Tabu Search and an Exact Algorithm for the Solutions of Resource-Constrained Project Scheduling Problems

Omer Atli*
Aeronautics and Space Technologies Institute, Turkish Air Force Academy
Yesilyurt, Bakirkoy, Istanbul 34149, Turkey

Received: 19-05-2010
Accepted: 15-10-2010

Abstract

When the project is scheduled with a given set of resources, it is difficult to find the optimal solution. Resource-constrained scheduling problems (RCPSP) are generally NP-hard. In this paper, a high level heuristic procedure "Tabu Search Algorithm (TSA)" is proposed to provide good solutions to resource-constrained, deterministic activity duration project scheduling problems. We present the application results of the computational tabu search and OPL-CPLEX algorithm and compare them with that of earlier applicable researches along with a discussion about further research. Our computational results are presented, which establish the superiority of tabu search over the existing heuristic algorithms. Two different solution strategies are also discussed, namely tabu search and OPL-CPLEX exact algorithm approach which can be used with the proposed model. Due to the execution time, we have shown that OPL-CPLEX's algorithm is a valid method with medium scale RCPSPs. For the considered deterministic problems, a good agreement has been obtained between theoretical and experimental results.

***Keywords*:** Project scheduling, Resource-Constrained, Tabu Search, OPL-CPLEX.

1. Introduction

Project scheduling has long been an area of intensive research, but until recently most efforts have been dedicated to developing solution procedures and generalizing model, while little work was done on generating problem instances.

Informally, a resource-constrained project scheduling problem (RCPSP) considers resources of limited availability and activities of known durations and resource requests, linked by precedence relations. The problem consists of finding a schedule of minimal duration by assigning a start time to each activity such that precedence relations and source availabilities are respected.

In this paper a mathematical model to solve Resource-Constrained Project Scheduling Problems (RCPSP), is developed. When the project is scheduled with a given set of resources, it is difficult to find the optimal solution. Resource-constrained scheduling problems are NP-hard. A higher level heuristic procedure "tabu search" is proposed to obtained good solutions to resource-constrained, deterministic project scheduling problems with deterministic activity duration. For the deterministic problems, most of the optimal schedules for the considered test problems have been obtained. Computational results which indicate the superiority of the results obtained by our approach are presented. Our objective is to minimize total project completion time subject to resource constraints in the project. Since traditional optimization techniques can't

*Corresponding author: atliomer2001@yahoo.com Tel: +90212 6632490.

cope with the RCPS problems, we present a new approach based on the tabu search algorithm (TSA). Tabu search algorithm is a metaheuristic procedure for solving combinatorial optimization problems. The approach was tested on a set of Patterson's 110 standard problems and compared with other approaches. The computational results validate the effectiveness of the proposed algorithm.

In project scheduling, one of the most common concerns is to attempt in order to complete a project as early as possible under conditions of limited resource availabilities. Hence, this is a typical NP-hard problem and linear programming and other exact approaches may be computationally impracticable when the project size increases. In practice, a project often contains hundreds of activities. Metaheuristic approaches are improved approaches that master strategies for controlling and guiding searches in order to find near optimal solutions globally.

In this study, a tabu search algorithm (TSA) and exact algorithm with OPL-CPLEX are developed. Both the Tabu search algorithm and the exact algorithm and their solutions have been integrated into a computer-aided system that has been written by Microsoft visual C++ 6.0 and OPL-CPLEX. The flowchart of the proposed tabu search algorithm and some parts of the algorithms are given in Section 4.3.

The heuristic methods for solving RCPS problems can be generally divided into serial and parallel. The serial approach derives its name from the fact that activities are considered sequentially. The parallel approach considers in one moment all activities which could be scheduled in that moment. In both approaches, the order in which the activities are considered for scheduling is defined by a priority list. The main difference between them consists in the way of solving the resource conflicts. If an activity can not be scheduled in moment *t* because of resource constraints, the serial procedure looks for the earliest moment in which all the required resources are available. In the same situation, the parallel procedure takes another activity ready for scheduling, with the next highest priority and tries to schedule it in moment *t*. Because of their better performance, our interest is focused on both procedures.

In this paper a heuristic procedure, tabu search algorithm is applied to the resource-constrained project scheduling problem in an attempt to improve the solution quality over existing heuristic algorithms. The 110 projects collected by Patterson (http://129.187.106.231/psplib/) which represent an accumulation of all multiple resource problems existing in the literature are used to measure the performance of the tabu search algorithm.

This paper deals with the resource-constrained project scheduling problems (RCPSP), where the activities of a project have to be scheduled with the objective of minimizing the makespan subject to resource constraints. Being one of the most intractable problems in the operations research area, RCPSP has often been a target and test bed for establishing new optimization tools and techniques. In order to efficiently solve this computationally complex problem in real time, we propose a parallel intelligent search technique named the tabu search heuristic.

Our adaptation of tabu search uses one tabu list, 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 deterministic problems.

Traditional project management techniques such as the critical path method (CPM) and the program evaluation and review technique (PERT) etc. 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 under the constrained resources, it is difficult to find the optimal solution.

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 some 3 heuristic rules, the minimum slack first, ROT and ACTIM heuristics, which are employed to provide starting solutions for tabu search algorithm.

We have used some priority rules like ROT, ACTIM, and Minslack 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 priority rules value. Activities with larger priority values have higher priorities for resource allocation. These priority techniques treated an iteratively scheduling technique. Under this technique, a project is scheduled forward and backward iteratively until there is no further

improvement in the project completion time. And in this study, we 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 remainder of the paper is organized as follows. The first section provides a short introduction to the study. In the second section RCPSP is described. The third section presents a wide literature summary and gap analysis of the resource constrained project scheduling. An illustrative example is given in order to show the application of the proposed models and algorithms and the foundations of the proposed solution approach are provided in the fourth section. Finally, in the fifth section, we present the computational tabu search and OPL-CPLEX algorithm's results; compare them with that of earlier applicable research and concluding remarks are made, along with a discussion about further research.

2. Resource-Constrained Project Scheduling Problems

A combinatorial optimization problem is defined by a solution space *X*, which is discrete or which can be reduced to a discrete set, and by a subset of feasible solutions $Y \subseteq X$ associated with an objective function $f : Y \to R$. A combinatorial optimization problem aims at finding a feasible solution $y \in Y$ such that $f(y)$ is minimized or maximized. A resource-constrained project scheduling problem is a combinatorial optimization problem defined by a tuple (V, p, E, R, B, b). Activities constituting the project are identified by a set $V = \{A_0,...,A_{n+1}\}$.

Activity A_0 represents by convention the start of the schedule and activity A_{n+1} symmetrically represents the end of the schedule. The set of non-dummy activities is identified by $A = \{A_1,..., A_n\}$. Durations are represented by a vector p in N^{n+2} where p_i is the duration of activity A_i, with special values $p_0 = p_{n+1} = 0$.

Precedence relations are given by *E*, a set of pairs such that $(A_i, A_j) \in E$ means that activity A_i precedence activity A_j. A precedence activity-on-node graph *G (V, E)* is defined where nodes correspond to activities and arcs correspond to precedence relations. In here, we will identify in the rest of the paper each activity with the corresponding node of the precedence graph. We assume that *G* contains no cycle; other wise the precedence relations are obviously inconsistent. Since precedence is a transitive binary relation, the existence of a path in *G* from node A_i to node A_j also means that activity A_i precedence activity A_j. Thus, all precedence graphs having the same transitive closure define the same precedence constraints. We assume that, taking account of the preceding remark, *E* is such that A_0 is a predecessor of all other activities and A_{n+1} is a successor of all other activities. Renewable resources are formalized by set $R = \{R_1,..., R_k\}$.

Availabilities of resources are represented by a vector *B* in N^q such that B_k denotes the availability of R_k. In particular, are source R_k such that R_k =1 is called a unary or disjunctive resource. Otherwise, as a resource may process several activities at a time, it is called a cumulative resource.

Demands of activities for resources are abstracted by *b*, *a* $(n+2)\times q$ integer matrix, such that b_{ik} represents the amount of resource R_k used per time period during the execution of A_i.

A schedule is a point *S* in R^{n+2} such that S_i represents the start time of activity A_i. C_i denotes the completion time of activity A_i, with $C_i = S_i + p_i$. S_0 is a reference point for the start of the project. Here we assume that S_0 = 0. A solution *S* is feasible if it is compatible with the precedence constraints (1) and there source constraints (2) expressed below, where;
$A_t = \{A_i \in A | S_i \le t \le S_i + p_i\}$ represents the set of non-dummy activities in process at time *t*.

$$S_j - S_i \ge p_i \qquad \forall (A_i, A_j) \in E \quad (1)$$

$$\sum_{A_j \in A_t} b_{ik} \le B_k \qquad \forall R_k \in R,\ \forall t \ge 0 \quad (2)$$

The makespan of a schedule *S* is equal to S_{n+1}, the start time of the end activity. The above-defined set A_t and constraints state that an activity can not be interrupted once it is started. This is referred to as not allowing preemption. The RCPSP can then be stated as follows:

Definition 1. *The RCPSP is the problem of finding a non-preemptive schedule S of minimal makespan* S_{n+1} *subject to precedence constraints (1) and resource constraints (2).*

An important preliminary remark is that, since durations are integers, we can restrict ourselves to integer schedules without missing the optimal solution. A non-integer feasible schedule can be transformed into an integer feasible schedule without increasing the makespan by recursively applying the following principle. Consider a non-integer schedule *S* and let A_i denote the first activity in the increasing start time order such that $S_i \notin N$. Then setting S_i to its nearest lower integer $\lfloor S_i \rfloor$ does not violate any precedence constraints, since the completion time of the predecessors of A_i are integers strictly lower than S_i. Left shifting an activity can violate a resource constraint only if it enters new sets A_t for $\lfloor S_i \rfloor \le t \le S_i$. Since we have $A_t \subseteq A_{S_i} \setminus \{i\}$ for $\lfloor S_i \rfloor \le t \le S_i$, no resource-constraint violation can appear by setting S_i to $\lfloor S_i \rfloor$. The set of integer schedules, containing at least one optimal solution, is said to be dominant.

Definition 2. *The resource-constrained project scheduling problem (RCPSP) refers to scheduling the activities of a project under precedence and resource constraints with the objective of minimizing total project throughput duration (makespan).*

The precedence constraints usually form the temporal constraints, whereas the renewable resources (e.g., manpower, material, and machines) which are necessary to carry out the project activities constitute resource constraints. Owing to the immense practical applications of the problem in various industries and organizations, it has attracted the significant attention of researchers in the last three decades. This can be treated as a generalized case of many scheduling problems, such as job-shop, flow-shop, and assembly line balancing. So RCPSP's are also special kind of job-shop and flow-shop scheduling problems [see. 26-27]. RCPSP has always been treated as a challenging combinatorial optimization problem.

3. Literature review

In this section, we aim at providing a snapshot of what was previously done for each optimization and/or heuristic approach regarding problem characteristics and researchers**.** Table 1. summarizes previous research on various resource-constrained projects scheduling with respect to author's contributions to the problem space.

In the literature many researchers 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.

Table 1. Summarizes previous research on various RCPSP

References	Problem	Problem Characteristics (PC)*								
		PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**	**PC9**
(21)	RCPSP	A		C		A	1	C	3	3
(4)	RCPSP	A		A	A	A		A	2	
(5)	MMRCPSP	A	C	A		A		B	2	
(22)	RCPSP	C			2	A				
(13)	RCPSP	A	A	A		A			2	
(26)	RCPSP	C		A+C		A	1	C	3	3
(11)	RCPSP		B	A		A+B			3	3
(18), (19)	RCPSP	A	B			A			3	3
(9)	RCPSP	C		C		A	1	C		
(27)	RCPSP	A		C		A	1	C	3	3
(25)	RCPSP	A		C		A	1	B	2	2
(7)	RCPSP	A		A	1	A		C	3	3
(15)	MMRCPSP	A		B		A		B	2	2
(2)	RCPSP	A	B			A		B	2	2
(10)	RCPSP	A		A		A			2	
Present Study	RCPSP	A	C	A+C		A	1	C	3	3

*

PC1: Problem Structure a. Deterministic b. Stochastic c. Both

PC2: Optimization Methods a. Dynamic Algorithm b. Integer Programming c. Exact Algorithms

PC3. Heuristic Methods a. Priority rules b. Evaluation Techniques (Genetic Algorithm) c. Tabu search Algorithm d. Simulating Annealing

PC4. Others Methods 1. Comsoal Method 2. TOC – Theory of Constraints

PC5. Objective a. Minimum Makespan b. Maximizing the net present value of the project cash flow

PC6. Strategic Oscillation 1. Randomized short term memory

PC7. Search diversification a. Low b. Mid. c. High

PC8. Quality of the solutions 1. Bad 2. Good 3. Best

PC9. Execution time 1. Low 2. Mid 3. High

So, many heuristics have been developed to find acceptable solutions for RCPSP in a reasonable amount of computation time.

The test results indicate that combined heuristics outperformed individual heuristics. However, the solution quality obtained using heuristic algorithms can vary from optimal to poor.

The heuristics-based approaches are the most studied approaches due to the flexibility. Execution time is the most considered statistical property in many articles. The exact algorithm approaches are the least considered approaches. But in this work, we can get a good solutions with OPL-CPLEX exact algorithm. This issue will be further explained in the conclusion section.

4. Problem description and mathematical formulation model

The RCPSP can be stated as follows. A project consists of (j + 2) activities where each activity has to be processed in order to complete the project. The (j + 2) activities compose of a set of activities j = {0, 1, 2… j+1}, where activities 0 and (j +1) are dummy and have no duration and no resources. These activities represent the initial and final activities. There exist a set of renewable resources which are represented by r = {1, 2… k}. The problem will be described using Fig.1. as an illustrative example.

4.1. *Standard resource-constrained project scheduling problem*

A typical project from Patterson (PAT3.RCP) is used to illustrate the solution found by using the tabu search. The project network and resource requirements are given in Figure 1 and Table 2. This project has three types of resources (type1, type 2 and type 3). Assume the amounts of resources available are 6 units of type 1, 7 units of type 2, and 6 units of type 3 resources.

In Table 2, a RCPSP example is given with n =11 real activities and R =3 resources with availabilities $\Re_1^{\rho} = 6$, $\Re_2^{\rho} = 7$, $\Re_3^{\rho} = 6$.

Table 2. A project with 11 real activities and 3 resources.

A_i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
p_i	3	5	6	2	3	3	4	5	4	2	3
r_{j1}	3	2	3	4	2	1	3	2	3	4	5
r_{j2}	2	4	1	3	0	1	1	2	2	1	4
r_{j3}	1	2	2	1	3	1	1	2	3	0	2

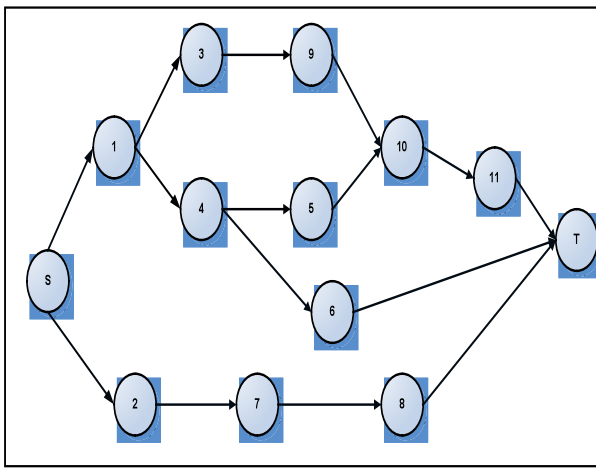

Figure 1. Precedence activity-on-node graph

The precedence constraints linking the activities $A_i \in V$ are displayed in Fig.1. as an activity-on-node graph. A schedule of minimal makespan S_{n+1}^{*} = 20 is displayed in Fig.2. as a 2-dimensional Gantt chart where the x axis represents the time and the y axis represents the resource occupation for all resource types.

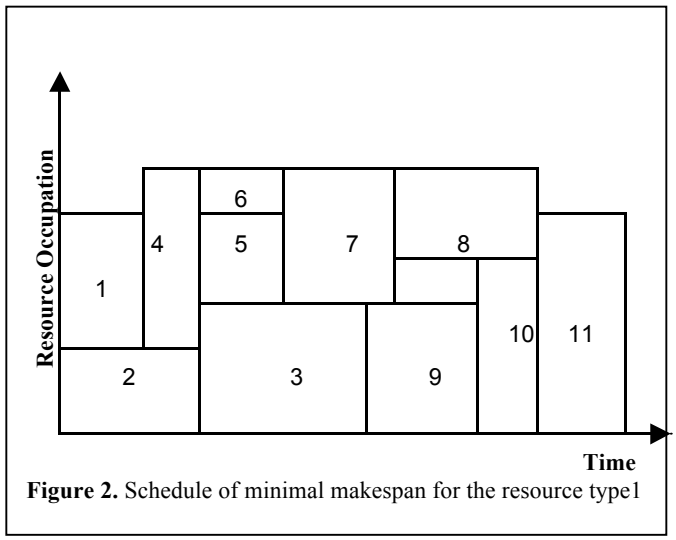

Figure 2. Schedule of minimal makespan for the resource type1

ROT, ACTIM and minslack rules are used to find a starting feasible sequence for the tabu search. The project duration for this sequence, 20 units, is determined using the procedure presented in the beginning of this section. The results of application of the tabu search to the illustrative project are the same as the priority rules' results and shown in Fig.2. for the resource type 1, 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; and Slack is the slack time of an activity. The solution has optimal project duration of 20 units.

The RCPSP can be defined by the following assumptions:

1) The processing times of activities are certain.
2) The start time of each activity is dependent upon the completion of some other activities.
3) The multiple resources are available in limited quantities but renewable from period to period.
4) Activities can not be interrupted.
5) The main objective is to minimize the total project time and makespan.

4.2. *Mathematical Formulation of RCPSP Model*

In RCPSP, the activities are interrelated by two kinds of constraints. First, the precedence constraints which force an activity not to be started, before all its predecessors have been finished. Second, processing of the activities is subject to the availability of resources with limited capacities. While being processed, the activity j in project requires $\Re_k^\rho$ units of resource $k \in$ K during every period of its processing time p_j. Resource type k has a limited availability r_{jk} at any point in time. Parameters p_j, r_{jk} and $\Re_k^\rho$ are assumed to be non-negative, and all are integer. For the start and end activities of a project, we have

$$\left.\begin{aligned} p_0 &= p_{n+1} = 0 \\ r_{0,k} &= r_{n+1,k} = 0 \end{aligned}\right\} \quad for\ all\ j \in \mathrm{I}.$$

where i stands for node; j stands for activity index; and r stands for resource index;

p_j: The processing time of activity j in project;

r_{jk}: The scheduling activity j in project consuming resource units per period from resource k;

$\Re_k^\rho$: The maximum limited resource k only available with constant period availability;

S: the set of activities being in the progress in period p.

In this formulation, the use of a dummy start as well as a dummy end activity is assumed. The decision variables denote the finish times of the different activities, while p_j denotes the duration of activity j, $\Re_k^\rho$ is the availability of the k^{th} resource type and r_{jk}^ρ is the resource requirement of activity j for resource type k. The set S_t denotes the set of activities that are in progress at time t.

The mathematical models for the RCPSP can be stated as follows:

$$\text{Min} \sum_{t=1}^{H} t x_{n+1,t} \qquad (3)$$

Subject to

$$\sum_{t=1}^{H} t x_{jt} + p_i - \sum_{t=1}^{H} t x_{it} \le 0 \qquad (4)$$

$$\sum_{j=1}^{n} \left(r_{jk}^{\rho} \sum_{u=t}^{t+p_i-1} x_{ju} \right) \le \Re_k^\rho \qquad (5)$$

$$\sum_{t=1}^{H} x_{jt} = 1 \quad \text{for j=1,...,n} \qquad (6)$$

$$H = \sum_{j=1}^{v} p_j \qquad (7)$$

In formulation (3)-(7), x_{jt} are 0-1 variables, which specify whether activity j finishes at time t or not. More specifically, for each activity j and for every feasible completion time $t \in [EFT_j, LFT_j]$, x_{jt} is defined as follows:

$$x_{jt} = \begin{cases} 1, & If \text{ activity } j \text{ finishes at time instant } t \\ 0, & otherwise \end{cases} \qquad (8)$$

The variable x_{jt} can only be defined over the interval between the earliest and latest finishing time of the activity under consideration. These limits are determined using the traditional forward and backward pass calculations. The backward pass recursion is started from a project length of T, which equals a feasible project length (e.g., determined by any of the heuristic procedures).

It should be clear to the reader that there is an immediate link between the binary decision variables and the completion times of the activities. The finishing time of activity j can be calculated by Eq.(8). The objective function given by Eq.(3) seeks to minimize the total project time that is the sum of the completion time for all activities. It minimizes the completion time of the dummy end activity and thus the completion time of the project. Eq.(6) specifies that only one completion time is allowed for every activity, which must lie in the interval between the earliest and latest finishing time. The precedence constraints are given by Eq. (4). The completion time of any activity cannot exceed the start time (completion time minus duration) of its successor.

The resource constraints for every resource type k are specified in Eq.(5) by considering for every time instant t and every resource type k, all possible completion times for all activities j such that the activity is in progress in period t. The weighted sum of the corresponding decision variables should not exceed the resource availability.

The constraints in Eq.(3) are referred to limit the resource demand imposed by the activities being processed at time p to the available capacity. The constraints in Eq.(4) are used to impose the precedence relations between activities.

4.3. *Proposed Tabu Search Algorithm*

Tabu search algorithms (TSA) have been broadly used in many areas such as planning, scheduling and other optimization problems where conventional mathematical techniques are inadequate. TSA was first developed by F.Glover in 1975, based on Artificial intelligence systems. Each search iteration stands for a possible solution to a problem. After several iterations, the algorithm that hopefully represents the optimal or near optimal solution to a problem can be found. TSA belongs to the class of heuristic optimization techniques, and it is very useful when a large search space with knowledge of how to solve the problem is presented. A balance between exploitation and exploration in the search space is one of the important factors when using TSA. To provide this balance, determination of the design strategy for TSA parameters like maximum iteration is one of the critical issues. In this paper, we present a new approach based on the improved tabu search algorithm for solving RCPSP. In the following, we give the fundamentals of tabu search.

The tabu list size is a parameter that is preset or determined experimentally. Our experiments have demonstrained that the better size of tabu list should be around activity size by $\sqrt{N}$.

Before the scheduling process is started, the relative priority ranking of all activities is done. The priority ranking is usually calculated with some heuristic rules or formula. 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 t, the feasible activity with the highest priority is scheduled if the available resources are sufficient. If the number of resources available is inadequate for the next feasible activity at time t, then time t+1 is considered. Note that activity preemption and partial resources assignments are not allowed.

When resources are constrained, determination of the critical path becomes more difficult. Bowers proposed the idea of using resource links for this purpose. These resource links are used to trace the use of resources, nothing 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. While computing the earliest starting time, the resource links and precedence links are noted. After the forward pass, the additional links are added to project network. The latest start times are then determined with the original links and the additional links using the same backward pass as utilized by CPM.

Three unique heuristics are employed to provide starting solutions for tabu search. The Minslack rule is very simple. The resource-unconstrained CPM/PERT slack time for each activity is calculated. Then when resources are assigned, top priority is given to the activity with the smallest CPM/PERT calculated slack time. In other heuristic rules, higher priority is given to the activity which has a higher priority value. For each activity i, the priority rule is calculated as a weighted and scaled sum of two components, resource and time allocation factors. In the following, we apply the proposed model using Patterson's 110 problem.

Let n be the number of activities in a project, I= {0,1,2,3…n} be the set of all activities; P be the set of all permutations defined on i^{th} any permutation $p \in P$ is defined as an n vector (p (1), p (2), p (3)….p (n)). When the order of activities in a permutation p is consistent with the precedence relationships in the project, the permutation p is called a feasible sequence. That means activities can be completed in the same order as in the sequence. Let F be the set of all feasible sequences, then F is a subset of P. A project can be scheduled under the resource constraints and hence the project duration can be determined. The goal of tabu search is to find a feasible sequence in F which gives the optimal or near-optimal schedule.

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

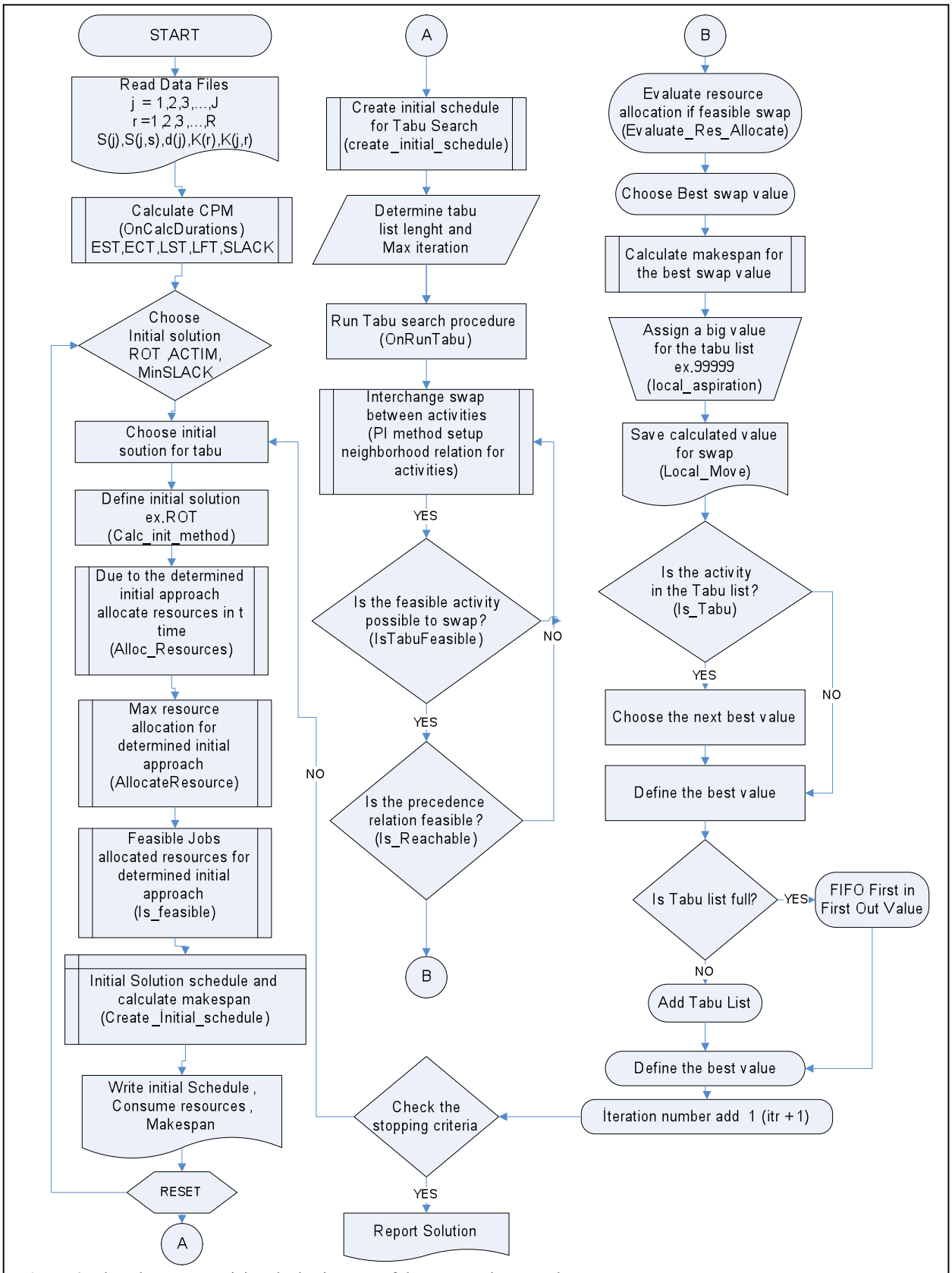

Figure 3. Flowchart summarizing the basic steps of the proposed approach

of a solution obtained by tabu search is considerably affected by the starting solution, we use the Minslack rule and other priority rules to try different starting solutions.

A move (swap) is a transition from one feasible sequence to another by interchanging the positions of two activities *i* and *j*. 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.

Tabu list is used to prevent a solution from being revisited for a certain number of iterations. The activities of a project are divided into two categories: critical and non-critical activities. 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.

In Fig.3. a flowchart summarizing the basic steps of the proposed approach is presented. The flowchart of tabu search algorithm is illustrated in Figure 3 as follows:

Application of tabu search to deterministic activity duration project scheduling problem is implemented.

Step 1. Read data file.

Step 2. Use CPM and calculate the project parameters (OnCalcDurations)

- Schedule the project without considering the resource requirements using CPM.
- Calculate EST, LST, ECT, LCT and Minslack (CalcES,CalcEF,CalcLF,CalcLS)
- Find makespan with unlimited resources (ptz)
- Identify the critical and non critical activities.(CalccriticalPath)
- Identify the resources loading level

Step 3. Choose initial solution method (Calc_init_method)

- Select a starting feasible sequence, denoted as *S* with priority rules.
- The starting solution can be determined using a heuristic rule, such as ROT, ACTIM, MINSLACK rules. (calcRot,CalcActim,CalcMinslack)
- Choose one of the priority rules (select Calc_init_method)
- Allocate the resources in time *t* for the selected initial solution method (Alloc_Resources)
- Do maximum resource allocation for determined initial approach (AllocateResources)
- Determine if Jobs are feasible when resources are allocated to the jobs for the determined initial approach (Is_feasible)
- Let the best sequence be S_b so far, using the initial feasible sequence S_i. (Create_Initial_Schedule_Order)

Step 4. Initialize variables used in tabu search.

- Create initial schedule for Tabu Search (create_initial_schedule)
- Construct the precedence matrix for the resource-constrained project.
- Let the Tabu list be empty lists. (CurrentTabuListLength = 0)
- Select values for tabulist (MaxTabuListLength = m_tabu_length)
- Let the elements of vectors tabustatus be zeros.
- Select a value for numofswapmove, the number of moves in the candidate list.
- Select values for stopping criteria maxiterations. (MAX_ITER = m_num_max_iteration)
- Reset the values of notfindadmissible, the number of iterations in which no admissible move, and notfindbetter, the number of iterations in which a feasible sequence better than S_b is found, to zero.

Step 5. Create a candidate list which consists of numofswap moves. (IsTabufeasible)

- 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. (Is_Reachable)

Step 6. Choose the best admissible candidate move. (Local_move) (Evaluate_Res_Allocate)

- Let findadmissible and findbetter be invalid (False)
- Do each candidate move in the candidate list
- Let s_j be a feasible sequence after making the move on s_i
- Evaluate the value of the 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 using Local_AspirationFound.
- Check tabu status of the move. If the activity is a critical activity in the backward calculation, then put that activity into tabulistC
- If the activity moved forward is a non-critical activity and is in tabulistNC then the move is tabu restricted.
- If this move is not tabu restricted then accept the move as the best admissible candidate move, denoted as M_b
- Change the value of findadmissible to a valid one(True)
- Reset notfindadmissible to zero
- Else
- The move passes the aspiration test if $f(s_j) < f(s_b)$
- If the move passes the aspiration test then
- Accept the move as bestadmissible
- Change the value of findadmissible to a valid one (True)
- Reset not findadmissible to zero.
- End if
- End do

Step 7. Make the best admissible candidate move.

- Let s_j be 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 find admissible is not true, then notfindadmissible = notfindadmissible + 1
- If findbetter is not true, then notfindbetter = notfindbetter + 1

Step 8. Check stopping criteria.

- If iter < MAX_ITER then s_b is reported as the solution, else go to Step 9.

Step 9. Update tabu restrictions and aspiration criteria and go to Step 4.

- Update tabu list tabulistC and tabulistNC.
- Update the vectors tabustatusC and tabustatusNC.

The solution determined using the 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.

5. Results and discussion

The tabu search algorithm was coded using Microsoft Visual C++ 6.0 for personal computer running under the XP 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 were used to evaluate the tabu search algorithm. In addition, a comparison of tabu search is made with the results of OPL-CPLEX exact algorithm applied to the same problems. In order to generalize the results obtained by the proposed methodologies, several examples from the published literature are studied regarding these dependencies. The performance of the assay was based upon four characteristics: Percentage of that research optimal solution, average difference from optimal, maximum differences from optimal and average execution time.

Table 3 shows the results of using the best combination of heuristics (BEST2, BEST3, and BEST4). BEST2 uses ACTIM and LFT; BEST3 uses ACTIM, LFT, and ROT; and BEST4 uses ACTIM, LFT, ROT, and ACTRES (DePuy et al., 2000).

The ACTIM value of an activity is calculated as the maximum time of the activity which controls the network on any path. Priority is given to the activity with the maximum ACTIM value. The LFT value gives priority to the activity with the earliest late finish time. The ROT value gives priority to the activity with the maximum ROT value. The ROT value is calculated by dividing the sum of each activity's resources requirements by the duration of the activity. Then, the maximum ROT that an activity controls through the network on any path is found. The ACTRESS value gives priority to the activity with the maximum ACTRESS value. The ACTRESS value is calculated by multiplying each activity's time by the sum of its resource requirements. Then the maximum ACTRESS

that an activity controls through the network on any path is found. The Minslack value gives priority to activity with the earliest CPM/PERT calculated early finish time.

Table 3. The results of using heuristics and exact methods for the Patterson's 110 problems

Using Heuristics and exact methods	Percentage of runs that is reached to optimal solution	Average difference from optimal %	Maximum difference from optimal %	Average Execution time
ACTIM(1)	30.9	5.0	23.7	Undefined
LFT(2)	26.4	6.1	23.3	Undefined
ROT(3)	10.0	11.4	36.4	Undefined
ACTRES(4)	27.3	5.2	25.0	Undefined
BEST2(1+2)	38.2	3.6	19.5	Undefined
BEST3(1+2+3)	40.9	3.1	15.8	Undefined
BEST4(1+2+3+4)	41.8	2.8	13.9	Undefined
COMSOAL 20 iterations both directions	42.18	2.34	11.3	Undefined
COMSOAL 50iterations forward direction	52.18	1.87	11.3	Undefined
Minslack	28	5.53	Undefined	Undefined
PSTSM	51	2.30	Undefined	Undefined
ROT	10	11.4	Undefined	5.07ms
ACTIM	31	4.53	Undefined	4.84ms
MINSLACK	17.3	11.6	Undefined	5.46ms
Tabu_ROT	39	3.54	Undefined	415ms
Tabu_ACTIM	61	1.38	Undefined	380ms
Tabu_minslack	45.5	2.97	Undefined	413ms
OPL-CPLEX	99.09	0	0	0,42s

110 test projects were solved in DePuy et al.(2000) utilizing the comsoal rule. Of the 110 projects, the average percentage increase above the known optimum was 52.18 % and the number of projects in which the optimal duration is found was 57 with the Comsoal rule. DePuy et al.(2000) also used a combination of heuristics to search for good solutions, and a portion of their results are summarized in table 3. The combination of heuristics procedure results that are significantly better than those obtained by using a single heuristic. Using a combination of ACTIM, LFT, ROT, ACTRESS and Minslack heuristics, 42 optimal solutions out of the 110 projects were found and on average the results were 2.8% above the known optimum. Using comsoal rules provides the best results in the paper of 1.87% above the known optimum.

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.

Tabu search was applied to each of the 110 test projects of Patterson. Experimental results of the tabu search algorithm for deterministic activity duration projects were presented.

Values of variables used in the tabu search algorithm were chosen based on experimental results. Then, the number of moves in the candidate list, numofmove, is sqrt (N), where N is the number of activities. Different values of stopping criteria, maxtbonadmissible and maxtbonbetter, were used to evaluate the performance of the tabu search algorithm. Since the neighborhood schedule is generated randomly, the tabu search algorithm was repeated iteratively for each parameter setting.

The results of trials for each of the 110 projects with starting solution generated using the ROT, ACTIM, Minslack rules and combination of them are summarized in Table 3. The improvement in project duration is the average percentage improvement of project duration over the initial schedule provided by Minslack. One can easily see that the experimental results are better than those of DePuy et al(2000). Comparing the results given in Table 3, the tabu search algorithm is better than Comsoal approach when the computation time is very short, say about 0.3 s on average. When the average computation time is greater than 0.3s, tabu search seems to have a greater ability to find the optimal project duration. Over 61% of the optimal solutions were found in these trials and the optimal solutions of 67 projects were found in all runs with an average execution time of 380 ms for tabu search. Over 99.09 % of the optimal solutions were found in these trials and the optimal solutions of 109 projects were found in all runs with an average execution time of 0.42 s for OPL-CPLEX exact algorithm.

6. Conclusion

The aim of this paper was to find an algorithm which can be 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 deterministic problems.

For Patterson's 110 test projects, the average solutions provided by the tabu search are only 1.38% above the optimum with average computation times of 380ms, for initial feasible solutions provided by Tabu_ACTIM and others. This is a very short average computation time, and final project durations are near to optimal. Summation and Tabu_ACTIM achieved statistically better results using modified tabu's combination of randomness and priority schemes than using either randomness exclusively (i.e. unmodified ACTIM) or only the priority scheme. Over 61 % of the optimal schedules were found in these trials and optimal project durations of over 67 projects were found in all trials for tabu. Over 99.09 % of the optimal solutions were found in these trials and the optimal solutions of 109 projects were found in all runs with an average execution time of 0.42 s for OPL-CPLEX exact algorithm. The performance of OPL-CPLEX exact algorithm on scheduling the deterministic problem was better than the comsoal and tabu search algorithm.

7. References

1. P. Brucker, A. Drexl, R. Mohring, K. Neumann and E. Pesch, Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. *European Journal of Operational Research*, **112** (1999) 3-41.
2. P. Brucker and S. Knust, A Linear Programming and Constraint Propagation-based Lower Bound For The RCPS. *European Journal of Operational Research*, **127** (2000) 355-362.
3. CPLEX Optimization. Using the CPLEX Callable Library, (CPLEX Optimization, 1995)
4. G.W. DePuy and G.E. Whitehouse, Applying the COMSOAL Computer Heuristic to the Constrained Resource Allocation Problem. *Computers and Industrial Engineering*, **38** (2000) 413-422.
5. A. Drexl, R. Nissen, J. Patterson and F. Salewski, PROGEN/X – An Instance Generator For Resources-Constrained Project Scheduling Problems With Partially Renewable Resources And Further Extensions. *European Journal of Operational Research*, **125** (2000) p. 59-72.
6. E.A. Elsayed, Algorithm for Project Scheduling with Resource Constraints. *International Journal of Production Research*, **20** (1982) 95-103
7. W.D. Gail and E. Whitehouse, A Simple and Efective Heuristic For The Resources Constrained Project Scheduling Problem. *International Journal of Production Research*, **39** (14) (2001)p. 3275-3287.
8. F.Glover, Tabu Search-Part I, ORSA. *Journal on Computing*, **1** (3) (1989) 190-206.
9. F. Glover, Tabu Search-Part II, ORSA. *Journal on Computing,* **2** (1) (1990) 10-32.
10. S. Hartmann and R. Kolisch, Experimental Evaluation of State-of-The-Art Heuristics for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research*, **127** (2000) 394-407.
11. O. Icmeli and W. Rom, 1995. Solving The Resource Constrained Project Scheduling With Optimization Subroutine Library. *International Journal of Production Research*, **23** (8) (1995) 801-817
12. R. Kolisch and A. Sprecher, PSPLIB A project Scheduling Problem Library. *European Journal of Operational Research*, **96** (1996) 205-216.
13. H. Khamooshi, 1999. Dynamic Priority-Dynamic Programing Scheduling Method (DP)2SM : a Dynamic Approach to Resources Constraint Project Scheduling. *International Journal of Production Management*, **17** (6) (1999) 383-391.
14. R. Klein, 2000. Project Sheduling with Time-varying Resource Contraints, *International Journal of Production Research*, **38** (16) (2000) 3937-3952.
15. M. Mori and C.C. Tseng, A Genetic Algorithm For Multi-Mode Resource-Constained Project Scheduling Problem. *European Journal of Operational Research*, **100** (1997) 134-141.
16. L.C. Morse and G.E. Whitehouse, Using Combinations of Heuristics to Schedule Activities of Constrained Multiple Resources Projects, *Project Management Journal*, (1996) 34-40
17. J.H. Patterson, Comparison of Exact Approaches for Solving the Multiple Constrained Resource Project Scheduling Problem. *Management Science*, **30**, (1984) 854-867.
18. J.H. Patterson, A Comparison of Exact Approaches For Solving The Multiple Constrained Resource, Project Scheduling Problem. *Management Science*, **30** (7) (1984) 854-867.
19. M. Pinedo, Scheduling Theory, Algorithms and Sysytems, (*Prentice- Hall International*, USA, 1995).
20. P.A. Punnen and Y.P. Aneja, A Tabu Search Algorithm for The Resource-Contrained Assigment Problem. *Journal of Operational Research Society*, **46** (1995) 214-220.
21. G.K. Rand, Critical Chain: The Theory of Constraints Applied to Project Management.

International Journal of Production Management, **18** (2000) 173-177

22. P.R. Thomas and S. Salhi, A Tabu Search Approach For The Resource Constrained Project Scheduling Problem. *Journal of Heuristics*, **4** (1998) 123-139.
23. Y. Tsai and D. Gemmill, Using Tabu Search To Scheduling Activities Of Stocastic Resource-Contrained Projects. *European Journal of Operational Research*, **111** (1998) 129-141.
24. M.G.A. Verhoeven, Tabu Search For Resource-Constrained Scheduling. *European Journal of Operational Research*, **106** (1998) 266-276.
25. D. White and J. Fortune, Current Practice in Project Management- An Empirical Study. *International Journal of Project Management,* **20** (2002) 1-11.
26. C. Kahraman, O. Engin, I. Kaya, M. Yilmaz, An Application of Effective Genetic Algorithms for Solving Hybrid Flow Shop Scheduling Problems. *International Journal of Computational Intelligence Systems*, **1** (2008) 134-147.
27. C. Kahraman, O. Engin, M. Yilmaz, A New Artificial Immune System Algorithm for Multiobjective Fuzzy Flow Shop Problems. *International Journal of Computational Intelligence Systems*, 2 (2009) 236-247.