algorithm and the number of multipliers are proportional to the time horizon, the approach becomes time and memory consuming for problems with long time horizons.

This paper presents a new LR approach for job shop scheduling problems to minimize a weighted earliness and tardiness criterion. Our goal is to find good schedules for problems of large size in a short computation time. First, a separable formulation of the problems is presented in Section 2. The new approach is then introduced in Section 3. In the approach, operation precedence constraints rather than machine capacity constraints are relaxed. The relaxed problem is decomposed into single or parallel machine scheduling subproblems. These subproblems are NP-complete in general, and are approximately solved by using fast heuristic algorithms as presented in Section 4. The dual problem is iteratively solved by using a recently developed "surrogate subgradient (SSG) method" that allows approximate optimization of the subproblems [20]. Feasible schedules are then constructed by using heuristics based on subproblem solutions or multipliers. The solution of the dual problem and the construction of feasible schedules will be presented in Section 5.

Because the subproblem algorithms do not depend on the time horizon and are very fast, our new LR approach is efficient, particularly for problems with long time horizons. For these problems, the new approach requires much less memory and computation time as compared to the LR approach based on part decomposition. This is demonstrated by numerical testing presented in Section 6. Testing also shows that our new approach significantly outperforms dispatching rules and can solve problems with thousands of parts and tens of machine types in less than 15 minutes on a personal computer, making it practical for shop-floor use.

## 2. Problem formulation

The job shop scheduling problem considered in this paper is to schedule $N$ parts on $H$ types of machines to minimize a weighted earliness and tardiness criterion following [19]. Each machine type $h$ $(1 \leqslant h \leqslant H)$ has $M_h$ identical machines, and the completion of each part $i$ $(1 \leqslant i \leqslant N)$ requires a series of $N_i$ operations, denoted by $(i,1),(i,2),\ldots,(i,N_i)$. Each operation $(i,j)$ is non-preemptive and can be performed on a machine belonging to a set of alternative machine types $H_{ij} \subseteq \{1,2,\ldots,H\}$. For simplicity, all parts are assumed to be available at time 0. For the case where some parts are not available at time 0, the following formulation and solution methodology are still applicable after a slight modification. The following symbols will be used in the problem formulation:

| | |
|---|---|
| $B_i$ | beginning time of part $i$ |
| $c_{ij}$ | completion time of operation $(i,j)$ |
| $C_i$ | completion time of part $i$ |
| $D_i$ | due date of part $i$ |
| $E_i$ | earliness of part $i$, defined as $\max(0, D_i - C_i)$ |
| $m_{ij}$ | machine type selected to process operation $(i,j)$, $m_{ij} \in H_{ij}$ |
| $M_{h\tau}$ | number of available machines of type $h$ at time $\tau$, $1 \leqslant h \leqslant H$ |
| $O_h$ | set of operations that can be performed on machine type $h$ |
| $t_{ijh}$ | processing time of operation $(i,j)$ on machine type $h \in H_{ij}$ |

| | |
|---|---|
| $T_i$ | tardiness of part $i$, defined as $\max(0, C_i - D_i)$ |
| $w_i$ | tardiness weight for part $i$ |
| $\beta_i$ | earliness weight for part $i$ |
| $\delta_{ijh}$ | operation index over time for machine type $h$ with $\delta_{ijh}(\tau) = 1$ if operation $(i, j)$ is performed on a machine of type $h$ at time $\tau$, and $\delta_{ijh}(\tau) = 0$ otherwise. That is, $\delta_{ijh}(\tau) = 1$ for $(i, j, h, \tau)$ with $m_{ij} = h$ and $c_{ij} - t_{ijh} \leqslant \tau \leqslant c_{ij}$, and $\delta_{ijh}(\tau) = 0$ for others |

With the above symbols, an optimization model for the scheduling problem is presented below following [19]:

*Objective*. The goal of on-time delivery and low work-in-process inventory is modeled as a weighted tardiness and earliness cost, i.e.,

$$\min_{\{m_{ij}\},\{c_{ij}\}} J, \quad \text{with } J = \sum_i w_i T_i + \sum_i \beta_i E_i, \tag{1}$$

where $w_i$ and $\beta_i$ are tardiness and earliness weights, $T_i$ and $E_i$ are tardiness and earliness for part $i$, respectively.

Since on-time delivery is the foremost criterion in (1), $\beta_i$ is usually an order of magnitude smaller than $w_i$.

*Operation precedence constraints.* An operation cannot be started until its preceding operation is finished, and it requires a specific amount of time for processing on the selected machine type, i.e.,

$$c_{i,j-1} + t_{ijm_{ij}} \leqslant c_{i,j}, \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, N_i, \tag{2}$$

where $c_{i,j}$ is the completion time of operation $(i, j)$, $t_{ijm_{ij}}$ is the processing time of operation $(i, j)$ on machine type $m_{ij}$.

*Machine capacity constraints.* The number of operations being processed on a machine of type $h$ at any time instant $\tau$ cannot exceed $M_h$, the number of machines for the type, i.e.,

$$\sum_{ij} \delta_{ijh}(\tau) \leqslant M_{h\tau}, \quad 0 \leqslant \tau < \infty, \ h = 1, 2, \ldots, H, \tag{3}$$

where $\delta_{ijh}$ is operation index over time for machine type $h$, $M_{h\tau}$ is the number of available machines of type $h$ at time $\tau$.

The above formulation differs from that of [19] in its time concept, as continuous time rather than discrete time is used. Consequently, the left side of the inequality (2) does not contain a term $-1$.

The overall problem is to minimize the cost function (1) subject to the above constraints by selecting appropriate machine types and completion times for individual operations. Since the problem is NP-hard and no algorithm can optimally solve the problem of practical sizes in a reasonable computation time, a near-optimal approach based on Lagrangian relaxation will be developed in the following sections.

## 3. Solution framework

Our new LR approach is based on machine decomposition, and is carried out by relaxing operation precedence constraints. The reason for relaxing the precedence constraints is that the number of such constraints does not depend on the time horizon so that relaxation can lead to a time horizon-independent approach. However, if only precedence constraints are relaxed, the relaxed subproblems, which are parallel machine problems to minimize the weighted earliness and tardiness criterion, are difficult to solve. For this reason, an additional effort is made by reformulating the original problem and relaxing also the earliness and tardiness constraints. The resulting subproblems are then parallel machine problems to minimize the weighted completion time, where efficient approximate algorithms exist.

### 3.1. Problem reformulation

The definition of the tardiness for part $i$ implies that

$$C_i - D_i \leqslant T_i, \text{ and} \tag{4a}$$

$$0 \leqslant T_i, \quad i = 1, 2, \ldots, N. \tag{4b}$$

Similarly, the definition of the earliness for part $i$ implies that

$$D_i - C_i \leqslant E_i, \tag{5a}$$

$$0 \leqslant E_i, \quad i = 1, 2, \ldots, N. \tag{5b}$$

With the above derivation, the problem presented in the previous section can be reformulated as follows:

$$\boldsymbol{P}\text{: } \min_{\{m_{ij}\},\{c_{ij}\},\{T_i\},\{E_i\}} J, \quad \text{with } J = \sum_i w_i T_i + \sum_i \beta_i E_i,$$

subject to (2), (3), (4a), (4b), (5a) and (5b), where earliness $E_i$ and tardiness $T_i$ are also treated as additional decision variables.

These two problems are equivalent because for any optimal solution of $\boldsymbol{P}$, $T_i = \max(0, C_i - D_i)$ and $E_i = \max(0, D_i - C_i)$ for all $i$. If this is not true, there exists some $i$ such that $T_i > \max(0, C_i - D_i)$ or $E_i > \max(0, D_i - C_i)$. A better solution can then be obtained by taking $T_i = \max(0, C_i - D_i)$ and $E_i = \max(0, D_i - C_i)$.

### *3.2. Relaxation and decomposition*

By introducing multipliers $\{\mu_i\}$, $\{v_i\}$, and $\{\lambda_{ij}\}$ to relax constraints (4a), (5a), and (2), respectively, the relaxed problem of $\boldsymbol{P}$ is obtained as:

$$\boldsymbol{RP}\text{: } \min_{\{m_{ij}\},\{c_{ij}\},\{T_i\},\{E_i\}} L(\{\{\mu_i\},\{v_i\},\{\lambda_{ij}\}\},\{\{m_{ij}\},\{c_{ij}\},\{T_i\},\{E_i\}\}), \tag{6}$$

subject to (4b), (5b), (3), with

$$L \equiv J + \sum_i \mu_i (C_i - D_i - T_i) + \sum_i v_i (D_i - C_i - E_i) + \sum_{ij} \lambda_{ij} (c_{i,j-1} + t_{ijm_{ij}} - c_{i,j}).$$

Let $\{\{m^*_{ij}\},\{c^*_{ij}\},\{T^*_i\},\{E^*_i\}\}$ be an optimal solution of $\boldsymbol{RP}$ for a given set of multipliers. The dual problem of $\boldsymbol{RP}$ is to maximize the dual function $q$:

$$\boldsymbol{DP}\text{: } \max_{\{\mu_i,v_i,\lambda_{ij}\} \geqslant 0} q \tag{7}$$

with $q \equiv L(\{\{\mu_i\},\{v_i\},\{\lambda_{ij}\}\},\{\{m^*_{ij}\},\{c^*_{ij}\},\{T^*_i\},\{E^*_i\}\})$.

Function $L$ can be rewritten as:

$$L = \sum_i (w_i - \mu_i) T_i + \sum_i (\beta_i - v_i) E_i + \sum_i \mu_i C_i - \sum_i v_i C_i + \sum_{ij} (\lambda_{i,j+1} - \lambda_{ij}) c_{i,j} - \sum_i \mu_i D_i + \sum_i v_i D_i + \sum_{ij} \lambda_{ij} t_{ijm_{ij}}.$$

Since constraints (4b), (5b) and (3) are independent of each other, $\boldsymbol{RP}$ can be decomposed into tardiness subproblem $\boldsymbol{P}_{\mathrm{T}}$, earliness subproblem $\boldsymbol{P}_{\mathrm{E}}$, and the subproblem to determine $\{m_{ij}\}$ and $\{c_{ij}\}$.

$$\boldsymbol{P}_{\mathrm{T}}\text{: } \min_{T_i \geqslant 0} \sum_i (w_i - \mu_i) T_i,$$

$$\boldsymbol{P}_{\mathrm{E}}\text{: } \min_{E_i \geqslant 0} \sum_i (\beta_i - v_i) E_i, \text{ and}$$

$$\boldsymbol{P}_{\mathrm{MC}}\text{: } \min_{\{m_{ij}\},\{c_{ij}\}} \underline{L},$$

subject to (3), with

$$\underline{L} \equiv \sum_i (\mu_i - v_i) C_i + \sum_{ij} (\lambda_{i,j+1} - \lambda_{ij}) c_{i,j} - \sum_i (\mu_i - v_i) D_i + \sum_{ij} \lambda_{ij} t_{ijm_{ij}}.$$

Tardiness subproblem $\boldsymbol{P}_{\mathrm{T}}$ has an optimal solution: $T_i = 0$ if $w_i - \mu_i \geqslant 0$, and $T_i = +\infty$ if $w_i - \mu_i < 0$. Earliness subproblem $\boldsymbol{P}_{\mathrm{E}}$ has an optimal solution: $E_i = 0$ if $\beta_i - v_i \geqslant 0$, and $E_i = +\infty$ if $\beta_i - v_i < 0$.

If each operation has only one eligible machine type to process, $m_{ij}$ will no longer be a decision variable and $t_{ijm_{ij}}$ will be a constant. In this case, subproblem $\boldsymbol{P}_{\mathrm{MC}}$ can be further decomposed into a set of subproblems, one for each machine type, by regrouping the terms in $\underline{L}$ according to machine types:

$$\begin{aligned}\underline{L} &= \sum_h \left\{ \sum_{(i,N_i)\in O_h} (\mu_i - v_i) C_i + \sum_{(i,j)\in O_h} (\lambda_{i,j+1} - \lambda_{ij}) c_{ij} + \sum_{(i,j)\in O_h} \lambda_{ij} t_{ijm_{ij}} \right\} - \sum_i (\mu_i - v_i) D_i \\ &= \sum_h \left\{ \sum_{(i,j)\in O_h} \tilde{w}_{ij} c_{ij} + \sum_{(i,j)\in O_h} \lambda_{ij} t_{ijm_{ij}} \right\} - \sum_i (\mu_i - v_i) D_i \\ &= \sum_h L_h - \sum_i (\mu_i - v_i) D_i,\end{aligned} \tag{8}$$

where

$$L_h \equiv \sum_{(i,j)\in O_h} \tilde{w}_{ij} c_{ij} + \sum_{(i,j)\in O_h} \lambda_{ij} t_{ijm_{ij}},$$

$$\tilde{w}_{ij} \equiv \begin{cases} \mu_i - \nu_i - \lambda_{ij}, & j = N_i, \\ \lambda_{i,j+1} - \lambda_{ij}, & 1 \leqslant j < N_i. \end{cases} \tag{9}$$

Since each $L_h$ in (9) depends only on decision variables related to machine type $h$, and $\sum_i (\mu_i - \nu_i) D_i$ is a constant for a given set of multipliers $\{\{\mu_i\}, \{\nu_i\}, \{\lambda_{ij}\}\}$, problem $\boldsymbol{P}_{\text{MC}}$ can be decomposed into a set of subproblems $P_h$, one for each machine type $h$, as follows:

$$P_h : \min L_h,$$
$$\text{subject to} \sum_{ij} \delta_{ijh}(\tau) \leqslant M_{h\tau}, \quad 0 \leqslant \tau < \infty. \tag{10}$$

Generally, some operations may have more than one eligible machine types to process. In this case, to decompose subproblem $P_{\text{MC}}$ into a set of subproblems, machine types are grouped in a way that any two machine types $h_1$ and $h_2$ with $O_{h_1} \cap O_{h_2} \neq \emptyset$ are clustered into the same group, where $O_{h_i}$ is the set of operations that can be performed on machine type $h_i$ as defined in Section 2. Subproblem $P_{\text{MC}}$ can then be decomposed into a set of subproblems, each for a group of machine types.

If a group of machine types contains only one machine type having only one machine, its corresponding subproblem, referred to as the "machine subproblem", is a single machine scheduling problem to minimize the weighted completion time of operations. If the group contains only one machine type but having multiple machines, its corresponding subproblem, referred to as the "machine type subproblem", is an identical parallel machine scheduling problem [12] to minimize the same criterion. Otherwise, the group contains more than one machine types. In this case, its corresponding subproblem, referred to as the "machine group subproblem", is an unrelated parallel machine scheduling problem [12] to minimize the sum of the weighted completion time and the weighted processing time of operations. The subproblem can be formulated as:

$$P_G : \min L_G \equiv \sum_{(i,j) \in O_G} \tilde{w}_{ij} c_{ij} + \sum_{(i,j) \in O_G} \lambda_{ij} t_{ijm_{ij}},$$

subject to

$$\sum_{ij} \delta_{ijh}(\tau) \leqslant M_{h\tau}, \quad 0 \leqslant \tau < \infty, \ h \in G, \tag{11}$$

where $G$ is a set of machine types, $O_G = \cup_{h \in G} O_h$ is the set of operations that can be performed on a machine type belonging to $G$. The formulations of machine and machine type subproblems can be viewed as two specific cases of this formulation.

For classic job shop scheduling, all subproblems of $P_{\text{MC}}$ are machine subproblems. Generally, however, the subproblems may contain machine, machine type, and machine group subproblems. The solution of these subproblems will be presented in the next section.

### 3.3. Decomposition and coordination structure

As most existing LR approaches, our new approach adopts a two-level decomposition and coordination structure as illustrated in Fig. 1. At the

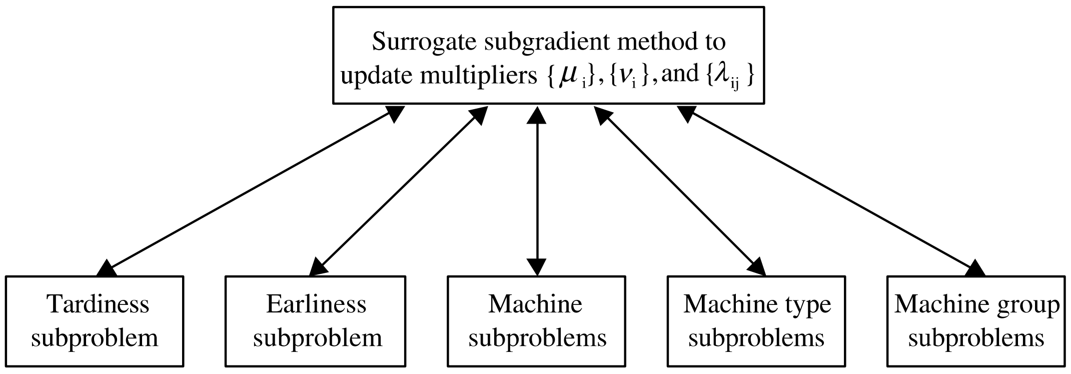
Surrogate subgradient method to update multipliers $\{\mu_i\}, \{\nu_i\}$, and $\{\lambda_{ij}\}$
Tardiness subproblem
Earliness subproblem
Machine subproblems
Machine type subproblems
Machine group subproblems

Fig. 1. The two level decomposition and coordination structure.

low level, all subproblems are solved for a given set of multipliers, and the set of multipliers are iteratively adjusted at the high level based on the degree of constraint violation. Simple heuristics are then applied to adjust subproblem solutions to obtain feasible schedules when such an iterative process terminates.

The differences between most existing LR approaches and the new approach are that in our approach, the subproblems are approximately solved by heuristic algorithms, and the recently developed SSG method [20] is used to update Lagrange multipliers at the high level. The resolution of the subproblems and the dual problem and the construction of feasible schedules will be presented in the following sections.

## 4. Subproblem solution

### 4.1. Computational complexity of the subproblems

For a given scheduling problem, the subproblems of $P_{\mathrm{MC}}$ may contain machine subproblems, machine type subproblems, and machine group subproblems. These subproblems have different computational complexities. For a machine subproblem, if the machine is available throughout the scheduling horizon, it is a single machine scheduling problem to minimize the weighted completion time criterion and is polynomially solvable [12]. Otherwise, it is NP-complete. Machine type and machine group subproblems, however, are strongly NP-complete even if the number of available machines of each machine type is constant, because identical or unrelated parallel machine problems to minimize the weighted completion time criterion are NP-complete [12].

The NP-complete complexity of machine type subproblems and machine group subproblems implies that optimal algorithms may be not appropriate for these subproblems, at least for these with practical sizes [5]. Instead, heuristic algorithms will be developed in the next subsection. The heuristic resolution of the subproblems is justified by a recently developed SSG method for the optimization of Lagrangian dual function [20], as SSG allows approximate optimization of subproblems under a convergence condition (see the next section for details). The disadvantage of solving subproblems approximately is that the lower bound property of LR is lost, and this is at an exchange of a great reduction in the computation time of our new LR approach.

### 4.2. Solving the subproblems

For simplicity, all machines are assumed to be available throughout the scheduling horizon in the following description of the heuristic algorithms. For subproblems where some machines are not available during certain time intervals, similar approximate algorithms that take account of machine availability constraints exist after appropriate modifications [18]. Also for simplicity of statement, two dimensional operation index (*i, j*) is replaced by one dimensional index $i$ after an appropriate index transformation. With the new operation index, the criterion of a subproblem can be written as $J \equiv \sum_i \tilde{w}_i c_i + \sum_i \lambda_i t_{im_i}$, where $\tilde{w}_i$, $m_i, c_i$ and $t_{im_i}$ correspond to $\tilde{w}_{ij}$, $m_{ij}, c_{ij}$ and $t_{im_{ij}}$, respectively.

In the following, three algorithms will be presented for machine, machine type, and machine group subproblems, respectively.

**Algorithm 1** (*Algorithm for machine subproblems*). Each machine subproblem is optimally solved by using the Smith's weighted shortest processing time (WSPT) rule [12]. The rule sequences all operations in a non-decreasing order of the ratio $t_i/\tilde{w}_i$, where $t_i$ and $\tilde{w}_i$ are the processing time and the weight of an operation $i$, respectively. The computational complexity of the rule-based algorithm is $\mathrm{O}(n\log(n))$, where $n$ is the number of operations.

**Algorithm 2** (*Algorithm for machine type subproblems*). Each machine type subproblem is approximately solved by using a parameter list scheduling heuristic [12]. In the heuristic, all operations are ordered according to a non-decreasing order of the ratio $t_i/\tilde{w}_i$ as in the Smith's WSPT rule for machine subproblems. The heuristic assigns the next operation to the machine that first becomes available. In most cases, the WSPT-based heuristic provides a near-optimal schedule for the machine type subproblem. The computational complexity of the heuristic is $O(mn\log(n))$, where $m$ is the number of identical machines and $n$ is the number of operations.

**Algorithm 3** (*Algorithm for machine group subproblems*). A machine group subproblem is approximately solved by using a similar parameter list scheduling heuristic. For each machine type $h$, all operations that can be processed by it are arranged in a non-decreasing order of the ratio $t_{ih}/\tilde{w}_i$. This forms $H$ operation lists, one for each machine type. These lists are indexed by machine type, where some operations may appear in more than one list. With these lists, the heuristic calculates the cost $\tilde{w}_i c_i + \lambda_i t_{ih}$ for each list $h$ and its first operation $i$ and selects an operation and a list (a machine type) with the minimum cost as the operation to be dispatched next and the machine type that the operation will be assigned to. In the calculation, the completion time $c_i$ is obtained assuming that operation $i$ will be assigned to the earliest available machine of type $h$ next. The selected operation is then assigned to the earliest available machine of the selected machine type. After the operation is assigned, its duplications in other lists are removed. This process repeats until all operations are assigned. The computational complexity of the heuristic is $O(mn\log(n))$, where $m$ is the total number of machines and $n$ is the total number of operations.

In few cases, the solution of the relaxed problem obtained by using the above algorithms does not satisfy the convergence condition for SSG at some iterations. In this case, Algorithm 3 is enhanced by a local search procedure to improve the solution quality. The local search procedure reallocates an operation to another machine type at each step until the condition holds or further improvement of a solution is impossible.

## 5. Dual problem and the heuristics

In this section, the dual problem is solved by using our recently developed SSG method [20], and feasible schedules are constructed using heuristics based on subproblem solutions or the multipliers. Before introducing the method, a property of the dual problem is explored to restrict the solution space of the problem.

### 5.1. Property of the dual problem

For an optimal dual solution, all coefficients in the objective functions of $P_T, P_E$, and $P_{MC}$ $(1 \leqslant h \leqslant H)$, i.e., $w_i - \mu_i, \beta_i - \nu_i$, and $\tilde{w}_{ij}$ should be non-negative for any $i$ and $j$. Otherwise, by taking $T_i = +\infty, E_i = +\infty$, or $c_{ij} = +\infty$, we have $L = -\infty$. Since the dual problem is a maximization problem with a bounded optimal objective value, such multipliers will not be an optimal solution. Consequently we have:

$$\mu_i \leqslant w_i, \nu_i \leqslant \beta_i, \tag{12a}$$

$$\lambda_{i,j+1} \geqslant \lambda_{ij}, \quad \text{if } 1 \leqslant j < N_i, \ i = 1,2,\ldots,N, \tag{12b}$$

$$\mu_i \geqslant \nu_i + \lambda_{iN_i}. \tag{12c}$$

Let $\Omega$ be the set of $\{\{\mu_i\},\{\nu_i\},\{\lambda_{ij}\}\}$'s satisfying constraints (12a)–(12c). We thus have the following property:

**Property 1.** *The optimal solution of DP is attained at a point in set $\Omega$.*

Intuitively, the multiplier $\lambda_{ij}$ can be interpreted as the price (marginal cost) for one time unit later completion of operation $(i, j-1)$ or the price for one time unit earlier completion of operation $(i, j)$ for part $i$. The inequality $\lambda_{i,j+1} \geqslant \lambda_{ij}$ implies that the late completion of operation $(i, j+1)$ is more costly (crucial) than the late completion of its preceding operation $(i, j)$. This is because the time slack for the completion of the part before its due date becomes smaller and smaller as time advances. The constraint (12c) is a variant of the constraint (12b) when the first and the last operations of a part are considered. The multiplier $\mu_i$

can be interpreted as the price for one time unit later completion of part $i$, while the weight $w_i$ is the price for one time unit later delivery of the part. The inequality $\mu_i \leqslant w_i$ implies that the late completion of the part is less costly (less crucial) than or as costly (crucial) as the late delivery of the part. This is because one unit later completion of a part does not imply one unit later delivery of the part if the part is completed before its due date. A similar explanation can be given to $v_i \leqslant \beta_i$.

### 5.2. Surrogate subgradient method

Lagrangian dual problems for separable integer programming problems are commonly solved by using the subgradient method, which requires *optimally* solving *all* subproblems at each iteration to obtain a subgradient direction. This may be time consuming for ones with many subproblems or some hard subproblems. Recently, the surrogate subgradient method has been developed to overcome the difficulty [20]. In the method, a proper direction can be obtained without optimally solving all the subproblems. In fact, only *approximate* optimization of one or several subproblems is needed to get a proper "SSG direction." The convergence of the method is proved. Because this method can obtain good directions with much less effort, it is powerful for problems of large size. In the following, a brief introduction of the method is given.

Consider a separable integer programming problem described as

$$(\mathrm{IP}) \quad \min_x J_{\mathrm{IP}} = \sum_{i=1}^{I} J_i(x_i), \tag{13}$$

$$\text{subject to} \quad Ax \leqslant b \quad \text{and} \quad x_i \in Z^{n_i}, \quad i = 1, \dots, I, \tag{14}$$

where $x = (x_1, x_2, \dots, x_n)^{\mathrm{T}}$ is an $n \times 1$ decision variable with $n = \sum_{i=1}^{I} n_i$ and $Z$ is the set of integers.

The LR of IP is given by

$$L(\lambda) \equiv \min_{x \in Z^n} \left[ \sum_{i=1}^{I} J_i(x_i) + \lambda^{\mathrm{T}}(Ax - b) \right], \tag{15}$$

and the Lagrangian dual problem is

$$(\mathrm{LD}) : \max_{\lambda \geqslant 0} L(\lambda), \tag{16}$$

where $\lambda$ is a vector of Lagrange multipliers.

As an extension of the Lagrangian dual, a surrogate dual is introduced:

$$\tilde{L}(\lambda, x) \equiv \left[ \sum_{i=1}^{I} J_i(x_i) + \lambda^{\mathrm{T}}(Ax - b) \right], \quad x \in Z^n, \tag{17}$$

and its corresponding SSG is defined as

$$\tilde{g}(x) \equiv Ax - b. \tag{18}$$

*SSG method*

*Step 0* (Initialize). Initialize $\lambda^0$ and minimize all subproblems to obtain $x^0$ i.e.,

$$x^0 = \arg\min_{x \in Z^n} \left[ \sum_{i=1}^{I} J_i(x_i) + (\lambda^0)^{\mathrm{T}}(Ax - b) \right]. \tag{19}$$

*Step 1* (Update multipliers). Given the current point $(\lambda^k, x^k)$ at the $k$th iteration, the Lagrange multipliers are updated according to

$$\lambda^{k+1} = \lambda^k + s^k \tilde{g}^k, \tag{20}$$

where $\tilde{g}^k$ is the SSG given by

$$\tilde{g}^k = \tilde{g}(x^k) = Ax^k - b, \tag{21}$$

with stepsize $s^k$ satisfying

$$0 < s^k < \left(L^* - \tilde{L}^k\right) \Big/ \left\|\tilde{g}^k\right\|^2. \tag{22}$$

Here $L^* = L(\lambda^*)$ is the optimal objective of dual problem *LD*, $\tilde{L}^k = \tilde{L}^k(\lambda^k, x^k)$ is the surrogate dual at the $k$th iteration.

*Step 2* (Perform approximate optimization). Given $\lambda^{k+1}$ perform "approximate optimization" to obtain $x^{k+1}$ such that $x^{k+1}$ satisfies

$$\tilde{L}(\lambda^{k+1}, x^{k+1}) < \tilde{L}(\lambda^{k+1}, x^k). \tag{23}$$

If such an $x^{k+1}$ cannot be obtained, set $x^{k+1} = x^k$.

*Step 3* (Checking stopping criteria). If the criteria given by

$$\|\lambda^{k+1} - \lambda^k\| < \varepsilon_1 \quad \text{and} \tag{24a}$$

$$\|x^{k+1} - x^k\| < \varepsilon_2 \tag{24b}$$

are met, then stop. Otherwise go to Step 1. The stopping criteria can also be based on CPU time or the number of iterations.

It has been proved that in the SSG method, the multipliers move closer step by step to optimal solution $\lambda^*$ of the dual problem, and if $\lambda^{k+1} = \lambda^k$, $x^{k+1} = x^k$, $(\lambda^k, x^k)$ is the optimal solution of the dual problem.

### 5.3. Solving the dual problem

Since subproblems $P_h$ (or $P_G$) in our new LR approach are only approximately solved, we use the SSG method to optimize the objective function of dual problem DP. In our implementation of the method, step size $s^k$ is taken as

$$s^k = \beta\left(L^* - \widetilde{L}^k\right) \Big/ \left\|\tilde{g}^k\right\|^2, \tag{25}$$

where $\beta$ is a parameter with $0 < \beta < 1$.

In the above formula, optimal dual $L^*$ is estimated by $L^U = (1 + \omega/\theta^\rho) \times \widetilde{L}^{[k]}$, where $\omega$, $\theta$, and $\rho$ are three parameters, $\widetilde{L}^{[k]}$ is the best surrogate dual obtained prior to iteration $k$. Parameters $\omega$ and $\rho$ are chosen within [0.1, 1.0] and [1.1, 1.5], respectively, and parameter $\theta$ is adaptively adjusted with $\theta = \max(1, \theta - 1)$ if $\widetilde{L}^k > \widetilde{L}^{[k]}$, and $\theta = \theta + 1$ otherwise.

Since an optimal solution of the dual problem is attained in set $\Omega$, the search of the solution is restricted to the set by projecting the SSG direction onto $\Omega$ at each iteration. The iterative process is terminated after a given number of iterations have been executed or a given computation time has been used up.

### 5.4. Construction of feasible schedules

Because of the discrete decision variables involved, the solutions to subproblems are generally associated with an infeasible schedule, i.e., some of the precedence constraints might be violated. One way to construct a feasible schedule is to use a list scheduling method similar to that presented in [10]. A list is created for each machine by ordering the operations assigned to the machine in the non-decreasing order of their completion times determined by the subproblem solutions. Operations are then scheduled on the assigned machines according to the lists as soon as all their preceding operations have been scheduled. This heuristic method, however, may result in deadlocks. For instance, if the first operation of Part 1 precedes the first operation of Part 2 on Machine 1 while the second operation of Part 2 precedes the second operation of Part 1 on Machine 2 in subproblem solutions, a deadlock will occur. Thus, more comprehensive heuristics have to be developed for the construction of feasible schedules.

The first heuristic developed is based on the orders of operations given by subproblem solutions. The orders are used to construct a directed graph, where each node represents an operation, and a directed arc represents either an operation precedence relation specified by an operation precedence constraint or by an order between two operations given by subproblem solutions. The graph may contain a loop (directed circuit). The heuristic dispatches operations from source nodes of the graph to sink nodes. An operation can be dispatched if all its preceding operations have been dispatched. If the graph has no loop, all operations can be successively dispatched in this way, leading to a feasible schedule. Otherwise, a loop will be detected. In this case, a repair policy that changes the order of two operations in the graph is invoked to break the loop, and the dispatching process is resumed after the repair.

The second heuristic is based on the multipliers given by the dual solution. The heuristic dispatches operations according to a priority defined by the cost $\tilde{w}_{ij}c_{ij} + \lambda_{ij}t_{ijm_{ij}}$ that depends on the multipliers, where $c_{ij}$ is calculated assuming that operation $(i, j)$ will be assigned to the earliest available machine of type $m_{ij}$ next. At each step, the heuristic calculates the cost for all dispatchable operations and their eligible machine types, and selects a dispatchable operation and its eligible machine type with the minimum cost as the operation to be dispatched next and the machine type that the operation is assigned to, respectively. This process continues until all operations are dispatched.

## 6. Testing results

Our new approach has been implemented in C++ on a PC with 450 MHz CPU. Numerical testing has been performed to compare the approach with the LR approach based on part decomposition [20] and dispatching rules. For the new approach, both of the heuristics presented in Section 5.3 for the construction of feasible solutions were tested, while the best feasible cost obtained by the heuristics is reported. Examples 1–3 are designed to compare our new approach with the part decomposition approach with problem sizes varying from small, medium to large. Example 4 is designed to compare our new approach with dispatching rules. For simplicity, the new approach relaxing precedence constraints is referred to as $\text{LR}_{\text{prec}}$ and the part decomposition-based approach relaxing machine capacity constraints as $\text{LR}_{\text{mach}}$.

**Example 1.** This example was designed to compare the performance of $\text{LR}_{\text{prec}}$ for three different types of job shops and to evaluate the impact of the tightness of the due dates on the performance. The three types are classical job shops, job shops with identical machines, and flexible job shops where some operations may be processed by more than one machine type.

Nine problems were constructed for this example from the benchmark ten job-ten machine problem provided in Fisher and Thompson [9]. Problems C1–C3 are *classical* job shops, which are the same as the benchmark problem except that weighted tardiness rather than makespan is to be minimized. The due date of part $i$ ($i = 1, 2, \dots, 10$) is generated according to $\alpha \sum_{i=1}^{10} t_{ij}$, where $\sum_{i=1}^{10} t_{ij}$ is the total processing time of the part, and $\alpha$ is the due date tightness factor taken as 1.0, 1.5, 2.0 for C1, C2, and C3, respectively. The tardiness weights of all parts are set to one.

Problems I1–I3 are job shops with *identical* machines constructed from the benchmark problem by replicating machines. Each machine is replicated 1–3 times. For these machines, the numbers of replication are 1, 2, 1, 3, 2, 2, 2, 3, 3, 3, respectively. The due dates and the tardiness weights for I$i$ are the same as those for C$i$, $i = 1, 2, 3$.

Problems F1–F3 are *flexible* job shops constructed from the benchmark problem too. For each operation, the number of alternative machines is randomly generated between 1 and 3. Each alternative machine is randomly generated between 1 and 10. The processing time of the operation on an alternative machine is set to the processing time of the operation in [9] multiplied by a rate randomly generated according to a uniform distribution $U[0.5, 2.0]$. The due dates and the tardiness weights for F$i$ are the same as those for C$i$, $i = 1, 2, 3$.

For a fair comparison between the two LR approaches, the same number of iterations is used as a stopping criterion. In this example, the number is taken as 100. Testing results are shown in Table 1, where the CPU time is the computation time of the approaches and the feasible cost is the cost of the best feasible schedule obtained. Since

Table 1
Testing results for $10 \times 10$ problems

| Problem no. | CPU time (seconds) | | Multiplier number | | Feasible cost | |
|---|---|---|---|---|---|---|
| | $\text{LR}_{\text{prec}}$ | $\text{LR}_{\text{mach}}$ | $\text{LR}_{\text{prec}}$ | $\text{LR}_{\text{mach}}$ | $\text{LR}_{\text{prec}}$ | $\text{LR}_{\text{mach}}$ |
| C1 | 1 | 7 | 110 | 11820 | 3291 | 3006 |
| C2 | 1 | 9 | 110 | 13510 | 835 | 800 |
| C3 | 1 | 11 | 110 | 14430 | 166 | 658 |
| I1 | 1 | 7 | 110 | 9770 | 1511 | 1215 |
| I2 | 1 | 8 | 110 | 10830 | 70 | 65 |
| I3 | 1 | 11 | 110 | 14430 | 0 | 45 |
| F1 | 1 | 12 | 110 | 11860 | 2045 | 1907 |
| F2 | 1 | 11 | 110 | 11700 | 423 | 671 |
| F3 | 1 | 16 | 110 | 14430 | 0 | 165 |

the memory requirement of the approaches is dominated by the number of multipliers especially for problems with long time horizons, it is reflected on multiplier number in the table.

From Table 1, we can see that the performance of $\text{LR}_{\text{prec}}$ for the three different types of problems are similar. Compared to $\text{LR}_{\text{mach}}$, the new approach needs much less computation time and memory for all cases. The performance of $\text{LR}_{\text{prec}}$ versus $\text{LR}_{\text{mach}}$ in terms of feasible cost, however, depends on how the due dates are set. When the dates are tight, $\text{LR}_{\text{mach}}$ performs better than $\text{LR}_{\text{prec}}$ The new approach, however, becomes better than $\text{LR}_{\text{mach}}$ as the due dates increase. One possible explanation for this is that $\text{LR}_{\text{mach}}$ is due date driven, it performs quite good when the due dates are tight, while the relaxation of the tardiness constraints (4a) deteriorates the performance of $\text{LR}_{\text{prec}}$ in this case. For this example, the first heuristic outperforms the second heuristic.

It should be noted that both of the LR approaches cannot be compared with well designed algorithms such as the shifting bottleneck procedure for classical job shop scheduling problems if we do not take account of the computation time. However, our goal is to find good schedules for problems of large size in a short computation time rather than to find an optimal schedule in several hours. In this case, our new LR approach has its own advantages, particularly on its computational efficiency and its flexibility to deal with a wider range of job shop scheduling problems with routing flexibility.

**Example 2.** This example was designed to compare the performance of the approaches for medium-sized problems. Thirty flexible job shop problems with 10 machine types and 100 parts were randomly generated for the example. For each problem, the number of machines per machine type is randomly generated from 1 to 3. Each part has 10 operations, with the number of alternative machine types randomly generated between 1 and 3. These alternative machine types are randomly generated between 1 and 10. The processing time of each operation $(i,j)$ on an alternative machine type is set to a nominal processing time multiplied by a rate. The nominal processing time and the rate are randomly generated according to uniform distributions $U[1,10]$ and $U[0.5,2.0]$, respectively. The due date $D_i$ is generated according to $D_i = \alpha \sum_{i=1}^{10} t_{ij}$, where $\alpha$ is the due date tightness factor. Because more parts are processed by a similar number of machines in the shops of this example, the tightness factors taken are larger than those in Example 1. Three sets of problems with tightness factors 4, 6, 8, respectively, are generated, with 10 problems for each set. For all problems, the tardiness weights of all parts are set to 1 and the earliness weights set to 0.1.

As in Example 1, both of the LR approaches are terminated after 100 iterations. Testing results are shown in Table 2, where F1, F2, and F3 denote the problem sets with $\alpha = 4$, 6, and 8, respectively. In this table, the CPU time is the mean computation time of the approaches for each set of problems, and $\text{RD}_J = (J_{\text{prec}} - J_{\text{mach}})/J_{\text{mach}}$ is the mean relative difference of the feasible costs for each set at 100 iterations.

From Table 2, we can see that the performance of the new approach for medium-sized problems is similar to that for small problems in Example 1. For this example, the second heuristic outperforms the first heuristic.

Table 2
Testing results for $100 \times 10$ problems

| Problem set | CPU time (seconds) | | Multiplier number | | $\text{RD}_J$ |
|---|---|---|---|---|---|
| | $\text{LR}_{\text{prec}}$ | $\text{LR}_{\text{mach}}$ | $\text{LR}_{\text{prec}}$ | $\text{LR}_{\text{mach}}$ | |
| F1 | 8 | 101 | 1100 | 7470 | 0.189 |
| F2 | 7 | 104 | 1100 | 8438 | −0.021 |
| F3 | 6.5 | 108 | 1100 | 8926 | −0.253 |

$\text{RD}_J$: mean relative difference of the feasible costs at 100 iterations.

Table 3
Testing results for $1000 \times 10$ problems

| Problem set | CPU time (minutes) | | Multiplier number | | $RD_J$ |
|---|---|---|---|---|---|
| | $LR_{prec}$ | $LR_{mach}$ | $LR_{prec}$ | $LR_{mach}$ | |
| F1 | 14.6 | 501 | 11000 | 658440 | 0.198 |
| F2 | 13.3 | 506 | 11000 | 658670 | −0.032 |
| F3 | 13.0 | 525 | 11000 | 659000 | −0.264 |

$RD_J$: mean relative difference of the feasible costs at 100 iterations.

**Example 3.** This example was designed to compare the performance of the approaches for large problems and to evaluate the impact of the problem size on performance. As in Example 2, three sets of flexible job shop problems with ten problems for each were randomly generated. The generation of the problems is similar to that in Example 2 except that the number of parts becomes 1000 and the due date tightness factors $\alpha$ for the three sets are taken as 20, 30, and 40, respectively.

As in Examples 1 and 2, the two LR approaches were tested with the iteration number as their stopping criterion. Testing results for 100 iterations are shown in Table 3, where F1, F2, and F3 denote problem sets with $\alpha = 20$, 30, and 40, respectively.

Since computation time is critical for practical scheduling, the two approaches were also tested for the three sets of problems with a limited computation time of 15 minutes. The mean relative difference $RD_J$ of the approaches for problem sets F1, F2, and F3 in this testing is −0.153, −0.302, −0.589, respectively.

From the above results, we can see that for large problems the new approach has a similar performance as for medium-sized problems in terms of feasible costs. As the time horizon increases caused by the increase of part number in this example, the reductions of the computation time and memory of the new approach versus $LR_{mach}$ become more significant. The reason for this is that the computation time and memory of the new approach do not depend on the time horizon. It is also seen that with 15 minutes of limited computation time, $LR_{prec}$ outperforms $LR_{mach}$ for all problem sets. For this example, the second heuristic outperforms the first heuristic.

The above results on the performance of the heuristics show that the second one becomes better as the size of problems increases, while the first one performs well for small problems. One possible explanation for this is that in the first heuristic, the completion time of an operation will be delayed if the precedence constraint with its preceding operation is violated. This effect will accumulate as more and more operations are dispatched and the accumulation will make the order of any two late dispatched operations given by the subproblem solutions useless if the number of operations is large. By contrast, the information of the multipliers used in the second heuristic is less sensitive to the number.

**Example 4.** This example was designed to further evaluate the performance of our new approach by comparing it with dispatching rules [2,16]. Six dispatching rules obtained by combining two routing rules with three sequencing rules are considered. The routing rules are SQ (select a machine type with the *shortest queue*) and LW (select a machine type with the *least work* in queue). The sequencing rules are SPT (select an operation with the shortest processing time), EDD (select an operation of a part with the earliest due date), and CR (select an operation of a part with the least ratio of the part slack time to the remaining processing time) rules [2]. Two sets of 30 randomly generated problems as in Examples 2 and 3 were tested.

Testing results are shown in Table 4, where $RD_J = (J_{\text{dispatch}} - J_{\text{LRprec}})/J_{\text{LRprec}}$ is the mean relative difference of the feasible costs of a dispatching rule and the new LR approach for each problem set. From the table, we can see that our new approach significantly outperforms all dispatching

Table 4
Testing results compared with dispatching rules

| Mean relative difference $RD_J$ | | Dispatching rules | | | | | |
|---|---|---|---|---|---|---|---|
| | | SQ/STT | SQ/EDD | SQ/CR | LW/SPT | LW/EDD | LW/CR |
| Problem set | $100 \times 10$ | 0.2892 | 0.1921 | 0.2733 | 0.2955 | 0.1934 | 0.2598 |
| | $1000 \times 10$ | 0.2810 | 0.2135 | 0.2649 | 0.2877 | 0.2126 | 0.2566 |

rules on the feasible cost for both problem sets. For the computation time, it is not surprise that dispatching rules are much faster than our new LR approach. They need only 0.1 second for 100 part problems and 1 second for 1000 part problems on average. By contrast, our new approach needs 7.2 seconds for 100 part problems and 13.6 minutes for 1000 part problems on average. However, it is worthy to spend more but reasonable computation time to obtain a significantly better schedule by using the new approach.

In summary, our new LR approach is much more effective than the LR approach based on part decomposition in terms of the computation time and memory. For large problems with long time horizons, the new approach outperforms the part decomposition approach on feasible cost with a limited computation time. Numerical testing also shows that our new approach significantly outperforms dispatching rules. These features render the new approach practical for shop-floor use, particularly in shops where a large number of parts and a long planning horizon are involved.

## 7. Conclusions

LR is frequently used for separable optimization. The efficiency of the approach, however, depends on how a problem is relaxed and decomposed, and how the relaxed subproblems and the dual problem are solved. In this paper, a new LR approach is developed for job shop scheduling based on machine decomposition through relaxing operation precedence constraints. By synergistically combining a recently developed SSG method for the dual problem with fast approximate algorithms for the subproblems, our new approach is efficient for large problems with long time horizons. For these problems, the new approach requires much less memory and computation time as compared to the LR approach based on part decomposition. Numerical testing shows that our new approach significantly outperforms dispatching rules and can solve problems with tens of machine types and thousands of parts in less than 15 minutes on a personal computer. All of these results demonstrate that an appropriate choice of the relaxation framework can lead to significant reductions in computation time and memory requirements for a LR approach, and that a good relaxation is a tradeoff between the number of the constraints relaxed and the complexity of subproblems.

## Acknowledgement

This work was supported in part by the National Science Foundation grant DMI-9813176.

## References

[1] J. Adams, E. Balas, D. Zawack, The shifting bottleneck procedure for job shop scheduling, Management Science 34 (3) (1988) 391–401.
[2] J.H. Blackstone, D.T. Phillips, G.L. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations, International Journal of Production Research 20 (1982) 27–45.
[3] J. Carlier, E. Pinson, An algorithm for solving the job-shop problem, Management Science 35 (2) (1989) 164–176.
[4] H. Chen, C. Chu, J.M. Proth, An improvement of the lagrangian relaxation approach for job shop scheduling: A dynamic programming method, IEEE Transaction on Robotics and Automation 14 (5) (1998) 786–795.
[5] Z.-L. Chen, W.B. Powell, Solving parallel machine scheduling problems by column, INFORMS Journal on Computing 11 (1999) 78–94.
[6] M. Dell'Amico, M. Trubian, Applying Tabu Search to the job-shop scheduling problem, Annals of Operations Research 41 (1993) 231–252.

[7] E. Falkenuer, S. Bouffouix, A genetic algorithm for job shop, in: International Conference on Robotics and Automation, CA, 1991, pp. 824–829.

[8] H.-L. Fang, P. Ross, D. Corne, A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems, in: Proceedings of the Fifth International Conference of Genetic Algorithm and Their Applications, 1993, pp. 375–382.

[9] H. Fisher, G.L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules, in: J.F. Muth, G.L. Thompson (Eds.), Industrial Scheduling, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[10] D.J. Hoitomt, P.B. Luh, K.R. Pattipati, A practical approach to job shop scheduling problems, IEEE Transactions on Robotics and Automation 9 (1) (1993) 1–13.

[11] P.J.M. Van Laarhoven, E.H.L. Aarts, J.K. Lenstra, Job shop scheduling by simulated annealing, Operations Research 40 (1) (1992) 113–125.

[12] E.L. Lawler et al., Sequencing and scheduling: Algorithms and complexity, in: S.C. Graves (Ed.), Operations Research and Management Science, Logistics of Production and Inventory North-Holland, vol. 4, 1993, pp. 445–522.

[13] K. Mesghouni, S. Hammadi, P. Borne, Evolution programs for job-shop scheduling, in: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, FL, vol. 1, 1997, pp. 720–724.

[14] E. Nowicki, C. Smutnicki, A fast taboo search algorithm for the job shop problem, Management Science 42 (1996) 797–813.

[15] W.P.M. Nuitjen, E.H.L. Aarts, A computational study of constraint satisfaction for multiple capacitated job shop scheduling, European Journal of Operational Research 90 (1996) 269–284.

[16] S.S. Panwalkar, W. Iskander, A survey of scheduling rules, Operations Research 25 (1977) 45–61.

[17] C.N. Potts, L.N. Van Wassonhove, Dynamic programming and decomposition approaches for the single machine total tardiness problem, European Journal of Operational Research 32 (1987) 405–414.

[18] G. Schmidt, Scheduling with limited machine availability, European Journal of Operational Research 121 (2000) 1–15.

[19] J. Wang, P.B. Luh, X. Zhao, J. Wang, An optimization-based algorithm for job shop scheduling, *SADHANA*, in: Journal of Indian Academy of Sciences, Special Issue on Competitive Manufacturing Systems 22 (1997) 241–256.

[20] X. Zhao, P.B. Luh, J. Wang, The surrogate gradient algorithm for Lagrangian relaxation method, Journal of Optimization Theory and Applications 100 (3) (1999) 699–712.