Schedule Generation and Reconfiguration for Parallel Machines

PETER B. LUH, MEMBER, IEEE, DEBRA J. HOITOMT, ERIC MAX, AND KRISHNA R. PATTIPATI, MEMBER, IEEE

Abstract—**Scheduling is one of the most important issues in the planning and operation of manufacturing systems. In practice, however, schedules are often generated by using either simple heuristic rules with questionable performance or expensive computer simulations. The interaction of jobs as they compete for limited resources is not visible, dynamic changes in the system (e.g., machines break down, jobs require more time than anticipated, etc.) cannot be easily accommodated, and job completion dates cannot be accurately predicted or controlled. This paper presents a methodology for scheduling independent jobs with due dates on identical, parallel machines. The jobs have different levels of importance and various proessing times on the machines, and the objective is to minimize the total weighted job tardiness of the schedule. Since the problem is n-p-hard, our goal is not to obtain the optimal schedule. Rather, we present an efficient near-optimal algorithm based on Lagrangian relaxation. A nice feature of this approach is that it provides a lower bound on the cost, which can be used as a measure of suboptimality. According to our implementation for a work center at Pratt and Whitney, most schedules generated are within 1% of the optima with reasonable CPU times. Furthermore, the method provides valuable job interaction information, which shop floor management uses to answer "what if" questions, to reconfigure the schedule to accommodate dynamic changes, and to schedule new jobs.**

I. Introduction

A. The Scheduling Problem

SCHEDULING is one of the most important issues in the planning and operation of manufacturing systems. It arises in situations wherein resources are to be assigned over time to perform a set of activities. Many productivity problems in the United States are associated with scheduling problems: not having the right items when they are needed, not having equipment available when it is needed, using excess inventory to "hide" problems, and inflexibility and lack of responsiveness. Resolution of these difficulties can translate into considerable direct and indirect savings. For example, the introduction of the Gantt charts during World War I reduced ship loading time by about half (see Preface of [30]).

Although it is an effective means of "representing" a schedule, a Gantt chart is not a scheduling method. Finding methodologies that consistently generate "good" schedules has concerned and frustrated researchers and practitioners alike. The difficulty is that most scheduling problems belong to the class of n-p-hard combinatorial problems, for which the development of optimal polynomial algorithms is unlikely. In view of the increasing complexity of production operations, scheduling problems faced even by small- to medium-sized companies are beyond the reach of most existing mathematical techniques. Consequently, practical scheduling problems are mostly "solved" by practitioners using simple heuristic algorithms and/or brute force computer simulations that find feasible, but not necessarily "good," schedules [15]. In these methods, the interaction of jobs as they compete for limited resources is not visible, the job completion dates cannot be accurately predicted or controlled.

In this paper, we consider the nonpreemptive scheduling of independent jobs with due dates on identical, parallel machines. For each job, a machine is required for a specified period of time, and the objective is to minimize the total weighted tardiness of jobs. Identical, parallel machines constitute a single type of resource and, for the purposes of this paper, will be referred to as one resource.

B. Research on Relevant Scheduling Problems

One common way of solving a scheduling problem is to determine a lower bound on the cost and then use a branch-and-bound method to determine the optimal solution. Picard and Queyranne [30] obtain a lower bound by finding the shortest paths in a multipartite network for a single machine scheduling problem. Gupta and Sen [16] and Amar and Vasilescu [1] present improvements to branch-and-bound algorithms by using the concept of "entrapment" for both the single and parallel machine problems. Fisher [7] uses Lagrangian relaxation to decompose a job shop problem into a set of subproblems, where each relates to a single job. He then uses the dual cost as a lower bound in the branch-and-bound method to obtain an optimal schedule. Another method for optimally solving a scheduling problem is dynamic programming (see, e.g., [10]). Unfortunately, both the branch-and-bound method and the dynamic programming method require exponential computation time in the worst case. Additionally, a problem must be solved again whenever changes occur (e.g., machines break down, jobs require more time than anticipated, etc.) during the course of the planning horizon.

Another approach to scheduling problems has been to obtain suboptimal solutions in polynomial time. Sahni [33] and Horowitz and Sahni [22] examine polynomial time approximation schemes based on dynamic programming methods for solving scheduling problems. These algorithms con-

Manuscript received November 14, 1988; revised April 17, 1990. This work was supported by the National Science Foundation under Grants ECS-8513163 and ECS-8717167.

P. B. Luh and K. R. Pattipati are with the Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06269-3157.

D. J. Hoitomt and E. Max are with Pratt and Whitney, East Hartford, CT 06108.

IEEE Log Number 9038620.

1042-296X/90/1200-0687$01.00

centrate on a few specialized areas (notably, completion time-based cost functions such as the weighted completion time and the average completion time). Similar results for the weighting tardiness cost function have not been forthcoming [15] and the prospect of developing polynomial time approximation schemes is considered unlikely [23].

In reality, most of the theoretical results cited above are not used in industry [15]. The problems of flexibility, schedule robustness, and efficiency for large-scale problems have not been satisfactorily addressed. Thus, in practice, a commercial planning package such as material requirements planning (MRP) is often implemented in job shops in conjunction with a shop floor control system [15], [32]. MRP acts as a reporting system while the shop floor control system determines schedules on the local level, such as a work center, by using heuristics or simple dispatching rules.

Heuristics have the advantage over branch-and-bound and dynamic programming in terms of efficiency and the ability to react to dynamic changes; heuristics also have the advantage over polynomial time approximation schemes in terms of widespread applicability. Examples include the use of earliest due date (EDD) rules to schedule jobs on parallel machines, sequencing based on the shortest or longest processing time (SPT, LPT), or ordering jobs based on the weighted shortest process time (WSPT) (see, e.g., [2]). Recent research in heuristics includes Fry *et al.* [11], where the so-called weighted absolute deviation is used as a cost function that penalizes earliness as well as tardiness. A method for inserting idle time into a schedule is then shown to improve the performance of the EDD heuristic. Zhao and Ramamritham [36] develop a rule that combines several rules (such as EDD and SPT) in a weighted fashion. For instance, schedule the job that has the smallest value of due date plus 20% of the processing time. They find that this works better than any single rule by itself. In general, however, heuristics offer no guarantee that the solution is within an acceptable margin of error when compared with the optimal solution, and if an error bound exists for a heuristic, it is often quite large. For example, the cost of LPT schedules may be 30% larger than the optimum, and one of the bin-packing algorithms used to solve parallel machine problems may produce up to 22% error in the worst case (see [4]).

It is generally concluded that a gap exists between scheduling theory and practice [15], [32]. Practical methods react to dynamic changes without the ability to reliably produce a good schedule, and theoretical methods produce good schedules without the ability to react to dynamic changes. In this paper, we hope to close this gap by introducing a scheduling methodology capable of producing a near optimal schedule with the ability to react to dynamic changes.

C. Scope of this Paper

In this paper, we consider the nonpreemptive scheduling of independent jobs with due dates on identical, parallel machines. For each job, a measure of importance (weight) is given, and a machine is required for a specified period of time. The objective is to minimize the total weighted tardiness of jobs. This problem description is an abstraction of a real production environment and will be formulated as an integer programming problem in Section II. Unfortunately, the problem, as most other scheduling problems, is n-p hard [24], and no polynomial time algorithms are known to exist. Our goal, therefore, is not to obtain the optimal scheduling solution. Rather, we want to obtain an efficient near-optimal solution with measurable performance as well as important job interaction information to accommodate dynamic changes and to handle new jobs.

To achieve this goal, a Lagrangian (dual) relaxation method is developed in Section III to solve the integer programming formulation of the problem. This technique allows for decomposition by job of the minimization problem into subproblems, each of which may be solved in linear time. The Lagrange multipliers, on the other hand, are updated by a subgradient algorithm which has linear convergence [31].

At the termination of the subgradient algorithm, the dual solution may not be feasible. To obtain a feasible schedule, the dual solution is used to form an ordered listing of jobs. A greedy method is then applied to the list to assign jobs on machines (listing scheduling). Since the dual solution yields a lower bound on the optimal cost [14], a schedule generated here can be measured quantitatively against the lower bound. According to our implementation for a work center of about 40 machines and 80 to 120 jobs at Pratt and Whitney, most results are within 1% of the lower bounds and are obtained in less than 20 CPU seconds on an IBM 3090 mainframe computer. Numerical testing results are provided in Section IV.

The optimized Lagrange multipliers reflect the sensitivity of the objective function with respect to resource levels and are sometimes referred to in this paper as "job interaction" variables. They are used in conjunction with the above-described method to provide quick answers to "what if" questions and to reconfigure an existing schedule whenever changes occur in job characteristics or in resource availability. Furthermore, the method can be extended to schedule new jobs. A relatively small number of new arrivals may be scheduled by using the existing job interaction variables. New jobs can also be scheduled with partial or full update of the job interaction variables. These capabilities have enormous value for researchers and practitioners alike and would result in considerable direct and indirect savings. The "what if" and schedule reconfiguration features of the algorithm are discussed in Section V. The corresponding numerical results are presented in Section VI. An updated version of the algorithm described in this paper, including the "what if" features, is currently in use to generate operational schedules for a work center at Pratt and Whitney.

This problem is the first in a set of increasingly difficult scheduling problems. The level of difficulty increases along three lines: single operation (SO) or multiple operation (MO) jobs; no precedence (NP) constraints, simple fork/join precedence (SP) constraints, or generic precedence (GP) constraints; and identical machines (IM) or nonidentical machines (NM). Lagrangian relaxation has been the crux of the investigation into practical scheduling methodologies for solving a series of these problems. Preliminary results for the

problem considered in this paper (SO/NP/IM) have been previously presented in Luh *et al.* [26], [27]. The results of this paper lead directly to the problem (MO/SP/IM), where precedence constraints are added to the problem formulation given in Section II of this paper. An enumeration technique with a simple bounding procedure ensures that the precedence constraints are obeyed [18]. The resulting solution methodology is then extended for the nonidentical machine problem with simple precedence constraints (MO/SP/NM) [19]. Finally, Hoitomt *et al.* [20] considers the general job shop problem (MO/GP/NM), where the precedence constraints are relaxed by using another set of Lagrange multipliers. The same subgradient method used here is combined with augmented Lagrangian relaxation techniques to obtain effective schedules.

II. Problem Formulation

An integer programming formulation is a common way to represent a scheduling problem. The following static, discrete-time, integer programming formulation has been influenced by the work of Bruvold and Evans [3] in some variable definitions, Norbis and Smith [28] in some constraint statements, and generally to Everett [6], Fisher [7], and Conterno and Ho [5]. We first define the following variables

- δ_{ik} 0-1 integer variable equals one if job i is active at time k
- B_i beginning time of job i (the first day of processing if time unit = day)
- C_i completion time of job i (the last day of processing if time unit = day)
- D_i due date of job i
- J objective function to be optimized
- K time horizon under consideration
- M_k number of machines available at time k, $k = 1, 2, \cdots, K$
- N number of jobs
- t_i time required from the resource by job i
- T_i tardiness of job i equals $\max\{0, C_i - D_i\}$
- w_i weight (or importance) of job i.

Among the above variables, the number of jobs N, the time horizon K, the weights of the jobs $\{W_i\}_{i=1}^N$, time requirements $\{t_i\}_{i=1}^N$, due dates $\{D_i\}_{i=1}^N$, and machine availability $\{M_k\}_{i=1}^K$ are assumed to be given. We also assume that job processing is nonpreemptive so that a contiguous block of time of length t_i is needed to process job i. Therefore, the decision variables are $\{B_i\}_{i=1}^N$, which is the beginning times of all jobs. Once $\{B_i\}_{i=1}^N$ are selected, $\{C_i\}_{i=1}^N$, $\{T_i\}_{i=1}^N$, and $\{\delta_{ik}\}_{i=1,k=1}^{NK}$ can be easily derived. For example, for $B_i = 2$, $t_i = 3$, and $D_i = 3$, we have $\delta_{i2} = \delta_{i3} = \delta_{i4} = 1$, $C_i = B_i + t_i - 1 = 4$, and $T_i = C_i - D_i = 1$. We also assume for simplicity that all jobs are available for processing at time 1 (to be relaxed later in Section III-B) and that the time horizon K is long enough to complete all the jobs (i.e., $C_i \le K$ for all i).

As mentioned, the objective function of interest is

$$J = \sum_i w_i T_i. \tag{1}$$

This objective function accounts for the weight 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. A static and deterministic parallel machine scheduling problem can now be formulated as follows:

$$P: \min_{\{B_i\}} J, \text{ with } J \equiv \sum_i w_i T_i \tag{2}$$

subject to capacity constraints:

$$\sum_i \delta_{ik} \le M_k, \; k = 1, 2, \cdots, K \tag{3}$$

and processing time requirements:

$$C_i - B_i + 1 = t_i, \; i = 1, 2, \cdots, N. \tag{4}$$

Note that in (4), adding 1 to $(C_i - B_i)$ is needed to obtain t_i in view of the definitions of B_i and C_i.

Lenstra *et al.* [24] showed that the single machine, weighted tardiness scheduling problem is n-p hard. Consequently, our parallel machine, weighted tardiness scheduling problem is also n-p hard.

III. Solution Methodology

A. The Lagrangian Relaxation Approach to the Scheduling Problem

Lagrangian relaxation has often been employed to obtain a set of decomposed subproblems from an original problem. This is clearly desirable if subproblems are much easier to solve than the original problem. The approach was pioneered by Everett [6], while Fisher [7] recognized the potential specifically for scheduling problems. Geoffrion [14] further developed the theoretical aspects for integer programming problems and corresponding solution methodology. Recently, Sandell *et al.* [34], Gavish [12] and Conterno and Ho [5] exploited the decentralized structure in obtaining solutions for related problems.

To decompose problem P according to jobs, we relax the capacity constraint (3) by using the Lagrange multiplier π_k to form the relaxed problem:

$$R: \min_{\{B_i\}} \left\{ \sum_i w_i T_i + \sum_k \pi_k \left(\sum_i \delta_{ik} - M_k \right) \right\} \tag{5}$$

$$\text{subject to (4)}. \tag{6}$$

Then, the dual problem is

$$D: \max_{\pi} L,$$

$$\text{with } L \equiv \left\{ -\sum_k \pi_k M_k + \min_{\{B_i\}} \sum_i \left\{ W_i T_i + \sum_k \pi_k \delta_{ik} \right\} \right\} \tag{7}$$

$$\text{subject to } \pi \ge 0 \text{ and (4)}. \tag{8}$$

This leads to the following decomposed subproblem for each job i (given π):

$$R_i: \min_{1 \le B_i \le K - t_i + 1} L_i, \text{ with } L_i \equiv \left\{ w_i T_i + \sum_k \pi_k \delta_{ik} \right\} \tag{9}$$

$$\text{subject to } C_i - B_i + 1 = t_i. \tag{10}$$

The assumption that K is long enough to complete all jobs ($C_i \leq K$) combined with the processing time requirement of (10) gives the upper bound on B_i in (9).

The above derivation presents a framework for solving the scheduling problem. There are several steps to obtaining an optimal solution: solving the subproblems, solving the dual problem, constructing a feasible solution, and finding an optimal solution. We shall briefly discuss each of them.

B. Scheduling Individual Jobs

From (9), the scheduling of job i becomes the selection of the optimal beginning time B_i^*. To do this, L_i is computed for each possible value of B_i, and B_i^* is the one yielding the lowest value of these L_i's. The computational complexity of problem R_i is linear in K since at most K evaluations of L_i is needed to determine B_i^*. This is in contrast to the n-p hardness of the original problem P. Note that the earliest start date constraint can be easily accommodated by requiring B_i to lie between the earliest start date of job i and $K - t_i + 1$ (rather than between 1 to $K - t_i + 1$).

C. Solving the Dual Problem

To solve the dual problem D (7), several methods of generating the dual solution π^* have been presented recently (see, e.g., Held and Karp [17] for a multiplier adjuster method, Fisher [8] for a column generation procedure, and Gavish and Srikanth [13] for a subgradient optimization procedure). We adapt a subgradient method originally presented by Shor [35], further exploited by Polyak [31], and commonly used to solve this type of problem (see, e.g., Fisher [8], Sandell *et al.* [34], and Pattipati *et al.* [29]).

In our algorithm, the variable π changes according to

$$\pi^{n+1} = \pi^n + \alpha^n g(\pi^n) \tag{11}$$

where n is the iteration index, $g(\pi)$ is the subgradient of L with respect to π (7) with the kth component equal to $(\sum_i \delta_{ik} - M_k)$, and α^n is the step size at the nth iteration. Polyak [31] proved that with the step size α^n given by

$$\alpha^n = \lambda \frac{\overline{L} - L^n}{g(\pi^n)^T g(\pi^n)}, \qquad 0 < \lambda < 2 \tag{12}$$

where $\overline{L}$ is an estimate of the optimal solution, and L^n is the value of L at the nth iteration, this method converges at the rate of geometric progression. Fisher [8] documented an adaptive step sizing mechanism that has been used with some success: decrease the step size by half whenever L^n fails to increase in some fixed number of iterations. Sandell *et al.* [34] modified the stepsize selection procedure as well. The stepsizing mechanism used here is a modified version of that suggested by Fisher [8] and incorporates features used in Sandell *et al.* [34], with parameters selected based on testing experiences. The subgradient algorithm terminates when the stepsize α^n remains small for a fixed number of iterations while L^n is not increasing.

D. Construction of a Feasible Schedule

Because of the stopping criterion used, the solution in the dual space is generally associated with an infeasible schedule, i.e., capacity constraints (3) might be violated for a few time slots. Note that processing time requirements (4) are always satisfied in view of how subproblems R_i (9) are solved. To construct a feasible schedule, a heuristic approach based on the "list scheduling" concept is developed as follows. In the optimal dual solution, each job is uniquely 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. For simplicity, the number of machines available at time k, M_k, is assumed to be monotonically decreasing over the time horizon. Thus, if a machine becomes available at time k, it remains available throughout the rest of the time horizon. The general case without restrictions on M_k will be discussed in Section V-B.

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 weighted tardiness of those conflicting jobs are calculated if they are delayed by one time unit. The jobs are then assigned to machines in the descending order of the costs until all the machines available at that time are assigned. The remaining jobs are delayed by one time unit and are considered together with those jobs originally scheduled to start at that time. Jobs are then assigned in descending order of the costs until all the available machines are assigned, and the process repeats. The pseudocode of this heuristic is given in Appendix A. The overall algorithm, consisting of procedures described in Sections III-B through III-D, is referred to in this paper generally as the algorithm or our algorithm.

E. Evaluation of the Feasible Solution via the Approximate Duality Gap

Once a feasible schedule is obtained, the corresponding value of the objective function J is an upper bound on the optimal objective J^*. The value of the dual function L^*, on the other hand, is a lower bound on J^* [14]. The difference between J^* and L^* is known as the duality gap. An upper bound of the duality gap is provided by $J - L^*$, which is a measure of the suboptimality of the feasible schedule with respect to the optimal schedule. To obtain an optimal solution, the branch-and-bound technique can be applied in conjunction with the upper/lower bounds obtained above (see, e.g., Fisher *et al.* [9]). However, such a step has exponential computational requirements in the worst case (see p. 575 of Horowitz and Sahni [33]). Furthermore, this step may not be justified because our computational experience shows that the approximate relative duality gap $(J - L^*)/L^*$ is usually very small (less than 1% for most tests; see Section IV).

IV. Test Results

Example 1: In the first example, there are 12 jobs and two identical machines. The planning horizon is 19 days (i.e., $K = 19$, and time unit = day), all machines are available on day 1 and throughout the planning horizon, and all jobs have the same weight ($w_i = 2$). Data are shown in Table I.

The value of the optimal dual solution is 31.82. The primal

TABLE I
DATA FOR EXAMPLE 1

Job	t_i	D_i		Job	t_i	D_i
1	3	6		7	1	12
2	1	5		8	3	11
3	3	7		9	3	6
4	2	9		10	3	6
5	4	8		11	2	14
6	4	8		12	4	12

TABLE II
JOB SEQUENCE FOR EXAMPLE 1

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

TABLE III
DATA FOR EXAMPLE 2

Job	w_i	t_i	D_i		Job	w_i	t_i	D_i
1	6	4	4		14	5	8	15
2	8	5	20		15	2	7	35
3	2	8	12		16	5	4	20
4	1	2	12		17	6	2	25
5	2	2	4		18	2	6	25
6	2	2	13		19	2	8	25
7	2	8	8		20	2	7	15
8	2	4	4		21	2	7	20
9	2	7	12		22	2	7	20
10	2	5	12		23	2	7	30
11	2	3	15		24	2	7	30
12	1	2	20		25	2	3	35
13	2	7	25					

solution at this point is, as expected, infeasible. Following the procedure outlined in Section III-D, the resulting feasible job sequence is given in Table II for each machine. The value of the objective function is 32.00, which is a relative difference of 0.57% compared with the value of the dual solution. Note that the value of the optimal objective function for this example should be an integer because the weights, process times, and due dates are all integer numbers. The minimal integer greater than 31.82 is 32; therefore, the optimal schedule has actually been attained here. Note also that jobs 1, 9, and 10 are completely interchangeable as are jobs 5 and 6. Thus, at least 12 equivalent solutions exist for this problem. The time to solve the problem was 2.2 CPU seconds on an IBM 3090 mainframe computer.

Example 2: The data listed in Table III are used in the second example. Here, there are four identical machines and 25 jobs with various weights. The planning horizon is 46 days, and all machines are immediately available from day 1 throughout the planning horizon. The data are sampled from an actual production system.

The value of the optimal dual solution is 37.66. Table IV shows the feasible job sequence with corresponding objective function 38.00, which is within 0.90% of the dual solution. Again, because the optimal objective function should be an integer, and the minimal integer greater than 37.66 is 38, the sequence shown in Table IV is actually the optimal sequence.

TABLE IV
JOB SEQUENCE FOR EXAMPLE 2

Machine 1	1	3	4	22	13	15		
Machine 2	7	20	2	18	24			
Machine 3	8	9	6	11	16	12	17	23
Machine 4	5	10	14	21	19	15		

TABLE V
RESOURCE AVAILABLE FOR EXAMPLE 3

Time k	M_k		Time k	M_k
0	8		6	38
1	16		7-9*	39
2	26		10-14	40
3	30		15-19	41
4	32		20-88	42
5	35			

*The resource availability is the same throughout this time period.

The problem is solved in 13.7 CPU seconds on an IBM 3090.

Example 3: This example takes data from the tool and die work center of Pratt and Whitney's Development Operations shop. Here, 89 jobs of four different weights are to be scheduled on 42 machines. Not all the machines are available at day 1 because some of them are busy processing jobs already started. The number of machines available increases monotonically over time ($M_k \leq M_{k+1}$, $k = 1, 2, \cdots, K-1$) as machines are freed up to process the 89 jobs. The planning horizon is determined to be 88 days. The "machines" actual refer to 42 skilled craftsmen of tool and die rather than real machines. It is assumed that the needed tools, equipment, or machinery are available upon request by a worker to perform a job. The resource availability is shown in Table V, and the job data are given in Table VI.

The value of the optimal dual solution is 1581.65, whereas the new feasible schedule has a value of 1583, which is within 0.085% of the dual solution. Table VII summarizes job classes and tardiness characteristics for the schedule. It can be seen from this table that by and large, average tardiness monotonically increases as weight decreases. The CPU time to generate the schedule is 2.5 s with all multipliers initialized at zero.

V. "WHAT IF" STUDIES AND THE SCHEDULING OF NEW JOBS

A. Job Interaction Variables

For convex programming problems, it is well known that the optimized Lagrange multipliers represent the sensitivity of the cost function with respect to the level of constrained resource [25], that is, if we suppose that the resource level M_k is a continuous parameter instead of an input parameter, we have

$$\pi_k^* = -\frac{dJ^*}{dM_k}. \tag{13}$$

For our problem P, machines are available in discrete units. As a result, π_k^* is only an estimate of the sensitivity of J^*

TABLE VI
DATA FOR EXAMPLE 3

Job	W_i	t_i	ESD*	D_i	Job	W_i	t_i	ESD	D_i
1	1	1	1	0	46	1	3	1	2
2	1	2	1	15	47	1	2	1	7
3	9	1	10	12	48	1	1	1	7
4	1	1	1	13	49	1	1	1	8
5	1	1	2	1	50	1	2	1	7
6	1	1	1	-3	51	1	2	1	7
7	1	2	1	-2	52	1	1	1	10
8	1	4	1	0	53	1	1	1	9
9	1	3	4	9	54	1	3	1	2
10	1	1	1	7	55	9	4	1	1
11	1	1	1	8	56	1	3	1	9
12	1	2	1	13	57	1	3	1	2
13	6	5	15	19	58	1	2	1	-1
14	1	1	1	2	59	1	1	1	10
15	1	1	1	7	60	1	20	1	24
16	1	2	1	-1	61	1	5	1	5
17	5	1	1	1	62	1	4	8	9
18	1	3	1	0	63	16	6	5	7
19	1	1	1	5	64	1	5	1	5
20	9	10	1	9	65	1	1	12	10
21	1	1	3	1	66	1	3	1	2
22	1	2	1	8	67	1	3	1	7
23	1	3	1	0	68	1	15	33	42
Job	W_i	t_i	ESD*	D_i	Job	W_i	t_i	ESD	D_i
24	1	7	1	3	69	1	8	1	10
25	6	8	56	60	70	1	16	1	17
26	1	9	1	8	71	1	10	16	20;
27	1	2	1	-1	72	1	3	1	-1
28	1	3	1	3	73	1	8	1	11
29	1	2	1	0	74	1	4	1	4
30	6	1	4	2	75	1	4	17	15
31	9	1	1	-1	76	1	3	8	5
32	1	1	1	7	77	1	3	1	2
33	1	11	1	11	78	1	3	1	1
34	1	2	4	15	79	1	1	1	-1
35	1	16	1	24	80	6	3	16	12
36	16	5	1	2	81	1	3	11	10
37	1	3	1	25	82	1	3	11	9
38	1	3	1	24	83	1	2	1	4
39	1	3	1	20	84	1	3	9	2
40	1	4	1	18	85	6	12	1	9
41	1	1	4	2	86	1	2	1	1
42	1	3	1	2	87	1	2	1	1
43	6	5	12	10	88	1	2	1	1
44	1	3	1	7	89	1	2	1	0
45	1	1	1	7					

*ESD is the earliest start date.

with respect to change in M_k, that is

$$\pi_k^* \approx -\frac{\Delta J^*}{\Delta M_k}. \tag{14}$$

Another well-established interpretation of the optimized Lagrange multiplier π_k^* is that it is the "price" of using the resource at time k, which is the so-called "shadow price" in economic terminology. Therefore, the cost of job i can be viewed as the sum of the cost of using the resource and the penalty for missing the due date (see (9)). The cost of using the resource of time k is higher if many jobs compete for the

TABLE VII
EXAMPLE OF EXAMPLE 3

		Algorithm Schedule	
Weight	No. of Jobs	No. Late	Avg. Late
16	2	2	3.0
9	4	3	2.0
6	6	5	4.0
1	77	48	3.27

resource at that time to meet their respective due dates. Consequently, the optimized Lagrange multipliers are also a measure of the competition among jobs for the resource. For this reason, the optimized Lagrange multipliers have been referred to as "job interaction variables."

The above properties of Lagrange multipliers can be exploited to estimate the impact of a wide variety of dynamic changes that may occur within the time horizon. This estimate allows "what if" questions to be answered without solving the problem again. If $\bar{J}$ denotes an estimate of the cost after a change and J is the cost of the nominal schedule, then the value $(\bar{J} - J)/J$ gives an indication of the magnitude of the change. For small changes, the estimates are usually fairly accurate. The accuracy, however, degrades as the magnitude of the change increases. This is caused by the fact that sensitivity information, which is local in nature, is used to generate our estimates.

To actually accommodate a change, an existing schedule can be reconfigured by simply using the heuristic outlined in Section III-D without updating the Lagrange multipliers if the change is relatively small. Otherwise, the schedule can be reconfigured after partially updating the Lagrange multipliers by running the subgradient algorithm for a fixed number of iterations or after fully updating the Lagrange multipliers until the convergence of the subgradient algorithm. In either case, the original multiplier values can be used as the initial conditions. The algorithm then usually converges within a few iterations, as opposed to the case when one starts with zero multiplier values. Schedule reconfiguration can thus be done efficiently and effectively.

We shall now derive formulas to estimate $\bar{J}$ for the following cases:

1) Adding a machine (or subtracting a machine) for certain time slots
2) increasing (or decreasing) the processing time requirement of job i
3) inserting a new job at certain time slots.

We shall then present methods to schedule new jobs. Relevant numerical testing results are given in Section VI.

B. Effect of Adding a Machine

We have assumed in Section III-D that M_k is monotonically nondecreasing over the time horizon. Thus, if a machine becomes available at time k', it remains available throughout the rest of the time horizon. Suppose now that machine becomes available at time $k' - 1$ instead of time k', making an additional machine available at time $k' - 1$. An additional machine at time $k' - 1$ implies that a job can be stated earlier. Since that job would then terminate earlier, the effect may propagate, and the total cost J may decrease. The exact impact on J, however, is difficult to quantify without solving the problem again. Equation (14) provides a quick estimate of the new cost, that is

$$\bar{J} = J - \pi^*_{k'-1}. \tag{15}$$

Now suppose that the start of a machine changes from k'' to $k'' + 1$. With one less machine at time k'', the total cost may increase. A quick estimate on the new cost is given by

$$\bar{J} = J + \pi^*_{k''}. \tag{16}$$

The above reasoning can be extended to the case where a machine becomes unavailable at time k'' for l time units:

$$\bar{J} = J + \sum_{k=k''}^{k''+l-1} \pi^*_k. \tag{17}$$

An analogous situation exists for the case where a machine becomes available at time $k' - l$, which is l time units earlier than previously. The quality of the estimate (17) depends on the magnitude of l and prices of the resource (or competitiveness of jobs) over that time interval.

If M_k is not monotonically nondecreasing, individual subproblems and the dual problem are solved in the same way as before. An additional checking step, however, must be included in the heuristic to ensure that jobs are scheduled on contiguous blocks of time.

C. Effect of Longer Processing Time

Suppose now job i requires longer processing time than originally anticipated. This is equivalent to the situation where a machine becomes unavailable or breaks down from $C_i + 1$ to the end of the new completion time $\bar{C}_i$. This reasoning can be justified from (5) in that the roles of M_k and δ_{ik} are symmetric. In addition to machine unavailability, job tardiness penalty might also be changed, and an estimate of the new cost is

$$\bar{J} = J + w_i(\bar{T}_i - T_i) + \sum_{k=C_i+1}^{\bar{C}_i} \pi^*_k \tag{18}$$

where $\bar{T}_i$ is the new tardiness of job i. Again, the quality of (18) depends on the magnitude of the change. Similar results hold for the case where a job finishes earlier than anticipated.

D. Effect of Inserting a New Job

Suppose we insert a new job, say job r, to begin processing at time k'. Job r has weight w_r, processing time t_r, and due date D_r. Inserting job r into the schedule with $B_r = k'$ and $C_r = B_r + t_r - 1$ will delay the completion of existing jobs. The impact on the total cost can be grouped into two parts: the increase in cost for existing jobs because of the reduction of resource availability during the period $[B_r, C_r]$ and the penalty associated with the new job if its due date cannot be met. An estimate of the new cost is therefore given by

$$\bar{J} = J + w_r T_r + \sum_{k=B_r}^{C_r} \pi^*_k. \tag{19}$$

The above analysis can be extended to examine the effect of 1) dropping an existing job from consideration, 2) shifting an existing job, 3) swapping two existing jobs, and 4) changing the weight or due date of a job. Since the extension is straightforward, it is omitted here.

E. Scheduling New Jobs

Equation (19) provides an estimate of the effect of starting the new job r at time k'. A reasonable guess for a good B_r is

therefore given by

$$\bar{B}_r = \arg\left\{\min_{1\le B_r\le K-t_r+1}\left\{w_r T_r + \sum_{k=B_r}^{B_r+t_r-1}\pi_k^*\right\}\right\}. \quad (20)$$

This is particularly true if the resulting $\bar{J}$ changes little with respect to J. Therefore, it is not necessary, as with many other scheduling methodologies (e.g., branch-and-bound and dynamic programming), to start from scratch when new jobs arrive. Rather, the information contained in the job interaction variables yields scheduling information that need only be updated periodically.

In summary, the above subsections present a methodology that provides quick answers for "what if" questions and updates a schedule whenever unforeseen changes (e.g., machine breakdown, unpredictable job processing times, the arrival of new jobs, and the departure of completed jobs) occur during the course of the planning horizon. It will be shown next that the estimates are accurate, the schedules can be efficiently reconfigured to accommodate these changes.

VI. Testing Results on "What If" Study and Handling New Jobs

A. "What If" Study

We shall use the data set of **Example 3** as the nominal data set and modify it in the ways described below to numerically examine the methods just presented. Recall that the original dual cost is 1581.654, and the original cost of the feasible schedule generated is 1583. The "what if" method described gives $\Delta\bar{J} \equiv \bar{J} - J$, which is the estimated cost difference. For comparison, the algorithm is also run from scratch (i.e., with all multiplier values initialized at zero) to generate the corresponding cost. The case where the multipliers are initialized based upon a previous schedule is examined in Section VI-B. In all cases, the magnitude of the change is small (less than 4%), the estimate is within 0.2% of the algorithm cost, and the time to acquire the estimate is at least an order of magnitude less than rerunning the algorithm from scratch.

Example 4: Changes in Resource Availability: In this example, a machine that was immediately available for scheduling (day 1) is now only available starting day 2. The estimate increase in the cost is 15.98, representing a 1.01% change from the original cost. When our algorithm is run from scratch with the change in data, a cost of 1602 is obtained, representing an increase by the amount 19. The algorithm requires 3.5 CPU seconds compared to almost no time to obtain the estimate.

Now, suppose a machine that was originally available beginning on day 1 becomes immediately available instead. The estimated change in the cost is -15.98, representing a 1.01% decrease in the cost. The algorithm gives a cost of 1569, which is an increase of 14.00. The algorithm requires 3.2 CPU seconds, compared with almost none for the estimate.

Example 5: Changes in Job Processing Time: Suppose a job with weight equal to 9 (second highest weight for this data set) requires one less day to process, which is a change from four to three days. The estimated change in cost is -49.25, which is a 3.11% decrease from the current schedule. The algorithm yields a cost of 1533, which is a decrease by the amount 50 and obtained in 2.7 CPU seconds.

Now, suppose the same job requires more time to process (from four to five days). The estimated new cost is 1646 (a 3.98% increase), and it is acquired in almost no CPU time. This is compared with the new cost of 1646 obtained in 2.5 CPU seconds of the algorithm, representing an increase by the amount 63.

Example 6: Change in Due Date and Priority: In this example, a job due date is changed from one to two days from present, and at the same time, the weight is reduced from nine to six. These changes resulted in an estimated decrease of 3.60% from 1583 to 1526. The cost obtained via the algorithm is 1526 in 2.9 CPU seconds.

B. Handling New Jobs

In this subsection, we demonstrate the new job scheduling technique described in Section V using the same nominal data set as above. In **Example 7**, a job is added, and the heuristic is used to schedule it. The algorithm is also run from scratch for comparison purposes. For an estimated change of less than 2%, the heuristic alone generates a schedule within 0.4% of the algorithm in almost no time as compared with rerunning the algorithm. In **Example 8**, with the first new job in the schedule, a second new job is added. The algorithm is run for a limited number of iterations with multipliers initialized at values before the first job was scheduled. When the algorithm is run from scratch for comparison, it gives a schedule in longer CPU time as expected.

Example 7: Scheduling a New Job without Update: A new job with the following characteristics arrives: Process time is seven days, job due date is nine days from today, and job weight is nine (second highest priority). By using (19) and (20), the estimated new cost is 1600, which is a 1.04% increase from the original schedule. A separate test run shows that the algorithm cost is 1594 (obtained in 2.2 CPU seconds), with the lower bound being 1593.07.

Example 8: Scheduling a New Job with Update: With the first new job scheduled by using the heuristic, the second new job now arrives with weight equal to one, process time 12, and due date 14. The algorithm is run for 40 iterations to schedule the second new job, which requires 0.6 CPU seconds. The result is 1599 compared with 1605 when the job interaction parameters are initialized at zero, which requires 100 iterations and 3.5 CPU seconds. Since the lower bound is 1598.72, the optimal schedule was obtained from the updated schedule. Even longer CPU times have been observed for other problems if one starts with zero multiplier values.

VII. Conclusions

The paper presents a methodology for scheduling independent jobs with due dates on identical parallel machines. The special integer programming formulation of the problem facilitates the application of the Lagrangian relaxation technique. Decomposition of the dual problem serves to simplify

the solution at the low level. The high-level problem is then solved via a subgradient method. A heuristic using the list scheduling concept is developed to construct a feasible solution based on the dual results. For most of the problems tested at Pratt and Whitney, results are within 1% of the lower bounds and obtained within practical amounts of CPU time. More importantly, the interaction of jobs as they compete for limited resources becomes visible. Based on the job interaction information, a methodology is developed to provide quick answers to "what if" questions, to reconfigure an existing schedule whenever changes occur in job characteristics or in resource availability, and to schedule new jobs. These capabilities are of enormous value for Pratt and Whitney engineers in their scheduling operations. We also believe that the approach will have further impact when extended to a more general production setting.

Appendix A

Pseudocode of the Heuristic Procedure

Define S_i as a sequence of $N - i + 1$ jobs $[1], [2], \cdots, [N - i + 1]$ associated with a set of $N - i + 1$ beginning times $\{B_{[j]}\}$, ordered from the smallest to the largest so that $B_{[1]} \leq B_{[2]} \leq \cdots \leq B_{[N-i+1]}$ and, if $B_{[l]} = B_{[l+1]}$, when $w_{[l](T[l]} + 1) \geq w_{[l+1]}(T_{[l+1]} + 1)$. Given the sequence S_1 and given $\{M_k\}$

Step 0. Set $k = 1$, $i = 1$.

Step 1. If $M_k \neq 0$, then schedule job $[i]$ to start processing at time k, set $M_j = M_j - 1$ for $j = k, k + 1, \cdots, k + t_{[i]} - 1$ and go to *Step 4*; otherwise, go to *Step 2*.

Step 2. Set $k = k + 1$; if $B_{[i]} \geq k$, go to *Step 1*; otherwise, go to *Step 3*.

Step 3. For all the unscheduled jobs such that $B_{[j]} < k$, reset $B_{[j]} = k$, reform sequence S_i, and go to *Step 1*.

Step 4. Set $i = i + 1$; if $i > N$, stop; otherwise, go back to *Step 1*.

References

[1] A. D. Amar and E. N. Vasilescu, "On the scheduling of identical processors with entrapment," *IIE Trans.*, vol. 20, no. 1, pp. 88-96, Mar. 1988.

[2] K. R. Baker, *Introduction to Sequencing and Scheduling*. New York: Wiley, 1974.

[3] N. T. Bruvold and J. R. Evans, "Flexible mixed-integer programming formulations for production scheduling problems," *IIE Trans.*, vol. 17, no. 1, pp. 2-7, March 1985.

[4] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "An application of bin-packing to multiprocessor scheduling," *SIAM J. Comput.*, vol. 7, no. 1, pp. 1-17, Feb. 1987.

[5] R. Conterno and Y. C. Ho, "Order scheduling problem in manufacturing systems," *Int. J. Production Res.*, vol. 26, no. 9, pp. 1487-1510, Sept. 1988.

[6] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Oper. Res.*, vol. 11, pp. 399-417, 1963.

[7] M. L. Fisher, "Optimal solution of scheduling problems using lagrange multipliers, Part I," *Oper. Res.*, vol. 21, pp. 1114-1127, 1973.

[8] M. L. Fisher, "Lagrangian relaxation method for solving integer programming problems," *Management Sci.*, vol. 27, pp. 1-18, 1981.

[9] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Management Sci.*, vol. 32, no. 9, pp. 1095-1103, Sept. 1986.

[10] S. French, *Sequencing and Scheduling*. New York: Wiley, 1982.

[11] T. D. Fry, R. D. Armstrong, and J. H. Blockstone, "Minimizing weighted absolute deviation in single machine scheduling," *IIE Trans.*, vol. 19, pp. 445-450, Dec. 1987.

[12] B. Gavish, "Optimization methods for configuring distributed computer systems," *IEEE Trans. Comput.*, vol. C-36, no. 7, pp. 773-793, July 1987.

[13] B. Gavish and S. K. Srikanth, "Optimal solution methods for large scale multiple salesmen problem," *Oper. Res.*, vol. 34, pp. 698-717, 1987.

[14] A. M. Geoffrion, "Lagrangian relaxation for integer programming," *Math. Programming Study*, vol. 2, pp. 82-114, 1974.

[15] S. C. Graves, "A review of production scheduling," *Oper. Res.*, vol. 18, pp. 841-852, 1981.

[16] S. L. Gupta and T. Sen, "On the single machine scheduling problem with quadratic penalty function of complete times: An improved branching procedure," *Management Sci.*, pp. 644-647.

[17] M. Held and R. M. Karp, "The traveling salesman problem and minimum spanning trees: Part II," *Math. Programming Study*, no. 1, pp. 6-25, 1971.

[18] 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.

[19] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "Job shop scheduling with simple precedence constraints," in *Proc. 1990 Automat. Contr. Conf.*, (San Diego, CA), May 1990.

[20] D. J. Hoitomt, P. B. Luh and K. R. Pattipati, "A Lagrangian relaxation approach to job shop scheduling problems," in *Proc. 1990 IEEE Int. Conf. Robotics Automat.* (Cincinnati, OH), May 1990.

[21] E. Horowitz and S. Sahni, "Exact and approximate algorithms for scheduling nonidentical processors," *J. Assoc. Comput. Mach.*, vol. 23, pp. 317-327, Apr. 1976.

[22] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*. Rockville, MD: Computer Science, 1978.

[23] E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Recent developments in deterministic sequencing and scheduling: A survey," in *Deterministic and Stochastic Scheduling*, M. A. H. Dempster, J. K. Lenstra, and A. H. G. Rinnooy Kan, (Eds.). Boston: D. Reidel, 1982.

[24] J. K. Lenstra, A. H. G. Rinnooy Kan, P. Brucker, "Complexity of machine scheduling problems," *Annals Discrete Math.*, I, pp. 343-362, 1977.

[25] D. G. Luenberger, *Linear and Nonlinear Programming* (2nd ed.). Reading, MA: Addison-Wesley, 1984.

[26] P. Luh, D. Hoitomt, E. Max, and K. Pattipati, "Parallel machine scheduling using Lagrangian relaxation," in *Proc. Int. Conf. Comput. Integrating Manu.* (Troy, NY), May 1988, pp. 244-248.

[27] P. Luh, D. Hoitomt, E. Max, and K. Pattipati, "Schedule generation and reconfiguration for parallel machines," in *Proc. 1989 IEEE Int. Conf. Robotics Automat.* (Scottsdale, AZ), May 1989, pp. 528-533.

[28] M. I. Norbis and J. M. Smith, "Two level heuristic for the resource constrained scheduling problem," *Int. J. Production Res.*, vol. 24, pp. 1203-1219, 1986.

[29] K. R. Pattipati, *et al.*, "CONFIDANTE: A computer-based design aid for the optimal synthesis, analysis and operation of maintenance facilities," *Proc. 1984 IEEE AUTOTESTCON*, Nov. 1984.

[30] J. Picard and M. Queyranne, "The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling," *Oper. Res.*, vol. 26, pp. 86-110, Jan./Feb. 1978.

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

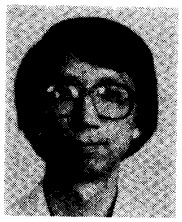
[32] F. A. Rodammer and K. P. White Jr., "A recent review of production scheduling," *IEEE Trans. Syst. Man Cybern.*, vol. 18, pp. 841-852, 1988.

[33] S. K. Sahni, "Algorithms for scheduling independent tasks," *J. ACM*, vol. 23, no. 1, pp. 116-127, Jan. 1976.

[34] N. R. Sandell, 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.

[35] N. Z. Shor, "On the structure of algorithms for the numerical solution of optimal planning and design problems," *Diss. kand. fiz.-matem. n.*, (Kiev, In-t kiberbetika AN USSR), 1964.

[36] W. Zhao and K. Ramamritham, "Simple and integrated heuristic algorithms for scheduling tasks with time and resource constraints," *J. Syst. Software*, no. 7, pp. 195-205, 1987.



Peter B. Luh (M'80) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, Republic of China, in 1973, the M.S. degree in aeronautics and astronautics engineering from the Massachusetts Institute of Technology, 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 an Associate Professor in the Department of Electrical and Systems Engineering. His major research interests include hierarchical planning and control of large-scale systems, schedule generation and reconfiguration for manufacturing systems, distributed decision making, game theory, and multitarget tracking. He has been a principal investigator and consultant to many industry- and government-funded projects in the above areas and has published about 80 papers. He has made significant contributions in large-scale dynamic optimization by developing hierarchical algorithms with parallel processing structures and coordination mechanisms, in manufacturing by designing and implementing an efficient and near-optimal schedule generation and reconfiguration system for a major manufacturing plant, in distributed decision making by developing normative and normative-descriptive models for dynamic team resource allocation problems, and in game theory by developing the inducible region concept and designing and implementing an incentive scheme for a large petrochemical company.

Dr. Luh is an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, a newly appointed Technical Editor for the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, and is Chairman of the Technical Committee on Computer-Aided Production Management of the IEEE Robotics and Automation Society. 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 major national, international, and intersociety conferences. He is a member of Sigma Xi and is listed in *Who's Who in Engineering* and *Who's Who in the East*.

Debra J. Hoitomt received the B.S. degree in education–natural science from the University of Wisconsin, Madison, in 1977, and the M.S. degree in systems engineering from the University of Arizona, Tucson, in 1984. She is currently a doctoral candidate at the Department of Electrical and Systems Engineering, University of Connecticut, Storrs.

In 1987, she joined the Manufacturing Information Technology group at Pratt & Whitney, East Hartford, CT, to become part of an effort to develop a large-scale, optimization-based, distributed scheduling methodology. Her current research interests are analysis and control of production systems, constrained optimization, numerical methods, and distributed processing.

Eric Max received the B.S. degree from Cornell University, Ithaca, NY, and the M.B.A. degree from Boston University, Boston, MA. He is the Inventory Manager at Pratt & Whitney's Overhaul and Repair Center. While Project Leader at Pratt's development shop, he saw a need to increase the accuracy of projected job completion dates and provide tools enabling more informed job priority decisions. He formed an interdepartmental project team combining the disciplines of production planning and control, artificial intelligence, mathematical optimization, and common sense. Under his guidance, and with academic support, a distributed scheduling methodology was developed, and it works within the natural decision-making process inherent in a multilayer organizational structure.

Krishna R. Pattipati (M'80) received the B.Tech degree in electrical engineering with highest honors from the Indian Institute of Technology, Kharagpur, in 1973, and the M.S. and Ph.D. degrees in systems engineering from the University of Connecticut, Storrs, in 1977 and 1980, respectively.

He was employed by Alphatech, Inc., Burlington, MA, from 1980 to 1986, where he supervised and performed research on human decision modeling, multitarget tracking, queueing networks, automated testing, and large-scale mixed-integer optimization. Since September 1986, he has been an Associate Professor in the Department of Electrical and Systems Engineering. He has also served as a consultant to Alphatech, Inc. and to the I.B.M. Thomas J. Watson Research Center.

Dr. Pattipati was selected by the IEEE Systems, Man, and Cybernetics Society as the Outstanding Young Engineer of 1984 and received the Centennial Key to the Future award. He won the best technical paper award at the 1985 IEEE AUTOTESTCON. He was a member of the administrative committee of the IEEE Systems, Man, and Cybernetics Society during 1986–1988. He served as the finance chairman of the International Conference on Control and Applications, Jerusalem, Israel, 1989, and as the Vice-Chairman for invited sessions of the IEEE International Conference on Systems, Man, and Cybernetics Conference, Boston, MA, 1989.