Scheduling of Manufacturing Systems Using the Lagrangian Relaxation Technique

Peter B. Luh, *Senior Member, IEEE,* and Debra J. Hoitomt, *Member, IEEE*

Abstract—**Scheduling is one of the most basic but the most difficult problems to be solved in the manufacturing industry. Many of the methodologies currently used in industry result in what one manager has referred to as "scheduling chaos," and overall shop goals such as on-time delivery of jobs are beyond control. There is a *great need* for improving the scheduling operation in complex and turbulent manufacturing environments. Unfortunately, problem characteristics generally translate into large-scale, discrete, dynamic, and / or stochastic scheduling formulations, which are generally considered impossible to solve in most practical situations. The logical strategy is to pursue scheduling methods which *consistently* generate good schedules efficiently. However, the schedule quality is often difficult to determine without actually knowing the optimum.**

In this paper, the practical solution of three manufacturing scheduling problems are examined. As each problem is formulated, constraints are added or modified to reflect increasing real world complexity. The first problem considers scheduling single-operation jobs on identical machines. The second problem is concerned with scheduling multiple-operation jobs with simple fork / join precedence constraints on identical machines. Lastly, the job shop problem is considered, where multiple-operation jobs with general precedence constraints are scheduled on multiple machine types. Lagrangian relaxation is used to decompose each of the scheduling problems into job- or operation-level subproblems. The subproblems are easier to solve than the original problem and have intuition appeal. This technique results in algorithms which generate near-optimal schedules efficiently, while giving a lower bound on the optimal cost. In resolving the scheduling problem from one time instant to the next, the Lagrange multipliers from the last schedule can be used to initialize the multipliers, further reducing the computation time.

Although individual results for the identical machine problems have been reported in the literature, this paper provides a unified framework for developing scheduling methodologies for large scale, diverse, and interdependent systems. In addition, an improved methodology for the job shop problem is presented, the most difficult of the systems studied here. The step-by-step modeling, resolution, intuitive explanation and comments provide much insight into the nature of manufacturing scheduling problems. All algorithms are illustrated with examples using actual factory data. This paper demonstrates that optimization methodologies are not only practical for scheduling in manufacturing systems, but produce consistently high quality results as well.

Manuscript received January 9, 1992; revised August 26, 1992. Paper recommended by Associate Editors, A. Benveniste and K. J. Åström. This research was supported in part by the Department of Higher Education, State of Connecticut, and Pratt & Whitney, East Hartford, CT, under the Cooperative High Technology Research and Development Grant Program, and by the National Science Foundation under Grants ECS-8717167 and DDM-9119074.

The authors are with the Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT, 06269-3157.

IEEE Log Number 9210263.

I. Introduction

SCHEDULING is one of the most important planning and operational issues in the manufacturing industry. It is concerned with the sequencing of operations on machines, and is directly linked with productivity. The resolution of scheduling problems can result in *significant* savings. For example, a scheduling system developed by IBM-Japan is estimated to save a major steel company over a million dollars a year [19]. Unfortunately, the consistent generation of high quality schedules remains a persistent problem in most manufacturing systems, despite the occasional success. Lead times and work-in-process inventories are often excessive, machine utilization low, and product completion dates cannot be accurately predicted, or controlled. It is not uncommon to find high level management tracking the status of high priority jobs on the shop floor.

In this paper, manufacturing scheduling is viewed as a decision problem with discrete variables, where an objective function is optimized subject to relevant constraints. Many objective functions may be considered, but the immediate practical concern is satisfying the customer demand. The decision variables are the beginning times of the various operations required to produce jobs. The practical solutions of three manufacturing scheduling problems are examined. As each problem is formulated, constraints are added or modified to reflect real world complexity. The first problem considers scheduling jobs on identical machines. Jobs require only a single operation on the machines, so that the machine capacity is the main constraint in this problem. The second problem addresses the scheduling of multiple-operation jobs on identical machines, where the production of a job may require a few operations to be performed in a particular sequence. In the third problem, jobs may require complicated networks of operations, each of which may be scheduled on a different type of machine, the so-called "job shop problem."

A. Issues and Literature Review

Given the economic and logistical importance of the scheduling problem, many of the early efforts centered on obtaining optimal schedules. Two prominent optimization methods are the branch and bound method [3] and dynamic programming (e.g., [5]). Unfortunately, it was discovered that the generation of *optimal* schedules is likely to require excessive computation time [14], independent of the methodology. Furthermore, problems in man-

ufacturing scheduling go beyond generating one schedule. Dynamic changes are inevitable elements of daily life. New jobs arrive, machines break down, processing times may be higher or lower than originally anticipated, and even the operational requirements may change in some shops. Since the pursuit of "pure" optimal scheduling methods is impractical, the challenges faced by the research community are to 1) develop efficient solution methodologies that can generate *near-optimal* solutions with *measurable* performance; 2) perform "*what if*" analysis to examine the impact of dynamic changes; and 3) develop effective methods for *schedule reconfiguration* to accommodate these changes. These are crucial issues in the day to day operations of any industrial scheduling system, but they have not been satisfactorily addressed in the literature in practice.

In industry, material planning systems (e.g., MRP) and simulation are often used for high level production and assembly planning requirements [8]. Heuristics or simple dispatching rules are then used to determine schedules at the local (work center) level. The local level generally responds quickly to new information, but the provision for feedback to the planning system is often insufficient. Data from the planning system may show that demand is satisfied by current production, but the reality on the shop floor may be quite different. For these reasons, it may be suggested that manufacturing systems should be wholly scheduled by heuristics. However, it is very difficult to evaluate the quality of heuristic schedules, and the consistency of performance may also be an issue. While a few heuristic methodologies may be capable of providing a measure of the *worst* possible schedule relative to the optimal schedule, the worst case may be more than 100% from optimal (see, e.g., the Nowicki and Smutnicki [18] flow shop heuristic). Even if one realized that a schedule was poor, most heuristics do not provide for iterative improvement of the schedule.

The hierarchical flow control approach of Kimemia and Gershwin [13], [7], is a control-theoretic approach aimed at resolving these issues for a certain class of problems. The hierarchy consists of a system-wide model, with conglomerate or aggregated machines and large time intervals, and migrates to the machine level using smaller time intervals. The mathematical formulation views the flow of jobs as continuous variables, and uses flow balance equations at the different levels to calculate local supply and demand conditions. This is an *approximation* approach since jobs and machines are actually discrete. Thus, the approach may not be suited for scheduling the production of a high variety of jobs, where the demand for each job type may be very small and the continuous variable approximation may be poor. In order to consider individual jobs, the discrete nature of the job flow must be retained.

B. Overview of the Paper

Lagrangian relaxation is a mathematical programming technique for performing constrained optimization. In this paper, it is used to decompose each scheduling problem under consideration into job- or operation-level subproblems. The objective function considered is *on time* delivery as measured by job tardiness, which is also decomposable by job. Since a Lagrangian relaxation algorithm is often stopped after a fixed number of iterations, some of the relaxed constraints may not be satisfied. The infeasible solution, however can be heuristically adjusted to produce effective schedules. The Lagrangian (the dual cost) also provides a lower bound on the optimal cost, and can be used as a performance measure for the schedule. Furthermore, in resolving the scheduling problem from one time instant to the next, the Lagrange multipliers from the last schedule can be used to initialize the multipliers, further reducing the computation time.

Although individual results for the identical machine problems have been reported in the literature, this paper provides a unified framework for developing scheduling methodologies into large scale, diverse and interdependent systems. In addition, an improved methodology for the job shop problem is presented, the most difficult of the systems studied here. The step-by-step modeling, resolution, intuitive explanation, and comments provide much insight into the nature of manufacturing scheduling problems. All algorithms are illustrated with examples using actual factory data. This paper demonstrates that optimization methodologies are not only practical for scheduling in manufacturing systems, but produce consistently high quality results as well.

Section II of this paper considers the scheduling of single operation jobs on identical machines. This problem is representative of a bottleneck work center, where the production of jobs is "controlled" by this work center. Using the Lagrangian relaxation technique, the "coupling" capacity constraints are relaxed using Lagrange multipliers. This decomposes the overall problem into a number of job-related subproblems, which are much easier to solve and have intuitive appeal. The Lagrange multipliers act as prices to regulate the use of machines. The beginning time of an operation is selected to strike the best balance between these machine prices and the tardiness of the job. The prices are updated according to the demand for the machines as a function of these beginning times. Heuristic adjustment then ensures schedule feasibility. An example using industry data produces a result within 0.1% of optimum.

Section III extends the methodology to scheduling *multiple* operation jobs on identical machines. For this problem, visits to nonbottleneck work centers are explicitly modeled as "timeouts." As previously, job-related subproblems are formed. The beginning times of operations again balance between prices and job tardiness. However, because the job tardiness is only associated with the last operation and precedence constraints must be obeyed, an enumeration of possible combinations of operation beginning times is performed. Because of the complexity associated with this technique, the methodology is not practical if the precedence structure involves many operations.

However, for jobs with fewer than three or four operations, the methodology can generate effective schedules for industrial size problems.

Building on these results, the scheduling of the job shop is studied in Section IV. In job shops, multiple operation jobs require *nonidentical* machines. The potentially large number of precedence-constrained operations per job requires the additional step of relaxing these precedence constraints with another set of Lagrange multipliers. Quadratic penalty terms are also added to enforce the constraint. The Lagrange multiplier and quadratic penalty together can be viewed as creating a "window" for operation scheduling, where the cost for being inside the window is less and increases as the operation moves outside the window. The Lagrange multipliers associated with the precedence constraints can be interpreted as prices for overlapping production. Using the decomposition and coordination method of [2], the job shop problem is decomposed into operation level subproblems. Since this methodology does not result in a lower bound, the evaluation of the schedules is accomplished by using the standard Lagrangian relaxation technique based on a different but equivalent problem formulation, as in [12]. In this paper, a new initialization of the dual variables is presented. This initialization scheme saves computation time, while generating better lower bounds. Two examples with factory data are given at the end of Section IV, both generating results within 10% of the optimum. These schedules compare favorably with those generated by a knowledge-based scheduler operating in the factory.

Scheduling Single-Operation Jobs on Identical Machines

In this section, the nonpreemptive scheduling of independent jobs with due dates on identical machines is considered (see also [16]). Each job can be completed in a single machining operation, and is associated with a specified period of processing time and a measure of importance (weight). The objective of the schedule is to meet the due dates.

A. Problem Formulation

An integer programming formulation is a common way to represent a scheduling problem. The quantity to be minimized J is the sum of a weighting problem w_i times the square of the tardiness T_i for each job, where the subscript i refers to the ith job, and the total number of jobs is I:

$$J \equiv \sum_{i=1}^{I} w_i T_i^2. \tag{1}$$

The tardiness refers to the time past the due date d_i, and can be mathematically equated to $\max[0, c_i - d_i]$, where c_i is the completion time of job i. The tardiness objective function accounts for the values of jobs, the importance of meeting due dates, and the fact that a job becomes more critical with each time unit after passing its due date. Note also that the objective function is jobwise additive.

Capacity constraints specify that the total number of jobs being processed (active) at a particular time k must be less than or equal to the number of machines M_k available at time k. Let the integer variable $\delta_{ik} \in [0, 1]$ equal one if job i is active at time k, and zero otherwise. Then the capacity constraint can be expressed for each time k:

$$\sum_{i=1}^{I} \delta_{ik} \le M_k (k = 1, \cdots, K), \tag{2}$$

where K is the time horizon under consideration. The time horizon K represents a discretization of some planning horizon T (e.g., one year) using a time step Δt (e.g., one day), so that $K = T/\Delta t$ (approximately 261 working days in one year). Without loss of generality, the time horizon K is assumed to be long enough to complete all the jobs. Note that time, in addition to jobs and machines, is discretized in obtaining this constraint.

The processing time requirement for jobs state that the elapsed difference between the beginning time b_i and the completion time c_i of job i should be the processing time required t_i, that is:

$$c_i - b_i + 1 = t_i (i = 1, \cdots, I). \tag{3}$$

In (3), the job is active on a machine during the entire time unit b_i and also during the time unit c_i.

Equations (1)–(3) represent a scheduling problem with single-operation jobs and identical machines. The scheduling problem is to select the job beginning times $\{b_i\}$ so as to minimize the weighted quadratic tardiness J subject to the capacity constraints and processing time requirements. Note that the processing time requirements relate to individual jobs, and the objective function is also jobwise additive. Only capacity constraints couple across jobs, and make the problem difficult.

B. Solution Methodology

As discussed in the Introduction, Lagrangian relaxation is a mathematical programming technique for solving constrained optimization problems [15]. For convex programming problems, the maximum of the Lagrangian (the dual cost) is equal to the minimum of the original objective function, and a "saddle point" exists. However, there are several difficulties in utilizing this method for solving discrete variable problems, such as the scheduling problem under consideration, First, the saddle point may not exist, and it may be difficult to determine exactly when an algorithm has converged. Second, the dual cost function is not everywhere differentiable. Specialized methods for optimizing such nondifferentiable cost functions must be employed, such as the subgradient method [21]. Third, even if the dual optimum were obtained, the corresponding schedule at that point may not be feasible. Heuristic adjustment is generally needed to ensure that the once-relaxed constraints are obeyed.

Lagrangian relaxation is used to relax the coupling capacity constraints (2) and obtain a set of decomposed

subproblems. After the decomposition framework is derived, several steps to obtaining a near-optimal solution will be presented: solving subproblems, solving the dual problem, and constructing a feasible solution.

B-1) The Lagrangian Relaxation Framework and Solving Subproblems

The capacity constraint (2) is relaxed by using the Lagrange multiplier π_k to form the relaxed problem:

$$\min_{\{b_i\}} \left\{ \sum_i w_i T_i^2 \sum_k \pi_k \left(\sum_i \delta_{ik} - M_k \right) \right\}; \quad (4)$$

subject to the processing time requirements (3). Then the dual problem is

$\max_{\pi \geq 0} L$, with

$$L \equiv \left\{ - \sum_k \pi_k M_k + \min_{\{b_i\}} \sum_i \left\{ w_i T_i^2 + \sum_k \pi_k \delta_{ik} \right\} \right\}; \quad (5)$$

subject to the processing time requirements (3). This leads to the following decomposed subproblem for each job i (given π):

$$\min_{1 \leq b_i \leq K - t_i + 1} L_i, \text{ with } L_i \equiv \left\{ w_i T_i^2 + \sum_{k=b_i}^{b_i + t_i - 1} \pi_k \right\}, \quad (6)$$

where the processing time requirement (3) has been incorporated into (6) by using the fact that δ_{ik} is zero when the job is not active. Equation (6) represents a decomposed scheduling subproblem for each job. The Lagrange multiplier π_k can be interpreted from a price perspective to represent the cost paid, in this case, for utilizing a machine at time k. The cost L_i is thus balanced between the job requirement for meeting its due date ($w_i T_i^2$) and the cost for utilizing a machine ($\sum_{k=b_i}^{b_i + t_i - 1} \pi_k$). The prices are then adjusted iteratively so that capacity constraints are gradually satisfied over the iterations (to be explained in the next subsection).

For a given set of prices (multipliers), the cost L_i is computed for each possible value of b_i, and the optimal beginning time b_i^* is the one which yields the lowest value of these L_i's. The specific number of evaluations depends on K, or how the planning horizon T is discretized, as described earlier. If a smaller time step Δt is selected, K would be corresponding increased for the same T. The computational complexity of a subproblem is therefore linear in K, in contrast to the NP-hardness of the original problem. If job i cannot be started before a particular time because materials are unavailable or for some other reasons, an "earliest start date" constraint is imposed on the job. This can be accommodated by requiring b_i to lie between the earliest start date of job i and $K - t_i + 1$ (rather than between 1 and $K - t_i + 1$).

B-2) Solving the Dual Problem

Let L_i^* denote the optimal cost of (6), then the high-level dual problem (5) can be rewritten as

$$\max_{\pi \geq 0} L, \text{ with } L \equiv \left\{ - \sum_k \pi_k M_k + \sum_i L_i^* \right\}. \quad (7)$$

Since the subproblems involve discrete variables, the objective function L in (7) is a piecewise linear concave function [4] and may not be differentiable at certain points in the π space. A subgradient method is therefore used to solve the dual problem [21], [9]. In the algorithm, the multiplier π is updated iteratively according to

$$\pi^{n+1} = \pi^n + \alpha^n g(\pi^n), \quad (8)$$

where n is the iteration index, $g(\pi)$ is the subgradient of L in (5) with respect to π with the kth component equal to $(\Sigma_i \delta_{ik} - M_k)$, and α^n is the step size at the nth iteration. From (8), it can be seen that π_k increases when machines are overutilized at time k and decreases otherwise, thus reinforcing the interpretation of π_k from the price perspective. Polyak [21] proved that, with the step size α^n given by

$$\alpha^n = \gamma \frac{\bar{L} - L^n}{g(\pi^n)^T g(\pi^n)} (0 < \gamma < 2), \quad (9)$$

where L is the optimal solution and L^n is the value of L at the nth iteration, this method converges at the rate of a geometric progression. The stepsizing mechanism used here is a modified version of that suggested by [4], and also incorporates features used in [22] and [20], with parameters selected based on testing experiences. The subgradient algorithm terminates after a fixed number of iterations.

Since there are K multipliers to update, the complexity of multiplier update is linear in K. If the operations are updated sequentially, the computational complexity for selecting all the beginning times for given multipliers is I^*K, where I is the number of jobs. However, the job subproblems are independent, and the beginning times could be obtained in parallel. The algorithm is therefore well-suited for distributed implementation, with concomitant reduction of overall computation time.

B-3) Constructing a Feasible Schedule

The dual solution is generally associated with an infeasible schedule, i.e., capacity constraints (2) might be violated at a few time slots. Note that processing time requirements (3) are always satisfied in view of how the subproblems are solved. To correct for these slightly overutilized time slots and construct a feasible schedule, a "list scheduling" algorithm is developed as follows. In the optimal dual solution, each job is associated with a beginning time, b_i^*. A list is created by arranging jobs in the ascending order of their respective beginning times, and jobs are scheduled on machines according to this list as machines become available. If the capacity constraint is violated at time k, a greedy heuristic determines which jobs should begin at that time slot and which jobs are to be delayed by one time unit. In this greedy heuristic, the incremental change in cost if a new job is delayed by

one time unit is calculated. Jobs are then assigned to machines in the descending order of incremental changes, subject to machine availability for the remaining processing period of a job. When all the machines available at that time slot are assigned, the leftover jobs are delayed by one time unit. The process then repeats.

The value of the objective function J for the feasible schedule is an upper bound on the optimal objective J^*. The value of the optimal dual function L^*, on the other hand, is a lower bound on J^* [6], [17]. The difference between J^* and L^* is known as the duality gap. An upper bound of the duality gap is provided by J-L^*, and $(J\text{-}L^*)L^*$ is thus a measure of the suboptimality of the feasible schedule with respect to the optimal one.

B-4) Implications

To examine the impact of dynamic changes, consider the situation where a machine breaks down at time k after a schedule has been generated. What is the impact of this breakdown on the objective function? This breakdown would likely cause an increase in the objective function because the job originally scheduled on this machine at time k could be pushed over or further behind its due date. This may also result in a "domino effect" of pushing other jobs over or further behind their respective due dates. It is generally difficult to estimate the impact of such a change on the objective function. As mentioned, the Lagrange multiplier π_k represents the price for utilizing a machine at time k. It also represents the sensitivity of the objective function with respect to small changes in machine capacity [15]. A good estimate of the increase in J is therefore given by π_k. If a job takes one more time unit to complete than anticipated, there will be one less machine available for other jobs at that extra time slot. The impact on J can also be estimated. Based on this sensitivity interpretation of Lagrange multipliers, an effective technique has been empirically demonstrated to answer a variety of "what if" questions [16].

Given the totality of the manufacturing scheduling problem with its many machines and jobs, most day-to-day changes in the system are relatively small (small changes about the characteristics of jobs to be scheduled and minor fluctuations in capacity). The prices of machines or Lagrange multipliers, therefore, should not change drastically from one day to the next day. Multipliers from the previous schedule can thus be used to initialize the reconfiguration process, and further reduce the computation time. Some examples can be found in [16].

C. Example

In this example,[1] 89 jobs of four different weights are to be scheduled on 42 machines. Not all the machines are available at day 1 (time unit = day) as some of them are busy processing jobs already started. The planning horizon is 88 days. The value of the optimal dual solution is 1581.7, while the feasible schedule has a cost of 1583, which is within 0.085% of the dual solution. The CPU time to generate the schedule is 2.5 seconds on an IBM 3090 mainframe with all multipliers initialized at zero. This example takes data from the tool and die work center at Pratt and Whitney's Development Operations shop. Data and schedules are available upon request. Other examples are discussed in [16].

[1]More examples can be found in [6].

III. Scheduling Multiple-Operation Jobs on Identical Machines

In this section, the model and solution methodology of the previous section are extended to consider the nonpreemptive scheduling of jobs consisting of a small number of operations on identical machines (see also [10]). The operations of a job must be undertaken in a particular order (precedence constraints or process plans) and, between some operations, the job may not be available for scheduling (a timeout) due to paperwork, inspection or other processing which does not require the use of the machines under consideration. The time requirements for processing and timeouts are assumed to be known. As discussed in the Introduction, this situation models a bottleneck work center, where a job may return for processing several times. For reasons which will become more clear as this section progresses, the number of operations per job is limited to a small number (say, 3 or 4), and the precedence of relationships between them are restricted to the simple fork/join type as shown in Fig. 1. In this graph, each operation is represented by a node, and the "forking" represents the case where several operations may be performed simultaneously (like an "and" graph). As in Section II, jobs have different due dates and levels of importance (or weights), and the objective is to meet the due dates. The problem is thus a step more complicated than that considered in Section II.

A. Problem Formulation

Many aspects of this problem formulation are described by the constraint statements of Section II with slight modifications. The capacity constraint can be stated as in (2), except that δ_{ik} represents the number of operations of job i active at time k. The processing time requirement is then restated for each operation j of job i:

$$c_{ij} - b_{ij} + 1 = t_{ij}, \qquad i = 1,\cdots, I; \qquad j = 1,\cdots, N_i; \quad (10)$$

where b_{ij} and c_{ij} are the beginning and completion times of operation j of job i [or operation (i, j)], respectively, t_{ij} is the corresponding processing time requirement, and N_i is the number of operations of job i. In addition, let I_{ij} represent the set of operations of job i immediately succeeding operation (i, j). Then the precedence constraint can be stated as follows:

$$c_{ij} + s_{ijl} + 1 \le b_{il}, \qquad i = 1,\cdots, I; \qquad j = 1,\cdots, N_i - l; \qquad l \in I_{ij}, \quad (11)$$

where S_{ijl} represents a required timeout or a period where the job is unavailable for processing after operation

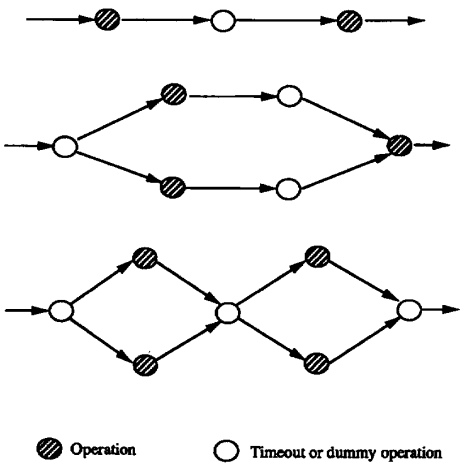

Fig. 1. Examples of simple fork/joint precedence structures.

(i, j). Without loss of generality, it is assumed that each job ends with a single operation so that $c_i = c_{iN_i}$.

The scheduling problem is to select operation beginning times $\{b_{ij}\}$ so as to minimize the weighted quadratic tardiness J of all jobs (1) subject to the capacity constraints (2), the precedence constraints (11), and processing time requirements (10).

B. Solution Methodology

As in Section II, Lagrangian relaxation is employed to relax the coupling capacity constraints (2) to obtain a set of decomposed subproblems. The subproblems are related to each job, as in Section II, and each job-level subproblem is solved by enumerating through all possible combinations of operation beginning times. The method becomes computationally intensive as the number of operations become larger. Therefore, it is only practical for problems where the number of operations per job is small.

B-1) The Lagrangian Relaxation Framework and Solving Subproblems

The decomposed subproblem looks very similar to (6), except that in this case all operations within the job are subject to the precedence constraints (11) as well as the processing time requirements (10). The subproblem for job i is:

$$\min_{\{b_{ij}\}} L_i, \text{ with } L_i \equiv w_i T_i^2 + \sum_k \pi_k \delta_{ik}. \tag{12}$$

As previously, the multiplier π_k represents the price of using a machine at time k. Define L_{ij} as the cost of performing operation (i, J) over the interval $[b_{ij}, c_{ij}]$, i.e.,

$$L_{ij} \equiv \sum_{k=b_{ij}}^{b_{ij}+t_{ij}-1} \pi_k. \tag{13}$$

Then (12) can be rewritten as the sum of the weighted quadratic tardiness and the cost of processing the operations of job i

$$\min_{1 \le b_{ij} \le K - t_{ij} + 1} L_i, \text{ with } L_i \equiv w_i T_i^2 + \sum_{j=1}^{N_i} L_{ij}, \tag{14}$$

subject to the precedence constraints (11). Note again that the processing time requirements have been embedded in (14). Because precedence constraints are restricted to the simple fork/join type with small N_i, the minimization of (14) is done by enumeration; that is, L_i is computed for each possible value of $\{b_{ij}\}_{j=1}^{N_i}$, and $\{b_{ij}^*\}_{j=1}^{N_i}$ are the ones yielding the lowest cost. A simple bounding step utilizing the monotonicity of the quadratic tardiness function can limit the range of this enumeration. The complexity of solving the subproblem, however, is related to the number of possible combinations of $\{b_{ij}\}_{j=1}^{N_i}$. For example, the complexity for a job with three consecutive operations is $O(K^3)$.

B-2) Solving the Dual Problem and Obtaining a Feasible Solution

Similar to (7), the high-level dual problem is

$$\max_{\pi \ge 0} L, \text{ with } L \equiv \left\{ -\sum_k \pi_k M_k + \sum_i L_i^* \right\}, \tag{15}$$

where L_i^* denotes the optimal value of (14). To solve the dual problem, the subgradient algorithm described in Subsection II-B is used. Again, there are K multipliers to update, so that the complexity of multiplier update is linear in K. The computational complexity to sequentially solve the subproblems is $I * K^{N_i}$, where I is the number of jobs. Since the job subproblems are independent, it is likely that this algorithm can be successfully adapted for distributed computing.

In view of how subproblems are solved, precedence constraints and processing time requirements are satisfied by the dual solution. Capacity constraints, however, may be violated at certain time slots. A feasible schedule is constructed by using the list scheduling technique similar to the one presented before. If several operations have the same optimal dual beginning times but there are not enough machines to start them all, a greedy heuristic determines which operations should begin at that time slot based on the incremental change in the weighted tardiness penalty. Subsequent operations of those delayed ones are checked to determine whether they should be delayed to preserve precedence constraints. A delay propagates through the precedence constraints so that the incremental change in tardiness remains updated. This process continues until all operations are scheduled. As previously, the difference between the objective function J for the feasible schedule and the value of the dual function L^* provides a measure of the suboptimality.

C. Example

For this example,[2] there are 44 machines and 112 jobs with a total of 210 operations. There are five weights, and

[2] More examples can be found in [10].

the planning horizon extends 247 days (time unit = day), almost a year of working days (excluding weekends and holidays). The lower bound on the optimal schedule is 1,018,130.8, while the cost of the feasible schedule is 1,018,432.5, a difference of 0.03%. This schedule was obtained in 17.0 CPU seconds on an IBM 3090 mainframe with the initial value of π_k equal to zero for all k. In this example, data from Pratt and Whitney's tool and die work center is again used.

IV. Job Shop Scheduling

Now consider the scheduling of job shops, a typical environment for the manufacture of low-volume/high-variety products. In a job shop, machines are grouped into work centers according to functions, and jobs with specified processing requirements travel to different work centers for processing. For simplicity, each work center is assumed to possess a set of identical machines. Each job consists of a sequence of operations to be performed in a particular order. Sample precedence constraints or process plans are shown in Fig. 2. Each operation can be performed at one of a number of work centers, and the operation time requirement depends on which machine type is selected to perform the operation. A routing decision is therefore required. The capacity of each work center is finite, and may be time varying. These differences can make the problem considerably more complex than that of Sections II or III (see [12]).

A. Problem Formulation

To formulate the scheduling problem with precedence constrained operations and nonidentical machines, let the set of machine types which can perform operation (i, j) be denoted as H_{ij}. Equation (2) is then rewritten to account for the capacity of each machine type h:

$$\sum_{i,j} \delta_{ijkh} \le M_{kh}. \tag{16}$$

In this case, δ_{ijkh} is one if operation (i, j) is scheduled on machine type h at time k, and zero otherwise. The processing time requirement holds for the machine type h^*_{ij} which is selected to perform the operation:

$$c_{ij} - b_{ij} + 1 = t_{ijh*}(1 = 1, 2, \cdots I; j = 1, 2, \cdots, N_i), \tag{17}$$

where, for simplicity of notation, the subscript for h^* is dropped. Similar to Section III, it is assumed without loss of generality that each job ends with a single operation so that $c_i = c_{iN_i}$. The job shop scheduling problem then consists of selecting beginning times $\{b_{ij}\}$ and machine types $\{H_{ij} \in H_{ij}\}$ to optimize the weighted quadratic tardiness of all jobs (1), subject to the capacity constraints (16), the precedence constraints (11) and the processing time requirements (17).

B. Solution Methodology

In Subsection III-B, the original problem is decomposed into job-related subproblems, where the scheduling of each job could be very time-consuming if the job consists of a large number of operations. In this section, the precedence constraints are relaxed by using another set of multipliers to obtain an operation-level decomposition of the scheduling problem.

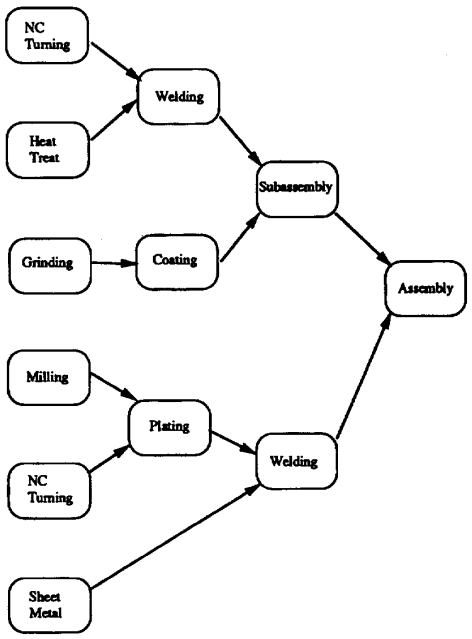

Fig. 2. A sample process plan.

B-1) The Augmented Lagrangian Decomposition and Coordination Method

The precedence constraints of (11) are first reformulated into equality constraints using the "interoperation time" s_{ijl} in preparation for performing an augmented Lagrangian relaxation of the scheduling problem

$$c_{ij} + s_{ijl} + 1 = b_{il}, \quad (i = 1, 2, \cdots I; j = 0, 1, 2, \cdots, N_i - 1; l \in I_{ij}), \text{ with} \tag{18}$$

$$s_{ijl} \ge S_{ijl}. \tag{19}$$

The augmented Lagrangian methodology consists of relaxing equality constraints (18) with associated multipliers as in Sections II and III, and also adding quadratic penalty functions to the objective function. The use of Lagrange multiplier and quadratic penalty terms together (also called "multiplier method") can be viewed as a combination of Lagrangian relaxation (as in Sections II and III) and a penalty method, and is generally used for continuous optimization problems. For that class of problems, the method has been shown to obtain better results than when either Lagrangian relaxation or the penalty method is used separately [1]. In the current integer optimization situation, the quadratic terms are motivated by difficulties with solution oscillation which ensue when the

Lagrangian relaxation method is used by itself. The augmented Lagrangian stabilizes the selection of the beginning times, as explained below and also detailed in [12].

The capacity constraints (16) are relaxed by using the nonnegative Lagrange multipliers π_{kh}. The precedence constraints (18) are relaxed by using Lagrange multipliers λ_{ijl}, and nonnegative penalty coefficients p_{ijl} scale the quadratic penalty function. There is no restriction on the sign of λ_{ijl}, since it relaxes an equality constraint. The augmented dual problem is:

$$\max_{\pi \geq 0,\, \lambda} \left\{ - \sum_{hk} \pi_{kh} M_{kh} + \sum_{i} \min_{\{b_{ij}\}, \{h_{ij} \in H_{ij}\}, \{s_{ijl}\}} \cdot \left\{ w_i T_i^2 + \sum_{j} \left[\sum_{k=b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh} + \sum_{l \in I_{ij}} \left[\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - b_{il}) + \frac{p_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - b_{il})^2 \right] \right] \right\} \right\}. \quad (20)$$

All dual variables and penalty coefficients are real numbers. The minimizations within (20) are job-level subproblems:

$$\min_{\{b_{ij},\, s_{ijl},\, h_{ij} \in H_{ij}\}} L_i, \text{ with}$$

$$L_i \equiv \left\{ w_i T_i^2 + \sum_{j} \left[\sum_{l \in I_{ij}} \left[\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - b_{il}) \cdot \frac{p_{ijl}}{2}(b_{ij} + t_{ijh} - s_{ijl} - b_{il})^2 \right] + \sum_{k=b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh} \right] \right\}. \quad (21)$$

Without the penalty method (i.e., $p_{ijl} \equiv 0$), the minimization within (21) can be decomposed into operation-level subproblems by appropriately regrouping individual terms. The solution oscillation asserted above can be seen by examining this operation-level subproblem:

$$\min_{\{b_{ij},\, h_{ij} \in H_{ij}\}} \left\{ w_i T_i^2 \Delta_{jN_i} + \sum_{k=b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh} + \left[\sum_{l \in I_{ij}} \lambda_{ijl} - \sum_{l:\, j \in I_{ilj}} \lambda_{ilj} \right] b_{ij} \right\}, \quad (22)$$

where $\Delta_{jN_i} = 1$ if j is the last operation of the job (i.e., $j = N_i$) and 0 otherwise. The selection of the beginning time is heavily dependent on the sign of the coefficient $[\sum_{l \in I_{ij}} \lambda_{ijl} - \sum_{l:\, j \in I_{ilj}} \lambda_{ilj}]b_{ij}$, e.g., b_{ij} tends to be large when this coefficient is negative and small when it is positive. As in Sections II and III, selection of the beginning times is followed by a multiplier update step (dual optimization). The dual optimization methodology, in turn, depends on the values of b_{ij}, perpetuating the difficulty (for a further discussion of oscillation in integer programming problems, see [23]). The quadratic penalty terms strike a balance between scheduling an operation early versus scheduling it late, and add the fine tuning necessary to fit all operations of a job together.

B-2) Obtaining Operation Level Subproblems

With the values of multipliers π and λ and penalty coefficients p given, each minimization subproblem within (21) involves one job only. As discovered in Section III, the solution of a job-level subproblem is impractical when there are a large number of operations per job. This remains a major difficulty in solving (21), and an operation level decomposition must be obtained for the method to be practical. Further decomposition seems to be hindered by the cross product terms in the quadratic penalty involving operations (i, j) and (i, l). Fortunately, the augmented Lagrangian decomposition and coordination method ([2, p. 236]) is designed to overcome this situation.

The augmented Lagrangian decomposition and coordination method was originally designed for continuous variable problems. A convergence result for that case is presented in [2]. For integer optimization problems, it has not been possible to show the convergence of the method.[3] It should be noted at the outset that the lower bound property of the augmented Lagrangian is lost when the method is used. This performance evaluation issue will be addressed in a later subsection. Because of the convergence and lower bound issues just mentioned, the augmented Lagrangian decomposition and coordination method has some disadvantages compared to a standard Lagrangian relaxation approach. Further, the performance of the standard technique is often good, as for the earlier problems. For the current job-shop problem, however, it is doubtful that good performance could be achieved by using the standard dual optimization methodologies because of the solution oscillation difficulties previously mentioned.

In order to extend the augmented Lagrangian decomposition and coordination method to the discrete job shop scheduling problem under consideration, some new notation is introduced to simplify the presentation. Let

$$c^s_{ijlh} \equiv b_{ij} + t_{ijh} + s_{ijl} \quad (23)$$

be defined as the "augmented completion time" of operation (i, j). The penalty term between two operations in (21) becomes $(c^s_{ijlh} - b_{il})^2 = (c^s_{ijlh})^2 - 2c^s_{ijlh}b_{il} + b^2_{il}$. The basic idea then is to linearize the middle cross product term. Define $\bar{c}^s_{ijlh}$ as a constant which represents an

[3] As one reviewer pointed out, since optimal schedules are an impractical goal for most manufacturing scheduling problems, there is no reason not to use the method.

estimate of c^s_{ijlh} and also $\bar{b}_{il}$ as a constant representing an estimate of b_{il}. The decomposition then consists of solving for c^s_{ijlh} by using the function $((c^s_{ijlh})^2 - 2c^s_{ijlh}\bar{b}_{il})$ and solving for b_{il} using the function $(-2\bar{c}^s_{ijlh}b_{il} + b^2_{il})$. If the estimates are exact, the optimal c^s_{ijlh} and b_{il} are obtained, but the value of the original cross product term is doubled.

In an iterative procedure, good estimates of $\{\bar{b}_{il}\}$ and $\{\bar{c}^s_{ijlh}\}$ are the beginning times and augmented completion times from the last iteration. In addition, another quadratic penalty is imposed against moving too far from the last computed beginning time of the operation. This is very useful in preventing the wide oscillations discussed in (22). The coefficient of this quadratic, $(1/\epsilon_{ijl})$, is restricted to be greater than p_{ijl} in the original methodology [2], but in this case they are set equal. Summarizing the above discussion, the operation subproblem (i, j) is presented as follows:

$$\begin{aligned}\min_{\{b_{ij},\, h_{ij}\in H_{ij},\, s_{ijl}\}} & w_i T_i^2 \Delta_{jN_i} + \frac{1}{2\epsilon_{ijl}}\left(b_{ij} - \bar{b}_{ij}\right)^2 \\ & + \sum_{l\in I_{ij}} \left\{ \lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl}) \right. \\ & \left. + \frac{P_{ijl}}{2}\left[(c^s_{ijlh})^2 - 2c^s_{ijlh}\bar{b}_{il}\right]\right\} \\ & + \sum_{l:\, j\in I_{il}} \left\{ \frac{p_{ilj}}{2}\left[\left(b^2_{ij} - 2\bar{c}^s_{iljh}b_{ij}\right] - \lambda_{ilj}b_{ij}\right\} \right. \\ & + \sum_{k=b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh}, \end{aligned} \tag{24}$$

where $\bar{c}^s_{ijlh}$ is computed using $\bar{h}_{il}$, the machine selected to perform operation (i, l) in the last iteration, and $\bar{s}_{ilj}$, the interoperation time between operation (i, l) and its succeeding operation (i, j) in the last iteration.

In order to solve (24), $\{s_{ijl}\}_{l\in I_{ij}}$ are first obtained as functions of b_{ij} and h_{ij}. An enumeration through all possible b_{ij} and h_{ij} is then performed. For particular candidates b_{ij} and h_{ij}, the function involving s_{ijl} is a symmetric quadratic, and it can be shown that the optimal value of s_{ijl} is $\max[S_{ijl}, s^r_{ijl}]$, where s^r_{ijl} is the rounded integer value of $[\bar{b}_{il} - (b_{ij} + t_{ijh} + (\lambda_{ijl}/p_{ijl}))]$. During the enumeration of candidate beginning times and machine types, the values of all possibilities are directly compared to obtain the lowest cost, solving (24). The complexity of this procedure at the subproblem level is K^*H in the worst case (where the operation can be performed on every possible machine type). To estimate the dual cost of (20), the individual operation cost of (23) is decreased by the following amount:

$$\frac{1}{2\epsilon_{ijl}}\left(b_{ij} - \bar{b}_{ij}\right)^2 + \sum_{l\in I_{ij}} \frac{p_{ijl}}{2} c^s_{ijlh}\bar{b}_{il} + \sum_{l:\, j\in I_{il}} \frac{p_{ilj}}{2}\bar{c}^s_{iljh}b_{ij} \tag{25}$$

before adding up.

B-3) Solving the Dual Problem

To solve the dual problem, the subgradient method is used to update the Lagrange multipliers π as in Subsection III-B. The Lagrange multiplier λ is adjusted according to the multiplier method update formulae [1], [2]:

$$\lambda^{n+1} = \lambda^n + p^n g(\lambda^n), \tag{26}$$

where p^n is the penalty coefficient at the nth iteration, and $g(\lambda)$ is the subgradient of (20) with respect to λ. The subgradient component relating to the jth operation to the lth operation of job i is $(b_{ij} + t_{ijh} + s_{ijl} - b_{il})$, where $l \in I_{ij}$. Some care must be taken in updating p_{ijl}. From (24), it can be seen that large values of p_{ijl} can easily dominate the selection of beginning times, referred to as ill-conditioning in [1].[4] To avoid the ill-conditioning problem, the penalty coefficient is only increased if the constraint is violated while the corresponding λ is positive (indicating the constraint has been violated in the past); *and* they may also be decreased if the constraint is not violated while the corresponding λ is not positive (indicating that constraint violation has not been a problem).

The computational complexity for the multiplier update is $K * H + \Sigma_i N_i$, while the subproblem solutions require $K * H * \Sigma_i N_i$. A related algorithm was implemented in a distributed environment [11] with eight workstations, where the computation time was reduced by half with minimal efforts at load balancing among the processors. This algorithm (the nondistributed version) will be discussed further later in this section. As in the algorithms presented in Sections II and III, the Lagrange multiplier π_{kh}, represents the price for utilizing a machine of type h at time k. The Lagrange multiplier λ_{ijl} relaxing precedence constraints (18) can be interpreted as the cost of violating this constraint by one time unit. From a slightly different perspective, it can be interpreted as the value for overlapping operations (i, j) and (i, l) by one time unit, or the value for reducing the processing time t_{ijh*} or the required timeout S_{ijl} by one time unit. From these, a variety of "what if" questions can be answered.

B-4) Obtaining a Feasible Schedule

The algorithm is stopped when a fixed number of iterations has been reached. As previously, the solution in the dual space is generally associated with an infeasible schedule, i.e., capacity constraints may be violated for a few time slots and/or a few precedence constrained operations may overlap slightly. Violations of precedence constraints are first corrected by pushing the operation beginning times forward in time if needed. A feasible schedule is then constructed by using the list scheduling technique similar to the one presented in Subsection III-B, based on the modified dual solution. The only difference is that the machine availability is tracked for all

[4] In applying multiplier methods (including the augmented Lagrangian decomposition and coordination method) to continuous variable optimization, the penalty coefficient is not required to be strictly increasing to obtain convergence.

machine types and, if a machine in unavailable for scheduling an operation, it may be scheduled on other eligible machine types. As in the heuristic presented in Subsection III-B, the incremental tardiness cost forms the basis of a greedy method for breaking ties and, if operations are delayed, propagation through all precedence constrained operations is performed to update this incremental tardiness. The difference between the heuristic cost and the dual cost can also be used as a stopping criterion for the dual optimization, in combination with limiting the number of iterations.

B-5) Obtaining a Lower Bound Via a Second Problem Formulation

As discussed earlier, it is necessary to construct a lower bound on the optimal cost to measure the performance of the algorithm. One way for obtaining a lower bound is to resort to the standard Lagrangian relaxation approach as in Sections II and III. Since the formulation presented in Subsection IV-A has an oscillation problem, a second but equivalent problem formulation is adopted. To avoid confusion, the first problem formulation and the related augmented Lagrangian decomposition and coordination algorithm will be referred to as the "schedule generation algorithm." This second problem formulation replaces the precedence constraint (11) with

$$\omega_{ijkl} + \sigma_{ilk} \le 1$$
$$(i = 1,\cdots, I; j = 0,\cdots, N_i; k = 1,\cdots, K; l \in I_{ij}), \quad (27)$$

where ω_{ijlk} is an integer variable equal to one for every time unit $k \le c_{ij} + S_{ijl}, l \in I_{ij}$, and zero otherwise; and σ_{ijk} is an integer variable equal to one for every time unit $k \ge b_{ij}$ and zero otherwise. It can be easily checked that (11) and (27) are equivalent representations of precedence constraints. The problem now is to maximize the weighted quadratic tardiness (1) subject to the capacity constraints (16), precedence constraints (27), and processing time requirements (17). This problem can in principle be solved by using the standard Lagrangian relaxation technique without suffering from the solution oscillation difficulty mentioned earlier. Prohibitive memory requirements, however, would be needed in view of the dimensionality of the nonnegative Lagrange multipliers μ_{ijlk} relaxing (27).

By fixing the capacity constraint multiplier π at the value obtained from the schedule generation algorithm π^*, this new dual problem can be decomposed into job-level dual subproblems (max–min subproblems).

$$D_i: \max\left\{-\sum_{jk, l\in I_{ij}} \mu_{ijlk} + \min_{(b_{ij}), (h_{ij}\in H_{ij})} \cdot\left\{w_i T_i^2 + \sum_j \left[\sum_{b_{ij}}^{b_{ij}+t_{ijh}-1} \pi^*_{kh} + \sum_{l\in I_{ij}}\sum_{k=1}^{c^s_{ijh}} \mu_{ijlk} + \sum_{l:\, j\in I_{il}}\sum_{k=b_{ij}}^{K} \mu_{iljk}\right]\right\}\right\}, \quad (28)$$

where $c^s_{ijh} = b_{ij} + t_{ijh} + S_{ijl}$ is the augmented completion time. To examine the characteristics of this equation in more detail and to simplify the presentation of operation level subproblems, the following notation is introduced:

$$\Pi^*_{ijh}(k) = \sum_{m=k}^{k+t_{ijh}-1} \pi^*_{mh}, \quad (29)$$

$$\Gamma_{ijl}(k) = \sum_{m=1}^{k} \mu_{ijlm}, \quad (30)$$

$$\Lambda_{ilj}(k) = \sum_{m=k}^{K} \mu_{iljm}. \quad (31)$$

For the precedence constraint between (i, j) and (i, l), $l \in I_{ij}$, Γ and Λ are mirror images (i.e., reversed) since they use the same multipliers μ. Using this notation, each job-related dual subproblem can be further decomposed into the following operation level subproblems

$$\min_{1\le b_{ij}\le K, h_{ij}\in H_{ij}} \left\{w_i T_i^2 \Delta_{jN_i} + \Pi^*_{ijh}(b_{ij}) + \sum_{l\in I_{ij}} \Gamma_{ijl}(c^s_{ijh}) + \sum_{l:\, j\in I_{il}} \Lambda_{ilj}(b_{ij})\right\}. \quad (32)$$

In (32), the monotonically increasing cost function $\Gamma_{ijl}(c^s_{ijh})$ is associated with the successor operation (i, l). Its monotonicity penalizes late completion times for the operation, and thus deters overlaps with succeeding operations. The Γ function will therefore be called the "successor penalty." The monotonically decreasing cost $\Lambda_{ilj}(b_{ij})$ is associated with a preceding operation. Its monotonicity penalizes early beginning times for the operation, deterring overlaps with preceding operations. The Λ function will thus be called the "predecessor penalty." There does not appear to be any oscillation difficulties associated with the minimization subproblem (32).

Based on the results of (32), the multipliers μ can be obtained *for each job* using the previously described subgradient method with reasonable computational requirements. The dual cost obtained by this procedure is a lower bound to the optimal cost, and can be used to quantitatively evaluate the quality of the schedule obtained from the schedule generation algorithm. This procedure will thus be referred to as the "performance evaluation algorithm." A good initialization for the multipliers μ can be obtained by using the monotonicity of the successor and predecessor penalties in conjunction with the known multipliers π^*. This initialization is explained in detail in the Appendix, while the effects of the initialization will be examined in test cases of Subsection IV-C.

B-6) Comparison with Our Previous Results

The augmented Lagrangian decomposition and coordination method is very closely related to the Gauss–Seidel

method of Hoitomt, Luh, and Pattipati [12]. The decomposition of the augmented penalty functions is very similar in both methods, and both use the beginning times, interoperation times and machines from the previous iteration. The major difference between the two methods is in the operation level subproblems where the quadratic penalty term $(1/\epsilon_{ijl})(\bar{b}_{ij} - b_{ij})^2$ is used in (24). The augmented Lagrangian decomposition and coordination method has also been shown to converge in the continuous variable case. While convergence in the discrete variable case is an open question for both methods, the extra stability obtained by additional quadratic term is important. Another improvement is a new initialization of μ, as described in the Appendix, which can also be used for the Gauss–Seidel method.

C. Examples

Three examples are given below. The first is a small test problem designed to demonstrate the kind of information required to use the scheduling methodology. The next two examples use data obtained from Pratt and Whitney. These data relate to the scheduling of approximately 20 work centers, including all the numerically controlled (NC) machines at the Development Operations (DO) shop of Pratt and Whitney. There may be more than one machine per work center. All the multipliers except μ were initialized at zero, and the penalty coefficients p were initialized at 2.0. The penalty coefficients were then increased by a factor of 1.02 (i.e., $p^{n+1} = 1.02p^n$) if the associated precedence constraint was violated and the associated λ is positive, and decreased by 0.05 (i.e., $p^{n+1} = p^n - 0.05$) if the associated λ was not positive and the associated precedence constraint was not violated. The penalty coefficient $(1/\epsilon)$ was always set to p (i.e., $(1/\epsilon^n) = p^n$). The stopping criteria are as follows: for the schedule generation algorithm, the algorithm is stopped when the dual cost is within a certain percentage of the heuristic schedule cost. This percentage varies with the size of the problem. For the performance evaluation methodology, each job-based dual problem is run for 25 iterations.

The augmented Lagrangian decomposition and coordination method with the initialization described in the last section will be the nominal for comparison purposes. In order to test the effects of the initialization, the nominal algorithm is run a second time, with the multipliers μ initialized at zero. The lower bounds resulting from the two initializations are then compared. The nominal is also compared against the Gauss–Seidel method. The initializations are all exactly like the nominal, including initialization of μ. Lastly, the nominal is compared with a knowledge-based scheduler which was being utilized in DO shop. The knowledge-based scheduler was developed to operate interactively with humans. The human scheduler for the NC machine work centers had years of experience as a scheduler in the shop, and a significant amount of experience in using the interactive scheduler. All results except those on the knowledge based scheduler were obtained using a SUN SPARC2 workstation. Data and schedules are available upon request.

Example 1: For this problem, there are three different machines and four equally weighted jobs ($w_i = 1$). The planning horizon is 25 days (i.e., $K = 25$, and time unit = day). All machines are available on day one and throughout the planning horizon. All jobs are available for processing on day one, but are due on day 0. Each job is composed of three serial operations to be processed on three machines without required timeouts between operations. Data are shown in Table I. The lower bound is obtained at 474.6, and the feasible schedule has a cost $J = 475.0$ with a relative duality gap of 0.09%. Since the weights, process times and due dates are all integers numbers, the schedule must have an integer cost. Therefore, this schedule must be optimal. The resulting schedule is shown in the form of a Gantt chart in Table II. In the table, the notation "i, j" refers to operation j of job i. The time to solve the problem is 0:09 CPU minutes (9 seconds).

Example 2: In this example, 140 jobs with 186 operations are to be scheduled on 29 different machines (there are no identical machines). A number of these jobs have 6 or 7 operations, and each operation may be scheduled on up to 6 different machines. The time horizon is 214 days. Using the augmented Lagrangian decomposition and coordination method, the lower bound is obtained at 149,529, and the feasible schedule has a cost of 151,382, with a relative duality gap of 1.24%. The time to solve the problem is 0:53 CPU minutes. Starting with μ initialized to zero, the lower bound obtained was 134,874, which is 9.80% worse than the nominal lower bound. The Gauss–Seidel method generated a schedule of 152,289, with a relative lower bound of 147,093, and a relative duality gap of 3.53%. This required 1:16 minutes of CPU time. The schedule generated by the knowledge based scheduler was evaluated at 174,264, which is 15.12% worse than the nominal.

Example 3: For the last example, there are a total of 32 machines and all of them are different (32 machine types). There are 228 jobs, each consisting of one to seven operations, for a total of 335 operations. An operation may be performed on one of a set of machine types, with the number of eligible machine types ranging from one to six. The planning horizon is 414 working days, or about 1(1/2) years. Although these data were also obtained from the NC machine work centers, the number of machines is different from Example 2 because only those machines which are specifically required by operations under consideration are counted in each case. The nominal lower bound was 4,634,495, and the schedule cost was 4,976,491 with a relative duality gap of 7.38%. The computation time was 3:52 minutes. If the μ multipliers are initialized to zero, the lower bound was 4,523,070, which was 2.4% lower the nominal lower bound. The Gauss–Seidel method gave a schedule cost of 4,963,270, and a lower bound of 4,623,536, for a duality gap of 7.35%. The computation time was 4:04 minutes. The

TABLE I
DATA FOR EXAMPLE 1

Job i	Op. j	Mach. h	t_{ijh}	I_{ij}
1	1	1	4	—
	2	2	3	1
	3	3	2	2
2	1	2	1	—
	2	1	4	1
	3	3	4	2
3	1	3	3	—
	2	2	2	1
	3	1	3	2
4	1	2	3	—
	2	3	3	1
	3	1	1	2

TABLE II
GANTT CHART OF THE SCHEDULE FOR EXAMPLE 1

Time	1	2	3	4	5	6	7	8	9	10	11	12	13
Machine 1	1,1	1,1	1,1	1,1	2,2	2,2	2,2	2,2	4,3	3,3	3,3	3,3	
Machine 2	4,1	4,1	4,1	2,1	1,2	1,2	1,2	3,2	3,2				
Machine 3	3,1	3,1	3,1	4,2	4,2	4,2		1,3	1,3	2,3	2,3	2,3	2,3

knowledge-based scheduler generated a schedule with a cost of 6,504,614, which is 30.71% from the nominal schedule.

Several conclusions can be drawn from these examples and other tests based on Pratt and Whitney data not reported here. First, the initialization of the multipliers μ used in the evaluation improves both the computation time and the quality of lower bound. Second, both optimization methods outperform the existing knowledge based scheduler. Third, the difference between the augmented Lagrangian decomposition and coordination method and the Gauss–Seidel method appears to be inconclusive. Sometimes one method gives the better schedule, and sometimes the other method gives the better schedule. However, the stability of the augmented Lagrangian decomposition and coordination method appears to become more important as the number of operations per job are increased.

V. CONCLUSIONS

Three manufacturing scheduling problems have been formulated and solved based on Lagrangian relaxation techniques. Numerical results show that the methods obtain near-optimal schedules in a timely fashion. In the day-to-day usage of algorithms, the computation time can be further reduced by using the Lagrange multipliers from the last schedule to initialize the optimization process [16]. The multipliers can also be used as "pricing" information for "what if" analysis. Moreover, the decomposition approach makes the methodology ideal for bottom-up implementation. Specialized work centers can be modeled and included under the broad decomposition umbrella. Empirical studies have also shown that a high proportion of these algorithms is distributable, and that distributed implementation will further improve the computation time [11]. These features place the Lagrangian relaxation based algorithms at the forefront of a new generation of scheduling methodologies for manufacturing systems, delivering high quality schedules efficiently while supporting management goals.

Several issues are currently under investigation. The first is improving the ability of the methodology to handle jobs with many complicated precedence constraints, such as those required by complicated assembly structures with bills of materials. The effects of the nondifferentiable dual cost function on the convergence of the algorithm are also being studied. Methodologies for overcoming the resulting solution oscillation are being developed. Another item on the agenda is to make the methodology relatively insensitive to the time scaling as compared to the time horizon (e.g., there are 16 one-hour time units in one day or two eight-hour shifts in one day, depending on the time scale). The modeling of a new type of work center is also underway, where jobs are processed concurrently in batches, such as in a heat treat oven. Finally, further investigation is being conducted into the use of the methodology to perform "what if" analysis.

APPENDIX

INITIALIZING THE DUAL VARIABLES FOR THE PERFORMANCE EVALUATION ALGORITHM

As discussed in Subsection IV-B, a good initialization of multipliers from the schedule generation algorithm can decrease the computation time. The idea here is to initialize μ so as to set the immediate solution of (32) close to $\{b_{ij}^*\}$, the operation beginning times associated with the dual solution as obtained by the schedule generation algorithm. The reason is that the convergence of (28) depends on how quickly the precedence constraints can be satisfied. First, note that the "successor penalty" (30) can be rewritten as

$$\Gamma_{ijl}(k+1) = \Gamma_{ijl}(k) + \mu_{ijlk}. \tag{33}$$

Therefore, once a value for μ at some time k_0 is established, it is carried in all $\Gamma(k)$, for $k > k_0$, and previous values of $\mu(k \le k_0)$ should be considered in initializing μ for all $k > k_0$. Likewise, note the "predecessor penalty" (31) may be rewritten as

$$\Lambda_{ijl}(k-1) = \Lambda_{ijl}(k) + \mu_{ijl(k-1)}, \tag{34}$$

and that it increases as the time index decreases. Similar to the successor penalty, when a value for μ at some time k_0 is established, it is contained in all $\Lambda(k)$, for $k < k_0$, and later values of $\mu(k \ge k_0)$ can be considered in initializing μ for all $k < k_0$. For illustration purposes, linear precedence constraints will be considered, that is, I_{ij} will be assumed to consist of a single operation. The extension to multiple successors will be briefly discussed at the end.

Suppose that all the multipliers μ are zero, and consider the first operation of job i. With μ at zero, the minimization of (32) considers only the Π function in

determining the best beginning time.[5] How can $\{\mu_{i1lk}\}$ be initialized to place the beginning time at or before b_{i1}^* and *prevent an overlap with succeeding operation* (i,l)? In particular, suppose that $\Pi_{ijh}^*(b_{ij}^*+1)$ is less than $\Pi_{ijh}^*(b_{ij}^*)$, and b_{ij}^*+1 may be selected as the best beginning time of (32). If $\mu_{i1l(c_{ijh}^{s*}+1)}$ is set equal to $[\Pi_{i1h}^*(b_{i1}^*+1)-\Pi_{i1h}^*(b_{il}^*)]$, the cost function at time b_{i1}^*+1 would no longer be smaller than the cost at b_{i1}^*. At b_{i1}^*+2, Γ_{ijl} is also nonzero according to (33), and $\mu_{i1l}(c_{ijh}^{s*}+2)$ can again be used to prevent its selection in (32). This leads to the following method for initializing μ_{i1lk} for $l\in I_{i1}$ and $k>b_{i1}^*$:

$$\mu_{i1l(k+t_{ijh}+S_{ijl})}=\max\Big[0,\big(\Pi_{i1h}^*(k)+\Gamma_{i1l}(k+t_{ijh}+S_{ijl})\big)-\big(\Pi_{i1h}^*(b_{i1}^*)+\Gamma_{i1l}(c_{ij}^{s*})\big)\Big]. \quad (35)$$

The beginning time of the first operation can thus be placed at some time slot $k\le b_{i1}^*$. Note, however, that (35) cannot be used to prevent operation $(i,1)$ from starting before b_{i1}^*. That is because such a multiplier μ_{i1lk}, for $k<b_{i1}^*$, would be used in the computation of $\Gamma_{i1l}(c_{ij}^{s*})$. Therefore, using (35) to initialize multipliers for $k<b_{i1}^*$ would be futile.

A similar argument can be followed to initialize the multipliers μ_{ijN_ik} for the last operation of job i. The cost function (32) of the last operation includes a cost due to the weighted quadratic tardiness penalty, a cost due to Π, and a cost due to Λ. Similar to the idea above, except in reverse, *the goal is to initialize the multipliers so that the last operation does not overlap with its preceding operation.* In this case, the last operation is to be penalized by Λ if it starts before $b_{iN_i}^*$. Define T_{ik} as the tardiness for job i when the last operation ends at k. This tardiness penalty is monotonically decreasing as it moves backward from c_{ij}^*. Therefore, for k starting from $b_{iN_i}^*-1$ and proceeding backwards to 1:

$$\mu_{ijN_ik}=\max\Big[0,\big(w_iT_{i,k+t_{iN_ih}-1}^2+\Pi_{iN_ih}^*(k)+\Lambda_{ijN_i}(k)\big)-\big(w_iT_{i,c_{iN_i}^*}^2+\Pi_{iN_ih}^*(b_{iN_i}^*)+\Lambda_{ijN_i}(b_{iN_i}^*)\big)\Big]. \quad (36)$$

This procedure places the beginning time of the last operation at some time slot $k\ge b_{iN_i}^*$. Again, if (36) is utilized in initializing a multiplier for $k>b_{iN_i}^*$, the value of $\Lambda_{ijl}(b_{iN_i}^*)$ would be increased by the same amount, for a net effect of zero.

From the above reasoning, the initialization of a multiplier μ_{ijlk} associated with two adjacent operations (i,j) and (i,l) for $l\in I_{ij}$ can be split into two parts: the first part is for $k\in[1,b_{il}^*-1]$, while the second part is for $k\in[c_{ij}^{s*}+1,K]$. Similar to (35), one obtains for $k>b_{ij}^*$:

$$\mu_{ijl(k+t_{ijh}+S_{ijl})}=\max\Big[0,\big(w_iT_{i,k+t_{iN_i}-1}^2\Delta_{jN_i}+\Gamma_{ijl}(k+t_{ijh}+S_{ijl})+\Lambda_{imj}(k)+\Pi_{ijh}^*(k)\big)-\big(w_iT_{i,c_{iN_i}^*}^2\big)\Delta_{jN_i}+\Pi_{ijh}^*(b_{ij}^*)+\Gamma_{ijl}(c_{ij}^{s*})+\Lambda_{imj}(b_{ij}^*)\big)\Big], \quad (37)$$

where operation (i,m) is a predecessor operation (m: $j\in I_{im}$).

To initialize μ_{ijlk} for $k\in[1,b_{il}^*-1]$, an equation similar to (37) can be used, but based on the reasoning behind (36). Starting at b_{il}^*-1 and moving backward to 1,

$$\mu_{ijlk}=\max\Big[0,\big(w_iT_{i,k+t_{iN_i}-1}^2\Delta_{lN_i}+\Gamma_{ilm}(k+t_{ilh}+S_{ilm})+\Lambda_{ijl}(k)+\Pi_{ilh}^*(k)\big)-\big(w_iT_{i,c_{iN_i}^*}^2\big)\Delta_{lN_i}+\Pi_{ilh}^*(b_{il}^*)+\Gamma_{ilm}(c_{il}^{s*})+\Lambda_{ijl}(b_{il}^*)\big)\Big], \quad (38)$$

where $m\in I_{il}$. Examining (37) and (38) together, there may be some small overlap (i.e., the intervals $[c_{ij}^{2*},K]$ and $1,b_{il}^*]$ overlap since $c_{ij}^{s}*$ is likely to be less than b_{il}^*), causing interaction between the two initializations. This means that one may not be able to set the beginning times to the points desired, but this minor difficulty may be left to the job-level subgradient optimization to resolve. In the event there is a gap with $b_{il}^*<c_{ij}^{s*}$, the multipliers μ_{ijlk}, for $k\in[c_{ij}^{s*},b_{il}^*]$ are initialized to zero.

In extending to the case with multiple successors, the right hand of (37) can be equally divided among $\{\mu_{ijlk}\}_{l\in I_{ij}}$. Likewise, for the case with multiple predecessors, the right-hand side of (38) can be equally divided among $\{\mu_{ijlk}\}_{j:\,l\in I_{ij}}$.

[5]There is no precedessor penalty on the first operation, the weighted tardiness cost only applies to the last operation, and with μ set at zero, the successor penalty is also zero. Therefore, only Π needs to be considered.

ACKNOWLEDGMENT

The authors would like to thank Mr. Scott Bailey, Mr. Curtis Cook, and Mr. Jack Gibbs of Pratt and Whitney for their invaluable suggestions and support. An earlier version of this work has been presented as a plenary paper at the 1991 IFAC Workshop on Discrete Event System Theory and Applications in Manufacturing and Social Phenomena, Shenyang, P.R. China. They would also like to thank an anonymous reviewer for citation [2] and noting the relationship between the methodology in this paper and the Gauss–Siedel method of Hoitomt, Luh, and Pattipati.

REFERENCES

[1] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. New York: Academic, 1982.
[2] G. Cohen and D. L. Zhu, "Decomposition and coordination methods in large scale optimization problems: The nondifferentiable case and the use of augmented Lagrangians," in *Advances in Large Scales Syst.*, vol. 1, pp. 203–266, 1984.
[3] M. L. Fisher, "Optimal solution of scheduling problems using Lagrange multipliers, Part I," *Op. Res.*, vol. 21, pp. 1114–1127, 1973.
[4] ——, Lagrangian relaxation method for solving integer programming problems," *Management Sci.*, vol. 27, pp. 1–18, 1981.
[5] S. French, *Sequencing and Scheduling*. New York: Wiley, 1982.
[6] A. M. Geoffrion, "Lagrangean relaxation for integer programming," *Math. Programming Study*, vol. 2, pp. 82–114, 1974.
[7] S. B. Gershwin, "Hierarchical flow control: A framework for

scheduling and planning discrete events in manufacturing systems," *Proc. IEEE*, vol. 77, no. 1, pp. 195–209, Jan. 1989.
[8] S. C. Graves, "A review of production scheduling," *Operations Res.*, vol. 18, pp. 841–852, 1981.
[9] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical programming*, vol. 6, pp. 62–68, 1974.
[10] D. J. Hoitomt, P. B. Luh, E. Max, and K. R. Pattipati, "Scheduling jobs with simple precedence constraints on parallel machines," *Contr. Syst. Mag.*, vol. 10, no. 2, pp. 34–40, Feb. 1990.
[11] D. J. Hoitomt, J. B. Perkins, and P. B. Luh, "Distributed scheduling of job shops," in *Proc. 1991 IEEE Int'l. Conf. Robotics and Automat.*, Sacramento, CA, Apr. 1991.
[12] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job shop scheduling problems," *IEEE Trans. Robotics Automat.*, vol. 9, no. 1, pp. 1–13, Feb. 1993.
[13] J. G. Kimemia and S. B. Gershwin, "An algorithm for the computer control of production in a flexible manufacturing system," *Proc. IEEE*, 1981, pp. 628–633.
[14] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Bruckner, "Complexity of machine scheduling problems," *Annals Discrete Math.*, vol. 7, pp. 343–362, 1977.
[15] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.
[16] P. Luh, D. Hoitomt, E. Max, and K. Pattipati, "Schedule generation and reconfiguration for parallel machines," *IEEE Trans. Robot. Automat.*, vol. 6, no. 6, pp. 687–696, Dec. 1990.
[17] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization.* New York: Wiley, 1988.
[18] E. Nowicki and C. Smutnicki, "Worse case analysis of an approximation algorithm for flow shop scheduling," *Operations Res. Lett.* vol. 8, pp. 171–177, 1989.
[19] M. Numao and S. Morishita, "A scheduling environment for steel-making processes," in *Proc. Fifth Conf. Artificial Intelligence Appl.*, IEEE Computer Society, Miami, FL, Mar. 6–10, pp. 279–286, 1989.
[20] K. R. Pattipati, J. J. Shaw, J. C. Deckert, L. K. Beean, M. G. Alexandridis, and W. P. Lougee, "CONFIDANTE: A computer-based design aid for the optimal synthesis," analysis and operation of maintenance facilities," in *Proc. 1984 IEEE AUTOTESTCON*, Nov. 1984.
[21] B. T. Polyak, "Minimization of unsmooth functionals," *USSR Comput. Math., and Math. Physics*, vol. 9, pp. 14–29, 1969.
[22] N. R. Sandell, Jr., D. P. Bertsekas, J. J. Shaw, S. W. Gully, and R. F. Gendron, "Optimal scheduling of large-scale hydrothermal power systems," in *Proc. 1982 IEEE Int. Large-Scale Syst. Symp.*, Virginia Beach, VA, Oct. 1982, pp. 141–147.
[23] R. N. Tomastik and P. B. Luh, "The facet ascending algorithm for integer programming problems, submitted for publication 1993.

Peter B. Luh (SM'93) received his B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, Republic of China, in 1973, M.S. degree in aeronautics and astronautics engineering from M.I.T., Cambridge, MA, 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, and is currently a Professor in the Department of Electrical and Systems Engineering, and Director of Microprecision Manufacturing Program within the Advanced Technology Center for Precision Manufacturing of the State of Connecticut. His major research interests include schedule generation and reconfiguration for manufacturing systems, scheduling of power systems, distributed decisionmaking, game theory, and hierarchical planning and control of large scale systems. He has been a principal investigator and consultant to many industry and government funded projects in the above areas, and has published more than 130 papers. He has made significant contributions in manufacturing by developing a near-optimal and efficient schedule generation and reconfiguration methodology to improve on-time delivery of products and reduce work-in-progress inventory. The method has been adopted as the backbone of a new scheduling system developed by the Development Operations Shop of Pratt & Whitney, is currently in use for the scheduling of selected work centers of the shop on a daily basis, and has reshaped the scheduling community. He has also made significant contributions in power systems by developing a near-optimal and efficient unit commitment and hydro-thermal coordination methodology that is currently in use by a major electric utility company in New England on a daily basis; in distributed decisionmaking by developing normative-descriptive models to describe human team decisionmaking and coordination processes in distributed task processing environments; in game theory by advancing the knowledge of incentives, and designing and implementing an incentive scheme for a large petrochemical company; and in large-scale dynamic optimization by developing hierarchical algorithms with parallel processing structures and coordination mechanisms.

Dr. Luh is a Technical Editor for the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION (1990–1993) and Program Vice Chairperson for Invites Session for the 1993 IEEE International Conference on Robotics and Automation. He was an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATIC CONTROL (1989–1991) and the Chairperson of the Video Proceedings Committee producing the first and second Video Proceedings for the 1991 and 1992 IEEE International Conference on Robotics and Automation. He won the Best Paper Award of the 1987 Joint Command and Control Research Symposium and has served on Program Committees and Operating Committees of many national, international and inter-society conferences. He is a member of Sigma Xi and is listed in *Who's Who in Engineering*, *Who's Who in the East*, and *Who's Who in American Education.*

Debra J. Hoitomt (M'93) received the B.S. degree in natural science education from the University of Wisconsin, Madison, in 1977, the M.S. degree in systems and industrial engineering from the University of Arizona, Tucson, in 1984, and the Ph.D. degree in electrical and systems engineering from the University of Connecticut, Storrs, in 1990.

Beginning in 1987, she has worked with Pratt & Whitney managers and engineers to realize a next generation scheduler for their strategically important Product Development Center, for which she received Pratt & Whitney's Eagle Achievement Award in 1993. She is currently an Assistant Professor in Residence at the University of Connecticut, facilitating the transfer of innovative new technologies from the Precision Manufacturing Center to local industries. Her research interests include operations research methods as applied to practical planning and scheduling, and managing change in manufacturing systems; concurrent engineering and manufacturing techniques, and integration of software systems using object oriented approaches; and system reliability, performability and evaluation techniques.

Dr. Hoitomt is the Chairperson of the Robotics and Automation Society's Computer Aided Production Systems Committee, and is a member of Eta Kappa Nu and the Institute of Industrial Engineers.