1996

Resource constrained project scheduling using simulated annealing and tabu search

Ying-Wei Tsai
Iowa State University

Resource constrained project scheduling using simulated annealing and tabu search

by

Ying-Wei Tsai

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Industrial Engineering

Major Professor: Douglas D. Gemmill

Iowa State University

Ames, Iowa

1996

Graduate College
Iowa State University

This is to certify that the Master's thesis of

Ying-Wei Tsai

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. GENERAL INTRODUCTION

General Background and Objective

In this thesis, we study how to schedule the activities and identify the critical path in a resource-constrained project. The critical path method (CPM) and the program evaluation and review technique (PERT) were developed to solve project scheduling problems with the assumption that resource availability is unlimited. However, the resource availability is usually limited due to some reason, such as budget, experienced workers, and equipment. The limited resource availability is one of the most important factors in scheduling a project. A scheduling problem becomes difficult to solve when resource constraints are applied. Resource constrained scheduling problems are generally NP-hard [3].

The concept of the critical path is an important feature of the CPM/PERT techniques. The time needed to complete the activities on the critical path will determine the project duration. Project managers pay most attention to the activities on the critical path in order to complete the project as early as possible. The critical path determined using the CPM/PERT technique considers only the technology constraints. This is no longer valid in a resource-constrained project. While determining the critical path of a resource-constrained project, there are two types of constraints, the technology constraints and the resource constraints, both of which need to be checked. Only a few researchers [4, 13] deal with this problem. In the paper presented in Chapter 2, we will add extra links, the resource links [4] and the precedence links, to the project network and then identify the critical path.

Since resource constrained scheduling problems are generally NP-hard [3], many heuristic [1, 2, 5, 10, 11, 12] rules are developed to find a good schedule for the resource-constrained project. Unfortunately, there is no heuristic rule that will always provide a good schedule. Instead of deriving a new heuristic rule, we focused on the application of existing methods which can be used to improve the schedule first determined using a heuristic rule. Two existing methods, simulated annealing and tabu search, were chosen for this purpose.

The simulated annealing [6] method is related to the way metals cool and anneal. The slow cooling process allows ample time for atoms to move around and achieve crystallization. Similarly, in the resource-constrained project scheduling problem, the objective is to prevent being trapped at a local minimum and continue the descent process. This is only possible if the method allows ascent; that is, accepts schedules which have longer project duration. We applied the simulated annealing procedure to the resource-constrained project scheduling problem and the experimental results show that simulated annealing can provide a near optimal schedule.

Tabu search [7, 8, 9] is capable of guiding a local heuristic search procedure to explore the solution space beyond local optimality. It uses flexible memory structures designed to permit evaluation criteria and historical search information to be exploited more thoroughly than by a memoryless method. Optimal and near optimal solutions have been obtained for a wide variety of classical and practical problems using tabu search. Utilizing the flexibility of tabu search, the resource-constrained project scheduling problem can be scheduled efficiently. Good solutions are obtained for both the deterministic and stochastic problem in short execution time.

Thesis Organization

This thesis is composed of three papers which may be suitable for publication. The first paper "Identifying the Critical Path in Resource Constrained Projects" in Chapter 2 was submitted to *Journal of the Operational Research Society* in November, 1996. The second paper "Using Simulated Annealing to Schedule Activities of Stochastic Resource-Constrained Projects" in Chapter 3 was submitted to *Project Management Journal* in September, 1996. The third paper "Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects" in Chapter 4 was submitted to *European Journal of Operational Research* in November, 1996.

In the first paper "Identifying the Critical Path in Resource Constrained Projects," the revised method of calculating the slack times of activities of a resource constrained project proposed by Bowers [4] is reinforced. In Bowers's revised method, resource links are explicitly created to trace the use of resources, noting the resource dependencies and hence identifying the critical path. This revised method is simplified by introducing new links called "precedence links." The precedence links defined in this paper ensure that the order in which activities are allocated resources is the same in both the forward pass and in the backward pass. With the addition of the resource links and the precedence links to the project network, the latest start times of activities can be calculated using exactly the same procedure as that of CPM/PERT without the need to consider resource requirements. The critical path can be identified by comparing the earliest start time and the latest start time of a activity.

The paper "Using Simulated Annealing to Schedule Activities of Stochastic Resource-Constrained Projects" is included in Chapter 3. The iterative process of "Simulated

Annealing" [6] starts with an arbitrary feasible schedule and performs an interchange of the positions of two randomly selected activities in the schedule and reallocates the resources with respect to the new schedule. We show its application to the resource allocation problem. The initial feasible schedule is calculated using two heuristic rules, the Composite Allocation Factor (CAF) method proposed by Badiru [1] and Badiru and Pulat [2] and the Minimum Slack First (MINSLK) method. The CAF method is one of the more recently proposed heuristic rules. Davis and Patterson [5] found the MINSLK method is one of the better heuristic rules for the resource constrained project scheduling problem. The simulated annealing algorithm is successfully applied to the resource-constrained project scheduling problem.

The last paper included in this thesis is "Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects." In this paper, tabu search is applied to the resource-constrained project scheduling problem. Tabu search provides good solutions for deterministic and stochastic problems. The experimental results based of Patterson's [12] 110 projects are better than the results provided by Morse *et al.* [10] for the deterministic problem. We also show that tabu search is capable of providing good schedules for the stochastic problem. Three aircraft maintenance projects from real world are also scheduled using tabu search. The experimental results show that the schedule calculated using the MINSLK rule or the CAF rule can be improved by 20% or more.

The reminder of the thesis is organized as follows. First, the three papers mentioned earlier are presented in Chapter 2-4 sequentially. Next, the general conclusions about this thesis are presented in Chapter 5 followed by a list of references cited in Chapter 1.

CHAPTER 2. IDENTIFYING THE CRITICAL PATH IN RESOURCE CONSTRAINED PROJECTS

A paper submitted to the Journal of the Operational Research Society

Ying-Wei Tsai and Douglas D. Gemmill

Abstract

The concept of the critical path is widely adopted by project managers to identify those activities most worthy of their attention. When resource availability is limited, the critical path is determined not only by the technology relationships but also the resource dependencies. The method of calculating resource constrained float proposed by Bowers is carefully examined and modified in this paper. A new type of link "precedence link" is proposed in this paper. By introducing the precedence links, the calculation of latest start times of activities is easier than Bowers' method. Experimental results indicate that the proposed method is able to identify the critical paths for the schedules of Patterson's 110 test projects.

Introduction

Over the years, several techniques have been developed for the purpose of planning, scheduling, and controlling projects. The critical path method (CPM) and the program evaluation and review technique (PERT) are the most widely used techniques. Using these techniques a critical path is determined which does not consider the resource availability of a

project. Since a project would be delayed if any one of the activities on the critical path is delayed, project managers pay most attention to the activities on the critical path in order to complete the project as early as possible. However, a striking shortcoming of CPM/PERT is the inability of dealing with a scheduling problem when the required resources are in limited supply.

Many researchers have worked on the resource constrained project scheduling problem. Badiru and Pulat[2] and Ozdamar and Ulusoy[9] provide a survey on this research. Different procedures have been developed for different goals. Demeulemeester and Herroelen[4], Badiru[1], Morse, McIntosh and Whitehouse[8] minimize the project completion time. Demeulemeester[5] and Lawrence and Morton[7] minimize the cost to complete a project. Other types of resource constrained scheduling problems have been researched and solved. However, only a few researchers have considered the criticality of an activity in a resource constrained project.

In a traditional network model of a project, the activities are connected by links which correspond to technology relationships. The critical path in the traditional CPM/PERT technique is therefore defined as the path connected by technology relations with the largest expected duration. This definition is no longer valid when the available resources are limited. The critical path found under the assumption of unlimited resources is not necessarily the critical path in a resource constrained project. The most damaging error is that activities initially found to have free slack time when resource constraints are not considered, become critical activities when constraints are included. Conversely, critical activities may be found to have slack time. To find the critical path in a resource constrained project, both technology

relationships and resource constraints must be considered. The critical path in a resource constrained project is defined as the path connected by technology relationships and resource sharing with the largest expected duration.

Badiru[1] calculated the probability of an activity falling on the critical path, but resource constraints are not considered in computation of the latest start times. Woodworth and Shanahan[12] claimed that they can correctly calculate slack times of activities with their proposed procedures. During the forward pass each activity is given a resource sequence label which records the resource being used and the order of its use. In addition to the resource labels, a new linkage mechanism is used to examine the history of resource utilization at each resource allocation during the backward pass to calculate the latest start time. The new linkage mechanism can require the creation of dummy activities. Continuing to examine history of the resource utilization and adopting the new linkage mechanism during the backward pass will cause extra burden as will be discussed later. Bowers[3] presented a revised method of calculating the slack times of activities of a resource constrained project. Some links called resource links are explicitly created to trace the use of resources, noting the resource dependencies and hence identifying the critical path. This revised method is introduced in the next section and also is simplified. New links called precedence links are defined in this paper to ensure that the order in which activities are allocated resources is the same in both the forward pass and in the backward pass.

The remainder of this paper is organized as follows. The methodology section includes discussion of the implementation of the resource links and the precedence links and

an example is given to illustrate the procedures. Experimental results are then shown in the results section. Finally, a conclusion is given in the last section.

Methodology

Typically, there are two graph techniques to model a resource constrained project: activity-on-arrow (AOA) and activity on node (AON). Dreger[6] states that the AOA technique has a dummy-activity problem in some applications. Bowers[3] found that the AOA technique has a dummy-activity problem and the dummy-activity problem never occurs in the AON technique. Bowers also found that the AON technique simplifies the process of deriving the slack time of activities of a resource constrained project. Therefore, the AON technique is adapted in this paper. One typical project network, HEUR3, constructed by Patterson[10] is given in Figure 2-1. The activity duration and resource requirements are given in Table 2-1. Three types of resources are used in this project (type A, type B and type C). The amount of each resource available is 6 type A units, 7 type B units, and 6 type C units.

Bowers' revised method for identifying the critical path of a resource constrained project includes six steps:

Step 1. Forward pass, ignoring resources. Calculate the earliest start time and the earliest completion time.

Step 2. Backward pass, ignoring resources. Calculate the latest start time and the latest completion time.

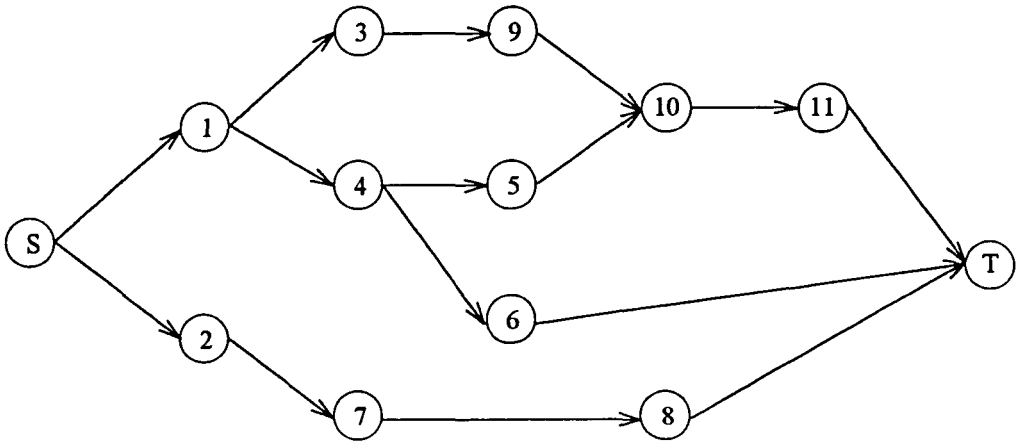

Figure 2-1. A typical project network, HEUR3.

Table 2-1. Activity duration and resource requirements the project HEUR3.

Node	Duration	Resource requirement
1	3	3,2,1*
2	5	2,4,2
3	6	3,1,2
4	2	4,3,1
5	3	2,0,3
6	3	1,1,1
7	4	3,1,1
8	5	2,2,2
9	4	3,2,3
10	2	4,1,0
11	3	5,4,2

* The resource requirements of type A, type B and type C resource respectively.

Step 3. Forward pass, with resources. Use a minimum latest start rule to assign resources to activities. Calculate the earliest start time and the earliest completion time and note resource utilization history.

Step 4. Add resource links.

Step 5. Backward pass, with resources, including resource links. Calculate the latest start time and the latest completion time.

Step 6. Compare earliest start time and latest start times.

The same calculation as used in basic CPM/PERT do are performed in **Step 1** and **Step 2**. In **Step 3**, the minimum latest start (MINLS) heuristic for resource constrained projects is used to assign resources to activities. Then the earliest start time (EST) and the earliest completion time (ECT) can be determined. In addition, the history of resource allocation for each activity is noted in **Step 3** and this information is then used to add resource links in **Step 4**. The resource links explicitly represent the resource dependencies in the project network. In **Step 5**, a backward pass incorporating the resource links provide the latest start time (LST) and the latest completion time (LCT) with the resource constraints. The critical activities and then the critical path are determined in **Step 6** by comparing the EST and LST.

Assume that many of the activities require the use of the same resources. A major drawback of the revised method proposed by Bowers is that the resource requirements must be checked and validated during the backward pass. To check the resource requirements and validate the resource availability, a resource database has to be maintained while scheduling. When resources are required to complete an activity, two operations must be completed.

First, query the resource database for the amount of resources which are not in use. Second, if the required resources are available, change the status of the resources to "in use." After an activity has been completed, the acquired resources must be released. The above process requires maintenance of a resources database will cause heavy burden when many kinds of resources are required or when a project contains a large number of activities. With the addition of the proposed precedence links, the resource requirements no longer need to be considered during the backward pass.

Bowers used the MINLS heuristic to assign resources to activities for resource constrained projects. Although, the concept of resource links can be applied to any resource allocation heuristic, we will use the minimum slack first (MINSLK) heuristic in our illustration of how to add resource links and precedence links. Before the introduction of precedence links, a brief review Bowers' method will be given. The HEUR3 project shown in Figure 2-1 is simplified by assuming that only resource type A is required. The simplified project is denoted as SIMPLE. The project schedule determined using the MINSLK heuristic is given in Table 2-2. The field MINSLK in Table 2-2 contains each activity's slack time from the schedule determined using CPM/PERT. The field Slack contains the slack times calculated when considering resource constraints. An activity is critical if the value of Critical is 1. Resource allocations and the resource links are given in Table 2-3. Type A resources are numbered in order to identify the resource dependencies. The nodes in Table 2-3 are rearranged in the order in which the resources are allocated.

Initially, none of the resources have been allocated. First, activity 1 and activity 2 are allocated resources. After activity 1 has been completed, resources A_1, A_2, A_3 are released and

Table 2-2. The schedule of the project SIMPLE calculated using the MINSLK heuristic.

Node	MINSLK	Activity Duration	EST	ECT	LST	LCT	Slack	Critical
1	0	3	0	3	0	3	0	1
2	4	5	0	5	1	6	1	0
3	0	6	3	9	3	9	0	1
4	5	2	13	15	13	15	0	1
5	5	3	15	18	15	18	0	1
6	10	3	15	18	17	20	2	0
7	4	4	5	9	6	10	1	0
8	4	5	9	14	10	15	1	0
9	0	4	9	13	9	13	0	1
10	0	2	18	20	18	20	0	1
11	0	3	20	23	20	23	0	1

Table 2-3. Resource allocations and resource links of the SIMPLE project calculated using the MINSLK heuristic.

Node	Resources Allocated	Resource Links
1	A_1, A_2, A_3	-
2	A_4, A_5	-
3	A_1, A_2, A_6	(1,3)*
7	A_3, A_4, A_5	(1,7),(2,7)
9	A_1, A_2, A_6	(3,9)
8	A_3, A_4	(7,8)
4	A_1, A_2, A_5, A_6	(7,4)(9,4)
5	A_3, A_4	(8,5)
6	A_5	(4,5)
10	A_1, A_2, A_3, A_6	(4,10),(5,10)
11	A_1, A_2, A_4, A_5, A_6	(5,11),(6,11),(10,11)

* The pair (a,b) means a resource link for node a to node b.

activity 3 and activity 4 could be started. Using the MINSLK heuristic, activity 3 should be scheduled next. Since activity 3 requires three type A resources, the three available resources, A_1, A_2, and A_6 will be assigned to activity 3. Since resources A_1 and A_2 were released by activity 1, a resource link from node 1 to node 3 is added. When activity 2 is completed, activities 4 and 7 are feasible to schedule. Activity 7 is chosen using the MINSLK heuristic and is assigned resources A_3, A_4 and A_5, where resource A_3 was released by activity 1 and resources A_4 and A_5 were released by activity 2. Two resource links, (1,7) and (2,7), are added. Following this procedure, one can find all of the resource links that need to be made.

Even though we have now added the resource links to the project network, resource requirements still need to be considered during the backward pass. The addition of precedence links will eliminate the need to consider resource requirements during the backward pass. The procedures for creating the resource links and precedence links during the forward pass are illustrated in the following paragraphs.

Assume that the project network has only one starting node denoted as S, and only one ending node denoted as T. Initially, no nodes are marked with start times. The fact that the start time for a node is unknown is denoted by saying that it is in set U. Two attributes are maintained for each resource, an attribute *AssignedTo* whose value is the number of the node to which this resource is assigned and an attribute *ReadyTime* which is the time when the resource will be available. The initial value for AssignedTo is the node S and the initial value of ReadyTime is zero.

Initially, the start time of node S is set to zero and node S is moved from set U to a list F. Nodes in the list F represent activities for which all predecessors have been scheduled,

waiting for resources to be available. The nodes in list F are ordered according to the priorities calculated by a specified rule. For example, the priority calculated by the MINSLK rule is that a smaller value of slack time has a higher priority. Once resources are allocated to an activity in list F based upon its priority, the activity will be moved from list F to list C. The nodes in list C are ordered according to the activity completion times. When a node becomes a member of set C, this means that the activity has been completed. The EST and the ECT are calculated and resource links and the precedence links are created. After the forward pass has been completed, the resource links and precedence links are added to the project network. During the backward pass, the LST and the LCT are calculated using exactly the same procedure as that of CPM/PERT without the need to consider resource requirements. The algorithm for determining resource links and precedence links is given in Table 2-4.

The forward pass through project SIMPLE is given in Table 2-5, where P-Link stands for the precedence link and R-Link stands for resource link. Resource links are noted as stated earlier in this section. Precedence links are noted following the algorithm in Table 2-4. At the beginning, starting activity S is moved to List C and resources are allocated to activities 1 and 2. Therefore, activities 1 and 2 are moved to List C at time 0 and ordered according to their ECTs. The clock is then advanced to time 3 which is the ECT of activity 1 and the value of C_ACT is 1. After activity 1 has been completed, two activities, 3 and 4, can be scheduled but must wait for resources and therefore are placed in List F. Since activity 3 has higher priority to acquire the resources, the available resources are allocated to activity 3 and the value of NEXT is 3. Based on the algorithm, a precedence link from activity 1 to activity 3 is noted. At time 5, C_ACT is 2 and NEXT is 7. Another precedence link from

activity 2 to activity 7 is noted. Following this procedure, one can find all of the precedence links that need to be made.

One can easily see that there are some resource links and precedence links which would be the same as existing technological links. In this case they don't need to be added to the project network explicitly.

Result and Discussion

The 110 test projects assembled by Patterson[10] were used to verify the proposed algorithm. According to Patterson, these 110 projects represent an accumulation of all multiple resource constrained problems existing in the literature. The number of resources required per activity vary between one and three. Of these 110 projects, 103 projects require 3 different types of resources, 3 projects require 2 different types of resources, and 4 projects require only one type of resource. The number of activities per project varies between 5 and 49. Each of the 110 projects have been successfully scheduled using the proposed algorithm.

Two comparisons were made to illustrate the relation between the number of precedence links created, the number of different resource types required and the number of activities in a project. Each of the 110 projects were scheduled using the MINSLK heuristic and activity durations were assumed to be deterministic. A comparison of number of resource links and precedence links required for each of the 110 projects based on the number of different resource types required is given in Table 2-6. The ratio of the average number of precedence links required to the sum of the average number of arcs in the original project plus

Table 2-4. The algorithm for figuring out the resource links and precedence links.

```
Place all nodes in set U.
Set the attribute AssignedTo of all resources to be node S.
Set the attribute ReadyTime of all resources equal to 0.
Set the initial value of variable ResReadyTime to be 0.
Move S from set U to list F.
Let the initial value of C_ACT, the completed activity, be S.
While not all activities have been scheduled do
        While the resources are available for the first node, denoted as NEXT, in list F do
                Set the EST of NEXT to be ResReadyTime.
                Compute ECT of NEXT.
                Move NEXT from list F to list C.
                If C_ACT is not equal to S and NEXT is not equal to T then
                        Add a precedence link from C_ACT to NEXT.
                endif
                Allocate resources to NEXT.
                For each resource allocated to NEXT do
                        Store the attribute AssignedTo in variable FROM.
                        Set the attribute AssignedTo to be NEXT.
                        if FROM is not equal to S then
                                Add a resource link from FROM to NEXT
                        endif
                        Set the attribute ReadyTime to be the ECT of NEXT.
                done
        done
        Remove the first node of list C, store in the variable C_ACT.
        Release the resources acquired by C_ACT.
        Let ResReadyTime be the ECT of the node C_ACT.
        Find the nodes whose all successors had been completed, move these nodes from set
        U to list F.
done
```

Table 2-5. The forward pass of the project SIMPLE.

Time	C-ACT	NEXT	Resource Allocated	Set U, List F and List C	P-Link & R-Link
0	---	S	---	U: 1,2,3,4,5,6,7,8,9,10,11,T F: C: S	---
0	S	1 2	A_1,A_2,A_3 A_4,A_5	U: 3,4,5,6,7,8,9,10,11,T F: C: 1,2	--- ---
3	1	3	A_1,A_2,A_6	U: 5,6,7,8,9,10,11,T F: 4 C: 2,3	P:(1,3) R:(1,3)
5	2	7	A_3,A_4,A_5	U: 5,6,8,9,10,11,T F: 4 C: 3,7	P:(2,7) R:(1,7) R:(2,7)
9	3	9	A_1,A_2,A_6	U: 5,6,8,10,11,T F: 4 C: 7,9	P:(3,9) R:(3,9)
9	7	8	A_3,A_4	U: 5,6,10,11,T F: 4 C: 9,8	P:(7,8) R:(7,8)
13	9	4	A_1,A_2,A_5,A_6	U: 5,6,10,11,T F: C: 8,4	P:(9,4) R:(7,4) R:(9,4)
14	8	---	---	U: 5,6,10,11,T F: C: 4	---
15	4	5 6	A_3,A_4 A_5	U: 10,11,T F: C: 5,6	P:(4,5) P:(4,6) R:(8,5) R:(4,6)
18	5	10	A_1,A_2,A_3,A_6	U: 11, T F: C: 6,10	P:(5,10) R:(4,10) R:(5,10)
18	6	---	---	U: 11, T F: C: 10	---
20	10	11	A_1,A_2,A_4,A_5,A_6	U: T F: C: 11	P:(10,11) R:(5,11) R:(6,11) R:(10,11)
23	11	T	---	U: F: C: T	---
23	T	---	---		

the average number of resource links required significantly decreases as the number of different resource types required increases. This indicates that in projects which need a large number of different types of resources the precedence links become a smaller part of the total links needed during the backward pass. Table 2-7 shows a comparison of the number of resource links and precedence links required for Patterson's 110 projects based on the number of activities. The ratio is lower for projects which have more activities.

Table 2-6. A comparison of the number of resource links and precedence links required for Patterson's 110 projects based on the number of different resource types required.

Resource type(s) required	*a.* Average number of links in the original project	*b.* Average number of resource links required	*c.* Average number of precedence links requi	$\frac{c}{(a+b)} \times 100\%$
1	20.25	20.75	13.50	32.93%
2	24.00	36.00	13.33	22.22%
3	36.24	105.58	22.61	16.48%

Table 2-7. A comparison of the number of resource links and precedence links required for Patterson's 110 projects based on the number of activities.

Number of activities	*a.* Average number of links in the original project	*b.* Average number of resource links required	*c.* Average number of precedence links requi	$\frac{c}{(a+b)} \times 100\%$
5,6,7	6.00	7.40	4.80	36.09%
11,12,16	15.33	18.33	10.33	33.88%
20,21,25	32.87	94.76	20.43	16.10%
33,49	71.00	202.10	44.08	17.10%

After carefully examining the precedence links required for each of the 110 projects, we found that most precedence links are identical to existing links. That is, most precedence links don't need to be added to the original project network which will simplify the calculation for the backward pass. In some cases, each of the precedence links are exactly the same as one of the original project network links or one of the required resource links.

Computational experiments also show that the precedence link works well for projects with stochastic activity duration. An example based on Patterson's HEUR3 project is given in Table 2-8 and Table 2-9. The activities have durations which have been changed from deterministic to random drawn from beta distributions. The results shown in Table 2-8 are the average of 100 runs. Table 2-9 shows the results for the run which has the shortest project duration of the 100 runs. The EST's and ECT's were calculated using the MINSLK rule. The LST's and LCT's of the schedule in Table 2-8 were calculated using CPM/PERT's backward pass method with the technology relationships, and the added resource links and the added precedence links.

Conclusion

Bowers proposed a revised method which can calculate the criticality of a activity by adding resource links and considering the resource requirements during the backward pass. For a project in which resources are highly utilized and many activities require the same resources, the maintenance of a resource database can be a complicated and time consuming job. An additional link called a precedence link was proposed to eliminate the need to

Table 2-8. The average of 100 runs for scheduling Patterson's HEUR3 project.

Node	Duration	EST	ECT	LST	LCT	Slack	Critical
1	3.225	0.000	0.000	0.202	3.427	0.202	0.64
2	5.244	0.000	5.244	0.401	5.645	0.401	0.36
3	6.355	5.288	11.643	6.092	12.447	0.804	0.30
4	2.064	3.225	5.288	3.427	5.490	0.202	0.64
5	3.126	5.577	8.703	5.753	8.880	0.176	0.70
6	3.200	13.003	16.204	13.505	16.705	0.502	0.33
7	4.243	8.703	12.946	8.880	13.122	0.176	0.70
8	5.192	13.003	18.196	13.551	18.743	0.548	0.38
9	4.234	11.643	15.877	12.448	16.682	0.805	0.29
10	2.084	16.484	18.568	16.682	18.766	0.199	0.62
11	3.176	18.766	21.943	18.766	21.943	0.000	1.00

The project duration is 21.943

Table 2-9. The shortest project duration of the 100 runs for scheduling Patterson's HEUR3 project.

Node	Duration	EST	ECT	LST	LCT	Slack	Critical
1	2.805	0.000	2.805	0.265	3.070	0.265	0.00
2	4.768	0.000	4.768	0.000	4.768	0.000	1.00
3	5.804	4.502	10.306	5.175	10.979	0.672	0.00
4	1.698	2.805	4.502	3.070	4.768	0.265	0.00
5	2.696	4.768	7.464	4.768	7.464	0.000	1.00
6	2.904	11.694	14.598	11.694	14.598	0.000	1.00
7	4.230	7.464	11.694	7.464	11.694	0.000	1.00
8	4.240	11.694	15.934	12.473	16.713	0.779	0.00
9	3.619	10.306	13.926	10.979	14.598	0.672	0.00
10	2.115	14.598	16.713	14.598	16.713	0.000	1.00
11	2.578	16.713	19.290	16.713	19.290	0.000	1.00

The project duration is 19.290.

maintain a resource database. This greatly reduces the complexity of the backward pass, allowing the standard CPM/PERT method to be used.

The 110 projects of Patterson were successfully scheduled using the proposed algorithm. The additional precedence links make up only a small part of the total number of links that need to be considered during the backward pass. In some cases, each of the precedence links are identical to some of the original network or resource links. Experimental results show that the algorithm can be used in both deterministic activity duration projects and stochastic activity duration projects.

References

1. A. B. Badiru (1991) A Simulation Approach to PERT Network Analysis. *Simulation*. **57**, 245-255.

2. A. B. Badiru and P. S. Pulat (1995) *Comprehensive Project Management*. PRENTICE HALL PTR, Englewood Cliffs, New Jersey.

3. J. A. Bowers (1995) Criticality in Resource Constrained Networks. *Journal of the Operational Research Society*. **46**, 80-91.

4. E. Demeulemeester and W. Herroelen (1992) A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. *Management Science*. **38**, 1803-1818.

5. E. Demeulemeester (1995) Minimizing Resource Availability Costs in Time-Limited Project Networks. *Management Science*. **41**, 1590-1598.

6. J. B. Dreger (1992) *Project Management - Effective Scheduling*. VAN NOSTRAND REINHOLD, New Work, New Work.

7. S. R. Lawrence and T. E. Morton (1993) Resource-Constrained Multi-project Scheduling with Tardy Costs: Comparing Myopic, Bottleneck, and Resource Pricing Heuristics. *European Journal of Operational Research*. **64**, 168-187.

8. L. C. Morse, J. O. McIntosh, and G. E. Whitehouse (1996) Using Combinations of Heuristics to Schedule Activities of Constrained Multiple Resource Projects. *Project Management Journal*. March, 34-40.

9. L. Ozdamar and G. Ulusoy (1995) A Survey on the Resource-Constrained Project Scheduling Problem. *IIE Transactions*. **27**, 574-586.

10. J. Patterson (1984) A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem. *Management Science*. **30**, 854-867.

11. K. Watkins (1993) *Discrete Event Simulation in C*. McGRAW-HILL Book Company, Europe.

12. B. M. Woodworth and S. Shanahan (1988) Identifying the Critical Sequence in a Resource Constrained Project. *Project Management*. **6**, 89-96.

CHAPTER 3. USING SIMULATED ANNEALING TO SCHEDULE ACTIVITIES OF STOCHASTIC RESOURCE-CONSTRAINED PROJECTS

A paper submitted to the Project Management Journal

Ying-Wei Tsai and Douglas D. Gemmill

Abstract

The goal of this paper is to demonstrate the application of a simple algorithm which can be easily applied to various kinds of resource-constrained, randomized activity duration project scheduling problems, and that will perform better in most cases than existing heuristics. The simulated annealing (SA) procedure is proposed in this paper. The SA algorithm proves to be an efficient way to find good solutions to both the deterministic and stochastic problems. Improved solutions to those provided by existing heuristics are provided in minimum computational time. In addition, SA finds the optimal solution to most of the test problems investigated.

Introduction

CPM and PERT are well known project planning techniques used to determine the critical path of a project and hence the total time required to complete the project. These methods can be used to efficiently schedule the sequence of activities that make up a project assuming that there are no resource availability limitations. However, the ability of

PERT/CPM to deal with variations in resource requirements when the availability of resources is constrained is extremely limited. It is important to consider how project duration will be affected by limited resources due to the fact that many projects must be completed with a limited set of resources. The number of resources available can be affected by many factors, such as budget constraints and number of skilled employees available. Resource constraints can be a key factor determining the project duration, and the duration can vary greatly with different levels of resource availability.

Consideration of resource allocation has emerged as an important aspect of project scheduling. Many researchers have worked on methods to minimize project duration given a limited set of resources. Ozdamar and Ulusoy [8] extensively surveyed the approaches used to solve the resource constrained project. For example, a zero-one programming approach proposed by Pritsker *et al.* [10] and a branch-and-bound procedure derived by Demeulemeester and Herroelen [5] are able to find the optimal solution of a project which has deterministic activity durations and resource requirements. Pritsker *et al.* [10] solve an example which includes three projects, eight activities and three resource types whose availability is constrained. The setup of this small example requires the definition of 33 variables and 37 constraints. As the size of each individual project and the number of total projects increases, it can be difficult to handle the variables and constraints used in this zero-one programming approach. Demeulemeester and Herroelen apply their the branch-and-bound procedure to the 110 projects assembled by Patterson [9] and are able to find the optimal solution to each of the problems in a short computation time.

Heuristic rules for project scheduling under resource-constrained conditions have also been developed, such as minimum slack first, shortest job first, etc. Many of these heuristics are discussed in [8]. Most of the rules are easy to implement and are able to find good, and sometimes optimal, solutions to the problems. Badiru [1] defines a heuristic rule in which the activities are scheduled according to what he calls the Composite Allocation Factor (CAF). Badiru is able to obtain reasonably good results when his CAF heuristic is applied to a project in which the activity durations are randomized. However, the CAF heuristic may generate poor schedules for some projects. Morse, McIntosh and Whitehouse [7] combine some heuristics to schedule activities of constrained multiple resource projects. The combined heuristics are applied to a project in which the activity durations are deterministic. Using different heuristic procedures on the same resource constrained project generates different project durations. A heuristic rule that performs well on one project might perform poorly on others.

Both the zero-one programming approach and the branch-and-bound procedure discussed are able to determine the optimal solution to resource-constrained project scheduling problems. In both cases, as the size of the project increases their application becomes more difficult. More importantly, both of these methods assume that the activity durations and resource requirements are deterministic. Heuristics are available which are easier to use but cannot guarantee optimality. Heuristics have been developed which can be applied to both deterministic and stochastic problems. However, a heuristic rule that performs well on one project might perform poorly on the others. The goal of this paper is to demonstrate the application of a simple algorithm which can be easily applied to various kinds

of resource-constrained, randomized activity duration project scheduling problems, and that will perform better in most cases than existing heuristics. The application of an optimization method called simulated annealing (SA) is developed in this paper. SA has been applied to many difficult optimization problems, such as the traveling salesman problem, computer (VLSI) design and graph partitioning problem, and good solutions have been obtained.

The remainder of this paper is organized as follows. The methodology section includes discussion of the implementation of SA and an example is given to illustrate the procedures. Experimental results are then shown in the results section for both deterministic and stochastic problems. Finally, a conclusion is given in the last section.

Methodology

Before presenting the implementation of SA we will briefly discuss the techniques used to model project networks. In addition, we will present the basic approach used by both of the heuristic methods that are used to provide initial feasible solutions.

There are two basic structures used to model project networks: activity-on-arrow (AOA) and activity-on-node (AON). We chose to employ the AON structure based upon the fact that the AON structure is growing in popularity and seems to possess significant advantages over the AOA structure. Bowers [3] states using the AON structure allows for a much more succinct algorithm for determining project duration than does the use of the AOA structure.

A typical project network selected from Patterson's 110 test projects [9] is shown in Figure 3-1. This project has three types of resources (type A, type B and type C). Assume the number of each resource available is 6 type A, 7 type B, and 6 type C resources. Then, given the deterministic activity durations and resource requirements shown in Table 3-1, the shortest possible project duration is 20 time units. The optimal schedule for this project is also shown in Table 3-1.

As stated earlier, many heuristic rules have been developed to solve resource constrained project scheduling problems such as the one described in Figure 3-1 and Table 3-1. The heuristics basically assign priorities to each of the activities based upon some rule. Then, for a given day k (or time unit k) the activity with the highest priority is scheduled first if its precedence requirements have been met and the available resources are sufficient for day k. When there are not enough free resources available for day k, day $k+1$ is considered.

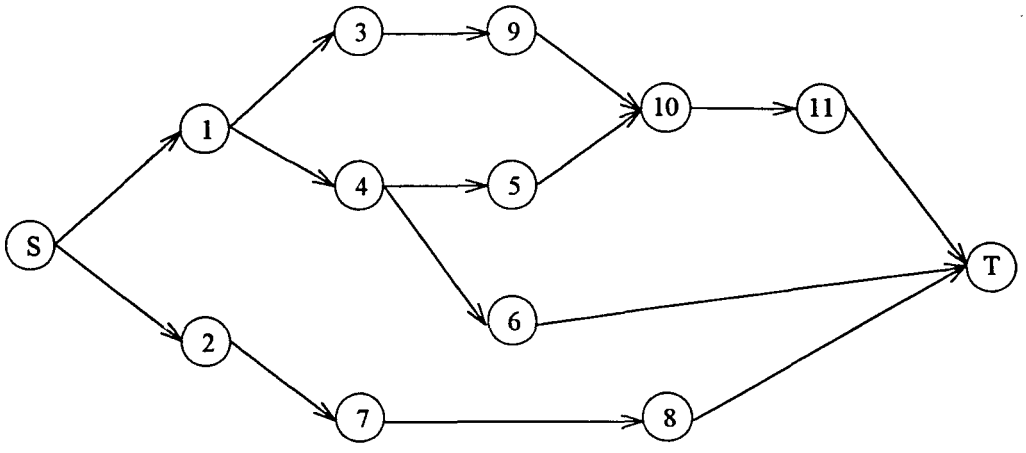

Figure 3-1. A typical project selected from the Patterson's 110 test projects.

Table 3-1. Duration, resource requirements and optimal schedule of the sample project.

Node (Activity)	Duration	Resource requirements	Start time in the optimal schedule
1	3	3,2,1[a]	0
2	5	2,4,2	0
3	6	3,1,2	5
4	2	4,3,1	3
5	3	2,0,3	9
6	3	1,1,1	12
7	4	3,1,1	5
8	5	2,2,2	12
9	4	3,2,3	11
10	2	4,1,0	15
11	3	5,4,2	17

[a] The resource requirements of type A, type B, and type C resources respectively.

Typically, ties between activities which have the same priority are broken arbitrarily. Also, most algorithms require that once an activity is started it will not be preempted.

In the SA approach that will be discussed later, we chose to use two heuristics to provide initial feasible solutions. SA is then used to improve upon the initial solutions provided. We chose to use the Composite Allocation Factor (CAF) method, proposed by Badiru [1] and the Minimum Slack First (MINSLK) method, used by Morse, McIntosh and Whitehouse [7] to provide initial feasible solutions. We chose to use MINSLK since Davis and Patterson [4] found it to be one of the better heuristic rules for the resource constrained project scheduling problem. The CAF method was chosen because it is one of the most recent methods developed and is a somewhat unique approach to the problem.

In the CAF rule proposed by Badiru [1], the priority given to an activity is determined by its CAF value. The greater the CAF, the higher the priority. The CAF for activity i is calculated as follows:

$$CAF_i = w * RAF_i + (1 - w) * SAF_i$$

where w is the resource allocation weighting factor between 0 and 1. RAF_i is defined as

$$RAF_i = \frac{1}{t_i} \sum_{j=1}^{R} \frac{x_{ij}}{y_j}$$

where x_{ij} is the number of resource type j units required by activity i, y_j is the maximum number of units of resource type j required by any activity in the project, t_i is the expected duration of activity i and R is the number of resource types involved. SAF_i is defined as

$$SAF_i = t_i + \frac{s_i}{t_i}$$

where s_i is the standard deviation of the duration for activity i. The set of RAF_i and SAF_i values are scaled from 0 to 100 to eliminate the time-based units. The scaling procedure is described in [2].

The MINSLK rule is very simple. The unconstrained PERT/CPM slack time for each activity is calculated. Then when resources are assigned, the top priority is given to the activity with the smallest PERT/CPM calculated slack time.

When resources are constrained, determination of the critical path becomes somewhat more difficult. Bowers [3] proposed the idea of using resource links for this purpose. These resource links are used to trace the use of resources, noting the significant resource dependencies that determine the project schedule and hence identify the critical path. We

have added the use of additional links which are referred to as precedence links. As with unconstrained problems, the earliest start times and latest start times are determined, which then identifies the critical path.

The earliest start times are determined as follows: the feasible activity which has the highest priority is scheduled if the available resources are sufficient for day k. If sufficient resources are not available for day k, then day $k+1$ is considered. During the forward pass of the construction of the resource constrained schedule, the particular unit(s) of each resource used for an activity are recorded and this information is then used to create resource links. Besides the resource links, additional links called precedence links are added. Precedence links are added to explicitly represent the precedence relationships in a specified schedule sequence. Addition of the precedence and resource links ensures that the activities are scheduled in the same order during both the forward pass and the backward pass. After the addition of the resource links and the precedence links to the original project, the latest start time (LST) and the latest completion time (LCT) are determined using the backward pass similar to the backward pass of CPM/PERT.

Suppose a feasible schedule order for the sample project of Figure 3-1 is as follows: 2-1-3-7-9-8-4-5-6-10-11. Then the project network with resource links for type C resources is shown in Figure 3-2. As illustrated in Table 3-2, resources are numbered and allocated to activities according to the feasible schedule order. The precedence links and resource links are shown in Table 3-3.

Table 3-2. Resource allocation for sample project.

Node (Activity)	Type A resource	Type B resource	Type C resource
1	A_3,A_4,A_5	B_5,B_6	C_3
2	A_1,A_2	B_1,B_2,B_3,B_4	C_1,C_2
3	A_3,A_4,A_6	B_7	C_4,C_5
4	A_2,A_3,A_4,A_5	B_1,B_6,B_7	C_6
5	A_1,A_6	---	C_1,C_2,C_3
6	A_2	B_2	C_4
7	A_1,A_2,A_5	B_1	C_6
8	A_1,A_6	B_4,B_5	C_4,C_5
9	A_3,A_4,A_5	B_2,B_3	C_1,C_2,C_3
10	A_3,A_4,A_5,A_6	B_3	---
11	A_1,A_2,A_3,A_4,A_5	B_4,B_5,B_6,B_7	C_5,C_6

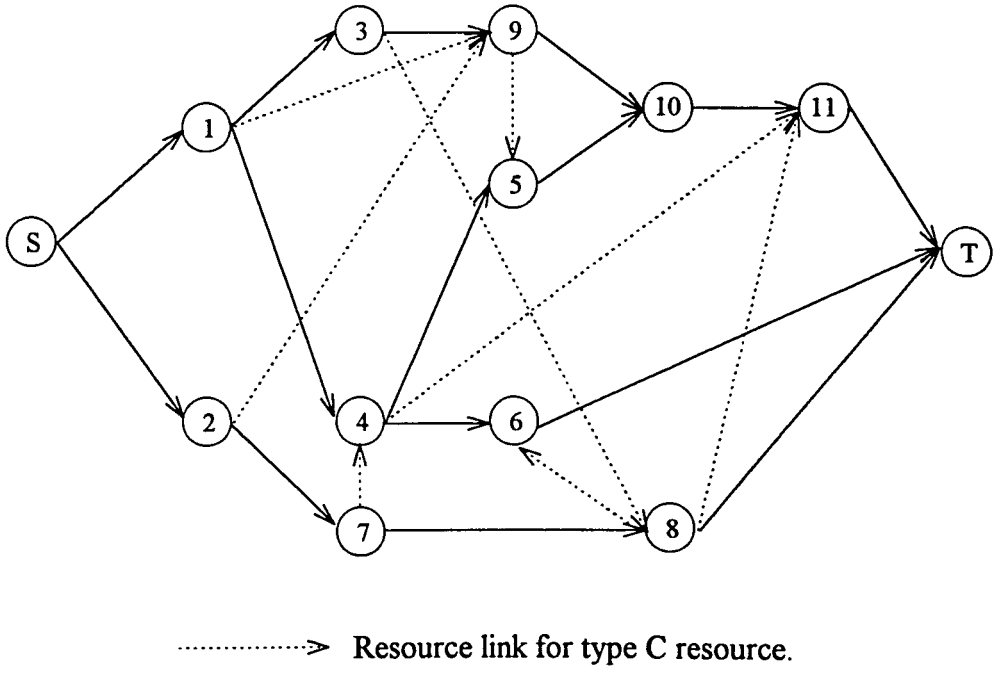

Figure 3-2. Project network with the resource links for type C resource.

Table 3-3. All precedence links and resource links for sample project.

Precedence Link	Resource link for Type A resource	Resource link for type B resource	Resource link for type C resource
(1,3)[a]	(1,3)	(1,4)	(1,9)
(2,7)	(1,7)	(1,8)	(2,9)
(3,9)	(2,7)	(2,7)	(3,8)
(4,5)	(3,8)	(2,8)	(4,11)
(4,6)	(3,9)	(3,4)	(7,4)
(5,10)	(4,6)	(4,11)	(8,6)
(7,8)	(4,10)	(7,4)	(8,11)
(9,4)	(5,10)	(8,11)	(9,5)
(10,11)	(5,11)	(9,6)	
	(6,11)	(9,10)	
	(7,4)		
	(7,8)		
	(7,9)		
	(8,5)		
	(9,4)		
	(10,11)		

[a] The pair (a,b) indicates a link from node a to node b.

A feasible schedule of this project determined with the CAF method is shown in Table 3-4. The weighting factor used for this example is $w = 0.1$. A schedule determined with the MINSLK rule is shown in Table 3-5. The notation is defined as:

CAF : priority value computed by CAF rule,

MINSLK : priority value computed by MINSLK rule,

EST : Earliest Start Time,

ECT : Earliest Completion Time,

LST : Latest Start Time,

LCT : Latest Completion Time,

SLACK : time between the LST and EST,

CRITICAL : CRITICAL = 1 if this node is on the critical path.

Application of Simulated Annealing to the Deterministic Problem

Simulated annealing is a random search technique that recently has been successfully applied to a number of problem types. Although SA can be shown to converge to the optimal solution under certain conditions, in general it is applied as a heuristic approach designed to give a good solution which might not be the optimal solution. The name SA comes from an analogous procedure in statistical mechanics where experiments that determine the low-temperature state of a material are performed by careful annealing, first melting the substance and then lowering the temperature slowly. In addition, controlled uphill steps are utilized in a manner similar to the Metropolis procedure in order to allow the process to escape from local

Table 3-4. A feasible schedule for the sample project determined with CAF.

Node	CAF	EST	ECT	LST	LCT	SLACK	CRITICAL
1	57.826	0	3	0	3	0	1
2	87.847	0	5	1	6	1	0
3	100.00	3	9	3	9	0	1
4	52.428	13	15	13	15	0	1
5	57.578	15	18	15	18	0	1
6	53.000	15	18	17	20	2	0
7	69.502	5	9	6	10	1	0
8	85.620	9	14	10	15	1	0
9	74.606	9	13	9	13	0	1
10	43.149	18	20	18	20	0	1
11	66.981	20	23	20	23	0	1

Schedule order : 2-1-3-7-9-8-4-5-6-10-11.
The project duration is 23.

Table 3-5. A feasible schedule for the sample project determined with MINSLK.

Node	MINSLK	EST	ECT	LST	LCT	SLACK	CRITICAL
1	0	0	3	0	3	0	1
2	4	0	5	1	6	1	0
3	0	3	9	3	9	0	1
4	5	13	15	13	15	0	1
5	5	15	18	15	18	0	1
6	10	15	18	17	20	2	0
7	4	5	9	6	10	1	0
8	4	9	14	10	15	1	0
9	0	9	13	9	13	0	1
10	0	18	20	18	20	0	1
11	0	20	23	20	23	0	1

Schedule order : 1-2-3-7-9-8-4-5-6-10-11.
The project duration is 23.

optimum points. In other words, SA will allow the search process to accept a configuration that results in a higher cost (climb a hill), but the likelihood of acceptance decreases with time as the temperature is lowered.

We use SA to search for the best schedule, or the schedule that minimizes the total time required to complete a project given precedence requirements and resource constraints. The set of all feasible schedules which satisfy the precedence relationships and resource constraints is a finite set of unknown size and is denoted as S. A function f, a real valued function, is defined as the project duration on the schedules of set S. The goal is to find a feasible schedule from set S which minimizes or nearly minimizes f over set S.

A pseudo-code for the SA algorithm based upon the implementation found in Eglese [6] is stated as follows:

Step 1. Select an initial feasible schedule, denoted as S_i.

Step 2. Set temperature change counter $t = 0$.

Step 3. Select an initial temperature $T(0) > 0$.

Step 4. Set repetition counter $n = 0$.

Step 5. Generate a feasible schedule S_j from the neighborhood of schedule S_i.

Step 6. Calculate $\delta = f(S_j) - f(S_i)$.

Step 7. If $\delta < 0$ then S_i is set equal to S_j;

else if *random* (0, 1) < $exp\ (-\delta / T(t))$ then S_i is set equal to S_j;

otherwise, S_i remains unchanged.

Step 8. $n = n + 1$.

Step 9. If $n < N(t)$ then go to **Step 5**.

Step 10. $t = t + 1$.

Step 11. $T(t) = \alpha T(t-1)$.

Step 12. If stopping criterion is true then stop; else go to **Step 4**.

In our application, we chose to use SA as an improvement algorithm. Therefore, an initial feasible schedule is required before the application of SA begins. As discussed earlier, the Composite Allocation Factor rule and the Minimum Slack First rule were chosen to provide initial feasible solutions. The temperature, $T(t)$, is started at a relatively high value and is decreased with a proportional function, $T(t) = \alpha T(t-1)$, where α is a constant. Typical values used for α are between 0.8 and 0.99. This procedure for decreasing $T(t)$ is called the cooling schedule. Initially, when the value of $T(t)$ is higher, the acceptance function $exp\ (-\delta / T(t))$ has a higher value which increases the probability of accepting "hill-climbs" and thus

avoids being trapped in a local minimum. A certain number, *N(t)*, of neighborhood schedules are tried at each temperature. The neighborhood of S_i is the set of all schedules which satisfy the precedence requirements and which differ from S_i in only two positions (i.e. the sequence of activities is different in only two positions).

The method used to determine whether or not a neighbor of S_i satisfies the precedence requirements is as follows. Assume the number of nodes (activities) is *N*. An *N* by *N* precedence matrix, denoted as P, was adopted to verify that the schedule satisfies the precedence requirements. Each element of precedence matrix P is set as:

$P_{i,j} = 1$, if node *j* is a successor of node *i*.

$P_{i,j} = 0$, otherwise.

An 11 by 11 matrix of the precedence requirements for the sample project of Figure 3-1 is shown in Table 3-6.

Table 3-6. An 11 by 11 matrix for the precedence requirements of the sample project.

NODE	1	2	3	4	5	6	7	8	9	10	11
1	0	0	1	1	1	1	0	0	1	1	1
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	1	1	0	0	0	1	1
5	0	0	0	0	0	0	0	0	0	1	1
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1
10	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0

An initial feasible sequence of 1-2-3-7-9-8-4-5-6-10-11 is found using the MINSLK rule. After the initial feasible sequence has been found, a neighborhood schedule can be found by interchanging two activities in the sequence. If a schedule generated doesn't satisfy the precedence requirements, a new schedule is generated by interchanging two different activities until a schedule is found in which the precedence requirements are satisfied.

Suppose there is a feasible schedule sequence

$$n_1, n_2, \ldots, n_{i-1}, n_i, n_{i+1}, \ldots, n_{j-1}, n_j, n_{j+1}, \ldots, n_N$$

and two nodes, n_i and n_j, $i < j$, are randomly selected for interchange. By interchanging these two nodes, a new sequence,

$$n_1, n_2, \ldots, n_{i-1}, n_j, n_{i+1}, \ldots, n_{j-1}, n_i, n_{j+1}, \ldots, n_N$$

is generated. To check the feasibility of the generated sequence, one examines

$$P_{k,j} = 0, \text{ for } k = i, i+1, \ldots, j-2, j-1$$

and

$$P_{i,k} = 0, \text{ for } k = i+1, i+2, \ldots, j-1, j$$

As an example of the SA process, two neighborhood schedules of the initial sequence 1-2-3-7-9-8-4-5-6-10-11 are given in Table 3-7, one with a shorter project duration and another with a longer duration. The SA process would accept the sequence 1-2-3-7-4-8-9-5-6-10-11 immediately due to its shorter duration. The second sequence 1-2-7-3-9-8-4-5-6-10-11 would be accepted with probability *exp (-2 / T(t))* since its duration is longer.

Even with resource constraints, it is possible that the available resources are sufficient to complete the project in the time determined by the PERT/CPM analysis. However, the final solution provided by the SA algorithm could be worse than this known best solution due to

Table 3-7. Two neighborhood schedules of the initial schedule calculated by MINSLK rule.

(A) Neighborhood schedule 1.

Node	Duration	EST	ECT
1	3	0	3
2	5	0	5
3	6	3	9
4	2	9	11
5	3	14	17
6	3	15	18
7	4	5	9
8	5	9	14
9	4	11	15
10	2	7	19
11	3	19	22

Schedule order : 1-2-3-7-4-8-9-5-6-10-11.
The project duration is 22.

(B) Neighborhood schedule 2.

Node	Duration	EST	ECT
1	3	0	3
2	5	0	5
3	6	5	11
4	2	15	17
5	3	17	20
6	3	17	20
7	4	5	9
8	5	15	20
9	4	11	15
10	2	20	22
11	3	22	25

Schedule order : 1-2-7-3-9-8-4-5-6-10-11.
The project duration is 25.

the hill climbing procedure that SA employs. For the same reason, it is possible that a better solution for a given project is found at an intermediate stage of the SA process than the final schedule S_i. Therefore, a mechanism is employed to maintain a record of the best solution found during the SA process, and this is returned as the final solution at the completion of the optimization process.

Two stopping criteria are used to terminate the SA iterative process. The main stopping criteria is based upon a minimum temperature *MinT*. When the temperature has been reduced to *MinT*, the iterative process is terminated. A secondary stopping criteria is based upon the number of consecutive iterations that have been performed without accepting a new sequence of activities. If the number of iterations performed without accepting a new sequence exceeds the value of *Stopping*, then the SA process is terminated. The values of *MinT* and *Stopping* were chosen based upon values that worked well during experimental runs, but no attempt was made to optimize the choice of these values.

An experimental result for the project of Figure 3-1 using the modified SA algorithm is shown in Figure 3-3 and Table 3-8. The MINSLK rule is used to provide the initial schedule. The project duration without considering resource constraints is 18 and the project duration for the initial schedule is 23. The initial temperature *T(0)* is set to one-half project duration of the initial schedule and *N(t)* is equal to the number of activities. In this way, both initial temperature *T(0)* and the number of neighborhood schedules generated *N(t)* depend on the particular project being optimized. The durations of the schedules generated during the SA process are shown in Figure 3-3. This figure clearly shows that schedules which generate longer project durations are accepted with some probability during the SA process, and that

this probability decreases with time. When the stopping criterion was met, the best schedule found and reported as the solution is 1-2-4-7-3-5-9-6-8-10-11, which is shown in Table 3-8. This is the same as the optimal schedule provided by Patterson [9]. The optimal project duration 20 includes no slack time for any of the activities. Therefore, the overall project will be delayed if any one activity is delayed.

Application of Simulated Annealing to the Stochastic Problem

Procedures and examples described to this point have dealt with projects which have deterministic activity durations. We will now extend the application of the SA algorithm to the resource-constrained, randomized activity duration project scheduling problem. Although SA was developed for use on deterministic problems, we will show that it works well on this class of stochastic problems.

The random variables in our problem are the individual durations of each activity. It has been common in PERT analysis to use three time estimates, an optimistic time estimate a, the most likely time estimate m, and a pessimistic time estimate b, in determining the distribution from which to sample for activity durations. The three time estimates might be

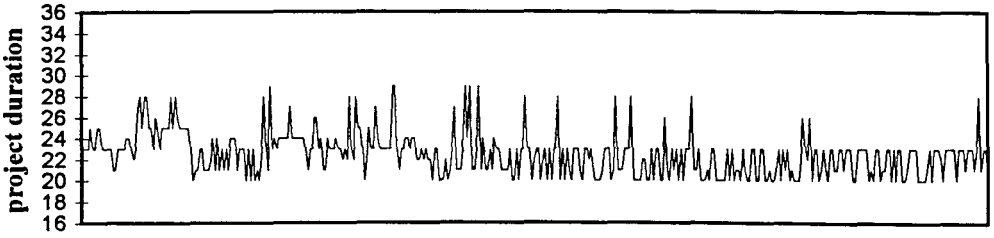

Figure 3-3. Duration of schedules generated during the SA process. The sample project has deterministic activity durations.

Table 3-8. The schedule determined by the SA algorithm.

Node	EST	ECT	LST	LCT	SLACK	CRITICAL
1	0	3	0	3	0	1
2	0	5	0	5	0	1
3	5	11	5	11	0	1
4	3	5	3	5	0	1
5	9	12	9	12	0	1
6	12	15	12	15	0	1
7	5	9	5	9	0	1
8	12	17	12	17	0	1
9	11	15	11	15	0	1
10	15	17	15	17	0	1
11	17	20	17	20	0	1

Schedule order : 1-2-4-7-3-5-9-6-8-10-11.
The project duration is 20.

estimated based on historical data, experience, customer requirements, simple forecasting or simulation. Although we could have used any distribution from which to sample for each activity, we have chosen to use the common historical method in this paper. Thus, the formulas for the expected activity duration t_e and the variance of activity duration s^2 are

$$t_e = \frac{a + 4m + b}{6}$$

$$s^2 = \frac{(b-a)^2}{36}$$

The beta distribution was then chosen to model activity duration and the formulas for parameters α and β of the beta distribution as derived by Badiru [1] are:

$$\phi = \frac{5a - 4m - b}{a + 4m - 5b}$$

$$\alpha = \phi\beta$$

$$\beta = \frac{-(\phi^2 - 34\phi + 1)}{(\phi + 1)^3}$$

Since we are now dealing with a project that has random activity durations, the overall project duration is also a random variable. Given a particular schedule or sequence of activities $n_1, n_2, n_3, \ldots, n_{N-1}, n_N$, the expected project duration for scheduling the activities in this order is computed as follows:

Step 1. Select number of repetitions $R > 0$ (sample size).

Step 2. Set the repetition counter $n = 0$.

Step 3. Set summation of project duration *SumDuration* = 0.

Step 4. Generate random activity durations from the beta distribution.

Step 5. Compute the project duration by allocating resources according to the schedule order $n_1, n_2, n_3, \ldots, n_{N-1}, n_N$, denoted as *Duration*.

Step 6. *SumDuration* = *SumDuration* + *Duration*.

Step 7. $n = n + 1$.

Step 8. If $n < R$ go to **Step 4**.

Step 9. The expected project duration for scheduling the activities in the order $n_1, n_2, n_3, \ldots, n_{N-1}, n_N$ is *SumDuration*/R.

The SA algorithm used to find the solution for a deterministic activity duration schedule is modified to find the solution for randomized activity durations. Instead of using function f for deterministic activity duration, a real valued function g is introduced. The function g is defined as the expected project duration on schedules of set S. The revised SA algorithm is as follows:

Step 1. Select an initial feasible schedule, denoted as S_i.

Step 2. Set the best schedule found $S_c = S_i$.

Step 3. Set temperature change counter $t = 0$.

Step 4. Select an initial temperature $T(0) > 0$.

Step 5. Set the not found better schedule counter *NotBetter* = 0.

Step 6. Set the repetition counter $n = 0$.

Step 7. Generate a feasible schedule S_j, from the neighborhood of schedule S_i.

Step 8. Calculate $\delta = g(S_j) - g(S_i)$.

Step 9. If $\delta < 0$ then S_i is set equal to S_j and *NotBetter* = 0;

else if *random* (0, 1) < $exp(-\delta / T(t))$ then S_i is set equal to S_j and *NotBetter* = 0;

otherwise, S_i remains unchanged and *NotBetter* = *NotBetter* + 1.

Step 10. If $g(S_i) < g(S_c)$ then $S_c = S_i$.

Step 11. $n = n + 1$.

Step 12. If $n < N(t)$ then go to **Step 7**.

Step 13. $t = t + 1$.

Step 14. $T(t) = T(t-1)$.

Step 15. If $T(t) <$ *MinT* or *NotBetter* > *Stopping* then S_c is the best schedule found; else go to **Step 6**.

As an example, the SA algorithm for randomized activity duration is applied to the sample project of Figure 3-1. The optimistic time estimate a is set to 0.8 times the deterministic activity duration, the most likely time estimate m is set to the deterministic

activity duration, and the pessimistic time estimate b is set to 1.5 times the deterministic activity duration. Using these values of a, m, and b gives a longer expected duration for each activity, equivalent to 1.05 times the deterministic activity duration. Before using SA, the MINSLK rule is used to find a schedule for the project, using the expected activity duration times as if they were deterministic. After the schedule is determined using the MINSLK rule in this fashion, the expected project duration is determined for this schedule using the procedure presented earlier in this section. Using a sample size of 100 (R = 100), the average project duration obtained using MINSLK is 24.372, as shown in Table 3-9. These results can now be used as a comparison with the results of the application SA.

The results of application of SA to this sample project are shown in Figure 3-4 and Table 3-10. For this example, the initial feasible schedule S_i is generated by using the MINSLK rule. Again, the average project duration is determined using a sample size of 100 (R = 100). Table 3-10 shows the sample averages of the variables associated with the schedule determined using the SA algorithm, and the solution has an average project duration of 21.746. This is an improvement of about 10.8% over the schedule determined using MINSLK. The results of the application of SA to a larger problem (25 activities) are shown in Figure 3-5, which provides a better illustration of the convergence towards the optimal solution.

Results & Discussion

The SA algorithm has been coded using Borland C++ Version 5.0 for a personal computer with Pentium-166 Mhz CPU running under the Windows 95 operating system. All

other applications are closed during SA program execution in order to measure the execution time of the SA algorithm. The 110 test projects constructed by Patterson [9] are used to evaluate the SA algorithm. 108 projects of the 110 test projects are solved in Morse, McIntosh and Whitehouse [7] utilizing the MINSLK rule. Of the 108 projects, the average percentage increase above the known optimum is 10.9% and the number of projects in which the optimal duration is found is 20 with the MINSLK rule. Morse, McIntosh and Whitehouse also use a combination of heuristics to search for good solutions, and a portion of their results are summarized in Table 3-11. The combination of heuristics produce results that are significantly better than those obtained by using a single heuristic. The best results found in that paper with the combination of seven heuristic rules provide a 2.68% increase above the

Table 3-9. The average duration of the sample project for the schedule determined with the MINSLK rule.

Node	Duration	EST	ECT	LST	LCT	SLACK	CRITICAL
1	3.156	0.000	3.156	0.461	3.617	0.461	0.44
2	5.282	0.000	5.282	0.291	5.574	0.291	0.56
3	6.243	3.156	9.399	3.618	9.861	0.462	0.44
4	2.081	13.988	16.069	13.988	16.069	0.000	1.00
5	3.100	16.069	19.169	16.069	19.169	0.000	1.00
6	3.143	16.069	19.213	18.136	21.280	2.067	0.00
7	4.182	5.282	9.464	5.574	9.755	0.291	0.56
8	5.260	9.464	14.724	10.654	15.915	1.190	0.00
9	4.126	9.861	13.988	9.861	13.988	0.000	1.00
10	2.111	19.169	21.280	19.169	21.280	0.000	1.00
11	3.093	21.280	24.372	21.280	24.372	0.000	1.00

Schedule order : 1-2-3-7-8-9-4-5-6-10-11.
The project duration is 24.372.

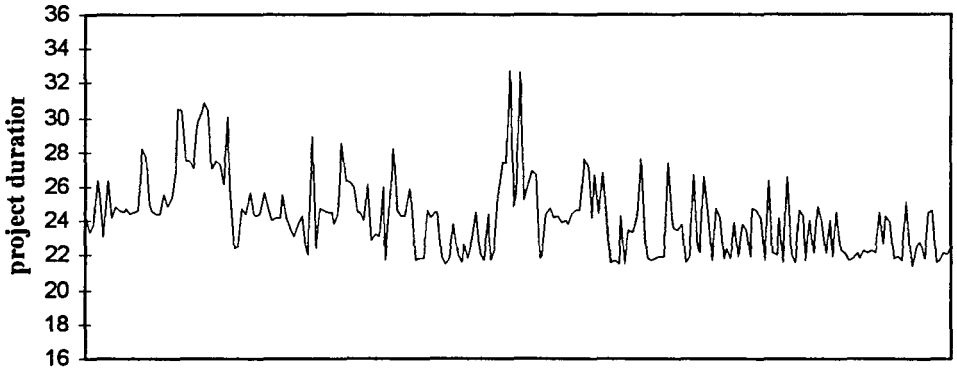

Figure 3-4. Duration of schedules generated during the SA process. The sample project has randomized activity durations.

Table 3-10. The average project duration of the schedule for the sample project determined by the SA algorithm.

Node	Duration	EST	ECT	LST	LCT	SLACK	CRITICAL
1	3.200	0.000	3.200	0.168	3.368	0.168	0.68
2	5.277	0.000	5.277	0.381	5.659	0.381	0.32
3	6.320	5.286	11.606	5.817	12.137	0.531	0.41
4	2.086	3.200	5.286	3.368	5.454	0.168	0.68
5	3.174	5.554	8.728	5.825	8.999	0.271	0.59
6	3.143	17.577	20.721	18.603	21.746	1.025	0.11
7	4.221	8.728	12.949	8.999	13.220	0.271	0.59
8	5.201	12.987	18.189	13.319	18.520	0.331	0.60
9	4.288	11.606	15.894	12.141	16.429	0.536	0.40
10	2.074	15.894	17.968	16.429	18.504	0.536	0.40
11	3.124	18.591	21.715	18.622	21.746	0.030	0.89

Schedule order : 1-2-4-3-5-7-9-8-10-6-11.
The project duration is 21.746.

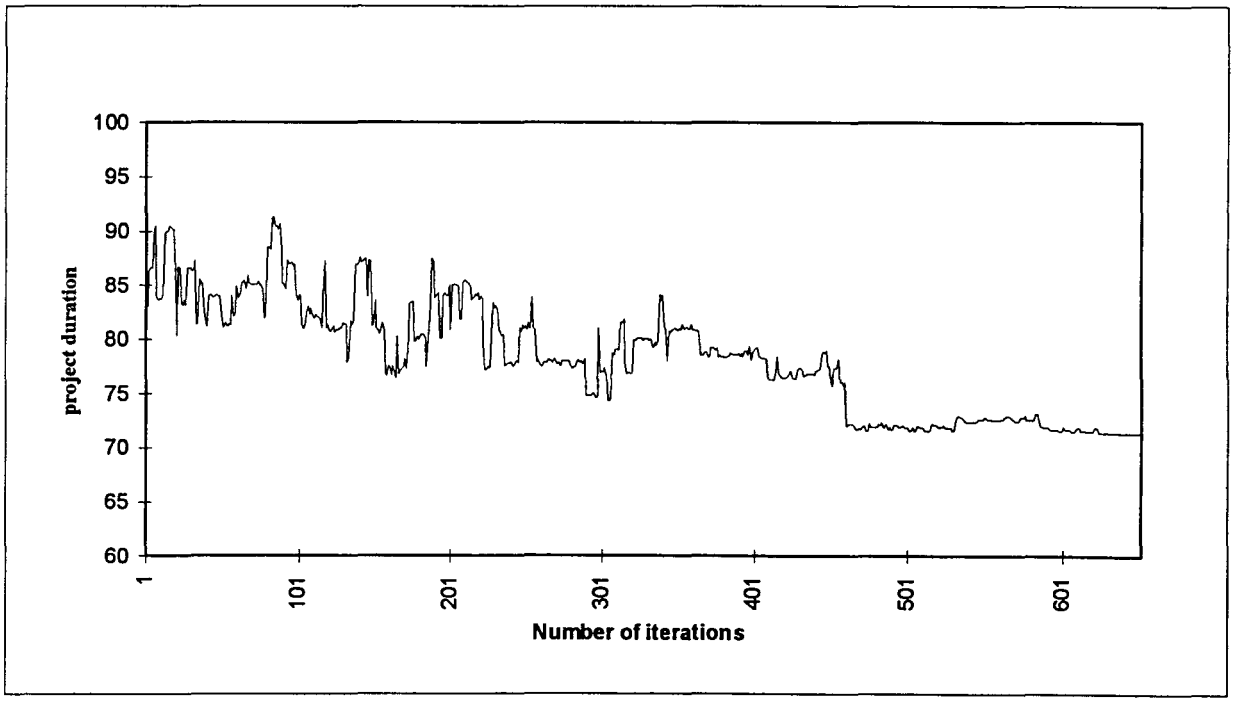

Figure 3-5. Duration of schedules generated during the SA process for a sample project with 25 activities.

known optimum. Only 48 optimal solutions out of the 108 projects are found with the combination of ACTIM, LFT, ROT, ACTRES, and MEF heuristics.

Experimental results of application of the SA algorithm will now be presented in two parts: the SA algorithm for deterministic activity duration projects and the SA algorithm for randomized activity duration projects. In both cases, SA is applied to each of the 110 test projects of Patterson.

For the SA algorithm for deterministic activity duration projects, initial temperature *T(0)* is set to be one-half of the project duration determined from the initial schedule S_c. Two methods are used to determine the initial schedule: MINSLK and CAF with $w = 0.5$. The

Table 3-11. Summary of results utilizing combination of heuristics.

	Combination of ACTIM[a], LFT[b] and ROT[c] heuristics	Combination of ACTIM, LFT, ROT and ACTRES[d] heuristics	Combination of ACTIM, LFT, ROT, ACTRES and MEF[e] heuristics
Average percentage increase above optimum	3.23%	2.91%	2.77%
Percentage of optimum obtained	41.67%	42.59%	44.44%
Number of projects in which optimum was found	45	46	48

[a] Priority given to the activity with the maximum ACTIM value. The ACTIM value of an activity is calculated as the maximum time that the activity controls the network on any one path.

[b] Priority given to the activity with the earliest PERT/CPM calculated late finish time.

[c] Priority given to the activity with the maximum ROT value. The ROT value is calculated by dividing the sum of each activity's resource requirements by the duration of the activity and then finding the maximum ROT that an activity controls through the network on any one path.

[d] Priority given to the activity with the maximum ACTRES value. The ACTRES value is calculated by multiplying each activity's time by the sum of its resource requirements and then finding the maximum ACTRES that an activity controls through the network on any one path.

[e] Priority given to the activity with the earliest PERT/CPM calculated early finish time.

temperature value $T(t)$ is decreased at a rate of 20%, i.e., $T(t) = 0.8T(t-1)$. Different values of $N(t)$ are used to evaluate the performance of the SA algorithm. Since the neighborhood schedule is generated randomly, the SA algorithm is repeated for 10 trials for each parameter setting. The results of the 10 trials with the initial feasible solution generated by the MINSLK

rule are summarized in Table 3-12. Table 3-13 shows the SA results when the initial schedule is generated using CAF.

The results in Table 3-13 show that the average percentage increase above the optimal solution is less than one percent in every case. The percent increase above the optimum decreases as the values of *N(T)* increase. In the best case, with *N(T)* = 10**N*, results of these experiments average only 0.197% above the optimal solution. The heuristics used by Morse, McIntosh and Whitehouse [7] find only 44.44% of the optimal solutions when using a combination of five heuristics while 77.00%, 83.64%, 86.18%,89.18%, and 91.91% of the optimal solution are found using the SA algorithm with an initial schedule given by MINSLK and with *N(t)* equal to N, 2N, 3N, 5N, and 10N respectively. 76.46%, 81.64%, 85.91%, 88.46%, and 93.27% of the optimal solutions are found using the SA algorithm with an initial schedule given by CAF. Naturally, execution times increase as *N(T)* is increased; however,

Table 3-12. Experimental results of 10 trials for deterministic activity duration projects using SA algorithm. Initial schedule is generated using the MINSLK rule.

	N(T)=1*N[a]	N(T)=2*N	N(T)=3*N	N(T)=5*N	N(T)=10*N
Average percentage of project duration increase above optimum	0.869%	0.550%	0.428%	0.325%	0.229%
Average percentage of optimum obtained	77.00%	83.64%	86.18%	89.18%	91.91%
Average execution time in seconds	0.3173	0.6276	0.9176	1.5130	2.9731
Average standard deviation of execution time in seconds	0.3806	0.7793	1.1376	1.8552	3.6544
Number of projects in which optimum solution is found in all 10 trials	47	56	62	73	84

[a] N is the number of nodes or activities.

Table 3-13. Experimental results of 10 trials for deterministic activity duration projects using SA algorithm. Initial schedule is generated using the CAF rule.

	N(T)=1*N	N(T)=2*N	N(T)=3*N	N(T)=5*N	N(T)=10*N
Average percentage of project duration increase above optimum	0.842%	0.630%	0.466%	0.353%	0.197%
Average percentage of optimum obtained	76.46%	81.64%	85.91%	88.46%	93.27%
Average execution time in seconds	0.3169	0.6108	0.9223	1.4827	2.9978
Average standard deviation of execution time in seconds	0.3737	0.7418	1.1275	1.8731	3.8133
Number of projects in which optimum solution is found in all 10 trials	47	53	64	73	80

the average execution time never exceeds 3 seconds. A similar performance results from the use of the SA algorithm regardless of whether the initial schedule is generated by the MINSLK heuristic or the CAF heuristic.

For the SA algorithm for randomized activity duration projects, the schedule which gives the lowest average project duration is reported as the best schedule found by the SA algorithm. Patterson's 110 test projects are again used to evaluate the performance of the SA algorithm. Instead of using the deterministic activity duration, three estimates are used in the same manner as discussed earlier, the optimistic time estimate a, the most likely time estimate m, and the pessimistic time estimate b. The expected activity duration is 1.05 times the deterministic activity duration.

When evaluating the results of the application of SA to these randomized problems, the optimal project duration is not known. Therefore, the known deterministic optimal solution is used as an approximate lower bound from which to compare the results using SA.

The results of application of SA to these random problems are shown in Table 3-14 and Table 3-15. Almost one-half of the 110 projects have their duration decreased over 10%, and some of projects have their duration decreased by over 20% from the initial solutions obtained by utilizing only the MINSLK or CAF heuristics. On average, the overall project duration is decreased by around 9% from the initial solutions.

Table 3-14. Experimental results of 10 trials for randomized activity duration projects using SA algorithm. Initial schedule is generated using the MINSLK rule.

	N(T)=1*N	N(T)=2*N
Average percentage of project duration increase above optimum[a]	3.40%	2.27%
Average percentage of project duration decrease from the initial schedule duration	8.64%	9.62%
Number of projects in which project duration decreases from the initial schedule duration over 10%	45	48
Number of projects in which project duration decreases from the initial schedule duration over 15%	22	27
Number of projects in which project duration decreases from the initial schedule duration over 20%	7	11
Average execution time in seconds	10.804[b]	21.414
Average standard deviation of execution time in seconds	9.868	19.708

[a] The optimum is the optimal project duration determined with the original deterministic activity durations times 1.05.

[b] The average execution time includes the time to generate randomized activity duration.

Table 3-15. Experimental results of 10 trials for randomized activity duration projects using SA algorithm. Initial schedule is generated using the CAF rule.

	N(T)=1*N	N(T)=2*N
Average percentage of project duration increase above optimum	3.36%	2.36%
Average percentage of project duration decrease from the initial schedule duration	8.55%	9.36%
Number of projects in which project duration decrease from the initial schedule duration over 10%	42	46
Number of projects in which project duration decrease from the initial schedule duration over 15%	18	20
Number of projects in which project duration decrease from the initial schedule duration over 20%	4	6
Average execution time in seconds	10.462	21.308
Average standard deviation of execution time in seconds	9.129	19.790

Conclusion

The goal of this paper was to find a simple algorithm which could be applied to the resource-constrained, randomized activity duration project scheduling problem. The SA algorithm proves to be an efficient way to find good solutions to both the deterministic and stochastic problems. For Patterson's 110 test projects, the average solutions provided by the SA algorithm were only 0.197 % and 0.229% above the optimum with average computation times of 2.998 and 2.973 seconds respectively for initial feasible solutions provided by MINSLK and CAF. This is a very short average computation time (about 3 seconds), and the final project durations are quite close to the optimum. Similar results are obtained when SA is

applied to the stochastic version of the problem. Experimental results show that the project duration of the 110 projects with random activity durations is decreased from the initial project duration by an average of 8.40% to 9.52%. While the computation time is much greater for the stochastic problems, the average time is still under 22 seconds. This amount of time is negligible when there is a potential to decrease overall project duration by almost 10% on average.

References

1. Badiru, A.B. 1991. A Simulation Approach to PERT Network Analysis. *Simulation*. 57, 245-255.

2. Badiru, A.B. and Pulat, P.S. 1995. *Comprehensive Project Management*. Englewood Cliffs, New Jersey: Prentice Hall.

3. Bowers, J.A. 1995. Criticality in Resource Constrained Networks. *Journal of the Operational Research Society*. 46, 80-91.

4. Davis, E.W. and Patterson, J.H. 1975. A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Management Science*. 21, 944-955.

5. Demeulemeester, E. and Herroelen, W. 1992. A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. *Management Science*. 38, 1803-1818.

6. Eglese, R.W. 1990. Simulated Annealing: A Tool for Operational Research. *European Journal of Operational Research*. 46, 271-281.

7. Morse, L.C., McIntosh J.O. and Whitehouse G.E. 1996. Using Combinations of Heuristics to Schedule Activities of Constrained Multiple Resource Projects. *Project Management Journal*. March, 34-40.

8. Ozdamar, L. and Ulusoy, G. 1995. A Survey on the Resource-Constrained Project Scheduling Problem. *IIE Transactions*. 27, 574-586.

9. Patterson, J. 1984. A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem. *Management Science*. 30, 854-867.

10. Pritsker, A.A.B., Watters, I.J. and Wolfe, P.M. 1969. Multiproject Scheduling With Limited Resources: a Zero-One Programming Approach. *Management Science*. 16, 93-108.

11. Watkins, Kevin. 1993. *Discrete Event Simulation in C*. London, England: McGraw-Hill.

CHAPTER 4. USING TABU SEARCH TO SCHEDULE ACTIVITIES OF STOCHASTIC RESOURCE-CONSTRAINED PROJECTS

A paper submitted to the European Journal of Operational Research

Ying-Wei Tsai and Douglas D. Gemmill

Abstract

In this paper, a higher level heuristic procedure "tabu search" is proposed to provide good solutions to resource-constrained, randomized activity duration project scheduling problems. Our adaptation of tabu search uses multiple tabu lists, randomized short-term memory , and multiple starting schedules as a means of search diversification. The proposed method proves to be an efficient way to find good solutions to both the deterministic and stochastic problems. For the deterministic problems, most of the optimal schedules of the test projects investigated are found. Computational results are presented which establish the superiority of tabu search over the existing heuristic algorithms.

Introduction

The critical path method (CPM) and the program evaluation and review technique (PERT) were developed to solve project scheduling problems with the assumption that resource availability is unlimited. However, resource availability is limited in most situations.

When the project is scheduled with a given set of resources, it is difficult to find the optimal solution. Resource constrained scheduling problems are generally NP-hard [4].

Ozdamar and Ulusoy [12] surveyed a wide variety of existing optimum-yielding techniques and scheduling heuristics for different types of resource constrained project scheduling problems. Because of the complexity involved in implementing the optimal techniques for large projects, the optimal techniques are not generally used in practice. Thus, many heuristics [1, 2, 10, 11, 13] have been developed to find acceptable solutions for resource constrained scheduling problems in a reasonable amount of computation time.

Badiru [1] and Badiru and Pulat [2] developed the Composite Allocation Factor (CAF) for each activity which is used to determine resource allocation. An activity that consumes more resources, lasts longer, and varies more in duration will have a larger CAF value. Activities with larger CAF values have higher priorities for resource allocation. Li and Willis [10] proposed an iterative scheduling technique. Under this technique, a project is scheduled forward and backward iteratively until there is no further improvement in the project completion time. Morse *et al.* [11] scheduled the project by combining existing heuristic rules. The test results indicated that the combined heuristics outperformed the individual heuristics. However, the solution quality obtained using heuristic algorithms can vary from optimal to poor. In this paper a heuristic procedure, tabu search, is applied to the resource constrained scheduling problem in an attempt to improve the solution quality over existing heuristic algorithms. The 110 projects collected by Patterson [13] which represent an accumulation of all multiple resource problems existing in the literature are used to measure the performance of the tabu search algorithm.

Tabu search [7, 8, 9] is a heuristic procedure for solving optimization problems. It provides a method which decreases the likelihood of becoming trapped at a local optimum solution. Tabu search employs a set of *moves* that transform one solution state into another and provides an *evaluation function* for measuring the attractiveness of these *moves*. The form of the tabu search procedure is highly flexible and often motivates the creation of new types of *moves* and *evaluation function* criteria for different problems.

Tabu search has obtained optimal or near optimal solutions to some types of problems, such as quadratic assignment problems [15, 16] and scheduling problems [6, 14]. Glover and McMillan [6] obtained near optimal solutions to employee scheduling problems whose integer programming formulations involved one to four million variables. This required 22-24 minutes on an IBM PC microcomputer. Skorin-Kapov's [16] quadratic assignment study used less CPU time than previously reported yielding the best known solutions for problems taken from the literature. Punnen and Aneja [14] proposed a tabu search which produced good quality solutions for the strongly NP-complete hard categorized assignment scheduling problems. Punnen and Aneja [15] presented a tabu search algorithm for resource constrained assignment problems which provided superior solutions than the existing algorithms.

In this paper tabu search is applied to the resource constrained scheduling problems for both deterministic and stochastic activity duration. The procedures are described in the following sections. The methodology section includes discussion of the implementation of tabu search and an example is given to illustrate the application of tabu search. Experimental results are then shown in the results section. Finally, a conclusion is given in the last section.

Methodology

Before presenting tabu search, a basic procedure for scheduling activities and identifying the critical path with respect to a specified priority ranking is introduced. In addition, we will briefly review two heuristic rules, the minimum slack first (MINSLK) and the CAF heuristics, which are employed to provide starting solutions for tabu search.

Before the scheduling process is started, the relative priority ranking of all activities is done. Usually, the priority ranking is calculated with some heuristic rules or formulae. Resources are allocated based on the priority ranking of activities. A feasible activity is an activity for which the required preceding activities have been completed. At each scheduling instant, only the feasible activities are considered for resource allocation. At time k, the feasible activity with the highest priority is scheduled if the available resources are sufficient. When the number of resources available is inadequate for the next feasible activity at time k, then time $k + 1$ is considered. Note that activity preemption and partial resource assignments are not allowed.

When resources are constrained, determination of the critical path becomes more difficult. Bowers [5] proposed the idea of using resource links for this purpose. These resource links are used to trace the use of resources, noting the significant resource dependencies that determine the project schedule and hence identify the critical path. We have added the use of additional links which are referred to as precedence links[17]. While computing the earliest starting time, the resource links and precedence links are noted. After the forward pass, the additional links are added to the project network. The latest start times

are then determined with the original links and the additional links using the same backward pass as CPM.

Two heuristics are employed to provide starting solutions for tabu search. The MINSLK rule is very simple. The resource unconstrained PERT/CPM slack time for each activity is calculated. Then when resources are assigned, the top priority is given to the activity with the smallest PERT/CPM calculated slack time. In the CAF rule proposed by Badiru [1] and Badiru and Pulat [2], higher priority is given to the activity which has a higher CAF value. For each activity i, the CAF is calculated as a weighted and scaled sum of two components, RAF (resource allocation factor) and SAF (stochastic activity duration factor).

Notation, Variables and Operations for Tabu Search

Let N be the number of activities in a project, $I = \{1, 2, ..., N\}$ be the set of all activities, and $\prod$ be the set of all permutations defined on I. Any permutation $\pi \in \prod$ is defined as an N vector $(\pi(1), \pi(2), \pi(N))$. When the order of activities in a permutation π is consistent with the precedence relationships in the project, the permutation π is called a feasible sequence. That means activities can be completed in the same order as the in the sequence. A permutation is not necessarily a feasible schedule. Let F be the set of all feasible sequences, then F is a subset of $\prod$. A project can be scheduled according to the order in a feasible sequence under the resource constraints and hence the project duration can be determined. The goal of tabu search is finding a feasible sequence in F which gives the optimal or near-optimal schedule.

We will adapt some definitions from references [7, 8, 9, 14]. A *starting solution* is a feasible sequence obtained from a heuristic approach and the starting solution will be improved by the tabu search algorithm. Since quality of a solution obtained by tabu search is considerably affected by the starting solution, we use the MINSLK rule and the CAF rule to determine different starting solutions.

A *move* is a transition from one feasible sequence to another by interchanging the positions of two activities i and j. In order to verify a permutation yielded by interchanging the positions of activities i and j is a feasible sequence, an N by N *precedence matrix* P is constructed. Elements of *precedence matrix* P are set as:

$P_{i,j} = 1$, if node j is a successor of node i.

$P_{i,j} = 0$, otherwise.

Assume $i < j$, the yielded permutation is a feasible sequence if the conditions

$$P_{k,j} = 0, \text{ for } k = i, i+1, \ldots, j-2, j-1$$

and

$$P_{i,k} = 0, \text{ for } k = i+1, i+2, \ldots, j-1, j$$

are true. Since an enormous number of *moves* has to be made during the tabu search process, the execution time of tabu search is significantly influenced by the time needed to validate that the interchange of two activities results in a feasible sequence. A *move* can be verified using the proposed method in $O(N)$ time. An *objective function* f is defined as the project duration of a feasible sequence. The *value of a move* is the difference between the *objective function* value of the feasible sequence yielded by making the interchange and the *objective function*

value of the current feasible sequence. If the *value of a move* is negative, then the *move* is called an *improvement move*.

In some applications, such as the applications in [14, 15], only one tabu list of recent *moves* is recorded. This tabu list is used to prevent a solution from being revisited for a certain number of iterations. In this study, two tabu lists are implemented due to the different characteristics of an activity. The activities of a project are divided into two categories: *critical activity* and *non-critical activity*. An activity is classified as a *critical activity* if it is on the critical path of the project scheduled using CPM/PERT. Otherwise it is a *non-critical activity*. A list named *TabuListC* consists of *critical activities* and another list named *TabuListNC* consists of *non-critical activities*.

When resource constraints are included, in general when a *critical activity* is delayed, the successors of the delayed activity will also be delayed. Consequently, the completion time of the resource constrained project will be delayed with a high probability. For the *critical activity*, once it is moved forward due to an interchange it can be completed earlier. It is then recorded in *TabuListC* and will remain there for *TabuTenureC* iterations. While a *critical activity* is in *TabuListC*, it is in *tabu status* and *TabuTenureC* is called the *tabu tenure* of a *critical activity* which is the duration the *critical activity* remains in *tabu status*. An N by 1 integer vector *TabuStatusC* is used to record the tabu status of the *critical activity*. The elements of *TabuStatusC* are the number of iterations that an activity has been in tabu status. Initial values for elements of *TabuStatusC* are zero. After each iteration, if there is a *critical activity*, say k, which became tabu, then the element $TabuStatusC_k$ is set to 1. All the elements of *TabuStatusC* which are not zero will be incremented by 1. When the value of an

element exceeds *TabuTenureC*, it is reset to zero. Thus, the tabu status can be checked in *O(1)* time. *TabuListNC*, an integer *TabuTenureNC*, and a vector *TabuStatusNC* are implemented in the same manner except that the *non-critical activity* is not moved forward in *TabuTenureNC* iterations.

An integer *NumOfMove* is the number of *candidate moves* which are generated and evaluated in each iteration. These *candidate moves* are recorded in a *candidate list*. Each *candidate move* in the candidate list is evaluated by an *evaluation function*. The *evaluation function* is a combination of *the value of a move* and a *penalty function*. A *penalty function* is applied when some predefined constraints are violated. Since only feasible sequences are considered in our scheduling application, no *penalty function* is needed. By giving a different definition of the *evaluation function*, tabu search can be used to find the solution for some other objective.

When a feasible sequence yielded by performing a *move* contains a tabu activity, the *move* is called a *tabu restricted move* and an *aspiration test* on this *move* has to be made. The *aspiration test* is an important element of flexibility in tabu search. The tabu status can be overruled if the *aspiration test* is satisfied otherwise the *move* is not permitted performing. The *aspiration test* used in this paper as follows: if the project duration of the new sequence is shorter than the best project duration found thus far, then a tabu status can be overruled. An *admissible move* is a *move* which is not a *tabu restricted move* or a *tabu restricted move* which satisfies the *aspiration test*. A *best move* is the *admissible move* that has the best value when evaluated by the *evaluation function*. The sequence yielded by making the *best move* serves as the starting solution of next iteration. The best sequence is not necessarily an

improvement over the present solution. This is the technique that tabu search employs to escape from a local minimum.

The performance of tabu search depends on several parameters. One of the most important parameters is the *tabu tenure*. The preferred choice of the value for tabu tenure is customarily based on an empirical test. A small tabu tenure will very likely introduce cycling into the search, and if the tabu tenure is too large the tabu search will be restricted to a small domain. Tabu tenure values for a scheduling problem typically can be expressed as a simple function of the number of activities, such as the square root of the number of activities. Another important parameter is the number of *candidate moves*, *NumOfMove*, considered in each iteration. Both solution speed and quality can be significantly influenced by the use of appropriate candidate list strategies.

Application of Tabu Search to the Deterministic Problem

Tabu search for a resource constrained, deterministic activity duration project scheduling problem is implemented as follows:

Step 1. Select a starting feasible sequence, denoted as S_i.

- The starting solution can be determined using a heuristic rule, such as the MINSLK rule.
- Let the best sequence found so far, denoted as S_b, be the feasible sequence S_i.

Step 2. Divide activities into two parts: *critical activity* and *non-critical activity*.

- Schedule the project, without considering the resource requirements, using CPM.
- Identify the *critical activities* and the *non-critical activities*.

Step 3. Initialize variables used in tabu search.

- Construct the *precedence matrix* for the resource constrained project.
- Let the lists *TabuListC* and *TabuListNC* be empty lists.
- Select values of *TabuTenureC* and *TabuTenureNC*.
- Let the elements of the vectors *TabuStatusC* and *TabuStatusNC* be zeros.
- Select value of *NumOfMove*, the number of *moves* of the *candidate list*.
- Select values of stopping criteria *MaxTryOnAdmissible* and *MaxTryOnBetter*.
- Reset value of *NotFindAdmissible*, the number of iterations in which no *admissible move* is found, and value of *NotFindBetter*, the number of iterations in which no feasible sequence better than S_b is found, to zero.

Step 4. Create a *candidate list* which consists of *NumOfMove* moves.

- Two activities are randomly selected and the positions of these two activities are interchanged.
- If the generated sequence is not feasible, select two other activities.
- Repeat the procedure until *NumOfMove* moves are found.

Step 5. Choose the best admissible candidate move.

- Let *FindAdmissible* and *FindBetter* be false.
- For each *candidate move* in the candidate list do
  - Let S_j be the yielded feasible sequence after making the *move* on S_i.
  - Evaluate *the value of a move*, $f(S_j) - f(S_i)$.

- If the *move* yields a better value than all other *moves* found admissible so far in the *candidate list* then
    - Check tabu status of the *move*. If the activity moved back is *a critical activity* and is in the list *TabuListC* or the activity moved forward is a *non-critical activity* and is in the list *TabuListNC* then the *move* is *tabu restricted.*
    - If this *move* is not *tabu restricted* then
        - Accept the *move* as best admissible candidate move, denoted as M_b.
        - Change the value of *FindAdmissible* to true.
        - Reset *NotFindAdmissible* to zero.
    - Else
        - The *move* passes the *aspiration test* if $f(S_i) < f(S_b)$.
        - If this *move* passes the *aspiration test* then
            - Accept the *move* as best admissible candidate move, denoted as M_b.
            - Change the value of *FindAdmissible* to true.
            - Reset *NotFindAdmissible* to zero.
        - Endif
    - Endif
- Endif

- Enddo

Step 6. Make the best admissible candidate move.

- Let S_j be the feasible sequence after making the *best move* M_b on S_i.
- If $f(S_j) < f(S_b)$ then S_b is replaced by S_j and change the value of *FindBetter* to true.
- If *FindAdmissible* is not true, then *NotFindAdmissible* := *NotFindAdmissible* + 1.
- If *FindBetter* is not true, then *NotFindBetter* := *NotFindBetter* + 1.

Step 7. Check stopping criteria.

- If NotFindAdmissible >= *MaxTryOnAdmissible* or NotFindBette >= *MaxTryOnBetter* then S_b is reported as the solution, else go to **Step 8**.

Step 8. Update tabu restrictions and aspiration criteria and then go to **Step 4**.

- Update the lists *TabuListC* and *TabuListNC*.
- Update the vectors *TabuStatusC* and *TabuStatusNC*.

The path that tabu search follows depends significantly on the starting solution. Therefore, the solution determined using tabu search also depends on the starting solution. In order to explore diversified paths so that a better solution may be found, one can repeat the procedure described above several times with different starting solutions.

A typical project from Patterson [13] is used to illustrated the solution found using tabu search. The project network and resource requirements are given in Figure 4-1 and Table 4-1. This project has three types of resources (type A, type B and type C). Assume the number of each resource available is 6 type A, 7 type B, and 6 type C resources.

The MINSLK rule is used to find a starting feasible sequence for tabu search. The project duration for this sequence, 23, is determined using the procedure presented in the beginning of this section. The results of application of tabu search to the sample project are shown in Table 4-2, where EST is the earliest start time, ECT is the earliest completion time, LST is the latest start time, LCT is the latest completion time, SLACK is the slack time of an activity, and CRITICAL is the indication of criticality. Table 4-2 shows the variables associated with the schedule determined using tabu search, and the solution has a project duration of 20 which is the optimal duration.

Application of Tabu Search to the Stochastic Problem

The existing optimal-yielding techniques, such as integer programming, are only able to solve deterministic scheduling problems. However, there is no way to know the exact activity duration before the activity is done in most situations. Using the expected activity

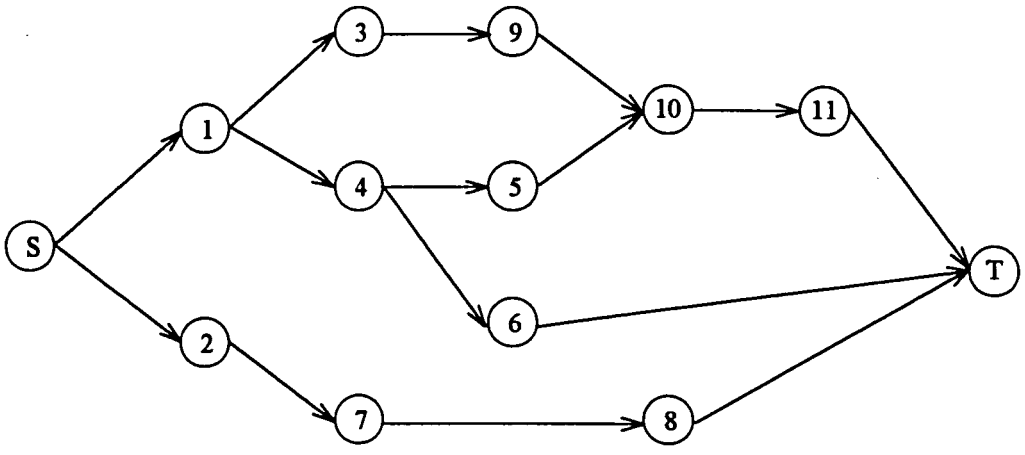

Figure 4-1. A typical project selected from the Patterson's 110 test projects.

Table 4-1. Duration, resource requirements and optimal schedule of the sample project.

Node (Activity)	Duration	Resource requirements	Start time in the optimal schedule
1	3	3,2,1*	0
2	5	2,4,2	0
3	6	3,1,2	5
4	2	4,3,1	3
5	3	2,0,3	9
6	3	1,1,1	12
7	4	3,1,1	5
8	5	2,2,2	12
9	4	3,2,3	11
10	2	4,1,0	15
11	3	5,4,2	17

* The resource requirements of type A, type B, and type C resources respectively.

Table 4-2. The schedule determined using tabu algorithm.

Node	EST	ECT	LST	LCT	SLACK	CRITICAL
1	0	3	0	3	0	1
2	0	5	0	5	0	1
3	5	11	5	11	0	1
4	3	5	3	5	0	1
5	9	12	9	12	0	1
6	12	15	12	15	0	1
7	5	9	5	9	0	1
8	12	17	12	17	0	1
9	11	15	11	15	0	1
10	15	17	15	17	0	1
11	17	20	17	20	0	1

Schedule order : 1-2-4-7-3-5-9-8-6-10-11.
The project duration is 20.

duration only and applying techniques for deterministic problems to schedule the project could ignore the effects of the random property of the project. With a modification on the *evaluation function*, the tabu search algorithm for a deterministic problem can be applied to a stochastic problem. Instead of computing the expected activity duration for scheduling the project, one can draw the activity duration from the historical data or a probability distribution. In this paper, we chose a beta distribution to model activity duration.

The three time estimates, an optimistic time estimate a, the most likely time estimate m, and a pessimistic time estimate b, used in PERT are used to calculated the parameters for the beta distribution. The formulae for parameters α and β of the beta distribution derived by Badiru [1] and Badiru and Pulat [2] are:

$$\phi = \frac{5a - 4m - b}{a + 4m - 5b}$$

$$\alpha = \phi\beta$$

$$\beta = \frac{-(\phi^2 - 34\phi + 1)}{(\phi + 1)^3}$$

Since the activity durations are randomly distributed, the overall project duration is also a random variable. Given a particular feasible sequence, the expected project duration is computed as follows:

1. One activity duration for each activity is drawn from the beta distribution with the parameters calculated based on the three time estimates associated with each activity.

2. The scheduling problem is then treated as a deterministic problem and the project duration is calculated.
3. The calculation of project duration is repeated a number of times with different sets of activity durations and then the average project duration is reported as the expected project duration.

As an example, tabu search is applied to a stochastic version of the sample project in Figure 4-1. The optimistic time estimate a is 0.8 times the deterministic activity duration, the most likely time estimate m is equal to the deterministic activity duration, and the pessimistic time estimate b is set to 1.5 times the deterministic activity duration. Using these values of a, m, and b gives a longer expected duration for each activity, equivalent to 1.05 times the deterministic activity duration. The MINSLK rule is used to find a starting feasible sequence for tabu search. The expected project duration of the starting feasible sequence is determined using the procedure presented at the beginning of this section. Using a sample size of 100, the average project duration is 24.346.

The results of application of tabu search to the sample projects are shown in Table 4-3. Table 4-3 shows the sample average of the variables associated with the schedule determined using tabu search, and the solution has an average project duration of 21.706. This is an improvement of about 10.8% over the schedule determined using the MINSLK rule. From this example and the experimental results shown in the next section, one can easily see that tabu search is capable of improving the project duration.

Table 4-3. The average duration of the sample project for the schedule determined using tabu search.

Node	Duration	EST	ECT	LST	LCT	SLACK	CRITICAL
1	3.170	0.000	3.170	0.154	3.324	0.154	0.63
2	5.199	0.000	5.199	0.355	5.554	0.355	0.37
3	6.273	5.270	11.544	5.854	12.127	0.584	0.36
4	2.100	3.170	5.270	3.324	5.424	0.154	0.63
5	3.180	5.529	8.708	5.705	8.884	0.176	0.64
6	3.157	17.621	20.778	18.578	21.736	0.957	0.10
7	4.283	8.708	12.991	8.884	13.167	0.176	0.64
8	5.288	13.032	18.320	13.229	18.517	0.197	0.66
9	4.193	11.544	15.736	12.132	16.325	0.588	0.34
10	2.121	15.736	17.857	16.325	18.446	0.588	0.34
11	3.150	18.557	21.706	18.586	21.736	0.029	0.90

Feasible sequence : 2, 1, 4, 3, 5, 7, 9, 8, 10, 6, 11.
The project duration is 21.706.

Result and Discussion

The tabu search algorithm was coded using Borland C++ Version 5.0 for a personal computer with Pentium-166 MHz CPU running under the Windows 95 operating system. All other applications are closed during tabu search program execution in order to measure the execution time of the tabu search algorithm. The 110 test projects constructed by Patterson [13] were used to evaluate the tabu search algorithm. A comparison of the results of the simulated annealing [17] and tabu search algorithms is given in this section. In addition, three projects, MPJ1, MPJ2, and MPJ3, adopted from an actual aircraft maintenance facility were scheduled using tabu search to illustrate its ability.

108 projects of the 110 test projects are solved in Morse, McIntosh and Whitehouse [11] utilizing the MINSLK rule. Of the 108 projects, the average percentage increase above the known optimum is 10.9% and the number of projects in which the optimal duration is found is 20 with the MINSLK rule. Morse, McIntosh and Whitehouse also use a combination of heuristics to search for good solutions, and a portion of their results are summarized in Table 4-4. The combination of heuristics produce results that are significantly better than those obtained by using a single heuristic. Using a combination of ACTIM, LFT, ROT, ACTRES, and MEF heuristics, 48 optimal solutions out of the 108 projects are found and on average the results are 2.77% above the known optimum. Using the combination of seven heuristic rules provide the best results in the paper of 2.68% above the known optimum.

Table 4-5 and Table 4-6 show the experimental results of 10 trials on each of the 110 projects using the simulated annealing (SA) algorithm [17]. N is the number of activities in a project and $N(T)$ is the number of schedules generated at each temperature in the annealing process. "Above optimum" is the average percentage increase in project duration above the optimal project duration provided by Patterson, "optimum obtained" is the percentage of optimal solutions found, "optimum found in all trials" is the number of projects whose optimal project duration is found in all trials, and "execution time" is the average execution time for scheduling all projects in all trials. When the initial schedule is generated using the MINSLK rule, the average percentage increase in project duration above the optimum is 0.229% with an average execution time 2.9731 seconds. The average percentage increase in project duration above the optimum is 0.197% with an average execution time of 2.9978 seconds when the initial schedule is generated using the CAF rule.

Tabu search is applied to each of the 110 test projects of Patterson. Experimental results of application of the tabu search algorithm will now be presented in two parts: the tabu search algorithm for deterministic activity duration projects and the tabu search algorithm for randomized activity duration projects.

Table 4-4. Summary of results utilizing combination of heuristics.

	Combination of ACTIM[a], LFT[b] and ROT[c] heuristics	Combination of ACTIM, LFT, ROT and ACTRES[d] heuristics	Combination of ACTIM, LFT, ROT, ACTRES and MEF[e] heuristics
Average percentage increase above optimum	3.23%	2.91%	2.77%
Percentage of optimum obtained	41.67%	42.59%	44.44%
Number of projects in which optimum was found	45	46	48

[a] Priority given to the activity with the maximum ACTIM value. The ACTIM value of an activity is calculated as the maximum time that the activity controls the network on any one path.

[b] Priority given to the activity with the earliest PERT/CPM calculated late finish time.

[c] Priority given to the activity with the maximum ROT value. The ROT value is calculated by dividing the sum of each activity's resource requirements by the duration of the activity and then finding the maximum ROT that an activity controls through the network on any one path.

[d] Priority given to the activity with the maximum ACTRES value. The ACTRES value is calculated by multiplying each activity's time by the sum of its resource requirements and then finding the maximum ACTRES that an activity controls through the network on any one path.

[e] Priority given to the activity with the earliest PERT/CPM calculated early finish time.

Table 4-5. Experimental results of Patterson's 110 projects for deterministic problem using SA algorithm. Initial schedule is generated using the MINSLK rule.

N(T)	N	2N	3N	5N	10N
Above optimum	0.87%	0.55%	0.43%	0.33%	0.23%
Optimum obtained	77.00%	83.64%	86.18%	89.18%	91.91%
Optimum found in all trials	47	56	62	73	84
Execution time (seconds)	0.317	0.628	0.918	1.513	2.973
Standard deviation of execution time (seconds)	0.381	0.779	1.138	1.855	3.654

Table 4-6. Experimental results of Patterson's 110 projects for deterministic problem using SA algorithm. Initial schedule is generated using the CAF rule.

N(T)	N	2N	3N	5N	10N
Above optimum	0.84%	0.63%	0.47%	0.35%	0.20%
Optimum obtained	76.46%	81.64%	85.91%	88.46%	93.27%
Optimum found in all trials	47	53	64	73	80
Execution time (seconds)	0.317	0.611	0.922	1.483	2.998
Standard deviation of execution time (seconds)	0.374	0.742	1.128	1.873	3.813

Values of variables used in the tabu search algorithm were chosen based on experimental results. Let N be the number of activities in a project. Then, the values of *TabuTenureC* and *TabuTenureNC* are $\sqrt{N}/2$. The number of moves in the candidate list, NumOfMove, is $\sqrt{N}$. Different values of stopping criteria, *MaxTryOnAdmissible* and *MaxTryOnBetter*, are used to evaluate the performance of tabu search algorithm. Since the neighborhood schedule is generated randomly, the tabu search algorithm was repeated for 10 trials for each parameter setting.

The results of the 10 trials on each of the 110 projects with the starting solution generated using the MINSLK rule are summarized in Table 4-7. Table 4-8 shows the tabu search results when the initial feasible sequence is generated using CAF. In Table 4-7 and Table 4-8, "project duration improved" is the average percentage improvement of project duration over the initial schedule provided by MINSLK or CAF. One can easily see the experimental results are better than those of Morse *et al.* [11]. Comparing the results given from Table 4-5 to Table 4-8, the simulated annealing algorithm is better than tabu search when the computation time is very short, say about 0.3 seconds on average. When the average computation time is greater than 0.3 seconds, tabu search seems to have a greater ability to find the optimal project duration. Over 93% of the optimal solutions are found in these trials and the optimal solutions of over 94 projects are found in all runs with an average execution time of 3.4 seconds for tabu search. This compares to over 91% of the optimal solutions found in these trials and the optimal solutions of over 80 projects found in all runs with an average execution time of 3.0 seconds for simulated annealing.

Table 4-7. Experimental results of Patterson's 110 projects for deterministic problem using tabu search. Initial schedule is generated using the MINSLK rule.

MaxTryOnAdmissible	1000	3000	6000	10000	20000
MaxTryOnBetter	100	300	600	1000	2000
Above optimum	1.40%	0.69%	0.40%	0.29%	0.19%
Project duration improved	9.29%	9.89%	10.13%	10.23%	10.32%
Optimum obtained	66.64%	81.36%	87.46%	90.64%	93.46%
Optimum found in all trials	43	62	76	87	95
Execution time (seconds)	0.283	0.687	1.173	1.853	3.396
Standard deviation of execution time (seconds)	0.353	0.813	1.303	2.074	3.694

Table 4-8. Experimental results of Patterson's 110 projects for deterministic problem using tabu search. Initial schedule is generated using the CAF rule.

MaxTryOnAdmissible	1000	3000	6000	10000	20000
MaxTryOnBetter	100	300	600	1000	2000
Above optimum	0.95%	0.52%	0.28%	0.26%	0.16%
Project duration improved	9.58%	9.96%	10.17%	10.19%	10.27%
Optimum obtained	71.36%	83.91%	90.00%	91.55%	95.36%
Optimum found in all trials	44	66	84	90	94
Execution time (seconds)	0.274	0.630	1.150	1.796	3.364
Standard deviation of execution time (seconds)	0.371	0.744	1.353	2.137	3.939

In the application of tabu search to randomized activity duration projects, the schedule which gives the lowest average project duration is reported as the best schedule found using tabu search. Instead of using the deterministic activity duration, three estimates are used in the same manner as discussed earlier, the optimistic time estimate a, the most likely time estimate m, and the pessimistic time estimate b. The expected activity duration is 1.05 times the deterministic activity duration.

When evaluating the results of the application of tabu search to these randomized problems, the optimal project duration is not known. Therefore, the known deterministic optimal solution times 1.05 is used as an approximate lower bound from which to compare the results using tabu search. Table 4-9 and Table 4-10 show the results using the simulated annealing algorithm and tabu search with different initial feasible schedules. In Table 4-9 and Table 4-10, "Above approximate lower bound" is the average percentage increase in project duration above the approximate lower bound, "project duration improved" is defined as before, "Improve over 10%" is the number of projects in which the value of "project duration improved" is over 10%, "Adm" is the value of MaxTryOnAdmissible, and "Better" is the value of MaxTryOnBetter.

From the experimental results, the execution time needed by tabu search is about half of the execution time needed by simulated annealing to find similar results. Almost one-half of the 110 projects have their duration decreased over 10%, and some of projects have their duration decreased by over 20% from the initial solutions obtained by utilizing only the MINSLK or CAF heuristics. On average, the overall project duration is decreased by around 9% from the initial solutions.

Table 4-9. Scheduling results of Patterson's 110 projects for randomized problem using SA and tabu search. Initial schedule is generated using the MINSLK rule.

	Simulated Annealing		Tabu Search	
	N(T)=N	N(T)=2N	Adm=25 Better=250	Adm=50 Better=500
Above approximate lower bound	3.40%	2.27%	3.71%	2.54%
Project duration improved	8.64%	9.62%	8.52%	9.53%
Improved over 10%	45	48	43	47
Improved over 15%	22	27	21	27
Improved over 20%	7	11	6	8
Execution time (second)	10.804	21.414	5.834	11.290
Standard deviation of execution time (second)	9.868	19.708	4.453	8.823

Experimental results of the three aircraft maintenance projects are given from Table 4-11 to Table 4-17. Project MPJ1 has 107 activities, project MPJ2 has 162 activities, and MPJ3 has 168 activities. MINSLK duration is the project duration determined using the MINSLK rule. CAF duration is the project duration determined using the CAF (w=0.5) rule. Improved duration is the project duration determined using the SA algorithm or tabu search. From Table 4-11 to Table 4-14 are the results of the deterministic problem and from Table 4-15 to Table 4-17 are the results of the randomized problem.

The results of the deterministic problem are discussed in this paragraph. An encouraging improvement, about a 20% or more reduction in the project duration determined

Table 4-10. Scheduling results of Patterson's 110 projects for randomized problem using SA and tabu search. Initial schedule is generated using the CAF rule.

	Simulated Annealing		Tabu Search	
	N(T)=N	N(T)=2N	Adm=25 Better=250	Adm=50 Better=500
Above approximate lower bound	3.36%	2.36%	3.03%	1.91%
Project duration improved	8.55%	9.36%	8.92%	9.85%
Improved over 10%	42	46	45	53
Improved over 15%	18	20	16	20
Improved over 20%	4	6	5	6
Execution time (second)	10.462	21.308	6.173	11.618
Standard deviation of execution time (second)	9.129	19.790	5.555	10.107

using the MINSLK rule or the CAF rule is achieved, and is accomplished in a short execution time. The average execution time for these three big projects is less than 23 seconds which is negligible when the project durations are improved about 20% or more. When the initial schedule is generated using the MINSLK rule, the SA algorithm and tabu search provided approximately the same improvement with the same execution time. The SA algorithm appears to perform somewhat better while the tabu search may be trapped at a local minimum when the initial sequence is determined using the CAF rule.

The need to calculate the average project duration when applying SA or tabu search to random activity duration projects results in long execution times. As we noted before, the

execution time needed by the SA algorithm is longer than the execution time needed by tabu search. From Table 4-15 to Table 4-17, one can see the significant difference in the execution time needed by the SA algorithm and tabu search. For example, when MINSLK is used to provide the initial sequence, the tabu search execution time is less than one quarter of the SA execution time while obtaining similar project duration results to those of SA. For the largest project, MPJ3, tabu search found an average project duration of 392.83 in an average of 224.03 seconds but SA found an average project duration of 403.60 in an average of 989.53 seconds. When CAF is used to provide the initial sequence, although the reduction in execution time is not as great as when using MINSLK, the tabu search execution time is still shorter than the SA execution time. Therefore, tabu search finds a similar or better schedule in a much shorter execution time than the SA algorithm does.

Table 4-11. Scheduling results of three projects for deterministic problem using SA algorithm. Initial schedule is generated using the MINSLK rule.

N(T)	N			2N		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	236	232	431	236	232	431
Improved duration	198.3	171.1	351.1	191.5	175.0	341.7
Project duration improved	15.97%	26.25%	18.54%	18.86%	24.57%	20.72%
Execution time (second)	2.56	8.87	6.82	5.56	13.56	12.11
Std dev of exec time	0.52	3.15	1.61	1.18	3.54	2.13

Table 4-12. Scheduling results of three projects for deterministic problem using SA algorithm. Initial schedule is generated using the CAF rule.

N(T)	N			2N		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	251	234	478	251	234	478
Improved duration	192.9	176.0	372.9	192.1	167.7	353.3
Project duration improved	23.15%	24.79%	21.99%	23.47%	28.33%	26.09%
Execution time (second)	3.08	8.26	5.89	5.21	16.84	11.57
Std dev of exec time	0.53	3.01	0.82	1.40	6.58	1.42

Table 4-13. Scheduling results of three projects for deterministic problem using tabu search. Initial schedule is generated using the MINSLK rule.

MaxTryOnAdmissible MaxTryOnBetter	100 1000			200 2000		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	236	232	431	236	232	431
Improved duration	193.3	175.0	350.3	191.2	173.3	338.2
Project duration improved	18.09%	24.57%	18.72%	18.98%	25.30%	21.53%
Execution time (second)	2.87	10.95	5.62	5.20	13.84	14.88
Std dev of exec time	0.86	3.53	2.57	2.12	4.02	4.85

Table 4-14. Scheduling results of three projects for deterministic problem using tabu search. Initial schedule is generated using the CAF rule.

MaxTryOnAdmissible MaxTryOnBetter	100 1000			200 2000		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	251	234	478	251	234	478
Improved duration	198.4	183.1	373.6	194.6	180.6	360.9
Project duration improved	20.96%	21.75%	21.84%	22.47%	22.82%	24.50%
Execution time (second)	3.86	22.69	10.91	7.14	22.51	18.95
Std dev of exec time	1.29	13.89	2.24	3.04	8.29	6.70

Table 4-15. Scheduling results of three projects for randomized problem using SA algorithm. Initial schedule is generated using the MINSLK and CAF rule.

N(T) = N	MINSLK			CAF		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	246.45	255.01	455.43	268.50	248.71	506.60
Improved duration	215.90	201.86	403.60	216.27	204.93	405.03
Project duration improved	12.40%	20.84%	11.38%	19.45%	17.60%	20.05%
Execution time (second)	395.62	740.69	989.53	390.45	706.21	1051.0
Std dev of exec time	30.99	59.62	81.32	36.56	49.06	72.35

Table 4-16. Scheduling results of three projects for randomized problem using tabu search. Initial schedule is generated using the MINSLK rule.

MaxTryOnAdmissible MaxTryOnBetter	25 250			100 1000		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	246.59	253.85	456.28	246.76	254.54	456.51
Improved duration	214.25	203.03	392.83	209.33	194.51	377.31
Project duration improved	13.12%	20.02%	13.91%	15.17%	23.58%	17.35%
Execution time (second)	64.14	134.60	224.03	235.50	406.57	640.99
Std dev of exec time	28.41	40.86	71.23	62.71	124.73	195.54

Table 4-17. Scheduling results of three projects for randomized problem using tabu search. Initial schedule is generated using the CAF rule.

MaxTryOnAdmissible MaxTryOnBetter	25 250			100 1000		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	267.72	249.58	506.61	268.97	249.92	507.74
Improved duration	224.05	223.73	424.84	216.70	205.70	404.46
Project duration improved	16.31%	10.36%	16.14%	19.43%	17.69%	20.34%
Execution time (second)	108.73	117.41	220.46	319.96	556.82	768.31
Std dev of exec time	36.53	33.94	79.06	94.59	208.61	278.25

Conclusion

The goal of this paper was to find an algorithm which can be used to improve project schedules initially determined using heuristic rules for the resource-constrained project scheduling problem. Based on the experimental results, tabu search is an efficient way to find good solutions to both the deterministic and stochastic problems.

For Patterson's 110 test projects, the average solutions provided by the tabu search are only 0.19 % and 0.16% above the optimum with average computation times of 3.396 and 3.364 seconds respectively for initial feasible solutions provided by MINSLK and CAF. This is a very short average computation time (about 3.5 seconds), and the final project durations are near to the optimal. Over 93.46% of the optimal schedules were found in these trials and the optimal project durations of over 94 projects were found in all 10 trials. From the results given in Table 4-5 to Table 4-8, the performance of the simulated annealing algorithm and tabu search on scheduling the deterministic problem were quite similar. Similar scheduling results using tabu search and the simulated annealing algorithm were obtained when both methods were applied to the stochastic version of the problem except that the execution times spent on tabu search were approximately half of the execution time spent on simulated annealing. Experimental results show that the project duration of the 110 projects with random activity durations was decreased from the initial project duration by an average of 8.52% to 9.58%. While the computation time was much greater for the stochastic problems, the average time is still under 11 seconds. This amount of time is negligible when there is a potential to decrease overall project duration by almost 10% on average.

For the three aircraft maintenance projects with deterministic activity duration, the schedules provided by the simulated annealing are better than the schedules provided by tabu search if the allowed execution time is similar. Tabu search performed better than SA for the randomized problem. When the initial feasible solution was determined using the MINSLK rule, tabu search provided a better schedule in a shorter execution time.

References

1. Badiru, A.B., "A Simulation Approach to PERT Network Analysis", *Simulation* 57 (1991) 245-255.

2. Badiru, A.B. and Pulat, P.S., *Comprehensive Project Management*, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

3. Bate N.G., "Network Criticality with Multiple Work-Patterns", *Journal of the Operational Research Society* 45 (1994) 927-933.

4. Blazewicz, J., Lenstra, J. K., and Rinnooy Kan, A.H.G, "Scheduling Subject to Resource Constraints: Classification and Complexity", *Discrete Applied Mathematics* 5 (1983) 11-24.

5. Bowers, J.A., "Criticality in Resource Constrained Networks", *Journal of the Operational Research Society* 46 (1995) 80-91.

6. Glover, F. and McMillan, C, "The General Employee Scheduling Problem: An Integration of Management Science and Artificial Intelligence", *Computer and Operations Research* 13 (1986) 563-593.

7. Glover, F., "Tabu Search, Part I", *ORSA Journal of Computing* 1 (1989) 190-206.

8. Glover, F., "Tabu Search, Part II", *ORSA Journal of Computing* 2 (1990) 4-32.

9. Glover, F., "Tabu Search: A Tutorial", *Interfaces* 20 (1999) 74-94.

10. Li, K.Y. and Willis, R.J., "An Iterative Scheduling Technique for Resource Constrained Project Scheduling", *European Journal of Operational Research* 56 (1992) 370-379.

11. Morse, L.C., McIntosh J.O., and Whitehouse G.E., "Using Combinations of Heuristics to Schedule Activities of Constrained Multiple Resource Projects", *Project Management Journal* March (1996) 34-40.

12. Ozdamar, L. and Ulusoy, G., "A Survey on the Resource-Constrained Project Scheduling Problem", *IIE Transactions* 27 (1995) 574-586.

13. Patterson, J., "A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem", *Management Science* 30 (1984) 854-867.

14. Punnen, A.P. and Aneja, Y.P., "Categorized Assignment Scheduling: A Tabu Search Approach", *Journal of the Operational Research Society* 44 (1993) 673-679.

15. Punnen, A.P. and Aneja, Y.P., "A Tabu Search Algorithm for the resource Constrained Assignment Problem", *Journal of the Operational Research Society* 46 (1995) 214-220.

16. Skorin-Kapov, F., "Tabu Search Applied to Quadratic Assignment Problem", *ORSA Journal of Computing* 2 (1990) 33-45.

17. Tsai, Y. and Gemmill D.D., "Identifying the Critical Path in Resource Constrained Projects", Working Paper No. 96-131, Department of Industrial and Manufacturing Systems Engineering, Iowa State University, IA, 1996.

18. Watkins, Kevin, *Discrete Event Simulation in C.* McGRAW-HILL Book Company, Europe, 1993.

19. Woodworth B.M and Shanahan S., "Identifying the critical sequence in a resource constrained project", *Project Management* (1988) 89-96.

CHAPTER 5. GENERAL CONCLUSION

The goals of the studies presented in the three papers were to identify the critical path and provide a good schedule for resource-constrained project scheduling problems with deterministic or stochastic activity duration. The proposed methods were implemented using Borland C++ Version 5.0 for a personal computer with Pentium-166 MHz CPU running under the Windows 95 operating system. Patterson's 110 projects were used to evaluate the performance of simulated annealing and tabu search. Experimental results indicate that the goals are successfully achieved. Both simulated annealing and tabu search provide better schedules than the schedules of Morse *et al.* [10].

In order to identify the critical path, the earliest start time of each activity is determined using a conventional heuristic rule for scheduling in the resource-constrained scheduling problem. During the forward calculation process, resource links and precedence links are noted. After the forward process is finished, the noted resource links and precedence links are added to the project network. The latest start time of each activity can then be determined using exactly the same process as the backward process of CPM/PERT. The critical path is determined by comparing the earliest start time and the latest start time of the activities. In this way we were able to identify the critical paths for each of Patterson's 110 test projects.

In the second paper, simulated annealing is applied to both the deterministic and stochastic scheduling problem. Using the simulated annealing algorithm, an initial schedule which is determined using a heuristic rule can be improved in a short time. The MINSLK rule

and the CAF rule were used to determine the initial schedule. Different cooling processes were used in this paper. Experimental results show that a longer cooling process will provide a better schedule.

The average percentage increase above the optimal solution is less than one percent in every case. The percent increase above the optimum decreases as the values of $N(T)$ increase. In the best case, with $N(T) = 10*N$, results of these experiments average only 0.197% above the optimal solution. 91.91% and 93.27% of the optimal solutions are found using the SA algorithm with an initial schedule given by MINSLK and CAF respectively with $N(t)$ equal to $10*N$. Application of SA to these random problems resulted in a decrease in project duration of over 10% in almost one-half of the 110 projects, and some projects had their duration decreased by over 20% from the initial solutions obtained by utilizing only the MINSLK or CAF heuristics. On average, the overall project duration is decreased by around 9% from the initial solutions.

Tabu search applied to the resource-constrained project scheduling problem is presented in the third paper. In the proposed tabu search, activities are classified in two categories, critical activities and non-critical activities. One tabu list was implemented for each category of activity. Once a critical activity is moved forward, that means it could be completed earlier, the activity will not be moved back for a certain number of iterations. In a similar manner, a non-critical activity will not be moved forward for a certain number of iterations.

For Patterson's 110 test projects with deterministic activity durations, the average solutions provided by the tabu search are only 0.19 % and 0.16% above the optimum with

average computation times of 3.396 and 3.364 seconds respectively when initial feasible solutions were provided by MINSLK and CAF. Over 93.46% of the optimal solutions were found in these trials, and the optimal project durations for over 94 projects were found in all 10 trials. The project duration for the 110 projects with random activity durations were decreased from the initial project duration by an average of 8.52% to 9.58%. For the three aircraft maintenance projects, project durations were improved about 20% or more for the deterministic activity durations. Even when keeping the execution time to a reasonable length, the project durations can be improved about 15% to 20% for randomized activity duration projects.

The performance of the schedules determined using the simulated annealing and tabu search are similar in both deterministic and stochastic problems. For the deterministic problems, both proposed methods require similar execution times. However, tabu search requires significantly shorter execution time to find a good schedule for the stochastic problem.

In the simulated annealing and the tabu search algorithm proposed in this thesis, there are some assumptions on the activities and resources. Activity preemption and partial resource allocations are not allowed. Also, there is no "look ahead" ability while allocating resources. That is, resources are allocated to activities solely based on their priorities. Further research could involve the relaxation of these assumptions.

REFERENCES

1. Badiru, A.B. (1991). A Simulation Approach to PERT Network Analysis. *Simulation*, **57**, 245-255.

2. Badiru, A.B. and Pulat, P.S. (1995). *Comprehensive Project Management*. Englewood Cliffs, New Jersey: Prentice Hall.

3. Blazewicz, J., Lenstra, J. K., and Rinnooy Kan, A.H.G. (1983). Scheduling Subject to Resource Constraints: Classification and Complexity. *Discrete Applied Mathematics*, **5**, 11-24.

4. Bowers, J.A. (1995). Criticality in Resource Constrained Networks. *Journal of the Operational Research Society*, **46**, 80-91.

5. Davis, E.W. and Patterson, J.H. 1975. A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Management Science*, **21**, 944-955.

6. Eglese, R.W. 1990. Simulated Annealing: A Tool for Operational Research. *European Journal of Operational Research*, **46**, 271-281.

7. Glover, F. (1989). Tabu Search, Part I. *ORSA Journal of Computing*, **1**,190-206.

8. Glover, F. (1990). Tabu Search, Part II. *ORSA Journal of Computing*, **2**,4-32.

9. Glover, F. (1990). Tabu Search: A Tutorial. *Interfaces*, **20**, 74-94.

10. Morse, L.C., McIntosh J.O., and Whitehouse G.E. (1996). Using Combinations of Heuristics to Schedule Activities of Constrained Multiple Resource Projects. *Project Management Journal*, March, 34-40.

11. Ozdamar, L. and Ulusoy, G. 1995. A Survey on the Resource-Constrained Project Scheduling Problem. *IIE Transactions*, **27**, 574-586.

12. Patterson, J. (1984). A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem. *Management Science*, **30**, 854-867.

13. Woodworth B.M and Shanahan S (1988). Identifying the critical sequence in a resource constrained project. *Project Management*, **6**, 89-96.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation and thanks to my major professor, Dr. Douglas Gemmill, for his advice, guidance, and patience throughout all phases of my graduate study and the preparation of this research.

I am grateful to Dr. Thomas Barta and Dr. James Rowings for their comments and insightful suggestions regarding my research as well as their participation on my graduate committee. I want to thank Mr. Jim Patterson for providing the 110 test projects.

Finally, I would like to express my warmest thanks to my family for their many years of support, love, and tremendous encouragement. In particular, I would like to thank my Dad, Sheng-Jr Tsai, and my Mom, Show-Nyu Wang.