A Macro-Level Scheduling Method Using Lagrangian Relaxation

Yuanhui Zhang, Peter B. Luh, *Fellow, IEEE*, Katsumi Narimatsu, Tetsuro Moriya, Tsuyoshi Shimada, and Lei Fang

***Abstract*—In this paper, a macro-level scheduling method is developed to provide high-level planning support for factories with multiple coordinating cells. The key challenges are large problem sizes, complicated product process plans, stringent cell coordination requirements, and possible resource overload. To model the problem with manageable complexity, detailed operations of a product within a cell are aggregated as a single operation whose processing time is related to the amount of resources allocated. "Overload variables" are introduced and penalized in the objective function. The goal is to properly allocate resources, efficiently handle complicated process plans, and coordinate cells to ensure on-time delivery, low working-in-process inventory, and small resource overload. The formulation obtained is "separable" and can be effectively decomposed by using Lagrangian relaxation. A combined backward and forward dynamic programming (BFDP) method is developed to solve a product subproblem after a novel transformation of its process plan. The BFDP is further simplified and solved approximately following the idea of the "surrogate subgradient method" to reduce the computation requirements for large problems. Numerical results show that near-optimal schedules can be obtained for problems with up to 50 000 operations within a reasonable amount of computation time.**

***Index Terms*—Lagrangian relaxation, manufacturing planning and scheduling, optimization.**

I. Nomenclature

a_i	Earliest beginning time of product i;
b_{ij}	Beginning time of operation (i, j);
b_{ij}^E	Earliest beginning time of operation (i, j);
b_{ij}^L	Latest beginning time of operation (i, j);
c_{ij}	Completion time of operation (i, j);
c_{ij}^E	Earliest completion of operation (i, j);
c_{ij}^L	Latest completion of operation (i, j);
d_i	Due date of product i;
h	Index of resource type from 1 to H;
H	Total number of resource types;
H_{ij}	A set of resources required simultaneously for the processing operation (i, j);
i	Index of product from 1 to I;
I	Total number of products;
I_{ij}	The set of successor operations of operation (i, j);
J_i	Number of operations in product i;
k	Index of time unit from 1 to K;
K	Total number of time units within the time horizon;
M_{kh}	Capacity of resource h at time unit k;
P_{ijh}	The total "resource hours" of a particular resource type h required by operation (i, j);
r_{kh}	Resource overload of resource h at time unit k;
t_{ij}	Processing time of operation (i, j);
π_{kh}	Lagrangian multiplier for resource capacity constraint of type h at time unit k;
δ_{ijkh}	Resource utilization of $h\,(h \in H_{ij})$ for operation (i, j) at time unit k;

Manuscript received November 18, 1999; revised August 24, 2000. This paper was recommended for publication by Associate Editor Y. Narahari and Editor N. Viswanadham upon evaluation of the reviewers' comments. This work was supported in part by Toshiba Corporation and the National Science Foundation under Grant DMI-9813176. The paper was presented in part at the 1999 IEEE International Conference on Systems, Man, and Cybernetics, Tokyo, Japan, October 1999.

Y. Zhang is with Oracle Corporation, Redwood Shores, CA 94065 USA (e-mail: dzhang@us.oracle.com).

P. B. Luh and L. Fang are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-2157 USA (e-mail: luh@engr.uconn.edu; fang@engr.uconn.edu).

K. Narimatsu and T. Moriya are with the Systems Engineering Laboratory, Research and Development Center, Toshiba Corporation, Kawasaki-Shi 212-8582, Japan (e-mail: katsumi.narimatsu@toshiba.co.jp; tetsuro.moriya@toshiba.co.jp).

T. Shimada is with the System Integration Technology Center, Toshiba Corporation, Tokyo 183-8512, Japan (e-mail: tsuyoshi.shimada@toshiba.co.jp).

Publisher Item Identifier S 1042-296X(01)03157-3.

II. Introduction

Many manufacturing systems are organized in cells, and products flow across cells for processing. Macro-level scheduling is to provide high-level planning support to decide when a product should be processed at which cell, and the amount of cell resources to be allocated for the processing. Effective scheduling is critical to improve on-time delivery, reduce inventory, cut lead times, and level resource utilization. Unlike cell-level scheduling, macro-level scheduling emphasizes the smooth flow of products across cells and the overall factory performance. Because of large problem sizes, complicated product process plans, stringent cell coordination requirements, and possible resource overload, it is very difficult to obtain good schedules within a reasonable amount of computation time.

Literature Review: In a conventional material requirement planning (MRP) system as described in [16], macro-level scheduling is performed by modules such as master production scheduling and rough-cut capacity planning. Material availability is the focus, and resource capacity is usually assumed to be infinite resulting in infeasible schedules and large resource overload. Moreover, several important issues such as product routing are missing in rough-cut capacity planning as discussed in [18]. There have been many efforts to improve MRP. A load leveling method using integer programming was developed in [1] to help human schedulers decide whether to increase capacity or to delay the production of some products. A linear programming model was constructed to seek the most profitable schedule subject to capacity constraints in [6]. A rule-based framework for master scheduling considering both material availability and resource capacity was presented in [17], and a knowledge-based system was described in [20]. However, in today's time-based and customer-oriented competition and the resulting low-volume and high-variety manufacturing, MRP-based methods usually have major difficulties to meet the important goals of on-time delivery and low working-in-process (WIP) inventory because of their material centric nature. A typical scenario is that products finished are not the ones immediately needed, and the system has an excessive amount of WIP inventory.

To improve the situation, job shop scheduling methods for on-time delivery and low WIP inventory were developed in [15] and [14] using Lagrangian relaxation. The goal of this paper is to extend the methods to the macro level, addressing many of the practical issues such as complicated process plans, simultaneous and partial resource utilization, and coordination across cells. In the literature dealing with complicated process plans, the components for final assemblies were scheduled via a combined sequencing and dynamic programming procedure so that the mean completion time is minimized in [3]. Assembly scheduling was formulated as a mixed integer linear programming problem in [8], and a heuristic approach was developed. In view of the complexity involved, research addressing complicated process plans with coordination requirements has been very limited. As for resource utilization, the simultaneous usage of multiple resources was formulated as an integer programming problem and solved by Lagrangian relaxation in [9] and [6]. In this paper, an operation may simultaneously require multiple resources and the utilization of the resources can be fractional. In addition, resource overload, which captures overtime and reserved resources, is allowed if deemed necessary. The key challenges are to model the problem with manageable complexity, to properly allocate resources, to efficiently handle complicated process plans, and to smoothly coordinate operations across cells. In view of the inherent complexity and the largeness of the problems, the aim here is to develop a near-optimal scheduling methodology with quantifiable quality in a computationally efficient manner.

Overview of the Paper: In the formulation to be presented in Section III, detailed operations of a product within a cell are aggregated as a single operation on key resources with bills of materials provided by MRP. An operation may simultaneously require multiple resources and the utilization of the resources can be fractional. Small resource overload is allowed in view of the aggregation, reserved resources, and possible overtime. The limited resource capacities are explicitly formulated, and the lead-time of a product within a cell or the processing time of an aggregated operation is flexible and related to the amount of resources allocated. The objective is to minimize a weighted sum of penalties on product tardiness, earliness, lead-time, and resource overload.

In Section IV, a solution methodology based on Lagrangian relaxation is developed. A combined backward and forward dynamic programming (BFDP) method is established to solve a product subproblem involving assembly (fan-in) and fan-out (an operation with multiple successors) after a novel transformation of its process plan. This approach results in a smaller number of multipliers, faster algorithm convergence, and better solution quality than further decomposing the subproblem through relaxation as in [23]. The part subproblem is then simplified and approximately solved following the idea of the "surrogate subgradient method" in [21] to reduce the computation requirements for large problems, and the coordination requirements across cells are explicitly considered within the dynamic programming (DP) context. Numerical results in Section V show that near-optimal schedules are obtained for large problems with up to 50 000 operations within a reasonable amount of computation time.

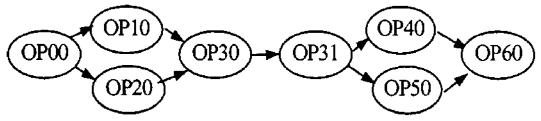

Fig. 1. A sample process plan.

III. Problem Formulation

The formulation presented in this section is built on our previous work on job shop scheduling [15], [14] with the following new features: flexible resource utilization, resource overload, and complicated process plans.

A. General Description

The scheduling time horizon is divided into K discrete time units, with index k ranging from 0 to K–1. There are H resource types, each consisting of a set of identical machines. The number of type h resources $(1 \leq h \leq H)$ at time k is given and denoted as M_{kh}. There are I products to be scheduled, and Product i $(1 \leq i \leq I)$ has a given earliest beginning time a_i (arrival time of raw materials) and a due date d_i. It has to go through multiple cells according to a specified process plan, and the processing at each cell is treated as an aggregated operation. Product i has a total of J_i operations, and the jth operation is denoted as (i, j). Complicated process structures such as assembly (fan-in) and fan-out (an operation with multiple successors) may exist as shown in Fig. 1. In the figure, Operation 00 fans out Operations 10 and 20, which can be processed in parallel. Operation 30 is an assembly operation, and can only be started after the completion of both Operations 10 and 20. When this process plan is seen as an undirected graph, Operations 00, 10, 20, and 30 form a cycle, which will require additional relaxation to be presented in Section IV. Without loss of generality, it is assumed that a process plan starts with a single node and ends with a single node.

In view of the aggregation of detailed operations in a cell and other factors such as reserved resources and possible overtime, small resource overload is permitted and resource capacities can be slightly violated. A nonnegative *resource overload variable* r_{kh} is introduced for each resource type at each time unit, representing the level of overload. It is then penalized in the objective function. In this paper, r_{kh} takes fractional and quantized values.

Resource Utilization Requirements: An aggregated operation can be performed by several alternative resource sets in a cell, and each possible set H_{ij} includes multiple resource types simultaneously required by the operation. The total "resource hours" for a particular resource type $h \in H_{ij}$ is given and denoted as P_{ijh}. The processing time required then depends on the amount of resource allocation, and a large amount often leads to a short time. Without loss of generality, an operation's processing time is assumed to be inversely proportional to the amount of resource allocated. This modeling has the flexibility to achieve various production speeds with different levels of resource allocation. For simplicity, the processing time of an operation can only take several discrete values. An equivalent viewpoint is that the resource utilization by an operation, denoted as δ_{ijkh}, is inversely proportional to its processing time t_{ij} within its processing period, and zero outside, i.e.,

$$\delta_{ijkh} = \begin{cases} \dfrac{P_{ijh}}{t_{ij}}, & \text{if } (i, j) \text{ is assigned to resource type } h \\ & \quad \text{and } b_{ij} \le k \le c_{ij} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

B. Modeling of Constraints

Resource Capacity Constraints: The resource capacity constraints require that the total utilization of a resource should be less than or equal to the amount of available resource plus possible overload r_{kh}, i.e.,

$$\sum_{i,j} \delta_{ijkh} \le M_{kh} + r_{kh} \qquad \forall k, h \tag{2}$$

where r_{kh} takes nonnegative fractional and quantized values if overload is deemed appropriate.

Operation Precedence Constraints: Given a product's process plan, the set of subsequent operations of (i, j) is denoted by I_{ij}. If $(i, l) \in I_{ij}$, (i, l) is a *successor* of (i, j), and (i, j) is a *predecessor* of (i, l). Operation precedence constraints state that the processing of an operation may start only after the completion of all its predecessors plus a possible required "timeout" S_{ijl}, i.e.,

$$c_{ij} + S_{ijl} + 1 \le b_{il}, \qquad (i, l) \in I_{ij}, \quad \forall i, j. \tag{3}$$

Arrival Time Constraints: As a special case of (3), the processing of a product cannot be started before the arrival of the required raw materials, i.e.,

$$a_i \le b_{i0} \qquad \forall i \tag{4}$$

where b_{i0} is the beginning time of the first operation of product i.

Processing Time Requirements: The completion time of an operation equals the beginning time plus the required processing time, i.e.,

$$c_{ij} = b_{ij} + t_{ij} - 1 \qquad \forall i, j \tag{5}$$

where b_{ij} and c_{ij} are the beginning and completion times of operation (i, j), respectively. Since the beginning time refers to the start of a discrete time unit, and completion time the end a discrete time unit, the term -1 is required in (5). Similar is the $+1$ in (3).

Cell Coordination Constraints: To coordinate across cells, an operation may be required to start and finish within a specified time window, i.e.,

$$b_{ij}^E \le b_{ij} \le b_{ij}^L \quad \text{and} \quad c_{ij}^E \le c_{ij} \le c_{ij}^L \qquad \forall i, j \tag{6}$$

where b_{ij}^E, b_{ij}^L, c_{ij}^E, and c_{ij}^L are the operation's earliest and latest beginning and completion times, respectively. If $b_{ij}^E = b_{ij}^L$ or $c_{ij}^E = c_{ij}^L$, the beginning or completion time of the operation is fixed. In addition, some consecutive operations may require "no wait" in between because of stringent physical constraints or coordination requirements. In this case, the processing of an operation must start immediately after the completion of its preceding operation, i.e.,

$$c_{ij} + 1 = b_{il}, \qquad (i, l) \in I_{ij}, \quad \forall i, j. \tag{7}$$

C. Objective Function

A good macroscheduler should achieve on-time delivery, low WIP inventory, and small resource overload. Inventory can be reduced by cutting short product lead-time, i.e., the time between the start of the first operation and the completion of its last operation. Similarly, short cell lead-times (or processing times) of aggregated operations will help reduce inventory within a cell. The objective function J is therefore the weighted sum of penalties on product tardiness, earliness, production/operation lead-times, and resource overload, i.e.,

$$J \equiv \sum_i \left[w_i T_i^2 + \beta_i E_i^2\right] + \sum_i \left[w_i^L (c_{i, J_i - 1} - b_{i0})\right] + \sum_{i,j} \left[w_{ij}^P (c_{ij} - b_{ij})\right] + \sum_h w_h^r \sum_k r_{kh}^2. \tag{8}$$

In the above, T_i is the product tardiness, i.e., $\max(0, c_i - d_i)$, with $c_i = c_{i, J_i - 1}$ being the completion time of product i. Earliness E_i is the amount of time that product begins earlier than the desired beginning time $\bar{b}_i$, i.e., $\max(0, \bar{b}_i - b_i)$, with $b_i = b_{i0}$ being the beginning time of product i. The second and third term in (8) penalize long product and operation lead-times, respectively. The last term is the cost of resource overload. Parameters w_i, β_i, w_i^L, w_{ij}^P, and w_h^r are weights associated with those penalty terms.

D. The Overall Problem

The overall problem is to minimize the above objective function subject to resource capacity, operation precedence, and cell coordination constraints. Key decision variables are operation beginning times, the type and amount of resources to be allocated, and resource overload required. Among the constraints, resource capacities are "coupling constraints" as they couple together decision variables belonging to different products. Since the objective function and coupling constraints are additive, the problem formulation is "separable," and Lagrangian relaxation can be effectively applied.

IV. Solution Methodology

Similar to the pricing concept of a market economy, Lagrangian relaxation replaces coupling resource capacity constraints by the payment of certain prices (i.e., Lagrange mul-

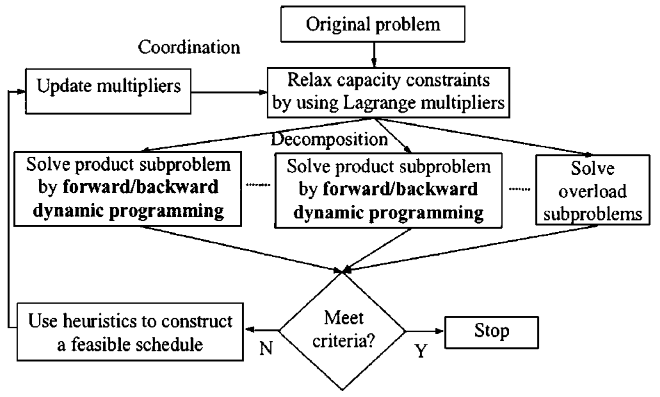
Original problem
Coordination
Update multipliers
Relax capacity constraints by using Lagrange multipliers
Decomposition
Solve product subproblem by forward/backward dynamic programming
Solve product subproblem by forward/backward dynamic programming
Solve overload subproblems
Meet criteria?
Use heuristics to construct a feasible schedule
N
Y
Stop

Fig. 2. The Lagrangian relaxation framework.

tipliers) for the use of resources at each time unit. The "relaxed problem" can then be decomposed into smaller subproblems. These subproblems are much easier to solve as compared to the original problem, and solutions can be efficiently obtained by using dynamic programming. The prices or multipliers are then iteratively adjusted based on the degrees of constraint violation following again the market economy principle (i.e., increase prices for over-utilized time units and reduce prices for under-utilized time units), while subproblems are resolved based on the new set of multipliers. In mathematical terms, a "dual function" is maximized in the multiplier updating process, and values of the dual function serve as lower bounds to the optimal feasible cost. At the termination of such price updating iterations, a few capacity constraints may still be violated. Simple heuristics are then applied to adjust subproblem solutions to form a feasible schedule satisfying all constraints. Heuristics can also be run after selected optimization iterations to check convergence or to generate candidate feasible schedules. Optimization and heuristics thus operate in a synergistic fashion, and quality of schedules can be quantitatively evaluated by comparing their costs to the largest lower bound obtained. The decomposition and coordination framework is illustrated in Fig. 2. The above approach has been successfully used to schedule job shops in [15] and [14]. The special features of flexible resource utilization, resource overload, and complicated process plans lead to new challenges in the formation and resolution of subproblems.

A. Lagrangian Relaxation

Resource capacity constraints are first "relaxed" by using Lagrange multipliers $\{\pi_{kh}\}$, and the Lagrangian is formed as

$$\begin{aligned} L \equiv & \sum_i \left[w_i T_i^2 + \beta_i E_i^2\right] + \sum_i \left[w_i^L (c_{i,J_i-1} - b_{i0})\right] \\ & + \sum_{i,j} \left[w_{ij}^P (c_{ij} - b_{ij})\right] + \sum_h w_h^r \sum_k r_{kh}^2 \\ & + \sum_{k,h} \left[\pi_{kh} \left(\sum_{i,j} \delta_{ijkh} - M_{kh} - r_{kh}\right)\right]. \end{aligned} \tag{9}$$

After regrouping relevant terms in L, the problem is decomposed into product subproblems and overload subproblems, which can be solved separately.

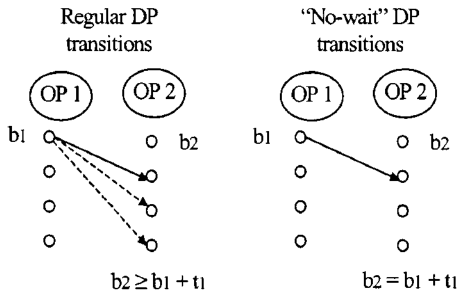

Fig. 3. Regular and "no wait" DP transitions.

B. Product Subproblems

Collecting terms in (9) related to Product i leads to the following product subproblem:

$$\begin{aligned} \min_{\{b_{ij}, t_{ij}, H_{ij}\}} L_i, \quad & \text{with } L_i \\ \equiv w_i T_i^2 + \beta_i E_i^2 + & w_i^L (c_{i,J_i-1} - b_{i0}) + \sum_j L_{ij}(b_{ij}, t_{ij}, H_{ij}) \end{aligned} \tag{10}$$

subject to constraints (3)–(7). In the above

$$\begin{aligned} L_{ij}(b_{ij}, t_{ij}, H_{ij}) & \equiv w_{ij}^P (c_{ij} - b_{ij}) + \sum_{h \in H_{ij}} \sum_k (\delta_{ijkh} \cdot \pi_{kh}) \\ & \equiv w_{ij}^P (c_{ij} - b_{ij}) + \sum_{h \in H_{ij}} \frac{P_{ijh}}{t_{ij}} \sum_{k=b_{ij}}^{c_{ij}} \pi_{kh}. \end{aligned} \tag{11}$$

It represents *operation-wise costs*. The multipliers $\{\pi_{kh}\}$ can be interpreted as prices for using resources. The subproblem objective L_i thus reflects the balance among on-time delivery, low WIP inventory, and the costs for utilizing resources. The resolution of the subproblems using DP is elaborated next.

1) DP for Product Subproblems: Similar to [14], a product's process plan can be mapped onto a DP diagram, with DP stages corresponding to operations and states corresponding to possible operation beginning times. Within a stage, each resource set H_{ij} with a particular level of resource allocation results in one column of DP states. As shown in Fig. 3, the "no wait" processing requirement and operation beginning and completion windows can be embedded within DP by restricting state transitions according to (6).

To solve the subproblem, backward DP (BDP) was developed in [14] by starting from the last operation and moving backward through the process plan. BDP can be extended to handle uncertainties as presented in [12]. It, however, *cannot* solve a subproblem with assembly because of possible decision conflicts as illustrated in Fig. 4. In the figure, the sequence of cost computation is from Operation 4 to Operation 3, and then to Operations 1 and 2. After obtaining the minimum costs for Operations 1 and 2, the subproblem solution is attained by forward tracing the stages. There may be a conflict if Operations 1 and 2 select different Operation 3 beginning times. Forward DP (FDP) presented in [5] has been used to solve deterministic product subproblems with assemblies. Similar to the shortcoming of BDP

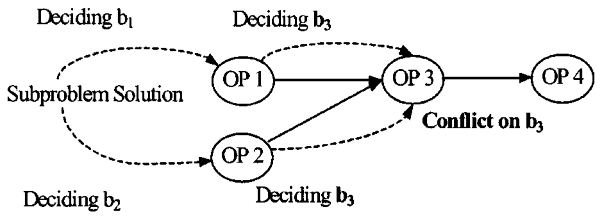

Fig. 4. BDP for a subproblem with assembly.

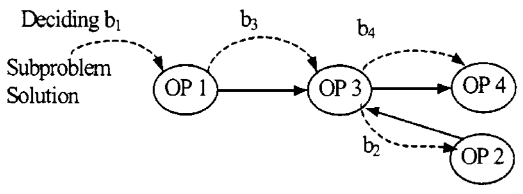

Fig. 5. Solving subproblem with BFDP.

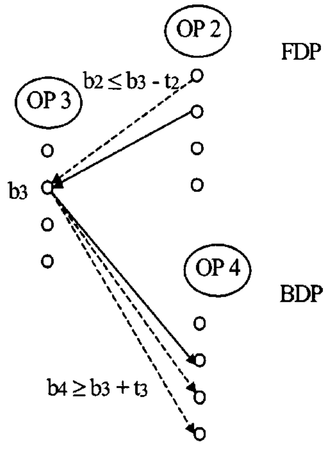

Fig. 6. Cost computation of BFDP.

for the assembly case, FDP cannot solve a subproblem when an operation has multiple successors.

To overcome the above difficulties, complicated process plans are decomposed into multiple simple plans by relaxing assembly precedence constraints in [23]. The additional relaxation, however, increases the number of multipliers, and the approach suffers from slow convergence.

2) Backward/Forward DP: In this paper, BDP and FDP are combined to solve a product subproblem after a novel transformation of the process plan. For the example in Fig. 4, Operation 2 is flipped to the right-hand side of Operation 3, and a *tree structure* is formed as shown in Fig. 5. This tree has a single root node Operation 1, and operation precedence relations are not changed as indicated by solid arrows. The tree is then solved from right to left, with Operations 4 and 2 in the first step. The key is that the beginning time of an operation is decided by at most one but not two operations. The detailed cost computation for Operation 3 is illustrated in Fig. 6. Since Operation 4 is a successor of Operation 3, possible transitions should satisfy $b_4 \geq b_3 + t_3$. The cost comparison is the same as that in BDP. For Operation 2, possible transitions should satisfy $b_2 \leq b_3 - t_2$. The cost comparison for Operation 2 is thus the same as that in FDP. Since this cost computation is similar to that of FDP, BFDP cannot handle uncertainties.

BFDP for the example in Fig. 5 is illustrated below. For simplicity, only operation-wise costs of (11) are considered, and decision variables are operation beginning times. The subproblem is

$$\begin{aligned}\min L, \quad &\text{with } L\\ &= \min_{\{b_1\}}\left\{L_1(b_1) + \min_{\{b_3 \geq b_1+t_1\}}\right.\\ &\quad \cdot\left\{L_3(b_3) + \min_{\{b_2+t_2 \leq b_3\}}\right.\\ &\quad \left.\left.\cdot\left\{L_2(b_2) + \min_{\{b_4 \geq b_3+t_3\}}\left\{L_4(b_4)\right\}\right\}\right\}\right\}. \qquad (12)\end{aligned}$$

$$\begin{aligned}\min L, \quad &\text{with } L\\ &= \min_{\{b_1\}}\left\{L_1(b_1) + \min_{\{b_3 \geq b_1+t_1\}}\left\{L_3(b_3) + \min_{\{b_2+t_2 \leq b_3\}}\right.\right.\\ &\quad \left.\left.\left\{L_2(b_2) + \min_{\{b_4 \geq b_3+t_3\}}\{L_4(b_4)\}\right\}\right\}\right\}. \qquad (12)\end{aligned}$$

From (12), it can be seen that a beginning time is directly decided by at most one but not two operations, there is no conflict in the process, and the subproblem can be optimally solved.

If cycles exist in a process plan, a tree cannot be formed, and relaxation of certain precedence constraints is needed before BFDP can be applied. Consider an operation *in a cycle* having multiple successors, e.g., Operation 00 in Fig. 1. All but one precedence constraints between this stage and its successors are relaxed. Without loss of generality, the precedence between such a stage (i, j) and its successor (i, l) is relaxed by using multipliers $\{\eta_{ijl}\}$ if (i, l) is not the first successor of (i, j) ("first" is defined in a very loose sense). The set of such (i, j) operations form a set R_i. With this relaxation, (9) becomes

$$\begin{aligned}L \equiv &\sum_i \left[w_i T_i^2 + \beta_i E_i^2\right] + \sum_i \left[w_i^L(c_{i,J_i-1} - b_{i0})\right]\\ &+ \sum_{i,j}\left[w_{ij}^P(c_{ij} - b_{ij})\right] + \sum_h w_h^r \sum_k r_{kh}^2\\ &+ \sum_{k,h}\left[\pi_{kh}\left(\sum_{i,j}\delta_{ijkh} - M_{kh} - r_{kh}\right)\right]\\ &+ \sum_{i,j}\sum_{(i,l)\in I_{ij}\cap R_i}\left[\eta_{ijl}(c_{ij} + S_{ij} + 1 - b_{il})\right] \qquad (13)\end{aligned}$$

and L_{ij} in (11) becomes

$$\begin{aligned}L_{ij}(b_{ij}, t_{ij}, H_{ij}) \equiv &\, w_{ij}^P(c_{ij} - b_{ij})\\ &+ \sum_{h\in H_{ij}} \frac{P_{ijh}}{t_{ij}} \sum_{k=b_{ij}}^{c_{ij}} \pi_{kh}\\ &+ \sum_{(i,l)\in I_{ij}\cap R_i} \eta_{ijl}(c_{ij} + S_{ijl} + 1)\\ &- \sum_{\{(i,l)|(i,j)\in I_{il}\cap R_i\}} \eta_{ilj} b_{ij}. \qquad (14)\end{aligned}$$

3) Specific Steps for BFDP: Using the transformation described above, the complicated process plan in Fig. 1 can be

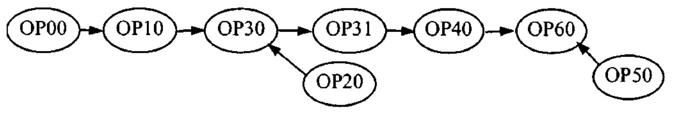

Fig. 7. A sample converted process plan.

converted to a tree structure with a single root node as shown in Fig. 7. In such a tree, the root node is at the left. Nodes having same number of hops from the root belong to the same "layer." Operation (i, j) is the *parent* of (i, l) if (i, j) has precedence relationship with (i, l) and is to the left of (i, l). If (i, j) is the *parent* of operation (i, l), then (i, l) is a *child* of (i, j). All the children of operation (i, j) form a set C_{ij}. Since an operation's beginning time is determined only by its parent operation, conflict does not exist. Further, since (10) is stage-wise separable, the subproblem can be optimally solved by dynamic programming according to [12].

The BFDP starts from computing terminal costs of tree leaves, i.e.,

$$V_{ij}(b_{ij}, t_{ij}, H_{ij}) \equiv \Delta_{ij}(w_i T_i^2 + w_i^L c_{ij}) + L_{ij}(b_{ij}, t_{ij}, H_{ij}) \qquad \forall j \in A_{i0} \tag{15}$$

where Δ_{ij} equals 1 if (i, j) is the last operation of product i and 0 otherwise, and A_{il} is the set of operations at the lth layer counting from tree leaves.

The cumulative cost when moving from children to a parent (i, j) is obtained by combining the minimum costs of children and stage-wise costs of the parent, i.e.,

$$\begin{aligned} V_{ij}(b_{ij}, t_{ij}, H_{ij}) &\equiv L_{ij}(b_{ij}, t_{ij}, H_{ij}) \\ &+ \sum_{(i,l)\in C_{ij}\cap I_{ij}} \min_{\{b_{il}, t_{il}, H_{il}\}} V_{il}(b_{il}, t_{il}, H_{il}) \\ &+ \sum_{(i,l)\in C_{ij},\,(i,j)\in I_{il}} \min_{\{b_{il}, t_{il}, H_{il}\}} V_{il}(b_{il}, t_{il}, H_{il}). \end{aligned} \tag{16}$$

In the above, the second term on the right-hand side is associated with (i, j)'s successors, and the third term is associated with (i, j)'s predecessors.

For operation (i, j)'s child (i, l), costs associated with all possible beginning times $\{b_{il}\}$, processing times $\{t_{il}\}$, and eligible resource sets $\{H_{il}\}$ are computed and compared, and the one with the lowest cost subject to precedence constraints is selected. Finally, the optimal subproblem cost is obtained as the minimum cumulative cost at the root. The optimal beginning times, processing times, and resource sets can then be attained by tracing the optimal path from root to leaves.

The minimization with respect to b_{il} on the right-hand side of (16) can be efficiently carried following [5]. For the second term on the right-hand side of (16) associated with successors, cost computation at b_{ij} includes the minimization of V_{il} for all $b_{il} \geq b_{ij} + t_{ij}$. If cost computation is carried out from bottom to top of a DP column, the minimization of V_{il} will only require one comparison between the previously obtained minimum V_{il} and the cost at $b_{il} = b_{ij} + t_{ij}$. Similarly, for the third term of

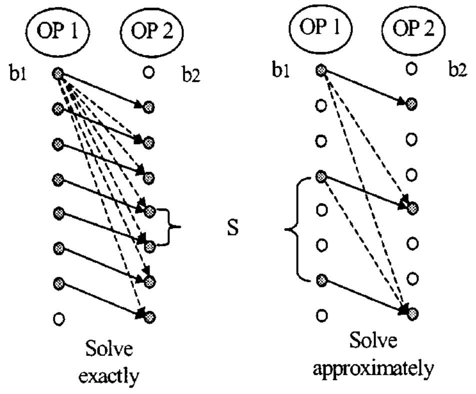

Fig. 8. Simplified dynamic programming.

(16) associated with predecessors, cost computation can be efficiently carried out from top to bottom. The complexity for the above is $O(\Sigma_j N \cdot K)$, where N is the largest number of possible combinations of discrete processing times and operation resource sets.

For a practical problem, the number of DP states can be large. To reduce the computational requirements, BFDP is simplified and solved approximately by computing and comparing the costs of selected DP states only. An adjustable parameter "DP simplification factor," denoted S, is used, and only the costs associated with one out of every S states are computed and compared as shown in Fig. 8. This simplified BFDP is valuable for solving large problems as will be shown in Section V.

C. Overload Subproblems

Collecting all the terms in (9) related to resource overload leads to the following overload subproblem:

$$\min_{\{r_{kh}\}} R_{kh}, \qquad \text{with } R_{kh} \equiv (w_h^r \cdot r_{kh}^2 - \pi_{kh} \cdot r_{kh}), \qquad r_{kh} \geq 0. \tag{17}$$

The solution of the above is

$$r_{kh}^* = \left\lfloor \frac{\pi_{kh}/(2 \cdot w_h^r)}{s} \right\rfloor \cdot s$$

where s is the smallest unit to adjust r_{kh}, and $\lfloor x \rfloor$ the floor function.

D. Dual Problem and Lagrange Multiplier Updating

Let L_i^* denote the minimum subproblem cost for product i, and R_{kh}^* the minimum overload cost for resource type h at time k. The high level dual problem is obtained as

$$\max_{\{\pi_{kh}, \eta_{ijl}\}} D, \text{ with } D \equiv \sum_i L_i^* + \sum_{k,h} R_{kh}^* - \sum_{k,h} \pi_{kh} M_{kh}. \tag{18}$$

Note that since capacity and precedence constraints are relaxed at the same level in (13), these two sets of multipliers are updated together. This dual function D is concave, piecewise linear, and consists of many facets. The *subgradient method* is commonly used for this kind of nondifferentiable optimization. It requires the minimization of all subproblems to obtain a subgradient direction to update multipliers. For large problems with many

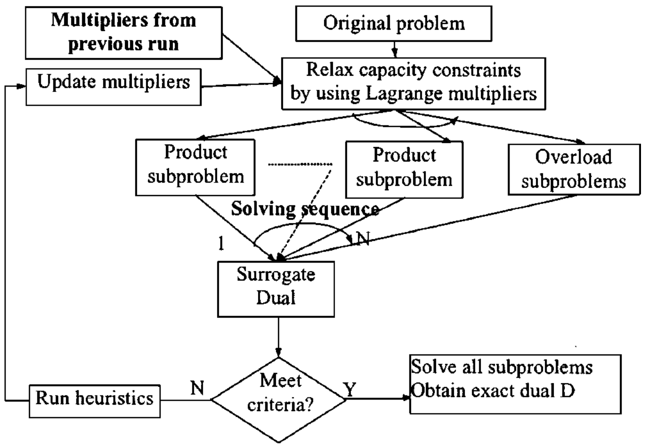

Fig. 9. Surrogate subgradient method and multiplier initialization.

subproblems and each subproblem requiring a large amount of computation, solving all subproblems is very time consuming. To overcome this, the recently developed "surrogate subgradient method" [21] is used to update the multipliers. In this method, only *approximate* optimization of several subproblems is needed to obtain a proper "surrogate" subgradient direction to update multipliers, as opposed to solving all the subproblems precisely (Fig. 9).

In our work, only one subproblem is approximately solved by using simplified BFDP before updating multipliers. If the *DP simplification factor* equals one, subproblems are solved exactly and the convergence condition in [21] holds. If the *DP simplification factor* S is greater than one, subproblems are not optimally solved. However, the dual problem can still converge by adaptively adjusting S and reducing it if necessary to satisfy the convergence condition.

E. Constructing Feasible Schedules

The updating of multipliers is stopped after a fixed amount of computation time or a fixed number of iterations have been executed. Since resource capacity and certain precedence constraints have been relaxed, subproblem solutions generally do not constitute a feasible schedule. Greedy heuristics based on the list scheduling concept were developed in [14] to quickly construct a feasible schedule by delaying some of the operations causing capacity violation. In our work, two heuristic methods were developed in view of the existence of resource overload. The first method extends the approach of [14] by first adding overload subproblem solutions to the original resource capacities. The adjusted capacities are then satisfied by delaying operations as needed to satisfy the relaxed precedence and capacity constraints. The second method uses product subproblems solutions as a basis, and delays certain operations as needed to satisfy precedence constraints. The required capacities and consequently overload are then obtained. In view of the "list scheduling" nature of both methods, cell coordination constraints (6) are only approximately satisfied in either method, although they are exactly satisfied within the BFDP process. These two methods are carried out alternatively during the multiplier updating iterations, and the schedule with the lowest feasible cost is recorded. The relative difference between this feasible cost J and the maximum dual value D is the relative *duality gap* $\equiv (J-D)/D \times 100\%$, and it quantifies the quality of the schedule obtained.

TABLE I
INPUT DATA OF EXAMPLE 1

Product	Oper 0	Oper 1	Oper 2
0,1,2	4 / 0 / 1*	3 / 1 / 1	2 / 2 / 1
3,4,5	1 / 1 / 1	4 / 0 / 1	4 / 2 / 1
6,7,8	3 / 2 / 1	2 / 1 / 1	3 / 0 / 1
9,10,11	3 / 1 / 1	3 / 2 / 1	1 / 0 / 1

* Processing Time/ Resource Type/ Resource Amount

F. Initialization of Multipliers for Rescheduling

Scheduling is usually performed periodically at the beginning of a shift based on a "snapshot" of the factory. Scheduling might also be needed after the arrival of major orders or the breakdown of critical resources. How to speed up rescheduling is an important issue. As mentioned, the values of capacity multipliers reflect the demand on resources. Since the status of the factory may not change significantly from one scheduling instant to the next, these multipliers may not change drastically. Rescheduling can thus be initialized by using multipliers obtained from the previous schedule. This initialization provides a better starting point for the optimization process, and significantly reduces the computational requirements as will be illustrated in the next section.

V. Numerical Results

The algorithm has been implemented using the object-oriented language C++ and tested on a Pentium III 500-MHz PC running the Windows NT operating system. The following examples demonstrate the key features of the method and testing results for large problems.

Example 1: This example is to demonstrate the value of resource overload, and to see that low overload can be achieved by properly selecting the overload penalty coefficients. There are three cells each with one unit of a key resource. Twelve products each with three operations are to be scheduled, and the processing is required to be finished before time unit 30. For all the products, due dates and weights are -1 and 5, respectively, and there is no earliness or lead-time penalty. Operation processing times and the required resource types/amount are provided in Table I.

There is no feasible schedule if resource overload is not allowed. By allowing overload, feasible schedules are generated in 30 s for different values of overload penalty coefficients w_h^r. The feasible costs and overload of the schedules are shown in Fig. 10. Very large penalty coefficients, which are similar to the no overload allowed case, result in small overload but large tardiness. If coefficients are very small, the schedule is similar to that in MRP with infinite capacity, resulting in large overload and small feasible cost J. It can be seen that resource overload can be controlled through the proper selection of penalty coefficients.

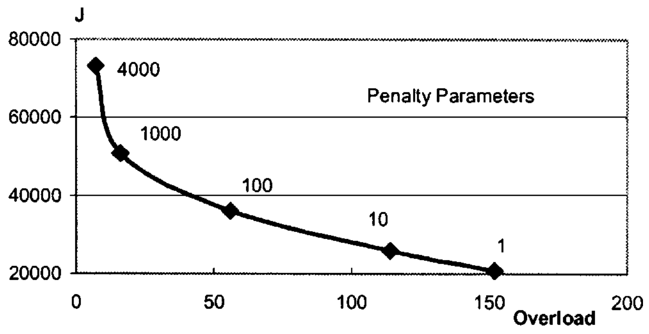

Fig. 10. Feasible costs and overload for Example 1.

TABLE II
TESTING RESULTS OF EXAMPLE 2

Method	Iteration Number	Duality Gap	CPU (sec)
BDP	1000	23.45%	579
BFDP	200	9.91%	116

Example 2: This example is to show that solving product subproblems by BFDP results in better algorithm convergence as compared to decomposing a process plan into multiple BDP subproblems through relaxation. There are 24 cells, each with one unit of a key resource. Ten products each with ten operations and multilevel assemblies are to be scheduled within a time horizon of 200 units.

The problem is first solved by using BDP after assembly precedence constraints are relaxed following [23], and then solved again using BFDP without relaxing such constraints. From the results in Table II, it can be seen that the new approach improves algorithm convergence, and a better schedule is obtained in a shorter amount of time. Although FDP can be used to solve this example, the purpose here is to demonstrate that reducing the number of relaxed constraints often leads to faster convergence and better results.

Example 3: This example is to demonstrate that the method can obtain near-optimal schedules for large problems within a reasonable amount of computation time. Based on sample data sets from Toshiba's gas insulated switchgear factory, 25 data sets were randomly generated and tested. For each data set, there are a total of 2000 products for a total of 20 000 operations to be scheduled on 24 cells each with 30 units of a key resource. The time horizon is 600 h. For each data set, the operation processing times, product due dates, and tardiness weights are deviated from the values of the previous data set by a random percentage uniformly distributed over [−70%, 70%]. Some of the products have "no wait" precedence relationships, or have fixed operation beginning or completion times.

With all multipliers initialized at zero and *DP simplification factor* $S = 6$, the average duality gap obtained in 10 min is 8.25%, implying that near-optimal schedules are obtained with in a reasonable amount of computational time. The smallest, largest, and standard deviation of duality gaps obtained are 7.37%, 9.4%, 0.55%, respectively, indicating that the method can consistently generate good schedules. The resulting schedules also satisfy most of the cell coordination requirements.

TABLE III
TESTING RESULTS FOR SIMPLIFIED DP

Iteration Number	S	Feasible Cost	CPU (sec)
200	1	20,856,456	2373
200	3	20,706,366	690
200	6	20,882,608	456

TABLE IV
TESTING RESULTS FOR RESCHEDULING

Multiplier Initialization	No. of Iteration s	Dualit y Gap	CPU (sec)
at 0	200	9.91%	456
from previous run	50	4.15%	109

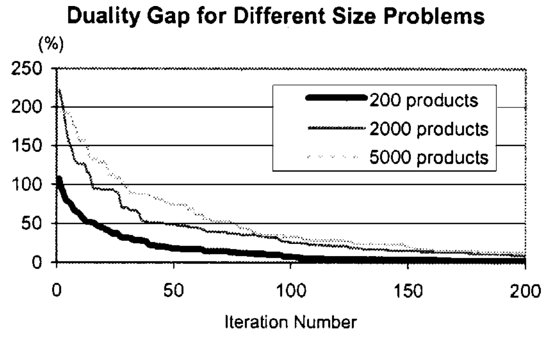

Fig. 11. Duality gap for different size problems.

To demonstrate the speedup obtained by using simplified BFDP, testing results for one of the 25 data sets with different values of *simplification factor* S are summarized in Table III. It can be seen that similar feasible costs can be obtained in a much shorter time by using simplified DP.

To demonstrate the value of using the previous multipliers in rescheduling, assume that 400 products are completed and 400 new products are added to the above data set. Rescheduling is then performed by initializing multipliers from the previous algorithm run. From the results in Table IV, it can be seen that the CPU time can be drastically decreased with this multiplier initialization process.

Example 4: This last example is to demonstrate the performance of the method for even larger problems. There are 5000 products for a total of 50 000 operations to be scheduled on 24 cells each with 60 units of a key resource. A near-optimal schedule with a duality gap of 13.21% is obtained in 40 min with all multipliers initialized at zero. If 400 product are assumed completed and 400 new products are added, the time for rescheduling can be reduced to 10 min by initializing multipliers from the previous algorithm run.

To highlight the convergence of the method, three data sets with 200, 2000, and 5000 products are tested with multipliers initialized at zero. The duality gaps at different iterations are plotted in Fig. 11. It can be seen that the convergence is similar for different sizes of problems although each iteration takes longer for larger problems. Near-optimal schedules are obtained within reasonable computation times for all the three data sets.

VI. Conclusions

A mathematical model with manageable complexity and an efficient solution methodology are critical for macro-level scheduling. The model presented in this paper considers practical issues such as flexible resource utilization, possible resource overload, complicated process plans, and stringent coordination requirements. In the solution process, the separable nature of problem formulation is fully exploited, and BFDP provides a systematic and effective approach to handle complicated process plans. The simplified DP and proper multiplier initialization further speed up the computation. Numerical results demonstrate that high quality schedules are generated for very large problems with up to 50 000 operations within reasonable amount of CPU times.

Acknowledgment

The authors would like to thank K. Yoneda of Fukuoka University (formerly with Toshiba Corporation), L. S. Thakur and H. Chen of University of Connecticut, and X. Zhao, D. Chen, and F. Liu (all three formerly with the University of Connecticut) of i2 Technologies, Inc., for insightful discussions in the development and implementation of this method.

References

[1] B. Andenso-Diaz and M. Laguna, "Modeling the load leveling problem in master production scheduling for MRP systems," *Int. J. Prod. Res.*, vol. 34, no. 2, pp. 493–493, 1996.
[2] K. R. Baker, *Elements of Sequencing and Scheduling*. New York: Wiley, 1974.
[3] ——, "Scheduling the production of components at a common facility," *IIE Trans.*, vol. 20, no. 1, pp. 32–35, 1988.
[4] D. P. Bertsekas, *Nonlinear Programming*: Athena Scientific, 1995.
[5] H. Chen, C. Chu, and J. M. Proth, "An improvement of the Lagrangian relaxation approach for job shop scheduling: A dynamic programming method," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 786–795, Oct. 1998.
[6] T. R. Chen and T. C. Hsia, "A Lagrangian relaxation approach for job shop scheduling with multiple resources," *Int. J. Intell. Contr. Syst.*, vol. 2, no. 4, pp. 511–529, 1998.
[7] S. C. K. Chu, "A mathematical programming approach toward optimized master production scheduling," *Int. J. Prod. Econ.*, vol. 38, pp. 269–279, 1995.
[8] D. H. Cummings McKoy and P. J. Egbelu, "Minimizing production flow time in a process and assembly job shop," *Int. J. Prod. Res.*, vol. 36, no. 8, pp. 2315–2332, 1998.
[9] G. Dobson and U. S. Karamarkar, "Simultaneous resource scheduling to minimize weighted flow times," *Oper. Res.*, vol. 37, no. 4, pp. 592–600, 1989.
[10] Q. Hao, B. Song, D. Wang, and Z. Yang, "Earliness/tardiness production planning by JIT philosophy for job shop manufacturing systems," *Prod. Planning Contr.*, vol. 9, no. 2, pp. 181–188, 1998.
[11] L. J. Krajewski, B. E. King, L. P. Ritzman, and D. S. Wong, "Kapan, MRP and shaping the manufacturing environment," *Manage. Sci.*, vol. 33, no. 1, pp. 39–57, Jan. 1987.
[12] R. E. Larson and J. L. Casti, *Principles of Dynamic Programming*. New York: Marcel Dekker, 1978.
[13] P. B. Luh, D. Chen, and L. S. Thakur, "An effective approach for job-shop scheduling with uncertain processing requirements," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 1–12, Apr. 1999.
[14] P. B. Luh, L. Gou, Y. Zhang, T. Nagahora, M. Tsuji, K. Yoneda, T. Hasegawa, Y. Kyoya, and T. Kano, "Job shop scheduling with group dependent setups, finite buffers, and long time horizon," *Ann. Oper. Res.*, vol. 76, pp. 233–259, 1998.
[15] P. B. Luh and D. J. Hoitomt, "Scheduling of manufacturing systems using the Lagrangian relaxation technique," *IEEE Trans. Automat. Contr.*, vol. 38, pp. 1066–1079, July 1993.
[16] H. W. Oden, G. A. Lagenwalter, and R. A. Lucier, *Handbook of Material & Capacity Requirements Planning*. New York: McGraw-Hill, 1993.
[17] J. Olhager and J. Wikner, "A framework for integrated material and capacity based master scheduling," in *Beyond Manufacturing Resource Planning (MRP II)*. New York: Springer, 1998, pp. 3–20.
[18] P. C. Pandey and M. A. A. Hasin, "Lead time and inventory cost for rough-cut capacity planning," *Int. J. Comput. Applicat. Technol.*, vol. 10, no. 5–6, pp. 280–288, 1997.
[19] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[20] G. A. Suer, M. Saiz, and O. Rosado-Varela, "Knowledge-base system for master production scheduling," *Beyond Manufacturing Resource Planning (MRP II)*, pp. 21–42, 1998.
[21] X. Zhao, P. B. Luh, and J. Wang, "The surrogate gradient algorithm for Lagrangian relaxation method," *J. Optim. Theory Applicat.*, vol. 100, pp. 699–712, 1999.
[22] Y. Zhang, P. B. Luh, K. Yoneda, T. Kano, and Y. Kyoya, "Mixed-model assembly line scheduling using the Lagrangian relaxation technique," *IIE Trans. Schedul. Logist.*, vol. 32, no. 2, pp. 125–134, Feb. 2000.
[23] Y. Zhang, P. B. Luh, K. Yoneda, Y. Kyoya, T. Kano, M. Tsuji, and Y. Kaminokado, "Job shop scheduling with multi-level assembly and lead time penalty," in *Proc. Rensselaer's Int. Conf. Agile, Intelligent and Computer-Integrated Manufacturing*. New York, NY, Oct. 7–9, 1998.

Yuanhui Zhang received the B.E. and M.S. degrees in control engineering from Tsinghua University, R.O.C., in 1993 and 1995, respectively, and the Ph.D. degree in electrical engineering from University of Connecticut, Storrs, in 1999.

Currently, he is a Senior Software Engineer at Oracle Corporation, Redwood Shores, CA.

Peter B. Luh (S'77–M'80–SM'91–F'95) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1973, the M.S. degree in aeronautics and astronautics engineering from the Massachusetts Institute of Technology, Cambridge, in 1977, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, MA, in 1980.

Since 1980, he has been with the University of Connecticut, Storrs. Currently, he is the SNET Endowed Professor of Communications and Information Technologies in the Department of Electrical and Computer Engineering, and the Director of Booth Center for Computer Applications and Research. He is interested in manufacturing planning, scheduling, and supply chain coordination, and has developed near-optimal and computationally efficient methods to improve on-time delivery of products and reduce work-in-progress inventory. He is also interested in schedule/bid optimization and load/price forecasting for power systems, and has developed near-optimal and computationally efficient unit commitment, hydro-thermal coordination, transaction/bidding, and load/price forecasting methods to minimize total cost or maximize the profit. He has been a Visiting Professor for National Taiwan University and a Visiting Researcher for Pratt & Whitney, Toshiba Corporation, and United Technologies Research Center. He owns one U.S. patent.

Dr. Luh received three Best Paper Awards. He is the Editor-in-Chief of IEEE Transactions on Robotics and Automation, an Associate Editor of *IIE Transactions on Design and Manufacturing, Discrete Event Dynamic Systems*, and *International Journal of Intelligent Control and Systems*. He was also an Editor of *IEEE Robotics and Automation Magazine* (1996–1998), and an Associate Editor for IEEE Transactions on Automatic Control (1989–1991).

Katsumi Narimatsu received the M.S. degree from Shizuoka University, Japan, in 1992.

He is a Research Scientist at System Engineering Laboratory, Corporate Research and Development Center, Toshiba Corporation. Since 1992, he has been engaged in research and development for Toshiba Corporation with systems and packaged software for production and project scheduling. He has developed some scheduling systems in several factories in Toshiba. He has also developed a packaged software for project resource scheduling. He is interested in scheduling, production management, project management, optimization, and computer science. He is especially interested in hierarchical and decentralized decision making system with scheduling facilities in manufacturing, design, and general business field. He is also interested in heuristics-based resource planning-scheduling systems as well as optimization-based systems.

Mr. Narimatsu is a member of Information Processing Society of Japan, Operations Research Society of Japan, and Scheduling Society of Japan.

Tetsuro Moriya received the M.S. degree in computer science from the Tokyo Institute of Technology, Tokyo, Japan, in 1995.

Since 1995, he has been working in the System Engineering Laboratory, Corporate Research and Development Center, Toshiba Corporation. He developed several job-shop scheduling systems for factories in Toshiba, and a packaged software of job-shop scheduling. He is interested in production planning, scheduling, project management, expert systems, and machine learning. His recent research is on heuristics to accelerate scheduling for machining shops. He is also working on the improvement of the software development processes with a software development support section, and has improved the software development processes for embedded systems in Toshiba.

Mr. Moriya is a member of the Information Processing Society of Japan and the Japanese Society for Artificial Intelligence.

Tsuyoshi Shimada received the M.S. degree in behavioral and cognitive science from Hokkaido University, Japan, in 1988.

He was with the System and Software Engineering Laboratory, Toshiba Corporation from 1988 to 2000, and studied pattern recognition by artificial neural networks and semiautomatic programming with genetic algorithm. His is currently a System Security Specialist at the System Integration Technology Center, Toshiba Corporation. He is interested in the evaluation methodology for security of network systems. He has been working on an international industrial standard ISO/IEC15408 (Common Criteria for Information Technology Security Evaluation) since 2000, and is a Researcher of System Security Evaluation Task Force in Japan Electronics and Information Technology Industries Association. He received the Certified Project Management Professional degree from the Project Management Institute, PA in 2000.

Mr. Shimada is a member of the IEEE Computer Society, the Japan Society for Artificial Intelligence, and the Project Management Institute.

Lei Fang received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 1996, and the M.S. degree in electrical engineering from the University of Macau, Macau, in 1999. He is currently working toward the Ph.D. degreee in the Department of Electrical and Computer Engineering, University of Connecticut, Storrs.

His research interests include optimization theory and algorithms and their applications to manufacturing and control systems.