Scheduling Flexible Manufacturing Systems For Apparel Production

Robert N. Tomastik, *Member, IEEE*, Peter B. Luh, *Fellow, IEEE*, and Guandong Liu

Abstract—**In mid to high volume apparel production, garments are typically grouped into production lots, and each lot is processed in its own manufacturing cell. A flexible manufacturing system used in this environment enables quick cell configuration, and the efficient operation of cells. The scheduling problem is to decide when to set up a cell and consequently begin garment production in the cell, and to decide the quantity of machines to allocate to each cell, under the constraints of limited machines. The time to process a production lot depends on the quantity of machines allocated to the cell in which the lot will be processed, and thus scheduling and resource allocation are highly coupled. Past approaches separate the resource allocation and scheduling decisions because the combined problem is too complex to be solved in a practical amount of time.**

In this paper, an accurate and low-order integer programming model is developed which integrates scheduling and resource allocation. Insight is provided into how the model relates to the operation of a real factory. The model is solved using the Lagrangian relaxation methodology, and a new bundle method is used for optimizing the Lagrangian dual function. The combination of an accurate low-order model, Lagrangian relaxation, and the bundle method is shown to be very practical. Testing is performed using data from a real factory producing 10 to 40 lots per week (between 4500 and 8900 garments total) on 105 machines of nine different types. Numerical results show that one-week schedules are generated in less than 3.5 CPU min on a 60 MHz personal computer, and the schedules are within 16–29% of optimal.

I. Introduction

In the apparel industry, a successful manufacturer must make efficient use of limited resources to meet dynamic customer demand as customer preferences change on a seasonal or monthly basis. To better meet customer demand, apparel producers organize their factories into product layouts (Fig. 1), where each cell manufactures a production lot composed of garments of a product family [1], and each operation of the lot requires a specific machine type(s). The advantages of a product layout are that product flow is simplified to flow lines, machines in stations are closer together to reduce travel time, and employees within a cell are more involved as a team. Use of a flexible manufacturing system (FMS) creates the additional advantages of quick cell creation for a new product family by simply re-programming the FMS to route garments, and efficient operation of each cell by automatically transferring garments between machines in stations. The backbone of an FMS for apparel production is a circular, unidirectional, automated material handling system as shown in Fig. 2. A cell in this environment is a cluster of stations, where each station is a physical location that accommodates an operator, one sewing machine, and a finite buffer. The FMS is responsible for routing garments between the stations.

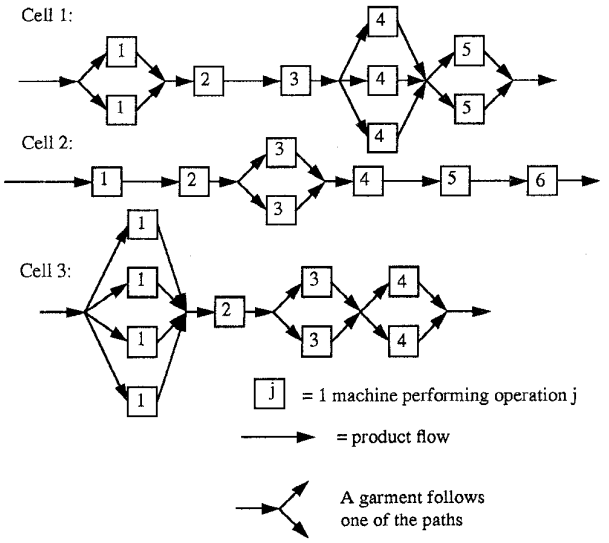


Fig. 1. Product layout of a factory.

To take full advantage of the FMS for apparel production, a good scheduling system is needed. Scheduling decides *when* to set up a cell for a production lot, and consequently begin releasing garments into the cell, and decides the *quantity of machines* to allocate to the cell. These decisions must be made with the consideration of limited machines of different types and limited stations. The objectives are on-time delivery of garments and low inventory. Each lot can represent a retailer's order, or the lots could be formed by a planning function as described in [2] or [3]. These planning functions look at demand from all retailers, form product families, and determine due dates and lot sizes to meet the demand.

Our research was motivated by the scheduling problem of Cannondale Corporation, a manufacturer of clothing for

Manuscript received February 20, 1995; revised August 10, 1995. This work was supported in part by the National Science Foundation grants DDM-9119074 and DMI-9500037 and the Advanced Technology Center for Precision Manufacturing at the University of Connecticut. This paper was presenetd at the IEEE CDC, New Orleans, LA, December 1995. This paper was recommended for publication by Associate Editor J.-M. Proth and Editor A. Desrochers upon evaluation of reviewers' comments.

R. N. Tomastik is with United Technologies Research Center, East Hartford, CT 06108 USA.

P. B. Luh and G. Liu are with the Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06269–3157 USA.

Publisher Item Identifier S 1042-296X(96)05148-8.

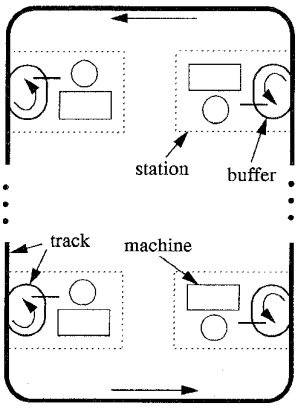


Fig. 2. Schematic diagram of material handling system.

bicyclists (Cannondale also manufactures bicycles, separate from apparel). Realistic FMS's such as Cannondale's have 50–100 stations and over 100 machines, and one week of production consists of about 10–40 lots, where lot sizes typically range from 100 to 1000. The large size of realistic FMS's makes the scheduling problem very difficult.

An integer programming approach is highly desirable to obtain optimal or near-optimal schedules. A complicating factor is that the time to process a production lot depends on the quantity of machines allocated to the cell in which the lot will be processed, and these scheduling and resource allocation decisions are highly coupled. Past approaches separate the resource allocation and scheduling tasks because the combined problem leads to a complex model which can not be solved in a practical amount of time. In this paper, we develop an accurate and low-order model which integrates scheduling and resource allocation. The model is solved using the Lagrangian relaxation methodology. The combination of an accurate low-order model, Lagrangian relaxation, and a new bundle method [4] generates good solutions in a reasonable amount of time using data from Cannondale's factory, and shows the approach is practical. In addition to solving this complicated problem, we provide insight into how the model relates to the operation of a real factory.

A complete and detailed description of the FMS and the scheduling problem is provided in Section II. Literature review is presented in Section III. An integer optimization model which accommodates the scheduling objectives, accounts for the system constraints, and provides for efficient solvability is developed in Section IV. A Lagrangian relaxation methodology is then presented in Section V for generating good solutions in a reasonable amount of time, and with quantifiable quality. Numerical results given in Section VI show that, using Cannondale's data, one-week schedules are generated in less than 3.5 min of CPU time on a 60 MHz personal computer, and the schedules are within 16–29% of optimal. Lastly, a conclusion is provided in Section VII.

II. Description of the FMS for Apparel Production

The backbone of the FMS for apparel production is a circular, unidirectional, automated material handling system, as in Fig. 2. The material handling system has a fixed number of stations. Sewing machines are on carts to facilitate quick movement in and out of stations, and a machine must be in a station to be used for production. Generally there are more machines than stations, and moving machines in and out of stations may be necessary to meet the demand. In this environment, a cell consists of a set of dedicated stations and machines where the FMS is programmed to route garments between stations in the cell. Furthermore, the cell exists only long enough to process its production lot, meaning that once the lot is complete, the stations and machines used by the cell are released for use by future cells. The setup time of a cell is the time to move machines (if necessary), and the time to change threads on machines. Programming the material handling system to perform the routings is not considered part of the cell setup, since the programming can be done off-line well before the above mentioned physical setup.

To process a production lot in its cell, a specified sequence of operations must be performed on each garment (as in Fig. 1). Each operation has a set of eligible machine type(s), and each operation requires a specified amount of time per garment. A machine assigned to an operation is set up and dedicated to that operation during the processing of the lot. Although a garment requires one machine to perform each operation, garments of the same lot can be processed in parallel by assigning multiple machines to that operation. The material handling system is programmed to route garments according to the sequence of operations and the stations containing the machine(s) assigned to the operations. As garments are loaded and released, they are automatically routed from the loading station to the appropriate stations in the prescribed order. The garments are released such that the cell bottleneck is kept busy, and work-in-process inventory of the cell is low. A flowline release policy such as kanban or drum-buffer-rope [5] meet these criteria.

During the routing, a garment and associated raw material are attached to a hanger and hangers are guided along a conveyor, which is a suspended track. When a garment arrives at a station, the hanger is automatically stored in the buffer. After the garments previously in the buffer are processed, the operator removes the garment and required raw material from the hanger, performs the operation, and puts the garment back on the hanger. If the buffer(s) for the next operation are not full, the material handling system takes the hanger and routes it to one of the machines assigned to perform the next operation. The machine chosen is the one with the fewest garments in its buffer. If the buffer(s) for the next operation are full, the current operation is blocked and cannot continue until an opening occurs. The inter-operation travel time depends on the distance between stations, and the time can be a few seconds to several minutes. Also, the conveyor is designed to have sufficient hanger capacity for the demands of clothing production.

The programmable routing and fast machine setups enable quick cell setup, thus giving the manufacturing system its flexibility. Also, as lots complete, the machines and stations allocated to the lots are re-configured to form new cells, and this process of forming and disbanding cells continues over

time. Other advantages include efficient routing, transferring garments in groups of one between operations for smoother flow of garments, and simultaneously operating multiple cells.

III. Literature Review

In this section, literature relevant to the integer programming approach to the scheduling problem under consideration is reviewed. For a detailed review of all scheduling approaches, please see [6].

Several papers on general FMS scheduling exist, such as [7]–[10]. These papers address the features unique to FMS's, such as routing flexibility and multipurpose machines. However, there is limited literature, either for FMS scheduling or for general manufacturing scheduling, on combining the scheduling and resource allocation tasks where the amount of resources allocated to a job affects the job processing time. This capability is required in the case of an FMS for apparel production, where the lot (or job) processing time depends on the number of machines (resources) allocated to the lot's cell. The majority of shop scheduling problems treat the resource requirements as inputs to the scheduling problem, and the job or operation processing times are fixed, as in [11]–[13]. Note that resource allocation may also mean which machines to route a part/garment through, given a set of eligible machines [14].

Several models exist considering the scheduling of a single machine where an additional type of resource (such as labor) is available to allocate to a job, and allocating this resource to a job reduces the job processing time [15]–[19]. These approaches capture the interdependence of scheduling and resource allocation, but the single machine approaches are not adequate for the FMS problem considered here. Scheduling and resource allocation are combined in a flow shop setting in [20] where a flexible resource (such as cross-trained labor) is allocated to jobs to minimize the makespan. Their paper addresses the specific constraints of a flow shop and is not general enough to be used here. However, our approach to the scheduling and resource allocation problem is similar in concept to what was presented there. In light of the current literature, our paper extends existing approaches by integrated scheduling and resource allocation in the general environment of job (i.e., production lot) scheduling, where the shop has multiple machines of many types, each type having limited capacity. Also, real-world application to the apparel industry is demonstrated.

IV. Scheduling Model

In this section, the scheduling problem is modeled as an integer optimization problem, similar to the job shop scheduling formulation in [21]. The decision variables are the beginning time to setup a cell for its production lot, and the quantity of machines to allocate to each operation of each lot. The objective (Section A) is to minimize the sum of weighted tardiness to help meet the due dates, and earliness to reduce inventory, where the weights account for lot priorities. In Section B, the time to process a production lot is derived and expressed in terms of the quantity of machines assigned to the lot. In Section C, the system-wide constraints on the limited number of machines and stations are modeled. The goals in developing the model are to formulate the problem simply but accurately, and to obtain a model that is additive in terms of the decision variables. The additivity of the decision variables, achieved among lots but not within a lot, gives the problem "separability" so that the system-wide constraints can later be relaxed to form lot subproblems, as described in the next section.

A. Objective Function

The objective function to be minimized is the weighted sum of production lot tardiness and lot release earliness to meet customer demand and to reduce work-in-process inventory. The due date (or promise delivery date), weights, and lot size of each lot are assumed given, where weights are numerical representation of the lot priority. The decision variables are the beginning time b_l to set up and process each lot l, $l = 1, 2, \cdots, L$, and the number of machines m_{ljh} of type h to allocate to operation j of lot l. The beginning time is an integer variable from 1 to K, where the scheduling horizon is divided into K equal time intervals; for example, each period represents one hour (the "time resolution") and the time horizon is one week, or 40 hours. The objective is expressed mathematically as

$$\min_{\{b_l\},\{m_{ljh}\}} J(b_l, m_{ljh})$$

where

$$J(b_l, m_{ljh}) = \sum_{l=1}^{L} (w_l T_l + \beta_l E_l) \tag{1}$$

where T_l is the tardiness of lot l defined as $T = \max[0, c_l - d_l]$, c_l is the completion time of lot l, d_l is the due date of lot l; E_l is the release earliness defined as $E_l = \max[0, s_l - b_l]$, s_l is the desired start date of lot l; and w_l is the tardiness weight/priority, and β_l is the earliness weight (see [22], for a method of selecting s_l and β_l). The beginning and completion times are related according to:

$$c_l = b_l + u_l + t_l - 1 \tag{2}$$

where the beginning time b_l is defined to be the beginning of a time period, the completion time c_l is defined to be at the end of a time period, u_l is the setup time for lot l, and t_l is the time to process all the garments of lot l.

The time t_l to process lot l is defined as the duration from the beginning of processing the first garment in the lot to the finish of processing the last garment in the lot. The time t_l depends on the number of machines assigned to the lot, as will be shown in the next subsection. The setup time accounts for moving machines in and out of stations (if necessary), and the time to change threads on the sewing machines. Although this setup time depends on the number of machines that have to be moved, for simplicity, the time will be assumed fixed and known.

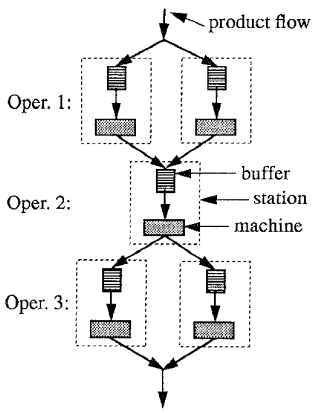


Fig. 3. Functional diagram of the processing of a production lot.

B. Lot Processing Time

The goal in finding an expression for t_l is to avoid detailing the exact behavior of every garment in lot l, as this would lead to a complicated model. An accurate approximation to t_l can be derived by observing that for a given allocation of machines, each lot has a bottleneck operation and that t_l essentially consists of three parts:

- t_l^1: The time to initially feed the bottleneck operation of lot l.
- t_l^2: The time for the bottleneck to process all the garments in the lot once the bottleneck is fed.
- t_l^3: The time for the garments to finish operations downstream of the bottleneck once all garments have been processed at the bottleneck.

The expression for lot processing time is:

$$t_l = \lceil t_l^1 + t_l^2 + t_l^3 \rceil \tag{3}$$

where $\lceil \cdot \rceil$ is the ceiling function, which numerically rounds-up the processing time to an integer multiple of the time resolution. The time resolution should be small enough such that the ceiling function creates only insignificant model inaccuracies.

As an example, Fig. 3 shows the functionality of a production lot of 500 garments, where the lot has three operations, the number of machines dedicated to each operation is 2, 1, and 2, respectively, the processing time of each operation is 1.5, 1.0, and 1.5 min, respectively, and, for simplicity in this example, the inter-operation travel time is zero. The maximum rate that garments are processed for each operation is 1.33, 1.00, and 1.33 garments per minute. Since operation two is the slowest, this operation is the bottleneck. Therefore, the components of the lot processing time are $t_l^1 = 1.5$ min, $t_l^2 =$ (500 garments)/(1 garment/minute) = 8.3 h, and $t_l^3 = 1.5$ min, and the total lot processing time is 8.4 h. The dependence of the lot processing time on the quantity of machines allocated to this lot can be seen by noticing that if a second machine is added to operation two, then the processing time would be cut to 6.3 h.

In the case of Cannondale, t_l^1 and t_l^3 are negligible, and we set them to zero. In applications where this is not the case, a suitable model for t_l^1 and t_l^3 can be formulated.

For a general lot l, the maximum rate of the bottleneck operation is

$$r_l = \min_j \frac{m_{lj}}{t_{lj}} \tag{4}$$

where t_{lj} is the per-garment processing time for operation j of lot l (t_{lj} is assumed to be independent of the eligible machine type selected or the operator assigned as is essentially the case for Cannondale), and m_{lj} is the total number of machines assigned to perform operation j. The equation for m_{lj} is

$$m_{lj} = \sum_{h \in \mathcal{H}_{lj}} m_{ljh} \geq 1 \tag{5}$$

where $\mathcal{H}_{lj}$ is the set of machine types that can perform operation j of lot l.

Note in (4) that more than one operation with rate r_l may exist, meaning there are multiple bottleneck operations. This case does not affect the analysis.

The time to process all the garments on the bottleneck is the lot size divided by the rate of the bottleneck, or

$$t_l^2 = \frac{N_l}{r_l} \tag{6}$$

where N_l is the lot size. Equation (6) assumes the bottleneck machines operate continuously during the time t_l^2, and keeping these machines busy during that time can be achieved by a flowline garment release policy as mentioned earlier.

The expression for t_l in (3) is an *approximation* and does not account for all factors. The accuracy of the expression was examined by simulating real cell configurations using a commercial software package SimFactory. Factors including processing time variations, finite buffers, inter-operation travel times, and lot sizes were considered. Ten lots, using actual Cannondale data, were simulated, where operation processing time variances equaled 0.125 times t_{lj}, buffer sizes were 10 at each station, inter-operation travel times were taken from Cannondale's FMS layout, and a "pull" policy was used for each cell [this policy requires each unstarved machine to produce as long as the immediate downstream buffer(s) is not full]. The simulation showed that the expression for t_l maintains an accuracy of $\pm 4.5\%$ on average. The approximation for t_l remains this accurate as long as the bottleneck operation of each lot is kept busy and work-in-process is not high. These conditions are usually met because a manufacturer seeks to keep the bottleneck busy to maximize throughput, and seeks to keep work-in-process low to minimize lead time. Again, these can be achieved through an effective garment releasing policy based on simple kanban or drum-buffer-rope systems. Consequently, there is no need to model individual garments explicitly.

The constraints (3) through (6) clearly show the dependence of t_l on m_{ljh}, and thus the scheduling and capacity allocation decisions are interdependent. Also, the decision variables m_{ljh} for the same lot are coupled by t_l in (3) through (6), and the minimization operator in (4) will later prevent the separation of these decision variables beyond the lot level. This problem will be addressed later.

C. Machine and Station Availability

As stated previously, the availabilities of machines and stations are limited. Also, machines and stations assigned to lot l are unavailable to other lots during the time from b_l to c_l. The total number of machines of type h, $h = 1, 2, \cdots, H$, being used in each time period k must be less than or equal to M_{kh}, the total number of machines of type h available at that time, or

$$\sum_{l=1}^{L} \sum_{j=1}^{J_l} \delta_{lk}\, m_{ljh} \le M_{kh}, \quad k = 1, \cdots, K; \quad h = 1, \cdots, H \tag{7}$$

where δ_{lk} is one if lot l is active at time k (or, equivalently, if $b_l \le k \le c_l$) and zero otherwise. In (7), the value of m_{ljh} for $h \notin \mathcal{H}_{lj}$ is implied to be zero.

Finally, the station availability constraints require that the total number of allocated machines of all types must be less than or equal to S_k, the number of stations available at each time k (each assigned machine requires one station), or

$$\sum_{h=1}^{H} \sum_{l=1}^{L} \sum_{j=1}^{J_l} \delta_{lk}\, m_{ljh} \le S_k, \quad k = 1, \cdots, K. \tag{8}$$

Again, m_{ljh} is zero for $h \notin \mathcal{H}_{lj}$.

The optimization problem is to minimize (1) subject to constraints (2)–(8). The objective function and the system-wide constraints (7) and (8) consist of sums of functions of individual lots. This characteristic will be exploited by decomposing the problem into lot subproblems using Lagrangian relaxation.

V. Schedule Generation

Our scheduling approach is to obtain good solutions in a reasonable amount of time and with quantifiable quality by using the Lagrangian relaxation method. This same approach has been successfully applied to job shop scheduling problems [21]. Although the schedule generated by this approach is not guaranteed to be optimal, Lagrangian relaxation generates a duality gap, which provides an upper bound on how far from optimal the schedule is.

A. Lagrangian Relaxation

The machine and station capacity constraints in (7) and (8) couple the decision variables among lots, making the problem difficult to solve. The method of Lagrangian relaxation forms a dual function by relaxing these complicating constraints with Lagrange multipliers $\{\pi_{kh}\}$ and $\{\tau_k\}$, respectively, and consequently decouples the lots. As a result, the original problem can be decomposed into lot subproblems that are easier to solve.

To start the relaxation process, the inequality machine and station capacity constraints are first converted to equality constraints by adding slack variables:

$$\sum_{l=1}^{L} \sum_{j=1}^{J_l} \delta_{lk}\, m_{ljh} + s_{kh} - M_{kh} = 0 \quad k = 1, \cdots, K; \quad h = 1, \cdots, H \tag{9}$$

$$\sum_{h=1}^{H} \sum_{l=1}^{L} \sum_{j=1}^{J_l} \delta_{lk}\, m_{ljh} + s_k - S_k = 0 \quad k = 1, \cdots, K. \tag{10}$$

In this way, the dual variables are unconstrained. The relaxed problem is

$$\begin{aligned} J_{LR}(\pi_{kh}, \tau_k) \equiv & \min_{\{b_l\}, \{m_{ljh}\}} \left\{ \sum_{l=1}^{L} (w_l T_l + \beta_l E_l) \right. \\ & + \sum_{k=1}^{K} \sum_{h=1}^{H} \pi_{kh} \left[\sum_{l=1}^{L} \sum_{j=1}^{J_l} \delta_{lk}\, m_{ljh} + s_{kh} - M_{kh} \right] \\ & \left. + \sum_{k=1}^{K} \tau_k \left[\sum_{h=1}^{H} \sum_{l=1}^{L} \sum_{j=1}^{J_l} \delta_{lk}\, m_{ljh} + s_k - S_k \right] \right\} \end{aligned} \tag{11}$$

subject to constraints (2)–(6).

The function J_{LR} is the Lagrangian dual function. The Lagrangian dual problem is [22]

$$\max_{\{\pi_{kh}\}, \{\tau_k\}} J_{LR}(\pi_{kh}, \tau_k) \tag{12}$$

where now

$$\begin{aligned} J_{LR}(\pi_{kh}, \tau_k) = & - \sum_{k=1}^{K} \sum_{h=1}^{H} M_{kh} \pi_{kh} - \sum_{k=1}^{K} S_k \tau_k \\ & + \sum_{l=1}^{L} R_l^1 + \sum_{k=1}^{K} \sum_{h=1}^{H} R_{kh}^2 + \sum_{k=1}^{K} R_k^3 \end{aligned} \tag{13}$$

and R_l^1 is the optimal cost of the subproblem for lot l:

$$\begin{aligned} R_l^1 = & \min_{b_l, \{m_{ljh}\}} \left\{ w_l T_l + \beta_l E_l \right. \\ & \left. + \sum_{h=1}^{H} \sum_{j=1}^{J_l} m_{ljh} \left[\sum_{k=b_l}^{c_l} (\pi_{kh} + \tau_k) \right] \right\} \end{aligned} \tag{14}$$

subject to (2)–(6). R_{kh}^2 and R_k^3 are the optimal costs of the slack subproblems:

$$R_{kh}^2 = \min_{0 \le s_{kh} \le M_{kh}} \{\pi_{kh}\, s_{kh}\} \tag{15}$$

$$R_k^3 = \min_{0 \le s_k \le S_k} \{\tau_k s_k\}. \tag{16}$$

Note that by adding slack variables s_{kh} and s_k in (9) and (10), the dual problem (12) becomes an unconstrained optimization problem. Thus, to solve (12), we can use the bundle method, which is a very effective method for unconstrained optimization. The added CPU time to solve slack subproblems (15) and (16) is negligible, since solving these subproblems does not require any enumerations.

Evaluating J_{LR} for a given set of multipliers is easier than solving the original problem since production lots are decoupled by the Lagrange multipliers. This decoupling allows the minimization in (14) to be solved for each lot independently

of other lots. Also, the dual cost is a lower bound to the optimal value of the primal problem (1) through (8). The difference between the objective function (1) evaluated at a feasible solution and the dual cost is called the duality gap, and provides a measure of how good the feasible solution is.

The Lagrange multipliers $\{\pi_{kh}\}$ and $\{\tau_k\}$ are continuous variables and are unconstrained in the dual problem. From (11), it can be seen that for fixed decision variables $\{b_l\}$, $\{m_{ljh}\}$, s_{kh}, and s_k, the dual function J_{LR} is linear with respect to the Lagrange multipliers. Therefore, the dual function is the minimum of many linear functions, and is thus piecewise linear and concave. For a given set of multipliers, a subgradient of the dual function can be computed by solving the subproblems, and substituting the subproblem solution into the vector of constraints (9) and (10), where elements of the vector correspond to individual constraints. The size of the vector equals the number of multipliers, or $KH+K$. In solving the subproblems and generating a subgradient for a given set of multipliers, only one solution to each subproblem is required, as is normally the case in a Lagrangian relaxation approach.

B. Solving the Subproblems

As can be seen from the model presented in Section IV, the number of possible cell configurations (i.e., allocation of machines to operations for a given lot) is enormous. However, several different configurations can produce the same lot processing time. For example, the configuration in Fig. 3 produces a processing time of $t_l = \lceil 8.4 \rceil = 9$, using a time resolution of one hour. Adding machines to nonbottleneck operations, such as operation 1 or 3, produces the same lot processing time. Other configurations with a different bottleneck operation may also produce the same lot processing time [could be but not necessarily caused by the rounding-up of processing times in (3)]. In general then, multiple configurations produce the same lot processing time. For a given lot processing time, the cell configuration with the fewest number of machines producing that processing time is called a "minimum-machine configuration." To simplify subproblem solving but without loss of generality, only minimum-machine configurations are considered. This simplification does not affect the optimal primal cost since the set of optimal schedules to the original problem always contains a schedule where all lots have a minimum-machine configuration. The minimum-machine configurations for each lot and associated lot processing times are stored in the form of a table, called the "cell configuration table," and its creation is described later. This table stores values of m_{lj}, but not m_{ljh}, and for a given lot and lot processing time, the minimum machine configuration is unique as will become clear later.

The method of solving the lot subproblems in (14) is then to enumerate all combinations of lot beginning time and lot processing time, where the lot processing time determines $\{m_{lj}\}$ via the cell configuration table. The worst-case number of enumerations is K^2, where all K lot beginning times and all K lot processing times are enumerated. For a given b_l and t_l (therefore $\{m_{lj}\}$), the values of m_{ljh} are determined so as to minimize the subproblem cost in (14). Equivalently, for each j, select $\{m_{ljh}\}$ to minimize $\sum_{h\in\mathcal{H}} m_{ljh}\left[\sum_{k=b_l}^{c_l}(\pi_{kh}+\tau_k)\right]$ subject to (5). In this minimization, relevant machine types are used in ascending order of "resource utilization cost," which is the term in brackets, subject to (5) and $m_{ljh} \leq \min_{b_l \leq k \leq c_l}\{M_{kh}\}$. Determining $\{m_{ljh}\}$ thus requires no further enumerations, and the overall complexity for solving all lot subproblems is LK^2 in the worst case.

The cell configuration table can be created prior to the optimization process and stored for later use, or can be created on-line as the subproblems are solved. In our implementation, the cell configuration table is created on-line, thus saving the memory needed to store the table, at a small cost of having to spend time re-creating the table every time the subproblems are solved. In solving the subproblem for lot l, m_{lj} is initialized to one for all j—this is the initial minimum machine configuration. We proceed by enumerating all lot beginning times for this configuration. Then, for the current bottleneck operation $\bar{j}$, $m_{l\bar{j}}$ is incremented by one (if the bottleneck operation is not unique, increment m_{lj} by one for all bottleneck operations). If this new configuration has a reduced lot processing time, then it is a minimum-machine configuration, and we proceed by enumerating all lot beginning times for this configuration. Otherwise, $m_{l\bar{j}}$ is again incremented by one, where $\bar{j}$ must be recomputed, and the new configuration is checked to determine if the lot processing time is reduced. After encountering a minimum-machine configuration and enumerating the beginning times with this configuration, the process repeats by adding a machine to the bottleneck operation. The process stops when all feasible minimum-machine configurations and lot beginning times are evaluated for the lot.

The slack subproblem for s_{kh} in (15) is easily solved by checking the sign of π_{kh}; the subproblem for s_k in (16) is solved similarly.

C. Solving the Dual Problem

The dual problem (12) is solved to obtain a good schedule and to provide a good lower bound to the optimal schedule cost. The most popular method for solving the dual problem is the subgradient method (e.g., [24] and [25]). This method starts with an initial set of multipliers, and iterates two basic steps until some stopping criteria is met: 1) set the multiplier updating direction to a subgradient of the current iteration point, and 2) move some distance along the direction to the next iterate. The subgradient method's popularity is based on its fast direction-computation, ease of implementation, and low computer-memory requirements. However, it suffers from slow convergence—in fact, it cannot guarantee that the dual cost will increase at each iteration.

An emerging competitor to the subgradient method is the bundle method, a variant of which is used here. Instead of determining the direction based on only one local subgradient, bundle methods (e.g., [26]–[28]) accumulate subgradients of points in a *neighborhood* of the current iterate. The set of subgradients used is called the ϵ-subdifferential, defined as (for maximization problems)

$$\partial_\epsilon J_{LR}(x) = \{g \in \mathcal{R}^n : J_{LR}(y) \leq J_{LR}(x) + \langle g, y - x\rangle + \epsilon, \forall y \in \mathcal{R}^n\} \quad (17)$$

which is the union of all subgradients at points in a neighborhood of the current iterate x, where x is an $n \times 1$ vector of the Lagrange multipliers with $n = KH + K$. The scalar parameter $\epsilon > 0$ specifies the extent of the neighborhood: a larger ϵ implies a larger neighborhood. The set $\partial_\epsilon J_{LR}(x)$ is a nonempty compact convex set in $\mathcal{R}^n$, and elements in $\partial_\epsilon J_{LR}(x)$ are called ϵ-subgradients.

Using the concept of an ϵ-subdifferential, bundle methods seek an ϵ-ascent direction d. An ϵ-ascent direction d is a direction which satisfies $J'_{LR,\epsilon}(x, d) > 0$, where $J'_{LR,\epsilon}(x, d)$ is the ϵ-directional derivative given as

$$J'_{LR,\epsilon}(x, d) = \sup_{t>0} \frac{J_{LR}(x + td) - J_{LR}(x) - \epsilon}{t}. \quad (18)$$

A direction satisfying $J'_{LR,\epsilon}(x, d) > 0$, if it exists at x, has the following important convergence property derived from (18):

$$J_{LR}(x + td) > J_{LR}(x) + \epsilon \quad \text{for some } t > 0 \quad (19)$$

which states that J_{LR} increases at least ϵ along d. Such a d exists if and only if $J_{LR}(x) < J_{LR}(x^*) - \epsilon$, where x^* is an optimal dual solution. If such a d does not exist, then $J_{LR}(x)$ is within ϵ of $J_{LR}(x^*)$, and x is called ϵ-optimal. Hiriart-Urruty and Lemarechal ([27], p. 345) show that an ϵ-ascent direction is normal to a hyperplane in $\mathcal{R}^n$ strictly separating the origin from $\partial_\epsilon J_{LR}$.

The availability of only one subgradient at each point implies unavailability of the complete ϵ-subdifferential. The unavailability of the ϵ-subdifferential motivated the bundle idea: at each iteration, accumulate ϵ-subgradients of the iterate in a "bundle" B, and substitute for $\partial_\epsilon J_{LR}$ the convex hull P of the elements in the bundle. The set P provides an inner polyhedral approximation to $\partial_\epsilon J_{LR}$, and the bundle method iteratively adds newly-computed ϵ-subgradients to B until P is a good enough approximation. For a given approximation P, a trial ϵ-ascent direction d is computed by finding a hyperplane separating the origin and P. Conventional approaches solve this problem by finding the point in P closest to the origin, and this approach requires computation time which increases exponentially as b increases, where b is the number of subgradients in the bundle (b generally increases with problem size). This excessive CPU time limited the practicality of bundle methods to small problems, where n and b are small. In [4], the complexity was reduced to $O(nb + b^2)$, a significant reduction for large problems such as the FMS scheduling problem considered here. The complexity was reduced by finding the point in the affine manifold of P (a linear subspace containing P) closest to the origin, and this approach requires a linear projection of a bundle element onto the affine manifold. Once the bundle method computes a trial direction, the direction is tested, via a line search, to determine if (19) can be satisfied. If (19) cannot be satisfied, then a new ϵ-subgradient is found along the trial direction such that the bundle is improved. A new trial direction is computed, and the process repeats until (19) is satisfied. The methods can in general detect an ϵ-optimal dual point.

Determining a new ϵ-subgradient requires function evaluations since the neighborhood $\partial_\epsilon J_{LR}$ depends on both y and $J_{LR}(y)$ in (17). It is more convenient and computationally better to use the "δ-subdifferential," which is the set of all subgradients evaluated at points in a ball of radius δ centered at the current iterate:

$$\partial_\delta J_{LR}(x) = \{g \in \mathcal{R}^n : g \in \partial J_{LR}(y), ||y - x|| \le \delta, \quad \forall y \in \mathcal{R}^n\}. \quad (20)$$

The advantage of using the δ-subdifferential is that computationally it is very easy to find points in this set, especially extreme points of the set, since no function evaluations are needed. Except for replacing $\partial_\epsilon J_{LR}(x)$ with $\partial_\delta J_{LR}(x)$, the method of computing an ϵ-ascent direction here is the same as in [4], where now ϵ equals the largest value of $\bar{\epsilon}$ such that $\partial_{\bar{\epsilon}} J_{LR}(x) \subseteq \partial_\delta J_{LR}(x)$, or, from (17) and concavity of J_{LR},

$$\bar{\epsilon} = \min_y \{J_{LR}(y) - J_{LR}(x) - \langle g, y - x\rangle : g \in \partial J_{LR}(y), \; ||y - x|| = \delta, \forall y \in \mathcal{R}^n\}. \quad (21)$$

Since $\bar{\epsilon}$ is computationally impossible to compute for the general case, an upper bound can be easily computed given the current bundle of subgradients:

$$\epsilon' = \min_{y \in \mathcal{Y}} \{J_{LR}(y) - J_{LR}(x) - \langle g, y - x\rangle : g \in \partial J_{LR}(y)\} \quad (22)$$

where $\mathcal{Y}$ is the set of points at which the bundle subgradients were evaluated and $||y - x|| = \delta$. In this implementation, as δ is increased, ϵ also increases.

The steps to optimizing the dual function follow the steps of conventional bundle methods (such as in [28], pp. 203–204), and are outlined below. We term this implementation the reduced-complexity bundle method, or RCBM.

S1) At a new iteration point x, compute the dual cost $J_{LR}(x)$ and a subgradient g, and initialize the bundle B to $B = \{g\}$.

S2) For the given $B = \{g^1, g^2, \cdots, g^b\}$ and its convex hull P, obtain a hyperplane separating the origin and P, where the normal to the hyperplane yields a trial direction d. Such a d is found by linearly projecting any $g^i \in B$ onto $A^\perp$, where A is the affine manifold of P:

$$d = [I - S^T (SS^T)^{-1} S]\, g^i \quad (23)$$

$$S = \begin{bmatrix} (g^1 - g^2)^T \\ (g^1 - g^3)^T \\ \cdots \\ (g^1 - g^b)^T \end{bmatrix}. \quad (24)$$

If $d = 0$ and an ϵ-optimal point is detected, or if some other stopping criteria is met, then Stop. Otherwise, if $d = 0$, then compute a new d by temporarily dropping an appropriate subgradient from B and re-projecting as in (23) (see [4], for complete details)

S3) At the point $x + \delta d/||d||$, compute the dual cost and the subgradient g. If $\langle g, d\rangle > 0$, then (according to [4]) d is an ϵ-ascent direction and go to S4. Otherwise, add g to B to improve P, and go to S2.

S4) Update the iteration point: $x \leftarrow x + \delta d/||d||$; Go to S1.

Generally the multiplier updating process terminates when some stopping criteria is met, e.g., the maximum number of iterations is reached, an ϵ-optimal dual cost is obtained, and/or the duality gap is smaller than a certain threshold. In choosing an appropriate value of δ, it may seem that larger values will result in faster convergence since the dual cost increases by ϵ at each iteration. However, a tradeoff exists between a larger δ and CPU time required to compute a direction. As δ becomes larger, the value of b required to compute an ϵ-ascent direction also becomes larger as more subgradients from the neighborhood are needed to get a good approximation to the δ-subdifferential. As b becomes larger, the number of computations, $O(nb + b^2)$, to compute a direction becomes larger. The value of δ, however, can be dynamically adjusted from one iteration to the next.

D. Obtaining a Feasible Schedule

Because of the stopping criterion and the relaxed constraints, the solutions from subproblems generally produce an infeasible schedule, i.e., machine and/or station capacity constraints are violated. Other constraints are always satisfied because they were carried through to the subproblems. To adjust subproblem solutions to form a feasible schedule, a heuristic list-scheduling procedure is used, similar to that presented in [21] for job shops. The method is different from the job shop method because in the FMS problem decision variables also include the number of machines, and there are station capacity constraints.

In the heuristic procedure here, a list is created by arranging all production lots in the ascending order of their beginning times. Production lots are then scheduled on their assigned machines and stations as machines and stations become available. If machine and/or station capacity constraints are violated at time k, a greedy heuristic based on the incremental change in J [the original cost function in (1)] determines which lots should begin at that time slot, which ones should be delayed, and which ones should have the number of assigned machines reduced but without delaying. The process then repeats. When the number of machines is reduced, the procedure checks if the bottleneck operation has been shifted and updates the lot processing time for the new cell configuration. Also, a "look back" feature is used to see if a production lot which previously had its number of machines reduced can be better scheduled with the original set of machines, but at a later time when capacity becomes available. The pseudocode of the heuristic is provided in the Appendix.

The heuristic is executed each time when RCBM computes the dual cost in S1 and S3. Recall that computing the dual cost requires solving all the subproblems. Computing the heuristic each time the dual cost is evaluated allows RCBM to monitor the duality gap, which the algorithm uses as part of its stopping criteria, as described in the next section.

VI. Numerical Results

In this section, the numerical results of several examples are presented. The purpose of the first example is to show in detail a simple problem and its solution. The other examples are scheduling problems based on Cannondale's system.

TABLE I
Example One

Lot	lot size	oper. no.	op. proc. time (hrs)	mach. type required
1	100	1	0.035	1
		2	0.052	2
2	100	1	0.035	1
		2	0.018	2
3	100	1	0.035	1
		2	0.152	2

A. Example One

In this example, three production lots are to be scheduled. The data for each lot is in Table I. The columns describe, respectively, the lot number, the lot size (in number of garments), the operation number (two operations per lot), the operation processing time (in hours) for one garment, and the machine type required to perform the operation. Each lot has a due date of zero, desired start date of zero, and weight of one. For this example, there are two machine types. There are two machines of type 1, and three of type 2. All five machines are available throughout the time horizon of ten periods (each period is one hour). The lots have a setup time of zero. For simplicity, station capacity constraint (8) is not considered in this example.

The RCBM was executed until the algorithm detected an ϵ-optimal dual point with $\epsilon' = 2 \times 10^{-4}$, which took 230 iterations and 2 CPU seconds on a PC (60 MHz Pentium microprocessor and 8 Mbytes of RAM). The schedule in Table II was found, which has a cost of $J^* = 16$, and the dual cost was found to be $J^*_{LR} = 15.33$ (see Table III for optimal Lagrange multipliers). For this simple example, the schedule in Table II was verified to be optimal by enumeration. Although an ϵ-optimal dual point was detected and the solution to subproblems evaluated at the optimal multipliers yielded the *optimal primal* solution, the duality gap is not zero [(16 − 15.33)/15.33 = 4.4%]. The optimal dual cost thus does not necessarily equal the optimal primal cost.

To understand the reasons, we draw an analogy from *linear* integer programming problems. For linear integer programming problems, the optimal Lagrangian dual cost equals the optimal cost of an associated integer programming problem where the integrality constraints are dropped, but the system-wide coupling are not relaxed (see [23], for complete explanation). Considering this fact, a duality gap exists when the optimal solution of this continuous-variable problem is not an integer solution, implying that the optimal cost of this problem must be different from the optimal cost of the original integer programming problem. If the vertices of the polytope formed by system-wide constraints are all integers, then there will be no duality gap. The same type of characteristic is believed to determine the existence of inherent duality gaps in

TABLE II
OPTIMAL DECISION VARIABLES FOR EXAMPLE ONE

Lot	begin time	oper. no.	machines assigned	proc. time
1	1	1	2	2
		2	3	
2	3	1	2	2
		2	1	
3	5	1	1	6
		2	3	

TABLE III
OPTIMAL LAGRANGE MULTIPLIERS FOR EXAMPLE ONE

k	π_{k1}	π_{k2}
1	1.01	0.16
2	1.01	0.16
3	0.50	0.17
4	0.50	0.17
5	0.0	0.16
6	0.0	0.16
7	0.0	0.0
8	0.0	0.0
9	0.0	0.0
10	0.0	0.0

nonlinear integer programs. It can be shown that for Example 1, not all the vertices of the polytope formed by system-wide constraints are integers.

B. Other Examples

The six examples presented here use data from Cannondale Corporation's FMS, which produces clothing for bicyclists. There are 9 machines types, 105 total machines, and 60 stations. All machines and stations are available throughout the time horizon of $K = 50$, where each time slot k represents one hour. The number of Lagrange multipliers is $HK + K = (9)(50) + 50 = 500$. Each of the six examples represents one week of work. The lots have various due dates and weights. Set-up times are negligible and are set to zero. RCBM was terminated when either of the following two conditions was met: C1) an ϵ-optimal point was detected, or equivalently, when an optimal dual point lies within the ball of radius δ centered at the current iterate, or C2) the duality gap is less than or equal to 1%. Table IV summarizes each example.

Table V shows the numerical results. In Table V, the duality gap equals (primal cost − dual cost)/(dual cost), where primal cost and dual cost reported were the best values found. In all the examples here, the stopping condition C1 was met. The dual costs reported in Table V are therefore ϵ-optimal, but not necessarily optimal. The number of function evaluations shows the number of times the dual cost was evaluated (i.e., the number of times the subproblems were solved). The CPU time is on the same 60 MHz personal computer as in Example One. The value of δ is the size of the neighborhood used in the bundle method. An appropriate value of δ was found through trial-and-error; however, the results are not sensitive to small variations in δ. Finally, the value of dualcost $+\epsilon'$ provides an upper bound to the optimal dual cost, where ϵ' was computed as in (22).

TABLE IV
DESCRIPTION OF EXAMPLES

Ex.	No. of lots	No. of garments	Ave. No. of operations/lot	No. of primal decision variables
2	10	7046	6.4	74
3	20	4486	6.9	157
4	25	7649	6.6	189
5	30	7798	6.6	228
6	35	7970	6.5	264
7	40	8908	6.7	309

The duality gap results in Table V show that the primal schedules are 16–29% from optimal. Although the dual cost provides a lower bound to the optimal primal cost, the optimal dual cost is not necessarily a tight lower bound, as described in Example One. Despite the larger-than-desired duality gaps, the method still produces near-optimal schedules, and does this in less than 3.5 CPU min on a personal computer. If desired, a tighter bound could be obtained by using a branch and bound enumeration procedure, where the primal and dual costs in Table V are used as initial bounds. A branch and bound method might not be practical due to the potentially large CPU time.

The above examples also demonstrate that now, with the new results of [4], the bundle method is practical for large problems. Of the CPU time reported in Table V, only 5–10% was used by the bundle method in computing a direction, indicating that the overhead needed in computing the direction is not significant. The remaining 90–95% of the CPU time was spent primarily in function evaluations. Also, the CPU time per function evaluation increases linearly with the number of lots, thus the scheduling method is not plagued by combinatorial explosion as the number of lots increases. The heuristic procedure for computing a feasible schedule requires an insignificant amount of CPU time.

VII. CONCLUSION

Scheduling of a flexible manufacturing system for apparel production was addressed using an integer programming approach. The approach decides the quantity of machines and stations to allocate to each cell, and the time to setup and begin processing the production lot for each cell. Schedules comply to the constraints of limited number of machines and stations. Testing on data from a real factory, where 10 to 40 lots (a total of 4500 to 8900 garments) are produced per week on 105 machines and 60 stations, shows that schedules are generated in less than 3.5 CPU min on a personal computer. The schedules, generated to meet lot due dates, are between 16–29% from optimal. The inherent existence of duality gaps

TABLE V
NUMERICAL RESULTS

Ex.	Primal cost	Dual cost	Duality gap	No. of RCBM iterations	No. of function evaluations	CPU time (min:sec)	δ	c'
2	221	191	16%	37	727	1:25	0.2	0.47
3	200	155	29%	37	543	1:29	0.2	0.64
4	341	283	20%	37	679	2:44	0.5	0.69
5	430	369	17%	30	621	3:01	0.5	3.4
6	380	316	20%	45	741	3:28	0.5	2.7
7	524	411	28%	33	450	2:26	0.5	3.9

in this problem suggests that the schedules are closer to optimal than the duality gap indicates.

The successful and practical application of the RCBM to a real-world scheduling problem is a significant development, with impact to other optimization-based scheduling approaches. The strong convergence property of the bundle method, in combination with the reduction of complexity in computing a direction, gives RCBM a strong advantage over subgradient methods. Other production scheduling techniques, such as [29], as well as power system scheduling, airline scheduling, etc., can benefit from this new development in optimization theory.

The scheduling method of this paper provides an excellent complement to a U.S. apparel industry initiative called "Quick Response," or QR for short [1]. The QR effort seeks to time-compress the supply chain from raw material to customer and increase the speed of information flow back from consumers, thus allowing manufacturers to deliver the right products in a timely manner. The supply chain consists of textile production (producing cloth and material), apparel production (producing garments from material), and retail. The means of QR include use of information technology such as electronic data interchange, and improved manufacturing methods such as modular sewing cells and flexible manufacturing technology. The scheduling issues addressed in this paper will help in reducing the lead time of apparel production by optimizing the use of limited resources.

Overall, the method gives an apparel producer a practical and highly-effective method of organizing and controlling their manufacturing operations. Currently, implementation of this scheduling system at Cannondale Corporation is almost complete. The primary issue in implementation has been developing user-interface software that allows the shop scheduler to easily use the scheduling methodology.

APPENDIX
PSEUDOCODE OF THE HEURISTIC

If a schedule generated from solving the subproblems is infeasible, it will be modified to generate a feasible schedule. Please refer to subsection V-D for a high-level description of the heuristic.

Let $\mathcal{B}$ be the set of lots that have already been scheduled by the heuristic, and $\mathcal{B}(i)$ the i'th lot of set $\mathcal{B}$. Let $\{l_1, l_2, \cdots, l_p\}$ be a schedule of p lots with $p \leq L$. The decision variables are assumed to be unchanged from the solutions obtained by solving subproblems unless otherwise stated.

S1) [Initialize.] Let $\mathcal{B} = \{\emptyset\}$ and $k = 1$.

S2) [Sort lots with $b_l = k$.] Let $\mathcal{C} = \{l : l \notin \mathcal{B}, b_l = k\}$. Sort the lots of $\mathcal{C}$ in the descending order of lot incremental cost ΔJ_l, where

$$\Delta J_l = [w_l T_l + \beta_l E_l]|_{b_l=k+1} - [w_l T_l + \beta_l E_l]|_{b_l=k}. \quad (25)$$

Let $i = 1$.

S3) [Schedule lots without modifying machine allocation.] If $\{\mathcal{B} \cup \mathcal{C}(i)\}$ is feasible at time k, then

S3a) $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{C}(i)$, $i \leftarrow i + 1$,

S3b) if $i \leq |\mathcal{C}|$, go to S3, else go to S7; else go to S4.

S4) [Swap machines.] If $\{\mathcal{B} \cup \mathcal{C}(i)\}$ violates a machine availability constraint for machine type h, determine the feasibility of swapping machine type h with an alternative machine type for lot $\mathcal{C}(i)$. If the swapping can be done such that $\{\mathcal{B} \cup \mathcal{C}(i)\}$ is feasible, then

S4a) perform swapping(s) by modifying $\{m_{ljh}\}$,

S4b) $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{C}(i)$, $i \leftarrow i + 1$,

S4c) if $i \leq |\mathcal{C}|$, go to S3, else go to S7; else go to S5.

S5) [Reduce machines.] Check if $\{\mathcal{B} \cup \mathcal{C}(i)\}$ can become feasible by reducing m_{lj} for lot $\mathcal{C}(i)$. If positive, then

S5a) reduce as few machines of $\mathcal{C}(i)$ as possible, such that $\{\mathcal{B} \cup \mathcal{C}(i)\}$ is feasible,

S5b) $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{C}(i)$, $i \leftarrow i + 1$,

S5c) if $i \leq |\mathcal{C}|$, then go to S3, else go to S7; else go to S6.

S6) [Delay the remaining lots.] For all remaining lots $i, i+1, \cdots, |C|$, delay their beginning time by 1.

S7) [Look back to reschedule the lots with their machines reduced.] If the current time k has available capacity, check for lots with $b_l < k$ and had their machines reduced. For each of such lots, if it can be better scheduled (in terms of J) at $b_l = k$ with its original machine configuration, then reschedule it accordingly.

S8) [Increase time index.] If $k < K$, $k \leftarrow k + 1$ and go to S2, else Stop.

ACKNOWLEDGMENT

The authors would like to thank Dr. R. Resch, Mr. D. Wells, Mrs. B. Miller, and Mr. M. Dower of Cannondale

Corporation, Georgetown, CT, for their valuable contributions to this project. Special thanks also goes to Mr. M. Renzulli for his contribution of developing the SimFactory model.

REFERENCES

[1] J. D. Blackburn, *Time-Based Competition, The Next Battleground in American Manufacturing.* Chicago: Business One Irwin, 1991.

[2] M. R. Bowers and A. Agarwal, "Hierarchical production planning: Scheduling in the apparel industry," *Int. J. Clothing Sci. Tech.*, vol. 5, no. 3/4, pp. 36–43, 1993.

[3] M. Fisher and A. Raman, *Reducing the Cost of Demand Uncertainty Through Accurate Response to Early Sales.* Philadelphia: Univ. of Pennsylvania Press, The Wharton School, Feb. 1994.

[4] R. N. Tomastik and P. B. Luh, "A reduced-complexity bundle method for maximizing concave nonsmooth functions," submitted, 1995.

[5] E. M. Goldratt and R. E. Fox, *The Race.* New York: North River, 1986.

[6] F. A. Rodammer and K. P. White, "A recent survey of production scheduling," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 6, pp. 841–851, 1988.

[7] A. Kusiak, "Aggregate scheduling in a flexible machining and assembly system," *IEEE Trans. Robot. Automat.*, vol. 5, no. 4, pp. 451–459, 1989.

[8] R. R. Inman and P. C. Jones. "Decomposition for scheduling flexible manufacturing systems," *Oper. Res.*, vol. 41, no. 3, pp. 608–617, 1993.

[9] M. A. Nascimento, "Giffler and Thompson's algorithm for job shop scheduling is still good for flexible manufacturing systems," *J. Oper. Res. Soc.*, vol. 44, no. 5, pp. 521–524, 1993.

[10] J. Choi and K. Hitomi, "A method of flexible scheduling for flexible manufacturing systems," *Int. J. Prod. Econ.*, vol. 33, no. 1–3, pp. 247–255, 1994.

[11] G. Dobson and U. S. Karmarkar, "Simultaneous resource scheduling to minimize weighted flow times," *Oper. Res.*, vol. 37, pp. 592–600, 1989.

[12] S. Ghosh and C. Gaimon, "Routing flexibility and production scheduling in a flexible manufacturing system," *Eur. J. Oper. Res.*, vol. 60, no. 3, pp. 344–364, 1992.

[13] J. Blazewicz and G. Finke, "Scheduling with resource management in manufacturing systems," *Eur. J. Oper. Res.*, vol. 76, no. 1, pp. 1–14, 1994.

[14] D. Y. Lee and F. DiCesare, "Scheduling flexible manufacturing systems using petri nets and heuristic search," *IEEE Trans. Robot. Automat.*, vol. 10, no. 2, pp. 123–132, 1994.

[15] R. G. Vickson, "Two single-machine sequencing problems involving controllable job processing times," *AIEE Trans.*, vol. 12, pp. 258–262, 1980a.

[16] ——, "Choosing the job sequence and processing times to minimize processing plus flow cost on a single machine," *Oper. Res.*, vol. 28, pp. 1155–1167, 1980b.

[17] L. N. Van Wassenhove and K. R. Baker, "A Bicriterion approach to time cost trade-offs in sequencing," *Eur. J. Oper. Res.*, vol. 11, pp. 48–54, 1982.

[18] R. L. Daniels and R. K. Sarin, "Single machine scheduling with controllable processing times and number of jobs tardy," *Oper. Res.*, vol. 37, pp. 981–984, 1989.

[19] R. L. Daniels, "A multi-objective approach to resource allocation in single machine scheduling," *Eur. J. Oper. Res.*, vol. 48, pp. 226–241, 1990.

[20] R. L. Daniels and J. B. Mazzola, "Flow shop scheduling with resource flexibility," *Oper. Res.*, vol. 42, no. 3, pp. 504–522, 1994.

[21] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job-shop scheduling problems," *IEEE Trans. Robot. Automat.*, vol. 9, pp. 1–13, 1993.

[22] C. Czerwinski and P. B. Luh, "Scheduling products with bills of materials using an improved lagrangian relaxation technique," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 99–111, 1994.

[23] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization.* New York: Wiley, 1988.

[24] B. T. Polyak, "Minimization of unsmooth functionals," *USSR Comput. Math. Phys.*, vol. 9, pp. 14–29, 1969.

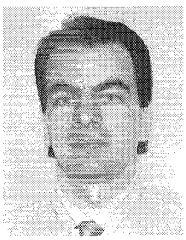
[25] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions.* Hiedelberg: Springer-Verlag, 1985.

[26] C. Lemarechal, "Bundle methods in nonsmooth optimization," in *Nonsmooth Optimization,* C. Lemarechal and R. Mifflin, Eds., *Proc. of a IIASA Workshop,* London: Pergamon, 1978.

[27] J.-B. Hiriart-Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithms I.* Berlin: Springer-Verlag, 1993a.

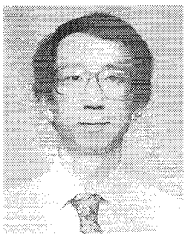
[28] ——, *Convex Analysis and Minimization Algorithms II.* Berlin: Springer-Verlag, 1993b.

[29] S.-C. Chang and D.-Y. Liao, "Scheduling flexible flow shops with no setup effects," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 112–122, 1994.



Robert N. Tomastik (M'91–S'93–M'95) received the B.S. degree from the University of Connecticut, Storrs, in 1989, the M.Eng. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1992, and the Ph.D. degree from the University of Connecticut in 1995, all in electrical engineering.

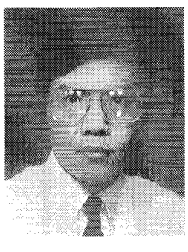
He is currently with United Technologies Research Center, East Hartford, CT. His research interests are in production scheduling, active control for machining, and optimization theory.



Peter B. Luh (S'76–M'80–SM'91–F'95) received the B.S. degree in electrical engineering from National Taiwan University, the M.S. degree in aeronautics and astronautics engineering from the Massachusetts Institute of Technology, Cambridge, MIT and the Ph.D. degree is from Harvard University, Cambridge, MA, in applied mathematics.

He is a Professor, Department of Electrical and Systems Engineering, University of Connecticut, Storrs, where he is also Director of Production Systems and Information Technology, Advanced Technology Center for Precision Manufacturing. His research areas are in the planning, scheduling, and coordination of manufacturing activities for on-time delivery of products and low inventory, and schedule and transaction optimization for power systems. He serves as a consultant to the United Technologies Research Center and Northeast Utilities. He also participates in the Holonic Manufacturing Systems project, a part of the international collaborative research program on Intelligent Manufacturing Systems (IMS). During his sabbatical leaves from University of Connecticut, he was a visiting researcher at Toshiba Corporation in Kawasaki, Japan, a consultant with Pratt & Whitney, and a Visiting Professor at National Taiwan University.

Dr. Luh is an Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, a member of the Connecticut Academy of Science and Engineering, a Senior Member of Society of Manufacturing Engineers, and a member of INFORMS (Institute for Operations Research and the Management Sciences).



Guandong Liu received the B.S. and M.S. degrees in electrical engineering from Huazhong University of Science and Technology (HUST), Wuhan, P. R. China in 1982 and 1985, respectively.

He was a faculty member , Department of Electrical Engineering, HUST, from 1985 to 1990, where he performed research on large-scale, multistage decision making applied to optimal control of hydropower stations. He is currently a doctoral candidate at the Department of Electrical and Systems Engineering, University of Connecticut, Storrs. His research interests include planning and scheduling of manufacturing systems, constrained optimization, and hierarchical solution methodologies for large-scale systems.