SCHEDULING OF MANUFACTURING SYSTEMS USING THE LAGRANGIAN RELAXATION TECHNIQUE[1]

Peter B. Luh and Debra J. Hoitomt[2]

ABSTRACT

Scheduling is one of the most basic but the most difficult problems encountered in the manufacturing industry. Generally, some degree of time-consuming and impractical enumeration is required to obtain optimal solutions. Industry has thus relied on a combination of heuristics and simulation to solve the problem, resulting in unreliable and often infeasible schedules. Yet, there is a great need for an improvement in scheduling operations in complex and turbulent manufacturing environments. The logical strategy is to find scheduling methods which consistently generate good schedules efficiently. However, it is often difficult to measure the quality of a schedule without knowing the optimum.

In this paper, the practical scheduling of three manufacturing environments are examined in the increasing order of complexity. The first problem considers scheduling single-operation jobs on parallel, identical machines; the second one is concerned with scheduling multiple-operation jobs with simple fork/join precedence constraints on parallel, identical machines; and lastly, the job shop scheduling problem is considered, where multiple-operation jobs with general precedence constraints are to be scheduled on multiple machine types.

The Lagrangian relaxation technique is used to decompose each problem into job-level or operation-level subproblems, which are much easier to solve than the original problem and have intuitive appeals. This technique then results in algorithms which generate near-optimal schedules efficiently, while giving a lower bound on the optimal cost. In addition, the Lagrange multipliers from the previous schedule can be used to initialize the algorithms in the day-to-day scheduling operations, further reducing the computation time. All algorithms are demonstrated with examples using actual factory data. The successful application of the

---

[1] 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 Grant ECS-8717167. The authors would like to thank Mr. Scott Bailey, Mr. Curtis Cook and Mr. Jack Gibbs of Pratt & 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.

[2] Peter B. Luh is with the Department of Electrical & Systems Engineering, University of Connecticut, Storrs, CT, 06269-3157, and Debra J. Hoitomt is with Pratt & Whitney, East Hartford, CT, 06108.

Lagrangian relaxation technique to solve these real-world problems demonstrates the advantages of the technique over existing scheduling methods.

1. INTRODUCTION

Scheduling is one of the most important planning and operation issues in manufacturing, and its resolution can result in significant production cost savings. For example, IBM-Japan recently announced the development of a new scheduling system which is estimated to save a major steel company over a million dollars a year (Numao and Morishita, 1989). Despite specialized successful instances, the consistent generation of effective schedules remains a persistent problem for most manufacturing systems. As a result, 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 see high level management devoting much time and effort tracking the status of high priority jobs on the shop floor (Ashton and Cook, 1989).

Given the logistical and economic importance of scheduling problems, most of the earlier efforts centered on obtaining optimal schedules. Two prominent optimization methods are the branch and bound method (Fisher, 1973) and dynamic programming (e.g., French, 1982). Unfortunately, it has been proved that most scheduling problems of interest belong to the class of NP hard combinatorial problems, and the computational requirements for obtaining an optimal solution grow exponentially as the problem size (e.g., number of jobs and/or number of machines) increases (Lenstra, et al., 1977). These pure optimization methods are therefore impractical for industrial applications. As a result, materials planning systems (e.g., "Materials Requirements Planning" (MRP)) and simulation are often used for the high level planning purpose, while shop floor control systems determine schedules at the local level by using simple dispatching rules (Graves, 1981). In this attempt to decompose a scheduling problem, it is generally difficult for the high level planning system to provide the coordination needed to prevent local scheduling conflicts. Consequently the overall schedule is often infeasible. In addition, qualities of resulting schedules are difficult to assess. Recent research in heuristics has focused on the development of worst case bounds, i.e., what is the worst schedule a heuristic can generate, and how far is it from an optimum one. These bounds are generally difficult to obtain and are often far from the optimum. In using the Nowicki & Smutnicki, 1989 heuristic, for example, schedules may be more than 100% worse than optimal.

Beyond generating a schedule, there are a few other issues that a good scheduler must consider. In a manufacturing environment, dynamic changes are inevitable elements of the daily life. New jobs arrive, machines break down and even the process plans of some jobs may change. What is the impact of these changes on a schedule? How should a schedule be reconfigured to accommodate these changes? Should schedules be regenerated from scratch or simply respond locally to changes? Can information from the previous schedule be used in regenerating a schedule to reduce computation time? These are crucial issues in the day to day operations of any industrial scheduling system, but have neither been satisfactorily address in the literature nor realized in practice.

Manufacturing scheduling can be viewed as a decision problem with discrete decision variables, optimizing an objective function subject to relevant constraints. 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.

Lagrangian relaxation is a mathematical technique for solving constrained optimization problems (Nemhauser & Wolsey, 1988), and has recently emerged as a powerful method for solving complex scheduling problems to meet the above mentioned challenges. It is assumed that *on time delivery* is the objective function to be optimized (see problem formulation in subsection 2.1 and also comments in Section 5). By relaxing complicating "coupling" constraints, e.g., machine capacity constraints, using Lagrange multipliers, the method decomposes a problem into a number of smaller subproblems which are much easier to solve. Intuitively, the "hard" capacity constraints are converted into "soft" prices, with Lagrange multipliers acting as prices to regulate the use of machines. At the low level, the scheduling of a job can be easily determined by balancing job tardiness penalty against machine utilization costs, and this process is independent of the scheduling of other jobs. After all the low-level subproblems have been solved, the multipliers are adjusted at the high level as in a market economy so that capacity constraints are gradually satisfied over the iterations. By synergistically combining Lagrangian relaxation with heuristics, this method can generate feasible schedules efficiently for practical sized problems (within a few seconds of CPU time for simple manufacturing environments and a few minutes of CPU time for more complex environments, see examples in Sections 2, 3 and 4). Since the cost of the relaxed problem is a lower bound on the optimal cost, the quality of resulting schedules can be quantitatively evaluated. The schedules we obtained are near-optimal, mostly within a few percent of their respective lower bounds. Furthermore, Lagrange multipliers have the meaning of marginal costs (i.e., the increase in the objective function caused by a breakdown of a machine), and can be used to answer a wide range of "what if" questions. A schedule can also be efficiently reconfigured to accommodate dynamic changes by using multipliers from the previous schedule, further reducing computation time. This is because the status of a shop usually do not change drastically from one day to the next, therefore the prices should not differ to much either. Lagrangian relaxation is thus a powerful scheduling methodology for today’s complicated and dynamic manufacturing environments.

This paper presents a unified framework and intuitive explanation of the Lagrangian relaxation approach for several generic manufacturing environments. These problems build in complexity in a step-by-step fashion. The first problem presented in Section 2 considers the scheduling of single operation jobs on parallel, identical machines. It models the scheduling of a bottleneck work center with a limited number of machines. The machine capacity constraints are relaxed by using a set of Lagrangian multipliers. The method is extended in Section 3 where each job may have several operations to be performed on this bottleneck work center, and these operations are separated by timeouts (representing non-bottleneck operations). The problem is the scheduling of multiple-operation jobs on parallel, identical machines with simple fork/join type precedence constraints and timeouts. The precedence constraints for this problem are handled by enumeration. In Section 4, the job shop scheduling problem is considered with multiple machine types, general precedence constraints and simple routing decisions. This problem is solved by relaxing capacity constraints for each machine type using a set of Lagrange multipliers, and also relaxing precedence constraints using another set of Lagrange multipliers. Examples using actual factory data are provided for all the three problems in their respective sections. Concluding remarks are then given in Section 5. Although individual results on the three problems have

been reported in the literature, this step-by-step modeling, resolution, intuitive explanation and comments shed additional insight on the important, complex and dynamic manufacturing scheduling problems.

2. SCHEDULING SINGLE-OPERATION JOBS ON PARALLEL, IDENTICAL MACHINES

In this section, the non-preemptive scheduling of independent jobs with due dates on parallel, identical machines is considered[3]. Each job can be completed in a single operation, and is associated with a specified period of processing time, a due date and a measure of importance (weight). The objective is to optimize the overall on time delivery performance. This problem models the scheduling of a bottleneck work center with a limited number of machines.

2.1 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 weight w_i times the square of the tardiness T_i for each job, where the subscript i refers to the ith job.

$$J \equiv \sum_i w_i T_i^2 \tag{2.1}$$

The tardiness refers to the time past the due date d_i, and is defined as $\max[0, c_i - d_i]$ with c_i being the completion time of job i. This 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.

Machine capacity constraints state 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 available at that time M_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:

$$\sum_i \delta_{ik} \le M_k, \qquad k = 1,...,K, \tag{2.2}$$

where K is the time horizon under consideration. Without loss of generality, K is assumed to be long enough to complete all the jobs.

The processing time requirement for job i states that the elapsed difference between the beginning time b_i and the completion time c_i should be the processing time required t_i, i.e.,

$$c_i - b_i + 1 = t_i, \qquad i = 1,...,I, \tag{2.3}$$

---

[3] See also Luh, et al., 1990.

where I is the total number of jobs.

In the above formulation, the time horizon K, the number of jobs I, the weights $\{w_i\}$, processing time requirements $\{t_i\}$ and due dates $\{d_i\}$ of jobs and machine availability $\{M_k\}$ are assumed given. The scheduling problem is to select job beginning times $\{b_i\}$ so as to minimize the weighted quadratic tardiness J subject to machine capacity constraints and processing time requirements. Once beginning times are selected, completion times $\{c_i\}$, tardiness $\{T_i\}$ and number of active jobs at a given time can be easily derived. Note that the processing time requirements relate to individual jobs, and the objective function is also job-wise additive. Only capacity constraints *couple across jobs* and make the problem difficult.

2.2 Solution Methodology

Lagrangian relaxation is employed to relax the coupling capacity constraints (2.2) to obtain a set of decomposed subproblems. After the decomposition framework is derived, several steps to obtain a near-optimal solution will be presented: solving subproblems, updating multipliers, and constructing a feasible schedule.

The capacity constraint at time k is relaxed by using the Lagrange multiplier π_k to form the relaxed problem:

$$\min_{\{b_i\}} \{ \sum_i w_i T_i^2 + \sum_k \pi_k (\sum_i \delta_{ik} - M_k) \}; \tag{2.4}$$

subject to the processing time requirements (2.3). Then the dual problem is

$$\max_{\pi \geq 0} L, \quad \text{with } L \equiv \{ -\sum_k \pi_k M_k + \min_{\{b_i\}} \sum_i \{ w_i T_i^2 + \sum_k \pi_k \delta_{ik} \} \}; \tag{2.5}$$

subject to the processing time requirements (2.3). In the above, π_k is a non-negative real variable, and $\pi \equiv [\pi_1, \pi_2, .., \pi_K]^T$. This then leads to the following decomposed low-level subproblems, one for each job (given π):

$$\min_{1 \leq b_i \leq (K - t_i + 1)} L_i, \quad \text{with } L_i \equiv \{ w_i T_i^2 + \sum_{k=b_i}^{b_i + t_i - 1} \pi_k \}, \tag{2.6}$$

where the processing time requirement (2.3) has been incorporated into (2.6) by using the fact that δ_{ik} equals one when the job is active and zero otherwise. Equation (2.6) represents a decomposed scheduling subproblem for job i, and can be interpreted from a pricing perspective. The Lagrange multiplier π_k represents the price for utilizing a machine at time k. The cost L_i is thus balanced between the job tardiness penalty $w_i T_i^2$ and the cost for utilizing

a machine $\sum_{k=b_i}^{b_i+t_i-1}\pi_k$. To solve (2.6) for a given set of prices (multipliers), the cost L_i is computed for each possible value of b_i, and the optimal b_i^* is the beginning time yielding the lowest value of these L_i's. After all the job-level subproblems have been solved, the prices are adjusted at the high level as in a market economy so that capacity constraints are gradually satisfied over the iterations. The computational complexity for solving a subproblem is linear in K, since at most K evaluations of L_i is needed to determine b_i^*. This is contrast to the NP-hardness of the original problem. Also if job i arrives at time a_i (> 0), this earliest start date constraint can be easily accommodated by requiring b_i to lie between a_i and K- t_i +1 (rather than between 1 to K- t_i +1).

Let L_i^* denote the optimal cost of (2.6), then the high-level dual problem (2.5) can be rewritten as

$$\max_{\pi \geq 0} L, \quad \text{with } L \equiv \{ -\sum_k \pi_k M_k + \sum_i L_i^* \}. \tag{2.7}$$

Since subproblems involve discrete decision variables, the objective function L in (2.7) may not be differentiable at certain points in the π space. A subgradient method is therefore used to solve the dual problem (Polyak, 1969, Held, Wolfe and Crowder, 1974). In the algorithm, the multiplier π is updated iteratively according to

$$\pi^{n+1} = \pi^n + \alpha^n g(\pi^n), \tag{2.8}$$

where n is the high level iteration index, $g(\pi^n)$ is the subgradient of L with respect to π with the kth component given by $(\sum_i \delta_{ik} - M_k)$, and α^n (≥ 0) is the step size at the nth iteration. From (2.8), it can be seen that π_k increases when machines are over-utilized at time k and decreases otherwise, reinforcing the pricing interpretation of π. With the step size α^n given by

$$\alpha^n = \gamma \frac{\bar{L} - L^n}{g(\pi^n)^T g(\pi^n)}, \quad 0 < \gamma < 2, \tag{2.9}$$

where L^n is the value of L at the nth iteration and $\bar{L}$ (> L^n) an estimate of the optimal solution, this method converges at the rate of a geometric progression (Polyak, 1969). The stepsizing mechanism used here is a modified version of that suggested by Fisher, 1981, and also incorporates features used in Sandell, et al., 1982, and Pattipati, et al., 1984, with parameters selected based on testing experiences. The subgradient algorithm terminates after a fixed number of iterations has been reached.

The dual solution is generally associated with an infeasible schedule, i.e., capacity constraints (2.2) might be violated for a few time slots. Processing time requirements (2.3),

however, are always satisfied in view of how subproblems are solved. To construct a feasible schedule, a "list scheduling" heuristic is developed based on the optimal beginning times $\{b_i^*\}$ of the dual solution. A list is first created by arranging jobs in the ascending order of their respective optimal beginning times, and jobs are scheduled on machines according to this list as machines become available. If several jobs have the same optimal beginning times but there are not enough machines to start them all, a greedy heuristic determines which jobs should begin at that time slot and which jobs should be delayed by one time unit. In this greedy heuristic, the incremental change in tardiness cost if a job is delayed by one time unit is calculated. Competing jobs are then assigned to machines in the descending order of incremental changes, subject to machine availability for the remaining processing period of the jobs. When all the machines available at that time slot are assigned, the leftover jobs are delayed by one time unit, and compete with those jobs originally scheduled to start at the new time slot, and the process repeats.

The value of the objective function J for the feasible schedule is an upper bound on the optimal cost J^*. The value of the optimal dual function L^* obtained from (2.7), on the other hand, is a lower bound on J^* (Geoffrion, 1974; Nemhauser & Wolsey, 1988). 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 $\frac{J - L^*}{L^*}$ is thus a measure of the suboptimality of the feasible schedule with respect to the optimal one.

To examine the impact of dynamic changes, consider the situation where a machine breakdowns 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. 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 (Luh, et al., 1990).

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 in the number of jobs to be scheduled, increases or decreases in processing times of a few jobs, 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 Luh, et al, 1990. Another example will be provided in subsection 4.3 for the job shop scheduling problem.

2.3 Example[4]

In this example, 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 1581.65, 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 computer with all multipliers initialized at zero. This example takes data from the Tool and Die work center of Pratt & Whitney's Development Operations (DO) shop at East Hartford, Connecticut. This data and the resulting schedule are available upon request.

3. SCHEDULING MULTIPLE-OPERATION JOBS ON PARALLEL, IDENTICAL MACHINES

Now consider the non-preemptive scheduling of jobs each consisting of a small number of operations on parallel, identical machines by extending the model and solution methodology of the previous section[5]. The operations of a job must be undertaken in a particular order (precedence constraints or process plans), and the job is not available for scheduling between operations (a timeout) because of inspection or other processing which does not require the use of the machines under consideration. The precedence structure is assumed to be of the simple fork/join type with small numbers of operations as shown in Figure 1. In the figure, each operation is represented by a node, and the "forking" represents the case where several operations may be performed simultaneously. The time requirements for processing and timeouts are assumed to be known. As in Section 2, jobs have different due dates and levels of importance (weights), and the objective is to optimize the on time delivery performance. The problem is thus one step more complicated than that considered in Section 2.

3.1 Problem Formulation

Many aspects of this problem can be described by the problem formulation of subsection 2.1 with slight modifications. The capacity constraint can be stated as in (2.2) except that δ_{ik} now represents the number of operations of job i active at time k. The processing time requirement for operation j of job i (or operation (i, j)) is given by:

$$c_{ij} - b_{ij} + 1 = t_{ij}, \qquad i = 1,...,I;\ j = 1,...,N_i; \tag{3.1}$$

where b_{ij}, c_{ij} and t_{ij} are, respectively, the beginning time, completion time and processing time requirement of operation (i, j), and N_i is the number of operations of job i. In addition, let I_{ij} denote the set of operations immediately following operation (i, j). Then the precedence constraints can be stated as follows:

---

[4] More examples can be found in Luh, et al. 1990.

[5] See also Hoitomt, et al., 1990a.

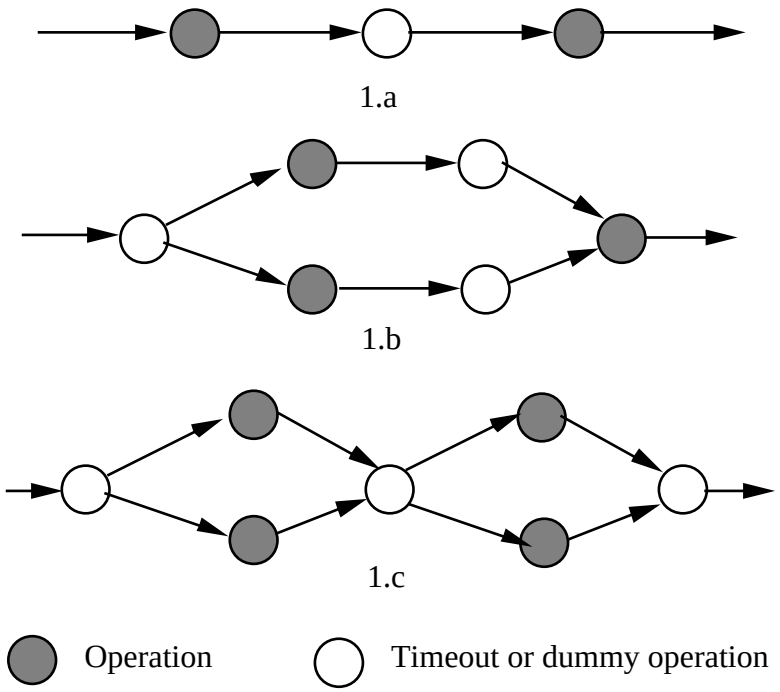


Figure 1 Examples of Simple Fork/Join Precedence Structures

(3.2) $c_{ij} + S_{ijl} + 1 \leq b_{il}, \quad i = 1,...,I; j = 1,...,N_i; l \in I_{ij};$

where S_{ijl} denotes the required timeout between operations (i, j) and (i, l) for $l \in I_{ij}$.

The scheduling problem is to select operation beginning times $\{b_{ij}\}$ so as to minimize the weighted quadratic tardiness penalty J (2.1) of all jobs subject to capacity constraints (2.2), precedence constraints (3.2), and processing time requirements (3.1).

3.2 Solution Methodology

To solve the problem, note that the Lagrangian relaxation equation (2.4) is still valid, except that individual operations are to be scheduled, and precedence constraints (3.2) and processing time requirements are to be satisfied. The decomposed low-level subproblem for job i is therefore given by

$$\min_{1\le b_{ij}\le (K-t_{ij}+1)} L_i, \quad \text{with } L_i \equiv \{ w_i T_i^2 + \sum_k \pi_k \delta_{ik} \}, \tag{3.3}$$

subject to precedence constraints (3.2) and processing time requirements (3.1). As previously, the multiplier π_k represents the price for 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.,

$$L_{ij} \equiv \sum_{k=b_{ij}}^{b_{ij}+t_{ij}-1} \pi_k, \tag{3.4}$$

where the processing time requirement has been embedded in the equation. Then (3.3) can be rewritten as the sum of the weighted quadratic tardiness penalty and the cost for using the machines, or

$$\min_{1\le b_{ij}\le (K-t_{ij}+1)} L_i, \quad \text{with } L_i \equiv w_i T_i^2 + \sum_{j=1}^{N_i} L_{ij}, \tag{3.5}$$

subject to precedence constraints (3.2). Because precedence constraints are restricted to the simple fork/join type with small N_i, the minimization of (3.5) is solved by enumeration. That is, L_i is computed for each possible value of $\{b_{ij}\}_{j=1}^{N_i}$ subject to precedence constraints, and $\{b_{ij}^*\}$ is the one yielding the lowest L_i. The complexity of solving (3.5) is therefore related to the number of operations, and is given by $O(K^{Ni})$.

Similar to (2.7), the high-level dual problem is

$$\max_{\pi\ge 0} L, \quad \text{with } L \equiv \{ -\sum_k \pi_k M_k + \sum_i L_i^* \}, \tag{3.6}$$

where L_i^* denotes the optimal value of (3.5). To solve the dual problem, the subgradient algorithm described in subsection 2.2 is used. 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 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

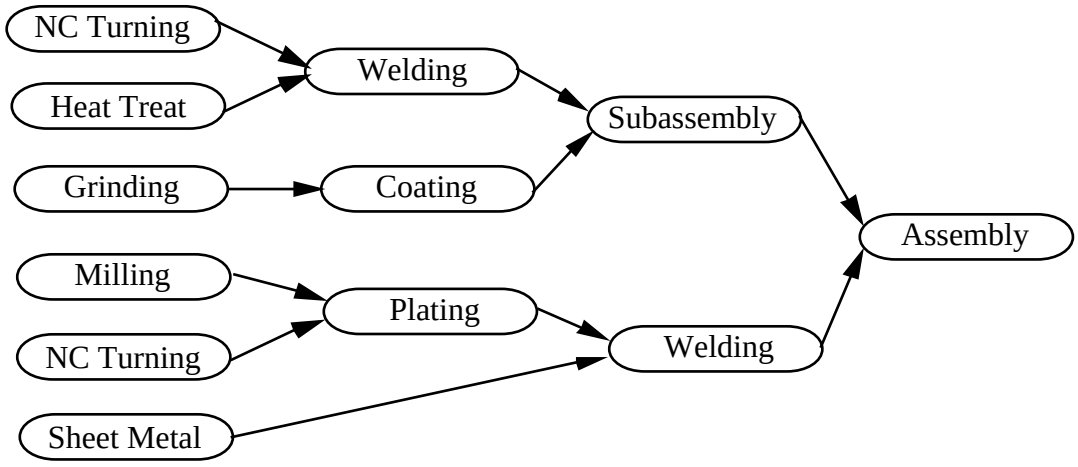
those delayed ones are checked to determine whether they should be delayed to preserve precedence constraints. As previously, the difference between the objective function J for the feasible schedule and the value of the optimal dual function L^* provides a quantitative measure of the quality of the resulting schedule.

3.3 Example

In this example, there are 44 machines and 112 jobs with a total of 210 operations. There are five weights, and the planning horizon is 247 days (time unit = day), almost a year of working days (excluding weekends and holidays). The lower bound on the optimal schedule is 1,018,131, while the cost of the feasible schedule is 1,018,433, a difference of 0.03%. The schedule was obtained in 17.0 CPU seconds on an IBM 3090 mainframe computer with the initial value of π_k equal to zero for all k. In this example, data from Pratt & Whitney's Tool and Die work center is again used. The data and the resulting schedule are available upon request.

4. JOB SHOP SCHEDULING

Now consider the scheduling of a job shop, a typical environment for the manufacturing of low-volume/high-variety products. In a job shop, machines are grouped into work centers according to functions, and without loss of generality, each work center is assumed to possess a set of parallel, identical machines with finite and maybe time varying capacity. Each job consists of a sequence of operations to be performed in a particular order, and travels to different work centers for processing. A sample process plan is shown in Figure 2. Each operation can be performed at one of a number of work centers, and the operation time requirement depends on the machine type selected to perform the operation. Compared to the problem of Section 3, the precedence constraints are not restricted now, and there are a number of different machine types. In addition, simple routing decisions must be made because operations may be performed on several machine types. These differences make the problem considerably more complex than that of Section 3[6].



[6] See also Hoitomt, et al., 1990b.

Figure 2. A Sample Process Plan

4.1 Problem Formulation

To formulate the scheduling problem with non-identical machines and precedence constrained operations, let H_{ij} denote the set of machine types which can perform operation (i,j). Equation (2.2) is first extended to account for the capacity of each machine type h:

$$\sum_{ij} \delta_{ijkh} \le M_{kh}, \tag{4.1}$$

where δ_{ijkh} is one if operation (i, j) is scheduled on machine type $h \in H_{ij}$ at time k, and zero otherwise. Let I_{ij} denote the set of operations immediately following operation (i, j), then the precedence constraints can be stated as:

$$c_{ij} + S_{ijl} + 1 \le b_{il}, \quad i = 1,...,I;\ j = 1,...,N_i;\ l \in I_{ij}; \tag{4.2}$$

where S_{ijl} is the required timeout between operations (i, j) and (i, l). Let t_{ijh} denote the processing time of operation (i, j) on machine type $h \in H_{ij}$. The processing time requirement holds for the machine type $h^* \in H_{ij}$ selected to perform the operation:

$$c_{ij} - b_{ij} + 1 = t_{ijh^*},\ i = 1,...,I;\ j = 1,...,N_i. \tag{4.3}$$

The job shop scheduling problem then consists of selecting operation beginning times $\{b_{ij}\}$ and machine types $\{h \in H_{ij}\}$ to optimize the weighted quadratic tardiness penalty of all jobs (2.1), subject to capacity constraints (4.1), precedence constraints (4.2) and processing time requirements (4.3).

4.2 Solution Methodology

In subsection 3.2, the original problem is decomposed into job-level subproblems. If a job has a large number of operations, the solution process would be very time consuming. In this section, capacity constraints are again relaxed by using Lagrange multipliers, one set of multipliers for each machine type. Furthermore, precedence constraints are relaxed by using another set of multipliers. In doing this, the original job shop scheduling problem is decomposed into much smaller operation-level subproblems.

In the first step of the solution process, the inequality precedence constraints (4.2) are converted into equality constraints using inter-operation times $\{s_{ijl}\}$ in preparation for the augmented Lagrangian relaxation of the problem (Bertsekas, 1982).

$$c_{ij} + s_{ijl} + 1 = b_{il}, \quad i = 1,...,I;\ j = 1,...,N_i-1;\ l \in I_{ij}; \text{ with} \tag{4.4}$$

$$s_{ijl} \ge S_{ijl}. \tag{4.5}$$

To decompose the problem, the augmented Lagrangian is formed as follows. Capacity constraints (4.1) are relaxed by using nonnegative Lagrange multipliers $\{\pi_{kh}\}$; precedence constraints (4.4) are relaxed by using Lagrange multipliers $\{\lambda_{ijl}\}$; and quadratic penalty terms with coefficients $\{p_{ijl}\}$ are added to the objective function. The augmented dual problem is then given by:

$$\max_{\lambda,\pi \geq 0} L, \quad \text{with } L \equiv \{ -\sum_{kh} \pi_{kh} M_{kh} + \sum_{i} \min_{\{b_{ij},\, s_{ijl},\, h \subset H_{ij}\}} \{ w_i T_i^2 + \sum_{j} [\sum_{k=b_{ij}}^{c_{ij}} \pi_{kh} + \sum_{l \subset I_{ij}} [\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - b_{il}) + \frac{p_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - b_{il})^2]] \ \} \ \}, \tag{4.6}$$

where the minimization within (4.6) are subject to (4.5). In the above, π_{kh} is a non-negative real variable, and λ_{ijl} and p_{ijl} are real variables. The minimization within (4.6) are actually job-level subproblems:

$$\min_{\{b_{ij},\, s_{ijl},\, h \subset H_{ij}\}} L_i, \quad \text{with } L_i \equiv \{ w_i T_i^2 + \sum_{j} [\sum_{k=b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh} + \sum_{l \subset I_{ij}} [\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - b_{il}) + \frac{p_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - b_{il})^2] \ ] \ \}. \tag{4.7}$$

The quadratic penalty terms are needed to prevent solution oscillation, as can be seen from the following reasoning. Without the penalty terms (i.e., $p_{ijl} \equiv 0$), the minimization within (4.7) can be decomposed into operation-level subproblems by appropriately re-grouping individual terms. If operation (i, j) is not the last operation of job i (i.e., $j \neq N_i$), its beginning time b_{ij} is selected as a tradeoff between the cost for utilizing a machine $\sum_{k=b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh}$ versus the cost for violating precedence constraints $[\sum_{l \subset I_{ij}} \lambda_{ijl} - \sum_{l\,:\,j \subset I_{il}} \lambda_{ilj}]\, b_{ij}$, a linear function of b_{ij}. Many times this linear term dominates the resource utilization cost, and b_{ij}^* would be very small if the coefficient of b_{ij} is positive (i.e., the multipliers associated with succeeding operations (the first term in the linear coefficient) is larger than then the multipliers associated with preceding operations (the second term in the linear coefficient)); otherwise, b_{ij}^* would be very large. The resulting b_{ij}^* is therefore sensitive to small changes in multipliers, and may oscillate over iterations. The quadratic penalty terms add the fine tuning necessary for the balance between scheduling early versus scheduling late to fit all operations of a job together.

Quadratic penalties, however, introduce cross product terms involving operations (i, j) and (i, l) for $l \in I_{ij}$. The job-level subproblem (4.7) therefore cannot be further decomposed into operation-level subproblems. Since each job may contain many operations, the solution

to (4.7) via enumeration as done in Section 3 is impractical. To overcome this difficulty, an *iterative approach* is taken. The operation beginning times of a job are solved in a sequential order, starting from the first operation. In selecting b_{ij}, all other relevant operation beginning times (b_{il}, $l \in I_{ij}$ for succeeding operations and b_{il}, $j \in I_{il}$ for preceding operations) are assumed fixed, taken from their latest computed values (or from initialization). With all other relevant beginning times fixed, b_{ij} can be easily determined. After all the operation beginning times of a job have been obtained, if the solutions to the current iteration are the same as the solutions to the previous iteration (or initialization), the process converges and $\{b_{ij}^*\}_{j=1}^{N_i}$ is obtained. Otherwise, these newly computed operation beginning times are used to start the next iteration, until the process converges or a fixed number of iterations has been reached. This is the so called "Gauss-Seidel iterative technique" (Bertsekas, 1989), where an operation-level subproblem is given by:

$$\min_{b_{ij},\, s_{ijl},\, h \subset H_{ij}} \{ w_i T_i^2 \Delta_{jN_i} + \sum_{b_{ij}}^{b_{ij}+t_{ijh}-1} \pi_{kh} + \sum_{l \subset I_{ij}} [\lambda_{ijl}(b_{ij} + t_{ijh} + s_{ijl} - \bar{b}_{il}) + \frac{p_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - \bar{b}_{il})^2]$$
$$+ \sum_{l:\, j \subset I_{il}} [\lambda_{ilj}(\bar{b}_{il} + t_{ilh^*} + \bar{s}_{ilj} - b_{ij}) + \frac{p_{ilj}}{2}(\bar{b}_{il} + t_{ilh^*} + \bar{s}_{ilj} - b_{ij})^2] \}, \tag{4.8}$$

subject to (4.5), where Δ_{jN_i} =1 if (i, j) is the last operation of job i and 0 otherwise; $\bar{b}_{il}$ and $\bar{s}_{ilj}$ are respectively the most recently computed beginning time and inter-operation time related to operation (i, l); and h^* is the machine type selected to perform the preceding operation (i, l) for $j \in I_{il}$.

In (4.8), there are three variables to be selected: operation beginning time b_{ij}, inter-operation time s_{ijl}, and machine type h. The idea is to obtain s_{ijl} as a function of the other two variables b_{ij} and h, and then enumerate through b_{ij} and h. It can be easily shown that with b_{ij}, h, $\bar{b}_{il}$, and $\bar{s}_{ilj}$ given, the optimal value of s_{ijl} is $\max[S_{ijl}, s_{ijl}^r]$, where s_{ijl}^r is the rounded integer value of $[\bar{b}_{il} - b_{ij} - t_{ijh} - \frac{\lambda_{ijl}}{p_{ijl}}]$. With s_{ijl} expressed in terms of the other two decision variables, b_{ij}^* and h^* can then be obtained by enumerating the beginning times from 1 through the time horizon K for each possible machine type $h \in H_{ij}$, and directly comparing the costs derived from all possibilities. As mentioned, the Gauss-Seidel iteration for job i converges if $\bar{b}_{ij} = b_{ij}$ and $\bar{s}_{ijl} = s_{ijl}$ for all j and l. If the Gauss-Seidel iteration does not converge within a fixed number of iterations, the solution which generates the minimal job cost of (4.7) is used. To obtain the job cost of (4.7), each operation cost of (4.8) must be reduced by the following amount before adding up to avoid double counting of precedence related costs:

$$\sum_{l \subset I_{ij}} [\frac{\lambda_{ijl}}{2}(b_{ij} + t_{ijh} + s_{ijl} - \bar{b}_{il}) + \frac{p_{ijl}}{4}(b_{ij} + t_{ijh} + s_{ijl} - \bar{b}_{il})^2] \tag{4.9}$$

$$+ \sum_{l:\, j \subset I_{il}} [\frac{\lambda_{il|j}}{2}(\bar{b}_{il} + t_{il h^*} + \bar{s}_{il|j} - b_{ij}) + \frac{p_{il|j}}{4}(\bar{b}_{il} + t_{il h^*} + \bar{s}_{il|j} - b_{ij})^2].$$

To solve the high-level dual problem, the subgradient method of subsection 2.2 is used to update the Lagrange multipliers π. The Lagrange multiplier λ is adjusted according to the multiplier method update formula (Bertsekas, 1982):

$$\lambda^{n+1} = \lambda^n + p^n g(\lambda^n), \tag{4.10}$$

where n is the high level iteration index, p^n the penalty coefficient at the nth iteration, and $g(\lambda^n)$ the subgradient of L in (4.6) with respect to λ. The subgradient component relating operations (i, j) and (i, l) for $l \in I_{ij}$ is $(b_{ij} + t_{ijh} + s_{ij|} - b_{il})$. The penalty coefficient $p_{ij|}$ is multiplied by a factor ζ (> 1) periodically (every five or ten iterations) if the subgradient component of $g(\lambda^n)$ is nonzero at that iteration. The parameter ζ is selected so that $p_{ij|}$ is non-decreasing but not so large that the problem becomes ill-conditioned (Bertsekas suggests $\zeta \in$ [4,10], p. 123).

The Lagrange multiplier π_{kh}, as before, represents the price for utilizing a machine of type h at time k. The Lagrange multiplier $\lambda_{ij|}$ relaxing precedence constraints (4.4) 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 time out $S_{ij|}$ by one time unit. From these, a variety of "what if" questions can be answered.

The quality of high-level multiplier updates depends on the convergence of the Gauss-Seidel iteration. Since the Gauss-Seidel method is not guaranteed to converge to an optimum, a limited Armijo type line search (Luenberger, 1984) is employed for the effective updating of λ and π. The algorithm is stopped when a fixed number of high-level iterations has been reached. As previously, the dual solution 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 operation beginning times forward in time if necessary. The list scheduling technique, as presented in subsection 3.2, is then applied to construct a feasible schedule.

Since the Gauss-Seidel technique may not generate an optimal solution for each job, the dual cost L of (4.6) is no longer a lower bound for the optimal cost J^*. To evaluate the quality of the resulting schedule, a second problem formulation is adopted. This problem formulation replaces the precedence constraint (4.2) by

$$\omega_{ij|k} + \sigma_{il k} \le 1, \qquad i = 1,..., N;\ j = 0,..., N_i;\ k = 1,...,K;\ l \in I_{ij}, \tag{4.11}$$

where $\omega_{ij|k}$ is an integer variable equal to one for every time unit $k \le c_{ij} + s_{ij|}$, $l \in I_{ij}$, and zero otherwise; and $\sigma_{il k}$ is an integer variable equal to one for every time unit $k \ge b_{il}$ and zero

otherwise. It can be easily checked that (4.2) and (4.11) are equivalent representations of precedence constraints. The problem now is to maximize the weighted quadratic tardiness (2.1) subject to capacity constraints (4.1), precedence constraints (4.11), and processing time requirements (4.3). This problem can in principle be solved by using the standard Lagrangian relaxation technique without the solution oscillation difficulty mentioned earlier. Prohibitive computational requirements, however, would be needed in view of the dimensionality of the Lagrange multipliers μ_{ijlk} relaxing (4.11). By fixing the capacity constraint multiplier π at the value obtained from the first problem formulation, this new dual problem can be decomposed into job-level subproblems. The multiplier μ can then be obtained *for each job* by using the 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 list scheduling technique.

4.3 Examples

This example draws data from about twenty work centers relating to numerical control (NC) machines of Pratt & Whitney's DO shop. There are a total of 29 machines and all of them are different (29 machine types). There are 140 jobs, each consisting of one to seven operations, for a total of 186 operations. An operation may be performed on one of several (up to six) machine types. Required timeouts of varying lengths also exist between operations. The jobs are characterized by different due dates and may have one of five different weights. The planning horizon is 214 working days. Not all the machines are immediately available as some of them are busy processing jobs already started.

All the multipliers were initialized to zero, and penalty coefficients p were initially set to 0.5 and doubled every five iterations for those coefficients associated with violated precedence constraints. The lower bound is obtained at 144,117, and the feasible schedule has a cost 151,382 with a relative duality gap 5.05%. The time to solve the problem is 4.17 CPU minutes on an IBM 3090 mainframe computer. The existing schedule generated by a knowledge-based scheduler was evaluated at 174,265, or 20.92% above the lower bound.

As another example, the NC machine group at the DO shop was scheduled over a three week period in Spring 1991, where the schedule was updated to reflect the arrival and departure of jobs and the latest information regarding machine status, process plans and processing times. As in the previous example, each machine type has one machine, each job may have up to seven operations, and each operation may be performed on one of several different machine types. With each schedule reconfiguration, the multipliers from the previous schedule were used to initialize the algorithm. The results are shown in Table 1 below. In the table, the problem size is MxN, where M is the number of machine types and N is the number of jobs. Note that M varies slightly because only those machines required by jobs are counted. The time horizon K is 300 days in each case, and CPU times are in minutes for an IBM 3090 mainframe computer.

Table 1. Schedule Generation and Reconfiguration for NC Machines

Date	Problem Size	Cost J	Lower Bound	Duality Gap	CPU Time
4/17	32x126	274,736	264,488	3.9%	2.6 min.
4/19	32x143	278,109	267,094	4.1%	2.8 min.
4/22	34x141	280,746	271,110	3.6%	2.9 min.
4/24[7]	33x133	531,851	531,203	0.1%	3.0 min.
4/26	32x135	268,828	266,665	0.8%	2.8 min.
4/29	32x128	268,504	266,122	0.9%	2.2 min.
5/1	32x124	270,585	267,442	1.2%	2.3 min.
5/3	33x127	247,843	243,192	1.9%	2.1 min.

The consistent near-optimal quality of the schedules is demonstrated by the fact that all the schedules are within 5% of their respective lower bounds. The computation time has also

[7] A few single-operation, high priority tardy jobs arrived on this day, driving up the schedule cost. Since these jobs immediately went into processing, the schedule cost returns to normal on 4/26.

been significantly improved over the previous example, since the multipliers and penalty coefficients were initialized based on results from the previous schedule.

5. CONCLUDING REMARKS

Three scheduling problems have been formulated and solved in the increasing order of complexity using the Lagrangian relaxation technique. It is worth mentioning that the approach relies heavily on the additive nature of the tardiness objective function. Since the objective function is job-wise additive, the problem can be decomposed into the scheduling of individual jobs once the coupling capacity constraints are relaxed. If the objective function were not additive, the relaxed problem could not be decomposed into the scheduling of individual jobs. This is the case for the most widely used objective function in the scheduling literature - the minimization of total production time, or the so called "makespan". The relaxation approach would fail if makespan were the objective function. Furthermore, the most important thing for a scheduling system is to meet on time delivery. If a schedule minimizes makespan but finishes a very important tardy job the last, the scheduling system is not doing its job. The tardiness related objective function therefore makes good sense for shop management, and also renders decomposability which is crucial for practical sized problems.

Numerical results for the three problems considered here show that the method generates near-optimal schedules in a timely fashion. The method can also answer a wide range of "what if" questions, and accommodate dynamic changes in the day-to-day scheduling operations. The success of this step-by-step modeling and resolution demonstrates the power of the approach for capturing the complexity, diversity and dynamics of many manufacturing systems, and places it at the forefront of a new generation of scheduling methodologies.

REFERENCES

1. Ashton, J. E., and F. W. Cook, Jr., "Time to Reform Job Shop Manufacturing," *Harvard Business Review*, March/April, 1989, pp. 106-111.
2. Bertsekas, D.P., and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice Hall, Englewood Cliffs, NJ, 1989.
3. Bertsekas, D.P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, NY, 1982.
4. Fisher, M.L., "Optimal Solution of Scheduling Problems Using Lagrange Multipliers, Part I," *Operations Research*, Vol. 21, 1973, pp. 1114-1127.
5. Fisher, M.L., "Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, Vol. 27, 1981, pp. 1-18.
6. French, S., *Sequencing and Scheduling*, Wiley, New York, 1982.
7. Geoffrion, A.M., "Lagrangian Relaxation for Integer Programming," *Mathematical Programming Study*, Vol. 2, 1974, pp. 82-114.
8. Graves, S.C., "A Review of Production Scheduling," *Operations Research*, Vol. 18, 1981, pp. 841-852.
9. Held, M., P. Wolfe and H. P. Crowder, "Validation of Subgradient Optimization," *Mathematical Programming*, Vol. 6, 1974, pp. 62-68.

10. Hoitomt, D.J., P.B. Luh, E. Max, K.R. Pattipati, "Scheduling Jobs with Simple Precedence Constraints on Parallel Machines," *Control Systems Magazine*, Vol. 10, No. 2, Feb. 1990a, pp. 34-40.
11. Hoitomt, D.J., P.B. Luh, K.R. Pattipati, "Job Shop Scheduling," *Proceedings of the First International Conference on Automation Technology*, Taipei, Taiwan, July 1990b, pp. 565-574.
12. Lenstra, J.K., A.H.G. Rinnooy Kan, P. Bruckner, "Complexity of Machine Scheduling Problems," *Annals of Discrete Mathematics*, Vol. 7, 1977, pp. 343-362.
13. Luenberger, D. G., *Linear and Nonlinear Programming*, second edition, Addison-Wesley, 1984.
14. Luh, P., D. Hoitomt, E. Max, K. Pattipati, "Schedule Generation and Reconfiguration for Parallel Machines," *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 6, Dec. 1990, pp. 687-696.
15. Nemhauser G.L., and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, Inc., New York, 1988.
16. Nowicki, E., and C. Smutnicki, "Worst Case Analysis of an Approximation Algorithm for Flow Shop Scheduling," *Operations Research Letters*, Vol. 8, 1989, pp. 171-177.
17. Numao, M., and S. Morishita, "A Scheduling Environment for Steel-Making Processes," *Proceedings of the Fifth Conference of Artificial Intelligence Applications*, IEEE Computer Society, Miami, Florida, March 6-10, 1989, pp. 279-286.
18. Pattipati, K.R., J.J. Shaw, J.C. Deckert, L.K. Beean, M.G. Alexandridis, W.P. Lougee, "CONFIDANTE: A Computer-Based Design Aid for the Optimal Synthesis, Analysis and Operation of Maintenance Facilities," *Proceedings of the 1984 IEEE AUTOTESTCON*, Nov. 1984.
19. Polyak, B.T., "Minimization of Unsmooth Functionals", *USSR Computational Math. and Math. Physics*, Vol. 9, 1969, pp. 14-29.
20. Sandell, N.R. Jr., D.P. Bertsekas, J.J. Shaw, S.W. Gully, R.F. Gendron, "Optimal Scheduling of Large-Scale Hydrothermal Power Systems," *Proceedings of the 1982 IEEE International Large-Scale Systems Symposium*, Virginia Beach, Virginia, Oct. 1982, pp. 141-147.