Solving Assembly Scheduling Problems With Tree-Structure Precedence Constraints: A Lagrangian Relaxation Approach

Jingyang Xu and Rakesh Nagi

***Abstract*—In this paper, we consider an assembly scheduling problem (ASP) with tree-structured precedence constraints. In our problem, there are a number of work centers. Each work center contains a number of machines of the same functionality. The job to be processed via this system is a job with tree-structure precedence constraints. Each operation in the job has a designated work center. We propose a mixed integer linear programming formulation and solve the problem with a Lagrangian relaxation (LR) approach. We solve the subproblems of the LR problem via a heuristic method and generate feasible solutions via a randomized list scheduling algorithm. Near-optimal results are obtained and the computational time is within a few seconds for problems with size up to 20 machines and 300 operations.**

***Note to Practitioners*—Most industrial products are complex assemblies made up of multiple manufactured products—organized as Bills-Of-Materials (BOMs)—that are produced in a shared manufacturing facility with possibly multiple functionally identical machines. There are a few rigorous approaches to deal with these kinds of problems in the literature. The job-shop scheduling problem has been extensively addressed, but these problems do not provide for any relationship between the multiple manufactured parts. In an assembly scheduling problem, the completion time of a child component becomes the release time of the parent part; hence, the multiple jobs become connected in a tree structure defined by the BOM. The second aspect that has been less considered in the scheduling literature is multiple, functionally identical machines—a common occurrence in industry. This paper takes on these two challenges and solves this problem using a LR approach and associated heuristics. Industrial strength problems of about 300 operations and 10 work centers are solved in a few seconds using this approach to near-optimality, while commercial strength mixed integer programming solvers are unable to find useful solutions in an hour.**

***Index Terms*—Assembly scheduling, Lagrangian relaxation (LR), makespan, parallel machine scheduling, randomized algorithm, subgradient search.**

I. Introduction

A. Background

SCHEDULING is one of the most important operational problems in the manufacturing industry. It is concerned with the assignment of operations to machines, sequencing them and/or providing them start and end times. Therefore, it is directly related to the productivity of the manufacturing enterprise. Generating high-quality schedules has been the most difficult problem to solve.

A low volume and high variety manufacturing environment is the most suitable for producing customized products. Such environments are most common in heavy-mechanical industry and defense-related high technology industry. Shorter life cycles of the products, customer choices, and technological advances have forced most manufacturing companies to shift the strategy from mass production to small batch production. In today's lean manufacturing environment, organizations are trying to produce products in one piece flow. Job shop is the environment where these kinds of complex assemblies are produced. As the variety of the products increases, coordination between different work centers becomes important for reducing makespan, as well as inventory holding cost. That is the reason efficient scheduling of production plays a key role in the cost competitiveness for an organization. Lead time and work-in-process inventories are often excessive; the machine utilization is low in high variety, but low volume manufacturing environment.

The objective of this research is to find good feasible solutions in a short time for industrial sized problems with an objective of minimizing the makespan. Also, the manager should have confidence in the quality of the solution generated. Our method provides such a quality characterization with extensive numerical results indicating that optimal or near optimal results are obtained in most test cases.

The assembly scheduling problem we consider is related to a famous class of scheduling problems called job shop scheduling problems. In a classical job shop scheduling problem, a number of jobs are to be processed and each job has a number of operations that must be performed sequentially. While there are no precedence constraints between the jobs, the operations within each job have sequential precedence constraints. The makespan for a job shop scheduling problem is considered to be the largest completion time of all jobs, which is equivalent to adding a dummy job with zero processing time that can only be processed when all other jobs are completed. In an assembly scheduling problem, we consider tree-structure precedence constraints among operations, which leads to a more general and complex precedence constraint system than that in a classical job shop scheduling problem. At the same time, an assembly scheduling problem has one final operation whose completion time, like the dummy job in a job shop scheduling problem, is the makespan of a schedule. Thus, the assembly scheduling problem can be considered as a generalization of classical job

Manuscript received April 23, 2012; revised October 17, 2012; accepted January 06, 2013. Date of publication May 21, 2013; date of current version June 27, 2013. This paper was recommended for publication by Associate Editor Y. Tang and Editor K. Goldberg upon evaluation of the reviewers' comments.

The authors are with the University at Buffalo (SUNY), Department of Industrial and Systems Engineering, Buffalo, NY 14214 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2013.2259816

shop scheduling problems. See also [1] for multiple assembly (tree) to single tree conversion.

Significant research has been done in the area of job shop scheduling. Most versions of this problem are NP-hard. There are methods for optimal solutions, such as dynamic programming [2] and branch and bound [3]. While these algorithms can obtain optimal solutions, they are inapplicable for solving large problems because of long computational times.

One popular research direction is to solve the scheduling problem via heuristic or metaheuristic algorithms. The aim of these algorithms is to find good feasible solutions within a short computational time. These algorithms include shifting bottleneck procedure [4], Tabu search and simulated annealing [5], memetic algorithm [6], genetic algorithms [7], artificial bee colony [8], neighborhood search algorithm [9], particle swarm optimization [10], and others. Among these approaches, the shifting bottleneck (SB) procedure is well studied and tested under various testing environments [11]. Though SB was originally developed for worker centers with a single machine, it is also extended to an environment where work centers have parallel machines [12]. Besides, there are heuristics proposed using Petri nets recently, such as [13] and [14]. For a comprehensive overview of all scheduling methodologies, one can refer to [15].

The heuristic methods mentioned above are able to generate good solutions within a short computational time for similar problems studied in this paper. The heuristic methods also have the advantage that they can be easily modified to solve problems with different objective functions and problem structures [16]. However, these methods do not provide information of the solutions' optimality gaps. Thus, it would be difficult to determine the stopping criteria for problems that are different from the testing cases. At the same time, some of them require properly tuned parameters for different problem structures. These disadvantages limit the general implementation of these heuristic methods. As it will be shown in the following sections, the approach proposed in this paper would be able to provide solution optimality gap information during the computation process and do not require properly tuned parameters from users.

Lagrangian relaxation (LR) belongs to the class of approaches for fast and good solutions. It has a successful history on solving job shop scheduling problems since the 1970s. It has advantages not restricted to generating tight lower bounds, which can be incorporated with branch and bound procedure, but also guidance on constructing good feasible solutions from relaxed problems' solutions. Compared to the heuristic methods mentioned above, LR has the advantage that optimality gap information of solutions are usually available during the computation process. Reference [17] conducted the first successful application of the LR approach on scheduling problems with precedence constraints on a min-sum objective. Reference [18] applied the LR framework on a parallel machine job shop scheduling problems with simple precedence constraints on problems up to 200 jobs, 40 machines, and 3 to 5 operations per job, with the objective of minimizing the weighted quadratic tardiness. Reference [19] considered an assembly environment, where a tree structure Bill-of-Materials (BOM) is used to define the precedence constraints. The problem of up to 30 machines and 150 operations, with an objective of minimizing the total weighted quadratic tardiness, is solved to around 4% optimality gap. Reference [20] studied the scheduling for BOM network scheduling a little further by modifying the problems with the objective of minimizing total weighted quadratic tardiness and quadratic earliness. Reference [21] proposed a different subgradient search schema and initial multiplier generation approach, which led to fast convergence of the subgradient search procedure. The problem objective is to minimize the summation of completion time. Reference [22] recently made an effort on combining LR and cut generation together for generating tighter lower bounds. Around 4% gaps are obtained for the objective of minimizing weighted tardiness with this approach for a problem size up to 100 jobs and 3 machines.

In all the LR applications mentioned above, the objective function belongs to a min-sum type and all the relaxation mechanisms are on relaxing capacity constraints of machines. The advantage with the min-sum objective is that LR could work pretty well as the problem gets decomposed into operation-level subproblems that can be solved to optimality quickly. However, for min-max type problems, such as minimizing makespan, this relaxation schema cannot be applied easily. At the same time, the time index formulation for the subproblem leads to a limit on the efficiency of the approaches. As operations' processing time is not integer or does not belong to a small range, the computation efforts would increase substantially in the subproblem level because the subproblems are solved by enumerating all solutions at every time slot.

Ten years ago, a surrogate subgradient search was proposed by [23]. This method is proposed for LR schema that leads to NP-hard subproblems. The surrogate subgradient search would allow approximate solutions to the subproblems, while the search procedure convergence is still maintained. Based on the surrogate subgradient search, [24] proposed an alternative LR framework of decomposition to machine level subproblems instead of relaxing machine capacity constraints. This framework is similar to the decomposition schema in this paper. The problem's objective is to minimize the summation of total weighted quadratic earliness and quadratic tardiness. [25] applied the framework of relaxing precedence constraints and solve instances of up to 10 machines and 20 jobs. A recent comment on surrogate subgradient search can be found in [26].

For these applications in which subproblems are not solved to optimality, the performance of LR depends on the number of relaxed constraints and the problem complexity. Though tight lower bounds are not always obtained, the solutions of LR problems can still help to generate good feasible solutions. We adopt the idea of solving LR problem approximately from surrogate subgradient search and employ it into our approach when searching for good solutions.

The remainder of this paper is structured as follows. In Section II, a mixed integer programming formulation is proposed for the problem and a LR schema is introduced. Section III describes the solution procedure for subproblems, the subgradient search for obtaining good multipliers, the relaxation schema for generating problem's lower bounds, and the randomized list scheduling algorithm to construct feasible solutions. In Section IV, we test the LR-based approach and compare it with shifting bottleneck procedure with randomly

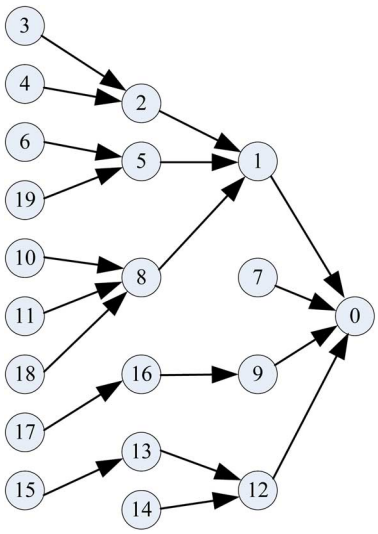

Fig. 1. BOM example.

generated input data under a number of scenarios. Section V presents the conclusions and directions for future research.

II. Problem Description and Formulation

The manufacturing facility producing multilevel assemblies consists of a set of production work centers $(W_1, W_2, \ldots, W_w)$. Each work center W_Y consists of f_Y functionally identical machines. In this manufacturing facility, all identical machines are grouped within the same work center, and no two work centers have machines which can perform the same function.

Given the Bill-of-Materials (BOM) of a final product and routing for each manufactured part, an operation network can be developed. This operations network has a tree-like precedence structure. A typical operations network is shown in Fig. 1. In the network, each node represents an operation. Each operation has a unique work center requirement, which means each operation is designated to be processed in one of the work centers. The scheduling problem of all the operations in the entire BOM is referred to as Assembly Scheduling Problem in the paper. To formulate the problem as a mixed integer linear program, the following assumptions are made.

- Back to back production strategy is adopted.
- A machine can perform at most one operation at a time.
- An operation can be performed on at most one machine at a given time.
- Preemption of operations is not permitted.
- Processing time of all operations are deterministic and known.
- Backlogging is not permitted.

It needs to be mentioned that the network in Fig. 1 can also represent an assembly scheduling problem with multiple end products. When operation 0 is considered as a dummy operation, there are four end products in Fig. 1 with final operations of 1, 7, 9, and 12. Each of the end product itself has its own BOM tree. Thus, minimizing the makespan for the one end product assembly scheduling problem is equivalent to minimizing the makespan of an assembly scheduling problem with multiple end products (see also [1]).

A. Mathematical Model for Assembly Scheduling Problem

The assembly scheduling problem with parallel machines is formulated into a Mixed Integer Linear Program (MILP), with notation and formulation as follows.

- There are n operations in set $N = \{1, 2, \ldots n\}$ to be processed. O_i is used to denote operation i.
- There are w work centers.
- Y is the index of work centers, where $Y \in \{1, 2, \ldots, w\}$.
- I_Y is the set of all operations that require work center Y.
- Each work center has a number of identical parallel machines. f_Y is the number of machines in worker center Y.
- Z is the makespan of a feasible schedule for all operations.
- $s(j)$ is successor of operation $j, \forall j \in N$.
- S_j denotes starting time of operation $j, \forall j \in N$.
- ES_j is the earliest starting time of operations $j, \forall j \in N$.
- F_j denotes finishing time of operation $j, \forall j \in N$.
- t_j is the processing time of operation $j, \forall j \in N$.
- h_j is the index of work center that operation j is designated to.
- $$\psi_{ij} = \begin{cases} 1, & \text{if } O_j \text{ precedes } O_i \text{ on some machine} \\ 0, & \text{otherwise. } \forall i, j \in I_Y, Y \in \{1, 2, \ldots, w\} \end{cases}.$$
- $$\phi_{jy} = \begin{cases} 1, & \text{if } O_j \text{ is performed on the } y^{th} \text{ identical machine of work center } Y \\ 0, & \text{otherwise. } \forall i, j \in I_Y, Y \in \{1, 2, \ldots, w\} \end{cases}.$$
- $$\chi_{ij} = \begin{cases} 1, & \text{if } O_i \text{ is a child of } O_j \text{ in the BOM tree, or } s(i) = j \\ 0, & \text{otherwise. } \forall i, j \in I_Y, Y \in \{1, 2, \ldots, w\} \end{cases}.$$

$$\text{Minimize}: \ Z$$

$$\text{Subject to}:$$

$$S_j \geq F_i, \quad \forall i, j \in N, \chi_{ij} = 1 \tag{1}$$

$$S_j \geq ES_j, \quad \forall j \in N \tag{2}$$

$$F_j \leq Z, \quad \forall j \in N \tag{3}$$

$$F_j = S_j + t_j, \quad \forall j \in N \tag{4}$$

$$\psi_{ij} + \psi_{ji} = 1, \quad \forall i, j \in I_Y, i \neq j, Y \in \{1, 2, \ldots, w\} \tag{5}$$

$$S_i - F_j \geq (\psi_{ij} + \phi_{iy} + \phi_{jy} - 3)M, \quad \forall i, j \in I_Y, i \neq j, \quad y \in \{1, 2, \ldots, f_Y\}, \quad Y \in \{1, 2, \ldots, w\} \tag{6}$$

$$\sum_{y=1}^{f_Y} (\phi_{jy}) = 1, \quad \forall j \in I_Y, Y \in \{1, 2, \ldots, w\} \tag{7}$$

$$\psi_{ij} \in \{0, 1\}, \quad \forall i, j \in I_Y, i \neq j, \quad Y \in \{1, 2, \ldots, w\} \tag{8}$$

$$\phi_{jy} \in \{0, 1\}, \quad \forall j \in I_Y, \quad y \in \{1, \ldots, f_Y\}, Y \in \{1, 2, \ldots, w\}. \tag{9}$$

The objective function minimizes the total production makespan (Z). Constraint (1) ensures that the starting time of a successor operation j cannot be less than the finishing time of predecessor operation i. $\chi_{ij} = 1$ means that operation i is the child of operation j in the BOM, which is equivalent to

operation i being the predecessor of operation j in a feasible schedule. Constraint (2) ensures that the starting time of each operation is greater than or equal to the earliest starting time of that operation (ES_j). The earliest starting time for operation j is obtained from the critical path of the BOM. Either a critical path algorithm or linear programming can be used to obtain the earliest starting times. Constraint (3) is the relation between makespan and finishing times of all operations, meaning that the makespan of a schedule is equivalent to the largest finishing time of all operations. Constraint (4) describes the relation between the starting time and finishing time of any operation j. Since no splitting operations are allowed, each operation's finishing time is equivalent to the operation's starting time plus its processing time. Constraint (5) is for unidirectional nature of precedence relation between any two operations done on the same work center. This constraint is redundant when operation i and j are not assigned to the same machine in a work center. When operation i and operation j are assigned to the same machine, constraints (5) and (6) together describe the capacity limit of corresponding machines. In constraint (6), M is a large number. Constraint (6) is a machine capacity constraint, which is redundant unless $\psi_{ij} = 1$, $\phi_{iy} = 1$, and $\phi_{jy} = 1$. This logic relation has two meanings. First, if either operation i or operation j are not assigned to the same machine y, the right-hand-side of the inequality would be a negative value with big absolute value, and the constraint is redundant. Second, when both $\phi_{iy} = 1$, and $\phi_{jy} = 1$, the inequality describes the precedence relation of operation i and operation j using ψ_{ij}, while constraint (5) describes the unidirectional precedence constraint. Constraint (7) shows that each operation is required to be performed on exactly one machine in the corresponding work center. Constraints (8) and (9) denote the binary nature of decision variables ψ_{ij} and ϕ_{jy}.

B. Lagrangian Relaxation of ASP

In our LR procedure, we relax the precedence constraints between operations using different work centers. Thus, in the relaxed problem, there will be precedence constraints between operations using the same work center. We denote the relaxed problem as $\mathbf{DL}$, whose formulation is as follows:

$\mathbf{DL}$:

$$\text{Minimize}: \ L = Z + \sum_{\forall i,j \in N, \chi_{ij}=1, h_i \neq h_j} (U_{ij}(F_i - S_j))$$

Subject to :

$$S_j \geq F_i, \quad \forall i,j \in N, \chi_{ij} = 1, h_i = h_j \tag{10}$$

and constraints (2)–(9).

The LR problem $\mathbf{DL}$ is not decomposable due to constraint (3), which is the constraint for operations' finishing time and makespan of the whole schedule. To make the problem decomposable into work center level subproblems, we break this relation by further relaxation. We first introduce a new variable Z_Y to represent the makespan for the partial schedule for all operations in the Y^{th} work center and replace constraint (3) with two new constraints (11) and (12), which represent the constraints between the makespan of a partial schedule for a single work center and makespan of a complete schedule

$$F_j \leq Z_Y, \quad \forall j \in I_Y, Y \in \{1,2,\ldots,w\} \tag{11}$$

$$Z_Y \leq Z, \quad \forall Y \in \{1,2,\ldots,w\}. \tag{12}$$

Then, we relax constraint (12) and obtain a new relaxed problem $\mathbf{DLN}$ from $\mathbf{DL}$. Another Lagrangian multiplier V_Y is introduced for the relaxed constraint (12). V_Y is also restricted to be positive. And the optimal solution's objective value of problem $\mathbf{DLN}$ is also a lower bound for the original problem. $\mathbf{DLN}$'s formulation is as follows:

$\mathbf{DLN}$:

$$\text{Minimize}: \ L = Z + \sum_{\forall i,j \in N, \chi_{ij}=1, h_i \neq h_j} (U_{ij}(F_i - S_j)) + \sum_{Y \in \{1,2,\ldots,w\}} V_Y(Z_Y - Z)$$

$$\text{Subject to}: \text{Constraints } (2),(4),(5),(6),(7),(8),(9),(10) \text{ and } (11).$$

The objective function in the relaxed problem minimizes not only the original objective, the makespan, but also the penalty for violating the precedence constraints. In a standard procedure, a subgradient search procedure is implemented to find optimal multipliers by solving the relaxed problems and updating the multipliers. A non-negative multiplier U_{ij} is associated with each pair of the relaxed precedence constraints. Positive values of these multipliers will help penalize violations of the relaxed constraints. Whenever the LR problem is solved to optimality in the subgradient search procedure for optimal multipliers, the optimal objective value of the LR problem would always be a lower bound of the original minimization problem. At the same time, when a LR problem cannot be solved to optimality, a lower bound for the LR problem is also a valid lower bound for the original minimization problem.

C. Decomposition and Subproblems

After relaxation of precedence constraints and single work center makespan constraints, the LR problem $\mathbf{DLN}$ can be decomposed into two different sets of subproblems: one makespan subproblem and w work center subproblems. We denote the makespan subproblem as $\mathbf{SPM}$ and work center subproblems as $\mathbf{SPW}$.

$\mathbf{SPM}$'s formulation is described as follows:

$$\text{Minimize}: Z\left(1 - \sum_{Y=1}^{w} V_Y\right)$$

Subject To :

$$Z \geq 0. \tag{13}$$

The above formulation is decomposed from the LR problem $\mathbf{DLN}$, where only the relations between multiplier V_Y and makespan of the complete schedule are involved. In the formulation, V_Y has a known value which is initialized and updated in the subgradient search process.

In **SPW**, there are w separate subproblems for work center 1 through w

$$\mathbf{SPW}\,(Y \in \{1,2,\ldots,w\}):$$

$$\text{Minimize}: \; L_Y = V_Y Z_Y + \sum_{\forall i \in N} \sum_{\forall j \in I_Y, \chi_{ij}=1} U_{ij} F_j - \sum_{\forall i \in I_Y} \sum_{\forall j \in N, \chi_{ij}=1} U_{ij} S_i$$

Subject To :

$$S_j \geq F_i, \quad \forall i,j \in I_Y, \chi_{ij}=1, h_i = h_j \tag{14}$$

$$S_j \geq ES_j, \quad \forall j \in I_Y \tag{15}$$

$$F_j \leq Z_Y, \quad \forall j \in I_Y \tag{16}$$

$$F_j = S_j + t_j, \quad \forall j \in I_Y \tag{17}$$

$$\psi_{ij} + \psi_{ji} = 1, \quad \forall i,j \in I_Y, i \neq j \tag{18}$$

$$S_i - F_j \geq (\psi_{ij} + \phi_{iy} + \phi_{jy} - 3)M, \quad \forall i,j \in I_Y, i \neq j, \; y \in \{1,\ldots,f_Y\} \tag{19}$$

$$\sum_{y=1}^{f_Y} (\phi_{jy}) = 1, \quad \forall j \in I_Y \tag{20}$$

$$\psi_{ij} \in \{0,1\}, \quad \forall i,j \in I_Y, i \neq j \tag{21}$$

$$\phi_{ij} \in \{0,1\}, \quad \forall j \in I_Y, y = 1,\ldots,f_Y \tag{22}$$

$$S_j \geq 0, \quad \forall j \in I_Y. \tag{23}$$

Constraints (14)–(23) have the same meanings as their corresponding constraints in the original problem's formulation. The difference is that all constraints for **SPW** are restricted to one work center, and the problem is smaller in dimension compared to the original ASP problem.

III. Methodology

A. Solving the Makespan Subproblem

Solving **SPM** is trivial. The optimal solution can be obtained by evaluating the values of multiplier V_Y as follows.

- If $\sum_{Y=1}^{w} V_Y > 1$, then $Z \to \infty$ and objective value goes to $-\infty$.
- If $\sum_{Y=1}^{w} V_Y = 1$, then Z can be any value and the objective value equals to 0.
- If $\sum_{Y=1}^{w} V_Y < 1$, then $Z = 0$ and objective value equals to 0.

To avoid **SPM** approaching an objective value of negative infinity, a normalization step is added such that when $\sum_{Y=1}^{w} V_Y > 1$ happens in the procedure for obtaining improved multipliers: If $\sum_{Y=1}^{w} V_Y > 1$, then let $V_Y^{\text{new}} = (V_Y^{\text{old}} / \sum_{Y=1}^{w} V_Y^{\text{old}}), \forall Y \in \{1,2,\ldots,w\}$, where V_Y^{old} denotes the value of V_Y before normalization, and V_Y^{new} is the value for V_Y after normalization.

B. Solving Work Center Subproblem

SPW can be denoted as $P_m|r_j, prec|\sum w_j C_j + wC_{\max}$, which is an identical parallel machine scheduling problem to minimize weighted completion time and weighted makespan with release time and precedence constraints. w_j is generated by summing up multipliers U_{ij} for operation j and the value can be positive, negative or zero. r_j is the earliest starting time ES_j, which is generated by the critical path algorithm on the original problem. Reference [27] proves that $1|r_j|\sum w_j C_j$ is NP-hard with positive w_j. Thus, **SPW** is also NP-hard in a strong sense. When the problem size gets large, it is time-consuming to obtain an optimal solution for **SPW** using a standard optimizer such as CPLEX. Experiments show that CPLEX cannot solve the subproblem within an hour for ten or more operations for a subproblem with a single machine. Thus, heuristics are constructed for obtaining feasible solutions to the subproblem **SPW**.

On solving $1|r_j, prec|\sum w_j C_j$, [28] proposes a greedy algorithm. However, that algorithm is limited to a single machine problem with $w_j > 0$. After modification, a new greedy algorithm is presented for solving $P_m|r_j, prec|\sum w_j C_j + wC_{\max}$. The new greedy algorithm is described as follows.

1) Sort all operations according to the Shortest Weighted Processing Time (SWPT) order and earliest release time following a similar rule as in [29], such that:
   - All operations with positive weight are ordered before operations with zero weight.
   - All operations with zero weight are ordered before operations with negative weight.
   - For any two operations i,j with positive weights, i is ordered before j if $(t_i/w_i) < (t_j/w_j)$.
   - For any two operations i,j with negative weights, i is ordered before j if $(t_i/w_i) < (t_j/w_j)$.
   - For any two operations i,j with zero weights, i is ordered before j if $r_i < r_j$.
   - For any two operations i,j with same weighted completion time, which is $(t_i/w_i) = (t_j/w_j)$, i is ordered before j if $r_i < r_j$.
2) Let $Q = \{i | i \in I_Y\}$ be the set of all operations in current work center; $B_j = \{i | i,j \in I_Y, \chi_{ij} = 1\}$ be the set containing a number of operations for operation j such that all the operations in the set are predecessors of operation j according to precedence constraints in the BOM; $T[y] = 0, WC[y] = 0, \forall y \in \{1,2,\ldots,f_Y\}$, where $T[y]$ is to store the current makespan for machine y in current work center, $WC[y]$ is the objective value for the partial schedule on machine y; $C[j] = 0, \forall j \in Q$, which initializes all operations' completion time with value 0.
3) Setup a set $Q' = \{j | j \in Q, B_j \cap Q = \emptyset\}$ that contains all operations available to be assigned; Find out the $T^* = \min_{\forall y \in \{1,2,\ldots,f_Y\}} T[y]$; Find out an operation j with minimum release time $r^* = \min_{\forall j \in Q'} r_j$. If $T^* < r^*$, set $T[y] = r^*, \forall T[y] < r^*, y \in \{1,2,\ldots,f_Y\}$.
4) Select a machine y^* such that $T[y^*] = \min_{y \in \{1,2,\ldots,f_Y\}}(T[y])$.
5) Select an operation j^* such that j^* ranks first in the set $\{j | j \in Q', r_j < T[y^*]\}$ according to the order in Step 1.
6) $T[y^*] = T[y^*] + t_{j^*}$, $WC[y^*] = WC[y^*] + w_{j^*} T[y^*]$, $Q = Q - \{j^*\}$, $C[j^*] = T[y^*]$.
7) If $Q = \emptyset$, go to Step 8, otherwise, go to Step 3.
8) Return $\sum_{y \in \{1,2,\ldots,f_Y\}} WC[y] + w \max_{y \in \{1,2,\ldots,f_Y\}} T[y]$ as objective value, and $C[j], \forall j \in I_Y$ as the solution of operations completion time in current schedule. The heuristic algorithm stops here.

C. Subgradient Search for Good Multipliers

In a typical LR approach, the LR problem or the decomposed subproblems are solved optimally. The optimal objective value of the relaxed problem is a valid lower bound for the original minimization problem. Through subgradient search, improved multipliers are obtained and tight lower bounds are generated. Meanwhile, heuristics are employed for constructing feasible solutions for the original problem from the solution of the LR problem. However, this procedure is modified for use in this paper. Since the subproblems cannot be solved optimally, the subgradient search can only generate relatively good multipliers and pseudo lower bounds. The pseudo lower bounds in each iteration in the subgradient search procedure are used as convergence criteria for the subgradient search as well as an indicator on the quality of multipliers' values. In this paper, we use subgradient search as an approach to find good multipliers, while lower bounds are constructed by other approaches in Section III-D. The subgradient search is implemented as follows.

1) Solve linear relaxation of the original problem; Obtain dual cost λ_{ij} for constraint $S_j - F_i \geq 0, \forall i,j \in N, \chi_{ij} = 1, h_i \neq h_j$ and dual cost δ_Y for constraint $Z - Z_Y \geq 0, \forall Y \in \{1,2,\ldots,w\}$.
2) Initialize multipliers at iteration $K = 1$ using the dual costs obtained in Step 1 (see [30] for rationale)

$$U_{ij}^1 = \lambda_{ij}, \quad \forall i,j \in N, \chi_{ij} = 1, \text{ and}$$
$$V_Y^1 = \delta_Y, \quad \forall Y \in \{1,2,\ldots,w\}.$$

3) Checking the value of V_Y, if $\sum_{Y\in\{1,2,\ldots,w\}} V_Y > 1$, a normalization procedure is applied (see Section III-A). Solving subproblems approximately using a heuristic (see Section III-B). The sum of objective values of all the subproblems provides a pseudo lower bound at iteration K.
4) Multiplier U_{ij} and V_Y can be updated as follows:

$$\|F - S\| = \sum_{\forall i,j\in N,\chi_{ij}=1} (F_j - S_i)^2$$
$$\|Z - Z_Y\| = \sum_{\forall Y\in\{1,2,\ldots,w\}} (Z - Z_Y)^2$$
$$U_{ij}^{K+1} = U_{ij}^K + \frac{sk(F_i - S_j)}{\sqrt{\|F - S\| + \|Z - Z_Y\|}}$$
$$V_Y^{K+1} = V_Y^K + \frac{sk(Z_Y - Z)}{\sqrt{\|F - S\| + \|Z - Z_Y\|}}$$

where $sk = 1/K$. Non-negativity of the U_{ij}^{K+1} is maintained by setting $U_{ij}^{K+1} = 0$ whenever $U_{ij}^{K+1} \leq 0$.
5) $K = K + 1$. If $K > 100$, stop; otherwise, go to Step 3.

D. Constructing Lower bounds

Since the subproblems are solved using heuristics, the objective values for the LR problem in the subgradient search procedure are not valid lower bounds for the original problem. However, a valid lower bound can be obtained for the LR problem using the lower bounds of the subproblems. The valid lower bound for the LR problem can be used as a valid lower bound of the original problem. Two sets of lower bounds are proposed in this section.

In the first set of lower bounds, we derive lowers bounds from existing lower bounds in the literature. Reference [31] derives a set of fast lower bounds for $P_m|r_j, q_j|C_{\max}$, which can be applied as a first set of simple lower bounds for the single work center subproblem

$$LB_Y^0 = \max_{j\in I_Y}(r_j + t_j)$$
$$LB_Y^1 = \min_{j\in I_Y}(r_j) + \frac{1}{f_Y}\sum_{j\in I_Y} t_j$$
$$LB_Y^2 = \frac{1}{f_Y}\left(\bar{r}_1 + \bar{r}_2 + \ldots + \bar{r}_{f_Y} + \sum_{j\in I_Y} t_j\right).$$

In LB_Y^2, $\bar{r}_k$ denotes the k^{th} smallest release time for all operations in I_Y.

Proposition 1: $\max_{Y\in\{1,2,\ldots,w\}}[\max_{k\in\{0,1,2\}}(LB_Y^k)]$ is a valid lower bound of the original problem.

Proof: First, a lower bound of the LR problem with multipliers $U_{ij} \geq 0$, $V_Y \geq 0$ is a valid lower bound for the original problem. Further, if multiplier V_Y satisfies the condition $\sum_{Y\in\{1,2,\ldots,w\}} V_Y \leq 1$, then the makespan subproblem's objective value is 0 and the summation of the w work center's subproblem is a valid lower bound of the original problem. Thus, the summation of the work center subproblems' lower bounds is a valid lower bound for the original problem. If we assign $U_{ij} = 0$ and $V_Y = 1/f_Y$, the work center subproblems become $P_m|r_j, prec|V_Y C_{\max}$. Then, after relaxing precedence constraints, the work center subproblems become $P_m|r_j|V_Y C_{\max}$. Until now, we can claim that $\sum_{Y\in\{1,2,\ldots,w\}}[V_Y \max_{k\in\{0,1,2\}}(LB_Y^k)]$ becomes a valid lower bound. Let $Y^* = \text{argmax}_{Y\in\{1,2,\ldots,w\}}\{\max_{k\in\{0,1,2\}}(LB_Y^k)\}$, then by assigning that $V_{Y^*} = 1$ and $V_Y = 0, \forall Y \neq Y^*, Y \in \{1,2,\ldots,w\}$, we have $\sum_{Y\in\{1,2,\ldots,w\}}[V_Y \max_{k\in\{0,1,2\}}(LB_Y^k)] = \max_{k\in\{0,1,2\}} LB_{Y^*}^k = \max_{Y\in\{1,2,\ldots,w\}}[\max_{k\in\{0,1,2\}}(LB_Y^k)]$. Thus, the proposition is proved and the proposed lower bound is a valid lower bound for the original problem.

Another set of lower bounds is valid only for work centers with a single machine. This lower bound is generated by relaxing the work center subproblem $1|r_j, prec|\sum_j w_j C_j + wC_{\max}$ to $1||\sum_j w_j C_j + wC_{\max}$. An easy lower bound can be constructed by scheduling the operations to the single machine according to SWPT rule when $\sum_{\forall j,j\in I_Y, w_j<0} w_j + w > 0$. When $\sum_{\forall j,j\in I_Y, w_j<0} w_j + w \leq 0$, operations with positive and zero weights are scheduled via SWPT from time 0, while the operations with negative weights are scheduled via SWPT from a known upper bound of the original problem.

E. Procedure for Generating Feasible Solutions

Feasible solutions for the original problem are generated from the LR problem solution at every iteration of the subgradient search using a randomized list scheduling algorithm. This randomized list scheduling algorithm makes use of the finishing times of all operations from the solution of the LR problem and

assigns operations to machines. The randomized list scheduling algorithm is described as following:

1) Sort all operations in a list L according to the ascending order of finishing times in the LR problem's solution, where the i^{th} member in the list is the operation that has the i^{th} smallest finishing time in the solution of the LR problem. In the list L according to this order, $L[i]$ is the index of the operation that has the i^{th} smallest finishing time.
2) Let $Q = \{i|i \in N\}$ be a set of all operations, $B_j = \{i|i, j \in N, \chi_{ij} = 1\}$ be a set for operation j containing all operations that are predecessors of operation j. Array T, with $T[y][Y]$ denoting the makespan of machine y in work center Y, is initialized with value 0, $\forall y \in \{1, 2, \ldots, f_Y\}, Y \in \{1, 2, \ldots, w\}$. Array C, with $C[j], j \in N$, denoting the completion time of operation j, is also initialized with value 0.
3) Setup a set $Q' = \{j|j \in Q, B_j \cap Q = \emptyset\}$ with $q = |Q'|$ being the total number of elements in set Q'. Set up a list L', such that L' is a sublist of L containing all operations in Q'. All operations in list L' are operations that have all of their predecessors assigned, thus themselves are available to be assigned.
4) Setup a set $P = \{p_j, \forall j \in Q'\}$ such that p_j is the probability of selecting operation j from set Q'. The probability of selecting operations follows a rule, such that if operation j_1 is before operation j_2 in the list L', then the probability of selecting operation j_1 is larger than that of j_2 or $p_{j_1} > p_{j_2}$. In this paper, we set $p_j = (2(q-k)/q(q+1))$, where $L'[k] = j$. Under the probability defined in P, an operation j^* from Q' is selected randomly to be assigned.
5) Find a machine y^*, which should be a machine in the work center to process operation j^*. At the same time, machine y^* should have the smallest makespan among all machines in the same work center, where the makespan is calculated with existing operation assignments. Mathematically, it can be stated that $y^* = argmin_{y \in \{1,2,\ldots,f_Y\}} T[y][h_{j^*}]$.
6) Assign operation j^* to machine y^* in work center h_{j^*}. The starting time for operation j^* and updated makespan of machine y^* in work center h_{j^*} is obtained from the maximum value among machine y^*'s makespan, operation j^*'s release time, and maximum finishing time of all operation j^*'s predecessors. Mathematically, it can be described as: $T[y^*][h_{j^*}] = \max[\max_{\forall j \in B_{j^*}}(C[j]), T[y^*][h_{j^*}], r_{j^*}]$.
7) Assign job j^* to machine y^* in work center h_{j^*}. Update the set for unassigned operations $Q = Q - \{j^*\}$, the value for machine y^*'s makespan $T[y^*][h_{j^*}] = T[y^*][h_{j^*}] + t_{j^*}$, and the completion time for operation j^*, $C[j^*] = T[y^*][h_{j^*}]$.
8) If $Q \neq \emptyset$, which means there are operations not assigned, go to Step 3; otherwise, return the operation assignment and starting time as solutions to the original problem, and $\max_{y \in \{1,2,\ldots,f_Y\}, Y \in \{1,2,\ldots,2\}} T[y][Y]$ as an upper bound value for the original problem.

An overview of the whole procedure to solve the network scheduling problem is shown, as in Fig. 2. The basic ideas driving the proposed logic are as follows.

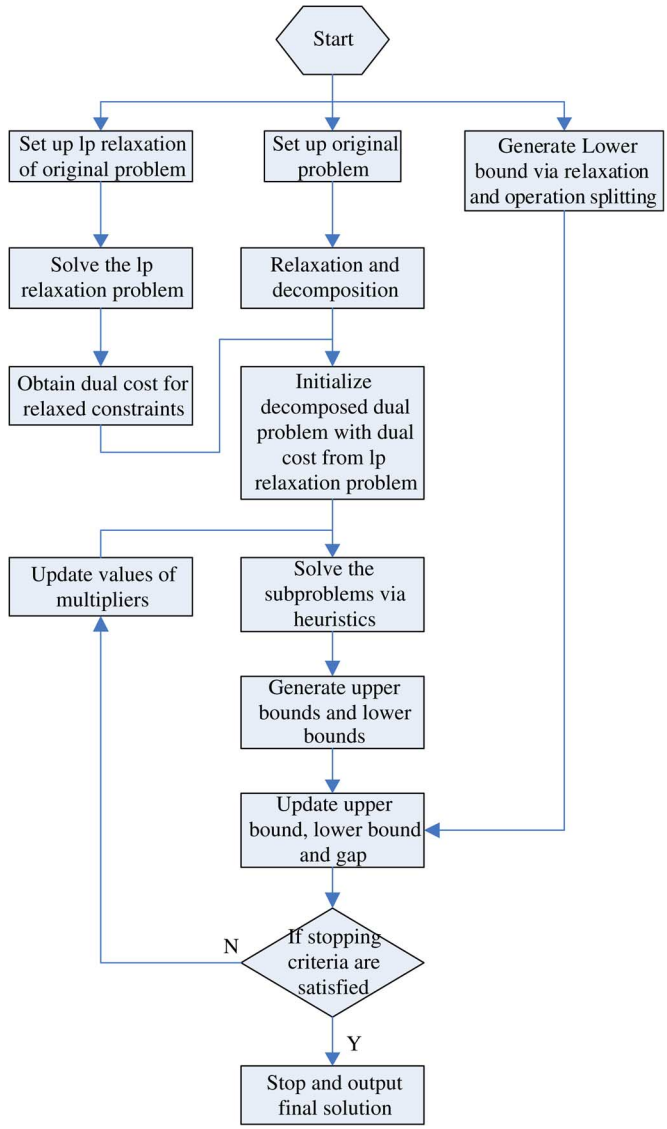

Fig. 2. Flowchart of the LR-based approach.

1) The original ASP problem is NP-hard and not easily solved. By using LR to decompose the original problem into a number of more tractable subproblems, we can solve the original problem based on solutions of simpler subproblems.
2) Since the subproblems are also NP-hard, and we are not able to obtain optimal solution with acceptable computational time, developing heuristics to solve subproblems to near optimality in a short time is needed, and deriving lower bounds from subproblems as the lower bound to original problem is also needed.
3) The solutions to the LR problem, which is decomposed to subproblems, are not feasible for the original problem. However, they contain operation assignment and sequencing decisions that are similar to optimal solutions for the original problem. By implementing a randomized list scheduling algorithm, we can make use of the information from the solutions of the LR problem and overcome local minimum to some extent.

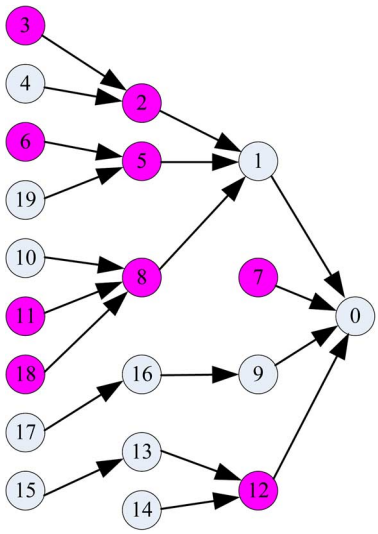

Fig. 3. A randomly generated BOM example.

TABLE I
EXAMPLE: OPERATION INFORMATION

Operation Index j	t_j	h_j	ES_j
0	3.701346	0	21.558947
1	2.456831	0	15.9223
2	6.37962	1	2.87817
3	2.87817	1	0
4	1.001373	0	0
5	6.736961	1	9.185339
6	9.185339	1	0
7	4.41905	1	0
8	5.80227	1	9.36964
9	7.28959	0	12.827906
10	7.847713	0	0
11	3.78402	1	0
12	5.549577	1	16.00937
13	7.635121	0	8.374249
14	2.61861	0	0
15	8.374249	0	0
16	5.048585	0	7.779321
17	7.779321	0	0
18	9.36964	1	0
19	6.413678	0	0

At this stage, additional control parameters for the subgradient search are added:

- α_1 is the number of subgradient search iterations where the pseudo lower bounds have not been improved. In the application stage, α_1 is initialized with value 0, added by 1 whenever a new iteration does not get any better pseudo lower bound, and reset to 0 whenever a new iteration gets a better pseudo lower bound.
- α_2 is the number of subgradient search iterations where the upper bounds have not been improved. α_2 is initialized with value 0, added by 1 whenever a new iteration does not get any better upper bound, and reset to 0 whenever a new iteration gets a better upper bound.
- α_3 is the number of times feasible solutions are generated by running the randomized list scheduling algorithm at each subgradient search iteration.

F. An Illustrative Example

As an illustration to the entire procedure, an example is presented with the following problem information.

- There are two work centers.
- Each work center has two identical machines.
- A randomly generated BOM structure with 20 operations is shown in Fig. 3.
- In Fig. 3, the operations are shaded according to their work centers; see also column h_j in Table I.
- The processing time of each operation is generated randomly and shown in column t_j in Table I.

As a first step, a critical path algorithm is executed for obtaining the earliest starting time for each operation. The earliest starting time for each operation is shown in column ES_j in Table I. After setting up the mathematical formulation of the assembly scheduling problem using the available BOM, operation and work center information, the linear programming relaxation is solved and the dual variable values are recorded.

Following the description in Section II-C, a number of precedence constraints are relaxed, and the relaxed network is shown in Fig. 4. At this point, the operations are grouped according to the work center they require, and the original assembly scheduling problem is decomposed into two work center subproblems and a makespan subproblem.

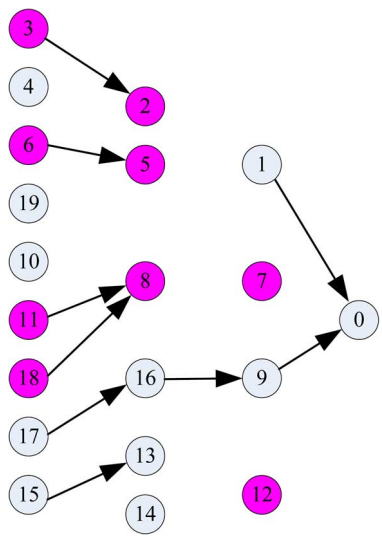

Fig. 4. BOM structure after relaxation.

Thereafter, the subgradient search procedure for improved Lagrangian multiplier values takes place, where the work center subproblems are solved using the heuristic algorithm in Section III-B. Based on the solution of the relaxed problem, the list scheduling algorithm, as described in Section III-E, is executed to generate a feasible solution to the original problem.

After a number of iterations, the subgradient search procedure converges and the final solution is achieved. In Fig. 5, the Gantt chart shows the schedule of all operations. This schedule has a makespan of 33.51445, while the lower bound is 30.08321.

IV. COMPUTATIONAL RESULTS

The computational experiments were conducted on a Thinkpad T500 laptop with 2.4 GHz Intel P2400 CPU and 4 GB memory. The programs were coded in C language with

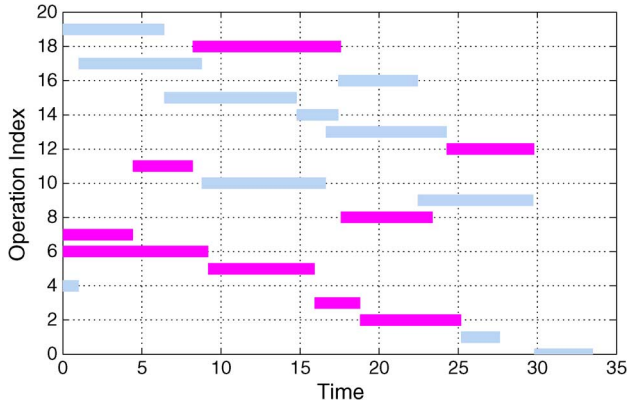

Fig. 5. Gantt chart for the final solution.

Visual Studio 2008. The optimizer used is CPLEX 12.1 32 bit version using one thread mode and default parameter settings. The stopping criteria for the subgradient search procedure is when $K = 100$, $\alpha_1 = 10$, or $\alpha_2 = 10$. Another control parameter is set that, $\alpha_3 = 100$. In the computational experiments, work centers with single machine and parallel machines are tested. The LR-based approach is compared with CPLEX and an implementation of the shifting bottleneck procedure.

A. Generating Test Cases

The test cases are tree-structured networks representing the precedence relations within the BOM structure. In each test case, the number of work centers, the number of operations, and the designated structure of the tree are given for generating random trees. While the number of work centers and number of operations are given, the BOM structure requirement is given in the form of the number of levels and maximum number of children. For example, a BOM requirement of "4 × 3" means that the randomly generated tree should have 4 levels and each node could have at most 3 children nodes. Operations' processing time follows continuous uniform distribution in [1,10].

All the test cases generated in this paper have a single end "product." This is because the assembly scheduling problem for minimizing the makespan with multiple end products can be transformed into a single end product case via a dummy end product (see also [1]). Thus, a test case with BOM structure "4 × 9," can also be considered as an assembly scheduling problem with (about) 9 end products.

The randomized procedure for generating a tree-structured network is as follows.

1) The generation procedure is started with a chain structure containing a number of nodes equal to the number of levels in the BOM. For each of the starting operations, generate a random real number in [1,10] as processing time and a work center is randomly selected and designated to process this operation.
2) If the number of operations exceeds the required amount, stop. Otherwise, an operation with random processing time and designated work center is generated. Then, the operation is added to the structure as a child of another operation randomly chosen among all the existing operations with number of children not exceeding the upper limit as in the BOM structure requirement. Repeat Step 2.

The test cases for comparison with CPLEX are generated under various types of BOM structures and work center profiles for the optimality evaluation of the LR-based approach. The test cases for comparison with the shifting bottleneck procedure have fewer BOM structures and work center profiles, but with ten replications of each BOM structure and work center profile combinations for comparison with shifting bottleneck procedure.

B. Comparison With CPLEX Results

The first set of experiments are conducted in an environment where each work center has a single machine. The test cases and corresponding results are shown in Tables II–V. In the layout of the four tables, each row contains information of the test problems, results from the LR-based approach, and results from CPLEX on the same problem. Column "WC," "n," and "BOM" are parameters representing the number of work centers, number of operations, and the scale of the BOM structure. BOM dimensions are in the format of "level × children" denoting the number of levels and maximum number of children for each operation in the tree structure. Column "m" exists only when there are work centers that have parallel machines. The value in column "m" is either a single number when all work centers have the same number of parallel machines, or a combination of several numbers for parallel machine numbers in each work center. For example, "2" means that every work center has two parallel machines and "$2 + 3 + 5$" means that the work centers have 2 machines, 3 machines and 5 machines, respectively. The columns "UB," "LB," "Gap," and "Time(s)" under "LR Results" represent the corresponding results of upper bound, lower bound, gap in percentage, and computational time in seconds for the original problem by the LR-based approach. "Gap" is calculated as in $\text{Gap} = (\text{UB} - \text{LB}/\text{UB}) \times 100\%$. The columns "UB," "Gap," and "Time(s)" under "CPLEX Results" represent the corresponding results of upper bound, gap, and computation time in seconds from CPLEX on the original problem. The computation time for CPLEX is limited to 1 hour for all the test cases, and CPLEX reports its own "Gap."

In Tables II and III, the computational experiments are compromised of test cases in narrow tree structures. As it is seen from Table II, the number of work centers ranges from 2 to 8 and the number of operations ranges from 10 to 200. Each BOM structure for the test cases in Table II has the same number of levels and limits on number of children for each operation. As it is shown in Table II, the LR-based approach is able to find several optimal solutions and provides optimality gaps of less than 2% for most of the cases. The computational time is within 5 seconds for most of the test cases in Table II.

CPLEX is supposed to have better performance on network structures with a large number of levels and a small number of children for each node, as the lower bounds from linear relaxation would be much tighter. In Table III, the limits on the number of children are 2 and 3, while the number of levels ranges from 4 to 15. The results show that the LR-based approach does not provide gaps as good as in Table II, though

TABLE II
SINGLE-MACHINE WORK CENTER PROBLEMS WITH NARROW TREE STRUCTURE: GENERAL CASES

Scenario			LR Results				CPLEX Results		
WC	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)	Time(s)
2	10	3×3	26.271*	26.271	1	0	26.271*	0	0.02
	25	4×4	69.018*	69.018	1	0	69.018	14.21	3600
	50	5×5	142.656	139.985	1	1.87	142.482	55.98	3600
	100	6×6	302.292	301.281	4	0.33	302.292	82.47	3600
	200	7×7	570.503*	570.503	4	0	570.503*	89.58	3600
3	10	3×3	19.233	17.865	1	7.11	19.233*	0	0
	25	4×4	48.738*	48.738	3	0	48.738*	0	30.11
	50	5×5	100.676*	100.676	4	0	100.676	36.32	3600
	100	6×6	195.513	191.762	1	1.92	195.513	74.14	3600
	200	7×7	397.302	394.562	5	0.69	394.562	85.71	3600
4	25	4×4	41.905*	41.905	1	0	41.905*	0	0.30
	50	5×5	75.249	74.169	1	1.44	75.249	32.11	3600
	100	6×6	161.164	160.153	3	0.63	161.164	65.6	3600
	200	7×7	327.593	324.853	14	0.84	324.853	81.99	3600
8	50	5×5	74.795	73.782	1	1.35	74.795	8.62	3600
	100	6×6	114.581	113.569	5	0.88	114.581	50.55	3600
	200	7×7	102.184	88.303	5	13.58	95.790	24.71	3600

TABLE III
SINGLE-MACHINE WORK CENTER PROBLEMS WITH NARROW TREE STRUCTURE: EXTREME CASES

Scenario			LR Results				CPLEX Results		
WC	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)	Time(s)
2	10	4×2	26.271*	26.271	1	0	26.271*	0	0.02
	25	6×2	72.787	69.018	1	5.18	72.787*	0	17.06
	50	7×3	140.996	139.985	1	0.72	140.996	53.36	3600
	100	7×3	302.292	301.281	1	0.33	302.292	81.42	3600
	200	15×2	576.768	570.504	13	1.09	576.768	83.53	3600
3	10	4×2	35.122	33.043	1	5.92	35.122*	0	0.02
	25	6×2	51.678	48.738	1	5.69	51.678*	0	0.06
	50	7×3	103.726	100.676	1	2.94	103.415	42.18	3600
	100	7×3	194.676	191.762	4	1.50	192.774	70.68	3600
	200	15×2	416.163	394.562	1	5.19	400.433	76.75	3600
4	50	7×3	80.907	74.169	4	8.33	76.909*	0	414.81
	100	7×3	163.904	160.153	1	2.29	161.164	62.35	3600
	200	15×2	343.765	324.853	18	5.50	338.011	72.55	3600
8	100	7×3	118.536	117.342	1	1.01	118.361	44.41	2600
	200	15×2	208.352	168.981	3	18.90	170	44.68	3600

TABLE IV
SINGLE-MACHINE WORK CENTER PROBLEMS WITH WIDE TREE STRUCTURE: GENERAL CASES

Scenario			LR Results				CPLEX Results		
WC	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)	Time(s)
2	10	3×5	26.271*	26.271	1	0	26.271*	0	0.06
	25	4×7	69.018*	69.018	1	0	69.018	69.018	30.39
	50	4×7	140.996	139.985	1	0.72	140.996	67.921	3600
	100	4×8	302.292	301.281	1	0.33	302.292	85.79	3600
	200	5×10	570.503*	570.503	1	0	570.503	91.05	3600
3	10	3×5	18.876	17.865	1	5.36	18.876*	0	0.02
	25	4×7	48.738*	48.738	1	0	48.738*	0	326.53
	50	4×7	100.676*	100.676	1	0	100.676	49.26	3600
	100	4×8	192.774	191.762	1	0.52	192.774	74.94	3600
	200	5×10	395.104	394.562	1	0.14	395.479	87.77	3600
4	25	4×7	57.933	56.921	1	1.75	57.933*	0	3600
	50	4×7	74.935	74.169	1	1.02	74.169	26.63	3600
	100	4×8	163.490	160.153	2	2.04	163.490	72.905	3600
	200	5×10	326.925	324.853	2	0.63	326.925	87.01	3600
8	50	4×7	44.628	38.974	1	12.67	44.519*	0	2.64
	100	4×8	114.581	113.569	1	0.88	114.581	60.18	3600
	200	5×10	173.195	167.802	1	3.11	171.219	73.5	3600

gaps for most test results are within 6%. Meanwhile, the upper bounds from LR have better performance on test cases with a relatively small number of work centers. On the aspect of computation time, our approach managed to keep the computational time within a number of seconds. The test cases in this table represents extreme cases when the tree is very narrow and our approach is still able to find good solutions very quickly.

Tables IV and V are for test cases with limits on the number of children larger than the number of levels, which leads to wide tree structures. In Table IV, the test cases scale from small prob-

TABLE V
SINGLE-MACHINE WORK CENTER PROBLEMS WITH WIDE TREE STRUCTURE: EXTREME CASES

Scenario			LR Results				CPLEX Results		
WC	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)	Time(s)
2	100	3×15	302.292	301.281	1	0.33	302.292	87.97	3600
	150	3×25	421.294*	421.294	1	0	421.294	92	3600
	200	4×15	570.500*	570.500	1	0	570.500	92.46	3600
	250	4×25	701.763*	701.763	1	0	701.763	93.47	3600
	300	5×15	844.010*	844.010	2	0	844.010	94	3600
4	100	3×15	161.164	160.153	2	0.63	161.164	76.7	3600
	150	3×25	241.909*	241.909	1	0	241.909	86.1	3600
	200	4×15	324.853*	324.853	1	0	324.853	87.29	3600
	250	4×25	368.828*	368.828	2	0	390.457	90.94	3600
	300	5×15	461.373*	461.373	2	0	461.373	89.53	3600
8	100	3×15	90.573	88.303	1	2.51	90.51	44.38	3600
	150	3×25	137.109	136.098	2	0.74	137.109	75.01	3600
	200	4×15	167.802*	167.802	1	0	167.802	76.54	3600
	250	4×25	188.209	186.256	1	1.04	187.267	73.43	3600
	300	5×15	252.178	249.095	1	1.22	251.185	80.84	3600

TABLE VI
PARALLEL-MACHINE WORK CENTER PROBLEMS WITH NARROW TREE: GENERAL CASES

Scenario				LR Results				CPLEX Results(1 hr)	
WC	m	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)
2	2	25	4×3	35.230	34.509	1	2.05	35.018	15.78
	2	50	4×4	73.272	69.992	1	4.48	72.041	61.76
	2	100	5×4	154.924	150.640	2	2.76	154.407	77.47
	2	200	5×4	286.886	285.252	4	0.57	288.727	90.01
	3	50	4×4	45.683	43.107	1	5.64	44.968	49.73
	3	100	5×4	107.065	103.707	2	3.14	105.811	71.70
	3	200	5×4	196.754	190.168	4	3.35	210.099	86.27
3	2	50	4×4	53.785	50.338	1	6.41	52.915	38.03
	2	100	5×4	103.685	95.881	2	7.53	101.391	65.68
	2	200	5×4	199.490	197.281	6	1.11	202.494	85.68
	3	50	4×4	39.257	33.559	1	14.52	36.579	27.28
	3	100	5×4	74.706	63.921	3	14.44	70.691	50.78
	3	200	5×4	134.110	131.521	4	1.93	137.881	79.08
4	2	50	4×4	41.938	37.084	1	11.57	39.127	23.69
	2	100	5×4	86.406	80.076	4	7.33	85.111	59.12
	2	200	5×4	168.200	162.427	4	3.43	167.950	82.83

lems to very large problems. The LR-based approach works very fast with computational time less than 2 seconds. Meanwhile, the LR-based approach manages to find better or same upper bounds as CPLEX in most test cases.

In Table V, test cases with wide tree structure and large number of operations are used. As we can see from the table, the LR-based approach is still able to find solutions as good as or better than CPLEX in most cases. The computational effort does not change, as the time for all test cases are within 2 seconds. The solution gaps from the LR-based approach are very tight, since about half of the test cases achieve optimality and the others have a gap less than 3%.

Tables VI and VII contain the results of test cases where work centers have identical parallel machines. The layouts of Tables VI and VII are different in describing the scenarios and test cases for work centers with parallel machines. A new column "m" is added to denote the machine number in each work center. For example, "2 + 2" in the "m" column means 2 machines for each of the 2 work centers. The column for computational time of CPLEX is removed because CPLEX is not able to find optimal solutions within an hour in all test cases.

As shown in Tables II–V, the LR-based approach tends to have better performance on a wide tree with a large number of operations and a small number of work centers. As seen from Table VI, the performance of the LR-based approach gets better or the same results as CPLEX on cases where the total number of machines is less than 4 or the number of operations is 200. This phenomenon is because the randomized list scheduling will need more iterations to search for potentially better combinations when the number of machines gets larger, and CPLEX's computational efforts become larger when the number of operations gets larger. As to the aspect of gap and computational time, the LR-based approach is still able to obtain small gaps (less than 8%) on most of the test cases.

Since we are mostly interested in the results of large problems, a set of test cases are set up in Table VII. All the test cases in this table are with a large number of operations (200) and a large number of machines (4–9). The BOM structure is designated to be "9 × 4" and "4 × 9" to examine the performance difference of the LR-based approach applied on the structure of narrow trees and wide trees. From the results in Table VII, the computational time are within 10 seconds and gaps within 6% for most cases, which illustrates the capability of the LR-based approach on problems with large sizes. As shown in the table, narrow tree structures and a large number of machines still affect the upper bound quality for the LR-based approach when

TABLE VII
PARALLEL-MACHINE WORK CENTER PROBLEMS: LARGE SIZE PROBLEMS

Scenario				LR Results				CPLEX Results(1 hr)	
WC	m	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)
2	2	200	9×4	293.648	285.252	5	2.86	295.946	81.18
	2	200	4×9	286.916	286.316	4	0.21	306.339	90.29
	3	200	9×4	201.401	190.168	5	5.58	195.033	71.44
	3	200	4×9	193.061	190.921	6	1.11	256.548	89.89
3	2	200	9×4	216.357	196.166	9	9.33	208.035	71.82
	2	200	4×9	198.916	197.542	4	0.69	205.501	86.21
	3	200	9×4	156.563	131.688	5	15.89	141.566	60.08
	3	200	4×9	149.113	147.001	6	1.42	149.805	82.06
4	2	200	9×4	167.164	146.961	9	12.09	157.062	61.88
	2	200	4×9	172.073	171.502	6	0.33	172.217	83.95
2	2	300	9×4	422.270	405.257	20	4.03	444.671	58.03
	2	300	4×9	421.896	421.300	5	0.14	639.612	95.48
	3	300	9×4	341.917	325.350	5	4.85	471.490	88.41
	3	300	4×9	277.291	273.603	3	1.33	411.206	93.81
3	2	300	9×4	339.575	315.768	5	7.01	361.173	81.38
	2	300	4×9	304.865	302.104	3	0.91	305.710	91.66
	3	300	9×4	224.964	213.375	7	5.15	306.35	78.46
	3	300	4×9	215.637	213.370	5	1.05	299.145	91.48
4	2	300	9×4	248.191	241.070	21	2.87	253.52	80.19
	2	300	4×9	251.606	250.525	3	0.43	254.041	88.79

TABLE VIII
MIXED WORK CENTER PROBLEMS: SINGLE AND PARALLEL MACHINES

Scenario				LR Results				CPLEX Results(1 hr)	
WC	m	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)
2	1+2	200	5×4	516.308	515.297	2	0.20	516.308	92.56
	1+3	200	5×4	541.123	540.111	2	0.19	541.123	92.06
	1+4	200	5×4	582.346	582.246	2	0.02	582.346	91.71
	1+5	200	5×4	573.777	572.763	2	0.18	573.777	91.51
	1+6	200	5×4	540.867	539.852	2	0.19	540.867	91.71
3	1+1+2	200	5×4	346.367	345.347	3	0.29	346.367	87.61
	1+2+2	200	5×4	364.986	359.977	2	1.37	364.986	87.24
	1+1+3	200	5×4	365.446	363.250	3	0.60	366.236	84.66
	1+2+3	200	5×4	332.059	331.036	2	0.31	332.059	85.04
	2+2+3	200	5×4	237.425	236.884	2	0.23	237.418	84.63
4	1+1+1+2	200	5×4	338.943*	338.943	2	0	338.943	85.80
	1+1+2+2	200	5×4	321.915	320.885	2	0.32	321.915	84.73
	1+2+2+2	200	5×4	224.954	221.239	2	1.65	225.595	79.04
	1+2+2+3	200	5×4	380.751*	380.751	2	0	380.751	83.93
	1+1+2+3	200	5×4	270.166	266.270	2	1.44	270.166	83.20

the number of operations is 200. When the number of operations is 300, our approach is always able to generate better upper bound than that obtained from running CPLEX for an hour.

Another set of experiments are also performed for work centers with a different number of machines. The results are shown in Tables VIII and IX. In Table VIII, the number of operations is 200, the BOM is 5×4 to represent a normal tree, the machine numbers are not the same in all work centers. The results show that the LR-based approach always gets the same or better upper bounds and lower bounds than CPLEX. At the same time, the computation time is still less than 4 seconds for all test cases. In Table IX, the number of operations is 300, the BOM is 9×4 and 4×9 to represent narrow trees and wide trees, the machine numbers are increased and all work centers have multiple machines in the same scenarios. As shown in Table IX, the LR-based approach has even more significant advantages in the aspect of upper bounds when compared with CPLEX. The lower bounds for the LR-based approach are still better and the computational time is still small.

C. Comparison With Shifting Bottleneck Procedure

The shifting bottleneck procedure has been proved efficient with various types of objective functions when implemented in a manufacturing environment where each work center has one machine [11]. When there are parallel machines, a most recent attempt is made by [12]. A typical shifting bottleneck approach as described in [11] has several major steps.

1) Select an unscheduled work center and generate single work center subproblems, which are either single machine scheduling problems or parallel machine scheduling problems with precedence constraints and release time to minimize maximum lateness $L_{\max}$.
2) Solve the single work center subproblem, either by branch and bound or heuristics, to obtain partial schedule of corresponding work center.
3) Re-optimize all partial schedules obtained so far.
4) Either repeat from Step 1 if there are still unscheduled work centers or stop if all work centers are scheduled.

TABLE IX
MIXED WORK CENTER PROBLEMS: SINGLE AND PARALLEL MACHINES

Scenario				LR Results				CPLEX Results(1 hr)	
WC	m	n	BOM	UB	LB	Time(s)	Gap(%)	UB	Gap(%)
2	2+3	300	9×4	368.010	362.304	4	1.55	382.190	83.35
	2+4	300	4×9	409.512	408.465	3	0.26	416.760	93.06
	2+5	300	9×4	445.849	442.711	13	0.70	457.310	89.57
	3+4	300	4×9	275.451	271.654	4	1.38	411.189	92.88
	3+5	300	9×4	334.587	321.788	16	3.83	446.558	88.25
3	1+1+4	300	9×4	499.971	494.608	7	1.07	499.971	87.07
	1+2+4	300	4×9	397.088	396.062	2	0.26	474.308	91.86
	2+3+4	300	9×4	233.035	215.661	14	7.46	235.950	74.66
	3+3+4	300	4×9	178.661	177.387	9	0.71	248.518	88.23
	2+2+5	300	9×4	337.644	328.623	10	2.67	342.675	84.13
4	1+1+1+3	300	9×4	414.656	413.622	5	0.25	414.656	82.74
	1+1+2+4	300	4×9	372.322*	372.322	3	0	372.322	415.01
	1+2+3+4	300	9×4	415.010	408.773	3	1.5	415.010	84.31
	2+2+3+3	300	4×9	194.247	193.081	4	0.6	199.322	86.84
	2+2+2+3	300	9×4	222.247	201.622	16	9.28	225.244	72.59

TABLE X
COMPARISON BETWEEN LR AND SB WITH SINGLE MACHINE WORK CENTERS

Scenario				LR Results			SB(EDD) Results		SB(CPLEX) Results	
WC	m	n	BOM	UB	LB	Time(s)	UB	Time(s)	UB	Time(s)
2	1	20	4×4	60.885	60.269	1	60.879	1	60.951	73
	1	20	4×9	62.468	61.365	1	64.157	1	62.2316	499
4	1	80	4×9	138.230	136.730	1	141.790	1	-	-
	1	80	9×4	143.580	134.540	2	165.970	1	-	-
	1	300	4×9	463.300	461.980	6	463.300	4	-	-
	1	300	9×4	480.580	461.430	12	486.250	6	-	-
10	1	300	4×9	211.100	208.050	9	333.030	22	-	-
	1	300	9×4	237.920	209.740	13	383.000	26	-	-

Tables X–XII show the comparison between our approach and the shifting bottleneck procedure. A number of test cases are generated randomly to represent a number of different problem structures. SB(EDD) is shifting bottleneck procedure that uses the early due date (EDD) dispatching rule to solve the subproblems according to [12]. SB(CPLEX) uses CPLEX to solve subproblems to optimality to represent the shifting bottleneck procedure suggested by [11], where subproblems are solved to optimality using branch and bound algorithm. In the experiments in [12], SB(EDD) generates the best solutions among all versions of shifting bottleneck procedure tested. Ten random test cases are generated for each scenario of the work center, machine number, operation number, and BOM structure. The values of UB, LB, and Time in each row are average values of the ten experiments under the same scenario.

Table X contains the results for scenarios where each work center only has one machine. When problem size is small, the LR-based approach and shifting bottle neck procedure have similar performance. SB(CPLEX) needs a significant amount of computational time due to the requirement of optimal solutions to the NP-hard subproblems. LR, SB(EDD) and SB(CPLEX) all achieve near optimal solutions. As the size of problems gets larger, SB(CPLEX) is not able to generate solutions within acceptable computational time due to the difficulty on solving NP-hard subproblems. Both LR and SB(EDD) have larger optimality gaps. LR has better performance when compared with SB(EDD). When the number of work centers gets to a large number of ten, SB(EDD) gets much worse results than those obtained using LR.

TABLE XI
COMPARISON BETWEEN LR AND SB WITH PARALLEL MACHINE WORK CENTERS

Scenario				LR Results			SB(EDD) Results	
WC	m	n	BOM	UB	LB	Time(s)	UB	Time(s)
2	2	50	4×4	76.562	75.226	1	85.227	1
	2	50	4×9	75.750	74.033	1	85.918	1
	4	100	6×6	82.857	71.698	3	129.296	1
	8	300	4×9	114.598	107.093	10	211.601	1
	8	300	9×4	135.525	107.382	10	318.095	1
4	2	100	6×6	97.092	92.230	2	176.149	2
10	2	300	4×9	115.866	110.900	9	338.195	17
	2	300	9×4	135.302	111.690	10	408.982	20

In Table XI, LR and SB(EDD) are compared using test cases where all work centers have parallel machines. It is observed that SB(EDD) is able to generate similar results as LR when the problem size is small. When problem size is large, LR has much better performance than SB(EDD). At the same time, we can also observe that LR requires more computational time before convergence as the number of operations increases. Meanwhile, SB(EDD) requires more computation time when the number of work centers gets larger.

In Table XII, the results are based on test cases where work centers do not have the same number of machines as each other. The results still show that SB(EDD) is able to compete with LR when the problem size is small, especially when there are fewer parallel machine work centers. When the size of the problems gets larger, the advantage of using LR becomes very significant.

TABLE XII
COMPARISON BETWEEN LR AND SB WITH MIXED WORK CENTERS

Scenario				LR Results			SB(EDD) Results	
WC	m	n	BOM	UB	LB	Time(s)	UB	Time(s)
2	1+2	30	4×4	80.738	79.412	1	80.738	1
	1+3	50	5×5	130.719	129.908	1	130.719	1
	2+3	80	6×6	109.968	106.184	1	178.232	1
3	1+1+2	80	6×6	163.403	160.403	1	214.851	1
4	1+2+2+3	200	4×9	290.958	289.851	3	315.631	2
	1+2+2+3	200	9×4	259.143	254.199	4	337.463	3
	2+3+4+5	300	4×9	235.615	233.246	8	412.141	3
	2+3+4+5	300	9×4	220.476	208.145	8	494.391	3

V. CONCLUSION AND FUTURE WORK

This paper proposes a LR-based approach to solve an assembly scheduling problem with a number of machines. The problem is decomposed into work center level subproblems and the subproblems are solved using heuristic algorithms. The lower bounds are generated by static analysis of subproblems and by dynamic analysis of subproblems during the subgradient search procedure. The upper bounds are generated via a randomized list scheduling algorithm using solutions of a LR problem. The results show that our approach is able to solve problems efficiently with small optimality gaps. Narrow tree structure and a large number of machines would affect the performance of the approach, but the impact is not significant. The approach we proposed has better performance for a large size problem when compared with shifting bottleneck procedure. This approach is perfectly suitable for solving large-scale assembly scheduling problems when fast and near-optimal solutions are desired.

The LR-based approach can be treated as a framework for other related problems. Since the subproblems are $P_m|r_j, prec|\sum_j w_j C_j + C_{\max}$, a general job shop scheduling problems with precedence constraints and release time to minimize weighted completion time and/or makespan would have the same type of subproblems when applying relaxation and decomposition schema as in this paper. Thus, a natural extension of this paper is to implement this approach to general job shop scheduling problems.

ACKNOWLEDGMENT

The authors would like to acknowledge the editor, associate editor, and anonymous referees whose comments have led to the improvement of this paper. Their advice is gratefully appreciated.

REFERENCES

[1] A. Agrawal, G. Harhalakis, I. Minis, and R. Nagi, "'Just-in-time' production of large assemblies," *IIE Trans.*, vol. 28, no. 8, pp. 653–667, 1996.
[2] J. Carlier and E. Pinson, "An algorithm for solving the job-shop problem," *Manage. Sci.*, vol. 35, no. 2, pp. 164–176, 1989.
[3] P. Brucker, B. Jurisch, and B. Sievers, "A branch and bound algorithm for the job-shop scheduling problem," *Discrete Appl. Math.*, vol. 49, no. 1–3, pp. 107–127, 1994.
[4] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Manage. Sci.*, vol. 34, no. 3, pp. 391–401, 1988.
[5] C. Y. Zhang, P. Li, Y. Rao, and Z. Guan, "A very fast TS/SA algorithm for the job shop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 1, pp. 282–294, 2008.
[6] S. M. K. Hasan, R. Sarker, and D. Essam, "Memetic algorithms for solving job-shop scheduling problems," *Memetic Comput.*, vol. 1, no. 1, pp. 69–83, 2008.
[7] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Syst. With Appl.*, vol. 38, no. 4, pp. 3563–3573, 2011.
[8] M. Yin, X. Li, and J. Zhou, "An efficient job shop scheduling algorithm based on artificial bee colony," *Sci. Res. Essays*, vol. 5, no. 24, pp. 2578–2596, 2011.
[9] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," *Expert Syst. With Appl.*, vol. 37, no. 1, pp. 678–687, 2010.
[10] T. Lin, S. Horng, T. Kao, Y. Chen, R. Run, R. Chen, J. Lai, and I. Kuo, "An efficient job-shop scheduling algorithm based on particle swarm optimization," *Expert Syst. With Appl.*, vol. 37, no. 3, pp. 2629–2636, 2010.
[11] I. M. Ovacik and R. Uzsoy, *Decomposition Methods for Complex Factory Scheduling Problems*, 1st ed. New York, NY, USA: Springer, 1996.
[12] K.-P. Chen, M. S. Lee, P. S. Pulat, and S. A. Moses, "The shifting bottleneck procedure for job-shops with parallel machines," *Int. J. Ind. Syst. Eng.*, vol. 1, no. 1/2, pp. 244–262, 2006.
[13] H. Hu and Z. Li, "Modeling and scheduling for manufacturing grid workflows using timed petri nets," *Int. J. Adv. Manuf. Technol.*, vol. 42, no. 5–6, pp. 553–568, 2009.
[14] H. Hu and Z. Li, "Synthesis of liveness enforcing supervisor for automated manufacturing systems using insufficiently marked siphons," *J. Int. Manuf.*, vol. 21, no. 4, pp. 555–567, 2010.
[15] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. New York, NY, USA: Springer, 2008.
[16] Y. Mati, S. Dauzere-Peres, and C. Lahlou, "A general approach for optimizing regular criteria in the job-shop scheduling problem," *Eur. J. Oper. Res.*, vol. 212, no. 1, pp. 33–42, 2011.
[17] M. L. Fisher, "Optimal solution of scheduling problems using Lagrange multipliers, Part (i)," *Oper. Res.*, vol. 21, no. 5, pp. 646–653, 1973.
[18] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "Scheduling jobs with simple precedence constraints on parallel machines," *IEEE Control Syst. Mag.*, vol. 10, no. 2, pp. 34–40, Feb. 1990.
[19] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job shop scheduling problems," *IEEE Trans. Robot. Autom.*, vol. 9, no. 1, pp. 1–13, Feb. 1993.
[20] C. S. Czerwinski and P. B. Luh, "Scheduling products with bills of materials using an improved Lagrangian relaxation technique," *IEEE Trans. Robot. Autom.*, vol. 10, no. 2, pp. 99–111, Apr. 1994.
[21] C. A. Kaskavelis and M. C. Caramani, "Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems," *IIE Trans.*, vol. 30, no. 11, pp. 1085–1097, 1998.
[22] T. Nishi, Y. Hiranaka, and M. Inuiguchi, "Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness," *Comput. Oper. Res.*, vol. 37, no. 1, pp. 189–198, 2010.
[23] X. Zhao, P. B. Luh, and J. Wang, "Surrogate gradient algorithm for Lagrangian relaxation," *J. Optim. Theory Appl.*, vol. 100, no. 3, pp. 699–712, 1999.
[24] H. Chen and P. B. Luh, "An alternative framework to Lagrangian relaxation approach for job shop scheduling," *Eur. J. Oper. Res.*, vol. 149, no. 3, pp. 499–512, 2003.
[25] P. Baptiste, M. Flamini, and F. Sourd, "Lagrangian bounds for just-intime job-shop scheduling," *Comput. Oper. Res.*, vol. 35, no. 3, pp. 906–915, 2008.

[26] T. S. Chang, "Comments on surrogate gradient algorithm for Lagrangian relaxation," *J. Optim. Theory Appl.*, vol. 137, no. 3, pp. 691–697, 2008.
[27] J. K. Lenstra and P. Brucker, "Complexity of machine scheduling problems," *Ann. Discrete Math.*, vol. 1, no. 1, pp. 343–362, 1977.
[28] H. Belouadah, M. E. Posner, and C. N. Potts, "Scheduling with release dates on a single machine to minimize total weighted completion time," *Discrete Appl. Math.*, vol. 36, no. 3, pp. 213–231, 1992.
[29] L. Tang, H. Xuan, and J. Liu, "A new Lagrangian relxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time," *Comput. Oper. Res.*, vol. 33, no. 11, pp. 3344–3359, 2006.
[30] R. L. Rardin, *Optimization in Operations Research*, 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1998.
[31] J. Carlier, "Scheduling jobs with release dates and tails on identical machines to minimize the makespan," *Eur. J. Oper. Res.*, vol. 29, no. 3, pp. 298–306, 1987.

Jingyang Xu received the B.S. degree in aerospace engineering from Beihang University (BUAA), Beijing, China, in 2006 and the Ph.D. degree in industrial engineering from the University at Buffalo (SUNY), Buffalo, NY, USA, in 2010.

He is a Senior Decision Science Consultant at The Walt Disney Company. His major research interests are in discrete optimization with application in scheduling and optimal control. His recent interests also include HVAC system optimization, and TV advertisement pricing.

Rakesh Nagi received the B.E. degree in mechanical engineering from the University of Roorkee (now IIT-R), Roorkee, in 1987 and the M.S. and Ph.D. degrees in mechanical engineering from the University of Maryland, College Park, MD, USA, in 1989 and 1991, respectively, while he worked at the Institute for Systems Research and INRIA, France.

He is a Professor of Industrial and Systems Engineering, University at Buffalo (SUNY), Buffalo, NY, USA. His has more than 70 journal publications and many conference presentations. His major research thrust is in the area of production systems and applied/military operations research. His research interests are in location theoretic approaches to facilities design, agile enterprises and information-based manufacturing, and just-in-time production of assemblies. His recent interest include sensor networks and Level2/Level3 data fusion using graph theoretic models.

Dr. Nagi is a Fellow of the Institute of Industrial Engineers (IIE).