ONDERZOEKSRAPPORT NR 8908

A DECISION SUPPORT SYSTEM FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING

BY

E. DEMEULEMEESTER & W. HERROELEN

D/1989/2376/11

A DECISION SUPPORT SYSTEM FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING

Erik DEMEULEMEESTER
Willy HERROELEN

Department of Applied Economic Sciences
Katholieke Universiteit Leuven
Dekenstraat 2, B-3000 Leuven, Belgium

A DECISION SUPPORT SYSTEM FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING

ABSTRACT

In this paper a decision support system is described for scheduling a project to minimize its total duration subject to technological precedence constraints and resource constraints. The system is centered upon a branch-and-bound procedure designed to yield optimal solutions to problems with up to ten different resource types. The software is programmed in C for personal computers running under the DOS operating system. It is user friendly and menu driven. Provisions exist in the software to obtain colour based Gantt charts and resource profiles. Maximum running times can be specified and the search process can be interrupted as soon as a user-specified project duration is obtained. Project data may be changed on screen, enabling the user to explore the impact of the various problem parameters on solution quality.

1. INTRODUCTION

The planning and control of large projects is a difficult and important problem of modern enterprise that many network planning techniques have tried to handle. Practical application of these techniques leads, however, to many difficulties. During the planning phase of a project, project management must solve a lot of technical problems as well as those involving time, cost and resource aspects.

Common network planning techniques, such as **PERT** (Program Evaluation and Review Technique) and **CPM** (Critical Path Method), essentially concern themselves with the time aspect only. These methods aim to minimize project duration, assuming that the various resources required for project completion are available. In practice, however, project completion requires the use of various resources, whose often limited availability directly influences planning objectives, time estimations, scheduling and progress control. When activities require resources for their execution (e.g. manpower, materials, equipment, capital, etc.) that are only available in a limited amount, bottlenecks may appear because activities cannot be started on time due to the unavailability of resources or activities requiring the same resource which is only available one unit at a time must be delayed, etc.

The various resource scheduling problems that may appear during project scheduling have been divided into three classes: time/cost trade-off, resource leveling and resource-constrained project scheduling (Herroelen 1972, Elmaghraby 1977, Moder et al. 1983). **Time/cost trade-off problems** may appear when there are no constraints imposed

on the availability of resources. The problem then consists of reducing the project completion time by adding additional resources to certain activities, so that the execution of these activities may be accelerated. When this is the case, there are many different ways in which activity durations may be selected so that project completion times of the resulting schedules are all equal. However, each schedule may yield a different value of total project direct cost. It would therefore be desirable to have some method for determining the least costly schedule for any given project duration. Several such optimal and suboptimal methods have been developed, each of which hinge upon various assumptions about the form of the activity direct cost-duration relationship. A recent comparison of commercially available project planning software packages revealed, however, that virtually none of the procedures made their way to software implementation (De Wit and Herroelen 1988). The same report also reaches the conclusion that none of the project planning software packages have features that would assist project management during the **project bidding stage**, where a contractor has to decide for example on his bid in terms of realization time and associated costs for critical network events. This interesting capability still has to await the commercialization of on-going research activities (Elmaghraby 1988).

The **resource leveling problem** (sometimes called the resource smoothing or resource loading problem) occurs when sufficient resources are available for the completion of the project, but one tries to keep resource usage as much as possible to a constant rate by rescheduling project activities within their available float subject to a fixed project duration. Many optimization and heuristic procedures are available for this problem. Commercial software packages use heuristic

procedures which are, unfortunately, rather weak and almost never explained in sufficient detail in the user and/or reference manual (De Wit and Herroelen 1988). It is interesting to note that the resource leveling problem is equivalent to the well-researched assembly line balancing problem (Herroelen 1980).

Under conditions of limited availabilities of resources, the objective of the project planners may be to allocate the various resources to the activities in such a way that the resulting increase in project duration beyond the critical path length is minimized. Reviews of this classical **resource-constrained project scheduling problem** can be found in Herroelen (1972) and Davis (1966, 1973). It is this type of problem that forms the subject of this paper.

The specific resource allocation problem addressed in this paper is the **multiple constrained-resource single project scheduling problem**, in which it is assumed that an activity is subject to technological precedence constraints (an activity can only be started if all its predecessor activities have been finished) and cannot be interrupted once begun (no job preemption allowed). Resources are assumed to be available per period in constant amounts, and are also demanded by an activity in constant amounts throughout the duration of the activity. The objective is to schedule the activities subject to the precedence and resource constraints in order to minimize the total project duration.

Comparisons of optimal and suboptimal procedures for the resource-constrained single project scheduling problem can be found in Cooper (1976), Davis and Patterson (1975), Patterson (1984), Alvarez-Valdés and Tamarit (1988). The multi-project scheduling problem has been

dealt with by Kurtulus and Davis (1982) and Kurtulus and Narula (1985). Early attempts to solve the problem concentrated in two areas: the formulation and solution of the problem as a mathematical programming problem and the development of heuristic solution procedures for obtaining satisfying solutions. Early attempts at using integer programming to solve the exact version of the problem were unsuccessful. Consequently, numerous enumerative (branch-and-bound) procedures for solving certain variants of the problem optimally were developed (Balas 1970, Christofides et al. 1987, Davis and Heidorn 1971, Elmaghraby and Herroelen 1977, Fisher 1973, Gorenstein 1972, Johnson 1967, Patterson and Huber 1974, Patterson and Roth 1976, Pritsker et al. 1969, Schrage 1970, Stinson et al. 1978, Talbot and Patterson 1978). It should be mentioned that the resource-constrained project scheduling problem is a generalization of the well-known job-shop scheduling problem, and as such is NP-complete (Blazewicz et al. 1983). In addition, recent efforts have been made to study the problem under the alternative objective of maximizing the Net Present Value of the project (see for example Elmaghraby and Herroelen 1988).

Computational results obtained by Patterson (1984) on a problem set of 110 test problems, seem to indicate that Talbot's solution procedure (Talbot and Patterson 1978) is an effective problem solving technique whenever resource constrainedness in a problem is low and would likely be the preferred solution approach where computer storage is a particularly limiting factor (Patterson 1984). The branch-and-bound solution procedure of Stinson (Stinson et al. 1978) was found to be the fastest in those instances in which computer memory is not limiting. Computational results obtained by Christofides et al. (1987) on a different set of test problems, indicate that their branch-and-bound procedure outperforms Stinson's procedure, at least for projects up to 25 activities and three resource types.

The purpose of this paper is to describe a decision support system for the multiple resource-constrained project scheduling problem which is centered upon a new branch-and-bound procedure. The software is programmed in C for personal computers equipped with EGA (Enhanced Graphics Adapter) card and running under the DOS operating system. Provisions exist in the user friendly, menu driven software to represent the optimal schedule using coloured Gantt charts and resource profiles. Maximum CPU running times can be specified and the search process can be interrupted as soon as a user-specified feasible project duration is obtained. Project data may be changed on screen, enabling the user to explore the impact of the various problem parameters on solution quality.

The optimal solution procedure is described in the next section. Section 3 focuses on computational results obtained by the branch-and-bound procedure on the well-known set of 110 test problems assembled by Patterson (1984). Section 4 dwells on the software characteristics of the decision support system. The last section is then reserved for our overall conclusions.

2. A BRANCH-AND-BOUND PROCEDURE

The single project, multiple resource-constrained project scheduling problem can be formulated using the notation

of Christofides et al. (1987) :

$$\min t_n \qquad [1]$$

subject to

$$t_j - t_i >= d_i, \qquad (i,j) \in H \qquad [2]$$

$$\sum_{S(t)} r_{ik} <= b_k, \qquad t = 1,2,...,T \quad k = 1,2,...,K, \qquad [3]$$

where

t_i = starting time of activity i, i = 1,2,...,n

H = set of pairs of activities with precedence constraints

d_i = processing time of activity i

r_{ik} = amount of resource k required by activity i

$S(t)$ = set of activities in process at time t

b_k = total availability of resource type k

It is assumed that activity i has a fixed processing time d_i (setup times are negligible or are included in the processing time). We further assume activity-on-the-node networks where activities 1 and n are dummy activities indicating the single start and end node of a project. The resource requirements r_{ik} are known constants over the processing interval of the activity and the resource availability of resource type k, b_k, is a known constant throughout the project duration interval. The precedence constraints given in Eq. [2], indicate that an activity j can only be started if all predecessor activities i are finished. Once started, activities run to completion (non-preemption condition). The resource constraints given in Eq. [3], prescribe that for each time period t and for each resource type k, the resource amounts

required by the activities in progress S(t) should not exceed the resource availability. The objective function is given as Eq. [1]. The project duration is minimized by minimizing the starting time of the dummy end activity **n**.

2.1 Depth-first branch-and-bound

The decision support software to be described in the next section is centered around a new branch-and-bound procedure based on **depth-first** search. Typical for a depth-first search is that a branch of the search tree is selected and systematically worked down to reach as quickly as possible a feasible solution. At each node of the branch-and-bound tree we try to put in process all the unscheduled activities satisfying the precedence constraints. Partial schedules, PS_t, only need to be considered at those time instants t which correspond to the completion time of one or more project activities. At every such time instant t, a corresponding partial schedule PS_t will contain some activities which have been completed, and others which are still in progress (the set S(t) defined above). Also at every time instant t, we define the **eligible set E**, as the set of activities which are not in the partial schedule and whose predecessor activities have finished. These eligible activities can start at time t if the resource constraints are not violated.

If it is impossible to schedule all eligible activities at time t, a so-called **resource conflict** occurs (a similar concept is described in Bellman et al. (1982) and Christofides et al. (1987)). Such a conflict will produce a new branching in the branch-and-bound tree. The branches describe ways to resolve the resource conflict; i.e., decisions about which activities are to be delayed.

We therefore define a **delaying set** D(p), which consists of all subsets of activities, either in process or eligible, the delay of which would resolve the resource conflict at level p.

The delay of a subset of activities, $D_q \in D(p)$, is introduced by adding extra arcs that force them to wait until the completion of some other activities. As such we determine for every task $i \in D_q \in D(p)$, the earliest finishing activity j, that is either in progress or eligible and that is not delayed, the completion of which would allow task i to start (ties are broken arbitrarily).

Backtracking occurs in the depth-first search when a schedule is completed or a branch is to be fathomed by the lower bound calculation. At the backtrack the added arc(s) corresponding to the last delaying set element $D_q \in D(p)$ is (are) removed and new arcs are added for the next one at the same level. If there is no delaying set element D_q left at that level, we backtrack to the previous one. When the level zero is reached in the search tree, the search process is completed and the optimal solution found.

Dominance pruning occurs whenever it can be shown that an activity can be left-shifted and the resulting schedule is both time and resource feasible. This rule is identical to the one used by Schrage (1970), Herroelen (1972) and Stinson et al. (1978).

The **lower bound computation** is based on the classical precedence-based and resource-based bounding arguments described by Schrage (1970), Herroelen (1972) and Stinson (1978). The **precedence-based lower bound** is computed as the earliest time that any unscheduled activity can be

started plus the critical path length of the remaining unscheduled activities, ignoring resource constraints. The **resource-based lower bounding** argument is based on the dynamically updated man-days requirements for the unscheduled activities divided by the per-period availability of the resources for the resource yielding the maximum remaining length. In updating the man-days requirements, explicit consideration is given to the resource-period availability which is unutilizable by the corresponding partial schedule (Elmaghraby 1977, Zaloom 1971).

The detailed algorithmic steps can now be described as follows:

Step 1. Let $T = 9999$ be an upper bound on the project duration. Set the level of the branch-and-bound tree $p = 0$. Initialize $m = 0$. For every activity i compute the remaining critical path length $RCPL_i$. For each resource type k, compute the resource work content $RWC_k = \Sigma_i (d_i * r_{ik})$. Initialize the activity completion times $t_i = 9999$. Schedule the dummy start activity at its earliest start time; i.e., set $t_1 = d_1$. Update the partial schedule $PS = \{1\}$ and the set of activities in progress $S = \{1\}$. Compute the lower bound as

$$LB(0) = \max \{RCPL_1; \max_k \lceil RWC_k/b_k \rceil\}$$

where $\lceil a \rceil$ denotes the smallest integer greater than or equal to a.

Step 2. Set $a = m$. Compute the number of free resources of type k as $f_k = b_k - \Sigma_{i \in S} r_{ik}$. Compute the next decision point, m, as the earliest completion time of all activities in progress: $m = \min\{t_i, i \in S\}$. For all activities $j \in S$ in progress for which $t_j = m$, update the set of activities in progress: $S = S - \{j\}$.

If the last scheduled activity is activity n, the schedule is completed. Update the schedule length $T = t_n$. If T is equal to LB(0), then stop with the optimal solution, else go to step 10 (backtrack).

For each resource type k, update the resource work content by adding the unutilizable resource-period availability: $RWC_k = RWC_k + f_k * (m - a)$. If $\lceil RWC_k/b_k \rceil$ is greater than or equal to T, then go to step 10 (backtrack).

Step 3. Initialize the set of eligible activities $E = \phi$ (the empty set). Construct the set of eligible activities: for each activity i which is not yet in the partial schedule PS, update the eligible set $E = E \cup \{i\}$ if all its predecessors are completed; i.e., if for all $(j,i) \in H$, $t_j <= m$. If there are no eligible activities (i.e., if $E = \phi$), go to step 2. If there are still some activities in progress, go to step 5.

Step 4. For each eligible activity $i \in E$, check whether there are other unscheduled activities j which can be processed simultaneously with activity i without violating the precedence and resource constraints. If none can be found, then put the eligible activity in progress: $PS = PS \cup \{i\}$, $S = \{i\}$, $t_i = m + d_i$ and go to step 2.

Step 5. Check whether it is possible to schedule all eligible activities together with the activities in progress within the resource availability of every resource k; i.e., for each k, check if $\Sigma_{i \in S} r_{ik} + \Sigma_{j \in E} r_{jk} <= b_k$. If there is at least one resource type k for which the sum of the resource requirements of all in-progress and eligible activities exceeds the resource availability, we have a resource conflict.

If there is no resource conflict, update the partial schedule : PS = PS U E, put the eligible activities in progress : S = S U E, set $t_i = m + d_i$ for all $i \in E$ and go to step 2.

Step 6. Update the branch level in the search tree: p = p + 1. Determine for each resource type k how many units have to be freed to resolve the resource conflict; i.e., for each k, set $tbf_k = \Sigma_{i \in S} r_{ik} + \Sigma_{j \in E} r_{jk} - b_k$. Define the delaying set
$D(p) = \{D_q \subset (S \cup E) \mid \Sigma_{i \in D_q} r_{ik} >= tbf_k$ for all k and D_q does not contain other D_q as a subset$\}$.

Step 7. For every task $i \in D_q \in D(p)$ determine the earliest finishing task j, that is either in progress or eligible and that is not delayed (i.e., $j \in (S+E-D_q)$), the completion of which would allow task i to start within the resource availabilities. Define the corresponding set of tuples $G_q = \{(j,i)\}$. Compute the lower bound L_q of each alternative D_q:

$$L_q = \max_{(j,i) \in D_q} \{ RCPL_i + \min (m + d_j, t_j)\}$$

Step 8. Branch into a new node of the branch-and-bound tree : store the activity completion times, the partial schedule, the set of activities in progress, the set of eligible activities, the resource work content for each resource type and the decision point m.

Select the $D_q \in D(p)$ with the smallest L_q. Update the delaying set : $D(p) = D(p) - D_q$.

Set $LB(p) = \max \{LB(p-1), L_q\}$. If $LB(p) >= T$, decrease the branching level : p = p - 1 and go to step 10.

Step 9. Define AS as the set of all eligible activities that must not be delayed : $AS = \{i \in E \mid i \notin D_q\}$. Define DS as the set of all activities that were in progress, but must be delayed : $DS = \{i \in S \mid i \in D_q\}$. Update: $PS = PS + AS - DS$, $S = S + AS - DS$, for all $i \in AS$ set $t_i = m + d_i$ and for all $i \in DS$ set $t_i = 9999$.

Add the extra precedence relations: $H = H \cup G_q$. If DS is not empty, invoke the left-shift dominance rule. If the added precedence constraints force a task to be started at time m, while the original precedence relations would allow it to be started earlier without violating the resource constraints, then this schedule is dominated: go to step 10 (backtrack); else, go to step 2.

Step 10. If the branching level $p = 0$, then STOP. Delete the extra precedence relations which were added at this branching level: $H = H - G_q$. If $D(p) = \emptyset$, set $p = p - 1$ and repeat step 10.

Select the $D_q \in D(p)$ with the smallest L_q. Update the delaying set : $D(p) = D(p) - D_q$. Compute the lower bound $LB(p) = \max \{LB(p-1), L_q\}$. If $LB(p) >= T$, decrease the branching level: $p = p - 1$ and repeat step 10.

Restore the activity completion times, the partial schedule, the set of activities in progress, the set of eligible activities, the resource work content for each resource type and the decision point m.

Go to step 9.

2.2 Numerical example

Consider the activity-on-the-node network given in Figure 1. The numbers above each node denote the fixed activity durations. The numbers below each node denote the daily

resource requirement for two resource types. The resources have a constant availability of 7, resp. 5 units per day.

==========
Insert Figure 1
==========

Initializing the branch-and-bound procedure, we set T = 9999 and find $RCPL_1$ = 8 as the critical path length. The resource work content is RWC_1 = 59 for the first and RWC_2 = 37 for the second resource type. Initialize the level of the branch-and-bound tree and the decision point: p = 0, m = 0. Initialize all activity completion times t_i = 9999. Schedule the dummy activity 1: t_1 = 0. The partial schedule is updated as PS {1} and the set of activities in progress is S = {1}. The lower bound is computed as LB(0) = $\max\{8;\max(\lceil 59/7 \rceil, \lceil 37/5 \rceil)\}=9$. This corresponds to node 1 at level 0 of the branch-and-bound search tree represented in Figure 2.

Proceeding with step 2, we set a = 0. The number of free units of resource type 1 and 2 equal f_1 = 7-0 = 7 and f_2 = 5-0 = 5, respectively. The earliest completion time of all activities in progress is m = 0, the next decision point. Update S = ϕ. The resource work contents are updated as RWC_1 = 59 + 7(0-0) = 59 for resource type 1 and RWC_2 = 37 + 5(0-0) = 37 for resource type 2. Applying step 3 of the procedure, the set of eligible activities is updated as E = {2,3,4,5}. Since it is possible to put eligible activities simultaneously in progress (step 4), we proceed with step 5. The resource requirements for type 1 and type 2, summed over all eligible activities, are 10, respectively 6 units. A resource conflict occurs. Proceed with step 6. Set p = 1 (level of the branch-and-bound tree in Figure 2), and compute the number of resource units to be released in order to resolve the

resource conflict: $tbf_1 = 10-7 = 3$ and $tbf_2 = 6-5 = 1$. The delaying set is $D(1) = \{\{2\},\{5\},\{3,4\}\}$. For $D_1 = \{2\}$, activity 5 is found to be the activity whose completion would allow activity 2 to start without violating the resource constraints. Set $G_1 = \{(5,2)\}$ with lower bound estimate $L_1 = \{RCPL_2+\min(0+d_5,t_5)\} = \{6+\min(2,9999)\}= 8$. In a similar fashion we find for $D_2 = \{5\}$ the tuple $G_2 = \{(2,5)\}$ with $L_2 = 8$. For both activities of $D_3=\{3,4\}$, we determine activity 5 as the activity whose completion would resolve the resource conflict. Define the set of tuples $G_3 = \{(5,3),(5,4)\}$ with corresponding lower bound estimate $L_3= \max \{[RCPL_3+\min(0+d_5,t_5)], [RCPL_4+\min(0+d_5,t_5)]\} = 10$.

Proceeding with step 8, select $D_1 = \{2\}$ with the smallest lower bound estimate $L_1 = 8$. Update $D(1) = \{\{5\},\{3,4\}\}$. Compute the lower bound for **node 2** of the branch-and-bound tree as $LB(1) = \max \{LB(0),L_1\} = 9$ and continue with step 9 of the algorithm. Set $AS = \{3,4,5\}$, $DS = \phi$ and $PS = \{1,3,4,5\}$. Put the activities 3, 4 and 5 in progress: $S = \{3,4,5\}$, $t_3 = 5$, $t_4 = 4$ and $t_5 = 2$. The extra precedence relation (5,2) is added. The left-shift dominance rule does not apply and the procedure continues with step 2.

Set $a = 0$. The number of free units of resource type 1 and 2 equal $f_1 = 7-7 = 0$ and $f_2 = 5-5 = 0$, respectively. The earliest completion time of all activities in progress is $m = 2$, the next decision point. Update the set of activities in progress $S = \{3,4\}$. The resource work contents are updated as $RWC_1 = 59 + 0(2-0) = 59$ for resource type 1 and $RWC_2 = 37 + 0(2-0) = 37$ for resource type 2. Applying step 3 of the procedure, the set of eligible activities is updated as $E = \{2\}$. Activity 3 and 4 are still in progress, so we continue with step 5. The resource requirements for type 1 and type 2, summed over

all eligible activities and activities in progress, are 7, respectively 5 units. No resource conflict occurs. Update PS = {1,2,3,4,5} and start activity 2: S = {2,3,4} and t_2 = 2+3 = 5.

Continue with step 2 of the procedure: set a = 2, f_1 = 0 and f_2 = 0. The earliest completion time is now m = 4 for activity 4. Update S = {2,3}. The resource work contents remain unchanged. There are no eligible activities at time m = 4, so E = ϕ. Repeat step 2: set a = 4, f_1 = 2 and f_2 = 0. The next decision point is m = 5. Activity 2 and 3 being completed, update S = ϕ. Update RWC_1 = 61 and RWC_2 = 37. Continue with step 3: E = {6,7,8}. Applying step 5 of the algorithm, a resource conflict is found to occur. Set p = 2, corresponding to the second level of the branch-and-bound tree in Figure 2. The number of resources to be released in order to resolve the resource conflict is tbf_1 = 3 for resource type 1 and tbf_2 = 0 for resource type 2. The delaying set is D(2) = {{6},{8}}. The completion of activity 8 would resolve the resource conflict with respect to activity 6. Hence G_1 = {(8,6)} with lower bound estimate $L_1 = RCPL_6+\min(5+d_8,9999) = 10$. For D_2 = {8}, we find G_2 = {(6,8)} with L_2 = 10.

Proceeding with step 8, select D_1 = {6} with the smallest lower bound estimate L_1 = 10. Update D(2) = {8} and compute the lower bound LB(2) = max {LB(1),L_1} = 10. Branch into **node 3** of the branch-and-bound tree of Figure 2. Set AS = {7,8}, DS = ϕ, PS = {1,2,3,4,5,7,8}. Put the activities 7 and 8 in progress: S = {7,8}, $t_7=t_8=8$. Add the extra precedence relation (8,6) and continue with step 2, since the left-shift dominance rule does not apply.

Set a = 5, f_1 = 1 and f_2 = 3. The next decision point is m = 8, the completion time of activities 7 and 8.

Consequently, $S = \phi$, $RWC_1 = 64$, $RWC_2 = 46$. Set $E = \{6\}$. There is no resource conflict and activity 6 is put in progress: $PS = \{1,2,3,4,5,6,7,8\}$, $S = \{6\}$ and $t_6 = 10$. Returning to step 2 of the procedure, we set $a = 8$, $f_1=3$ and $f_2= 2$. The next decision point is $m = 10$, the completion time of activity 6. $S = \phi$, $RWC_1=70$ and $RWC_2=50$. The set of eligible activities is updated: $E = \{9\}$. The dummy end activity is scheduled: $S = \{9\}$, $PS = \{1,2,3,4,5,6,7,8,9\}$, $t_9 = 10$. The next decision point is $m = 10$. Since the last activity has been scheduled, the schedule length is updated as $T = 10$, and the procedure backtracks to level 2 of the branch-and-bound tree.

The extra precedence relations at level 2 are deleted and $D_2 = \{8\}$ is selected with $L_2 = 10$. This corresponds to the descendant node 6 <. 8 at level 2 of Figure 2. Compute $LB(2) = 10$. Since this lower bound value is equal to T, the node is fathomed. Set $p = 1$ and backtrack to level 1 of the search tree. Delete the extra precedence relations at level 1 and select $D_2 = \{5\}$ with $L_2 = 8$ corresponding to node 4 of the search tree. Update $D(1) = \{\{3,4\}\}$ and compute the lower bound $LB(1) = \max \{LB(0), L_2\} = 9$. Now $LB(1) < T$. Continuing with step 9 of the procedure: $AS = \{2,3,4\}$, $DS = \phi$. Activities 2,3 and 4 are put in progress: $PS = \{1,2,3,4\}$, $S = \{2,3,4\}$ and $t_2 = 3$, $t_3 = 5$ and $t_4 = 4$. The extra precedence relation (2,5) is added, and since the left-shift dominance rule does not apply, we continue with step 2.

Set $a = 0$, $f_1 = 0$ and $f_2 = 0$. The next decision point is $m = 3$ corresponding to the completion time of activity 2. Update $S = \{3,4\}$, $RWC_1 = 59$ and $RWC_2 = 37$. The eligible set is $E = \{5,6,7\}$. A resource conflict occurs. Set $p = 2$ and compute the number of resource units to be released

in order to resolve the resource conflict: $tbf_1 = 6$ and $tbf_2 = 4$. The delaying set is $D(2) = \{\{5,6\},\{6,7\},\{3,6\},\{3,4,5\},\{3,4,7\},\{3,5,7\}\}$. For $D_1 = \{6,7\}$ we find the set of tuples $G_1 = \{(3,6),(3,7)\}$ with lower bound estimate $L_1 = \max\{RCPL_6+\min(3+d_3,t_3);RCPL_7+\min(3+d_3,t_3)\} = 8$. In a similar fashion we find for $D_2=\{5,6\}$ the set of tuples $G_2 = \{(3,5),(3,6)\}$ with $L_2 = 10$. In addition we find $G_3 = \{(5,3),(5,6)\}$ with $L_3 = 13$, $G_4 = \{(6,3),(6,4),(6,5)\}$ with $L_4=13$, $G_5 = \{(5,3),(5,4),(5,7)\}$ with $L_5=13$, and $G_6 = \{(6,3),(4,5),(4,7)\}$ with $L_6=13$.

Select $D_1 = \{6,7\}$ with smallest $L_1 = 8$. Update the delaying set $D(2) = \{\{5,6\},\{3,6\},\{3,4,5\},\{3,4,7\},\{3,5,7\}\}$. Compute $LB(2) = \max\{LB(1),L_1\} = 9$. Continue with step 9 of the algorithm. Set $AS = \{5\}$, $DS = \phi$ and put activity 5 in progress: $PS = \{1,2,3,4,5\}$, $S = \{3,4,5\}$ and $t_5 = 5$. Add the extra precedence relations (3,6) and (3,7) corresponding to **node 5** in the branch-and-bound tree of Figure 2. Continue with step 2.

Set $a = 3$, $f_1 = 0$ and $f_2 = 0$. The next decision point is $m = 4$ corresponding to the completion of activity 4. Update $S = \{3,5\}$, $RWC_1 = 59$ and $RWC_2 = 37$. The set of eligible activities is now empty: $E = \phi$. We set $a = 4$, $f_1=2$ and $f_2=0$. The next decision point is $m = 5$. Update $S = \phi$, $RWC_1 = 61$, $RWC_2 = 37$. Now $E = \{6,7,8\}$. A resource conflict occurs. Step 6 of the algorithm prepares for level 3 of the search tree: $p = 3$, $tbf_1 = 3$ and $tbf_2 = 0$. The delaying set is $D(3) = \{\{6\},\{8\}\}$. For $D_1 = \{6\}$ we find $G_1 = \{(8,6)\}$ with lower bound estimate $L_1 = 10$ and for $D_2 = \{8\}$ we have $G_2 = \{(6,8)\}$ with $L_2 = 10$. Select $D_1=\{6\}$ with the smallest lower bound estimate. This corresponds to the node 8 <. 6 at level 3 of the search tree in Figure 2. The lower bound $LB(3) = 10 = T$. The node is fathomed. Set $p = 2$ and backtrack.

Now select D_2 = {5,6} with L_2 = 10, corresponding to the node 3 <.5,6 at level 2 of the search tree in Figure 2. Since LB(2) = 10 = T, the node is fathomed and we backtrack to level 1. Select D_3 = {3,4} with L_3 = 10, corresponding to the node 5 <. 3,4 at level 1 of the search tree. Now LB(1) = 10 = T. Set p = 0 and stop.

==========
Insert Figure 2
==========

3. COMPUTATIONAL RESULTS

The branch-and-bound procedure described in the previous section has been programmed in Microsoft C, Version 5.10 for a personal computer IBM PS/2 Model 70 (or compatibles) running under the DOS operating system. The one hundred and ten test problems assembled by Patterson (1984) were used to validate the procedure. These problems represent an accumulation of all multi-resource problems existing in the literature today that are readily available. The number of activities included in these test problems varies between 7 and 50, with the number of resource types required per activity varying between one and three. The majority of the projects (103) consists of activities which require the full complement of three different resource types for their performance.

The computational results are given in Table I. This table lists for each of the 110 test problems, the optimal project duration and the CPU time in seconds needed to find the feasible schedule corresponding to the optimal project duration (as confirmed later by the backtracking stage of the algorithm). It also lists the total CPU time in seconds required by the branch-and-bound procedure and the maximal value of p, the number of levels in the branch-and-bound tree.

As can be seen from this table, the procedure was able to solve all 110 test problems within the CPU time limit of 5 minutes per problem imposed. The average total CPU time (excluding input and output) obtained on a personal computer IBM PS/2 Model 70 A21 with 25 MHz processor was 2.75 seconds with standard deviation of 15.49 seconds. This relatively high standard deviation was mainly caused by problem 72 which required 162.08 seconds to solve. The remaining 109 test problems required an average CPU time of only 1.29 seconds with a standard deviation of 2.24 seconds.

In 104 out of the 110 test problems, the time required to generate the feasible solution which, upon backtracking, proves to be the optimal one, was less than 3 seconds. This is a strong indication of the excellent quality of the program if it is not allowed to run to completion but used as a heuristic.

Table II compares the summary operating characteristics of the procedure with the solution procedures of Davis (1971), Stinson (1978) and Talbot (1978) as obtained by Patterson (1984) on an Amdahl 470/V8. Both the procedure described in this paper and Stinson's method are the only ones capable of solving all 110 test problems within a 5 minute CPU time limit. The average CPU time obtained by our procedure is of the same order of magnitude as the one obtained by Stinson's method, where it is to be understood that the Stinson results were obtained on a mainframe.

==========
Insert Table II
==========

The promising computational results obtained by the branch-and-bound procedure make it a valuable tool to be included in a decision support system for resource monitoring in activity networks.

4. THE RESOURCE MONITORING DECISION SUPPORT SYSTEM

The branch-and-bound procedure described in the previous section lies at the heart of a decision support system (DSS) for resource monitoring in project activity networks. The DSS has been programmed in Microsoft C, Version 5.1 and runs under DOS on IBM PS/2 (or compatibles) equipped with enhanced graphics (EGA) capabilities. The software is menu driven allowing the user to activate the menu items in graphics mode, either by clicking the mouse or by using the arrow keys on the keyboard. It is currently dimensioned to deal with activity-on-the-node networks within a maximum of sixty activities and constant resource requirements for up to ten different resource types.

Project data are entered on screen or through an unformatted ASCII input file. In both cases, the data consist of the number of activities, the number of resource types, the fixed activity durations and constant activity requirements for each of the resource types, the resource availabilities, and the immediate successor activities of each project activity, based on the common finish-start precedence relationships.

By choosing the option **Change parameters** from the input screen, the user may specify the time limit in seconds imposed on the DSS solution procedure (the default option is a time limit of 300 seconds). In addition, the user may specify an acceptable project duration for the problem to be solved. As soon as the procedure has found a feasible solution with a project duration smaller than or equal to the specified project length, it will stop. This is an interesting what-if capability in those

instances where the user has a feasible solution on hand and quickly wants to check the possibility of improving on it.

By choosing the option **Compute solution** from the menu, the branch-and-bound procedure described earlier in this paper is activated and tries to obtain the optimal solution for the corresponding resource-constrained project scheduling problem within the specified time limit. Choosing the option **Output** from the menu, an output screen may be generated which lists the activities with their corresponding start times. If the time limit imposed did allow the branch-and-bound algorithm to run to completion, this listing is headed by an indication of the optimal schedule length and the CPU time (in seconds) needed to find the optimal resource-constrained solution. If not, the activity listing and start times correspond to the best feasible solution found within the imposed time limit. The header then specifies the "time-constrained" schedule length and the CPU time needed to find this feasible solution. This is an utmost interesting what-if analysis characteristic of the DSS software package, enabling the user to obtain in a relatively fast manner good feasible solutions for those problems which cannot be solved optimally within the imposed time limits.

Choosing the option **Gantt chart** from the **Output data menu** invokes a colour-based Gantt chart, listing the activities at their start times corresponding to the reported solution (optimal or feasible). Activities which are delayed beyond their earliest start time have their coloured activity bars preceded by a black line. The Gantt chart scale automatically expands or shrinks depending on the number of activities to be plotted.

The option **Resource profile** from the **Output data menu** generates a coloured resource histogram for each of the resource types. Again, the scale of the histograms automatically changes depending on the number of activities and the resource limits imposed on the problem.

The **Change data** option offers the possibility for changing the project data on-line. This involves changing activity durations and precedence relations, adding and deleting resource types. Changing the various resource constraints allows the user to explore the impact of resource-constrainedness on solution quality and provides a de facto resource leveling heuristic.

5. CONCLUSIONS

In this paper a decision support system was described for scheduling a project to minimize its total duration subject to technological precedence constraints and resource constraints. The system is centered around a branch-and-bound procedure for the multiple constrained resource project scheduling problem. Computational experience gained with the algorithm on a personal computer was very promising. The 110 Patterson-test problems could be solved optimally within an average CPU time of 2.75 seconds on a personal computer IBM PS/2 Model 70 A21 with 25 MHz processor. Feasible solutions which are close to the optimum can be obtained in a small amount of CPU time.

The decision support capabilities of the developed software are extensive. The optimal schedule can be represented by coloured Gantt charts and resource profiles. Maximum CPU running times can be specified and the search process can be interrupted as soon as a user-specified feasible project duration is obtained. Project

data may be changed on-line, providing an extensive what-if analysis capability. Resource types may be added or deleted on screen. The availabilities of up to ten resource types may be altered, enabling the user to explore the impact of resource-constrainedness on solution quality.

ACKNOWLEDGEMENT

The authors are gratefully indebted to Professor James Patterson, Indiana University. Without his cooperation and assistance in providing us with the source data for the 110 test problems, this paper would not have been possible.

REFERENCES

ALVAREZ-VALDES, R. and J.M. TAMARIT (1988), Computational Comparison of Classical and New Heuristic Algorithms for Resource-Constrained Project Scheduling, Paper presented at the First International Workshop on Project Management and Scheduling, Lisbon, 11-13 July, 1988.

BALAS, E. (1970), Project Scheduling with Resource Constraints, in **Applications of Mathematical programming Techniques** (Beale ed.), American Elsevier, New York.

BELLMAN, R., A.O. ESOGBUE and I. NABESHIMA (1982), **Mathematical Aspects of Scheduling and Applications**, Pergamon Press, Oxford.

BLAZEWICZ, J., J.K. LENSTRA and A.H.G. RINNOOY KAN (1983), Scheduling Projects to Resource Constraints: Classification and Complexity, **Discrete Applied Mathematics**, 5, 11-24.

CHRISTOFIDES, N., R. ALVAREZ-VALDES and J.M. TAMARIT (1987), Project Scheduling with Resource Constraints: A Branch and Bound Approach, **European Journal of Operational Research**, 29, 262-273.

COOPER, D.F. (1976), Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation, **Management Science**, 22(11), 1186-1194.

DAVIS, E.W. (1966), Resource Allocation in Project Network Models - A Survey, **Journal of Industrial Engineering**, 17(4), 177-188.

DAVIS, E.W. (1973), Project Scheduling Under Resource Constraints: Historical Review and Categorization of Procedures, **AIIE Transactions**, 5(4), 297-313.

DAVIS, E.W. and G.E. HEIDORN (1971), An Algorithm for Optimal Project Scheduling Under Multiple Resource Constraints, **Management Science**, 27(12), B803-816.

DAVIS, E.W. and J. PATTERSON, (1975), A Comparison of Heuristic and Optimal Solutions in Resource-Constrained Project Scheduling, **Management Science**, 21(8), 944-955.

DE WIT, J. and W. HERROELEN (1988), An Evaluation of Microcomputer-based Software Packages for Project Management, Paper presented at the First International Workshop on Project Management and Scheduling, Lisbon, 11-13 July, 1988.

ELMAGHRABY, S.E. (1977), **Activity Networks - Project Planning and Control by Network Models**, Wiley, New York.

ELMAGHRABY, S.E. (1988), Project Bidding Under Deterministic and Probabilistic Activity Durations, Paper presented at the First International Workshop on Project Management and Scheduling, Lisbon, 11-13 July, 1988.

ELMAGHRABY, S.E. and W.S. HERROELEN (1977), A Cost Minimization Problem with Resource-Duration Interaction, Research Report, Department of Applied Economic Sciences, Katholieke Universiteit Leuven, Belgium.

ELMAGHRABY, S.E. and W.S. HERROELEN (1988), The Scheduling of Activities to Maximize the Net present Value of Projects, OR Report N° 222, North Carolina State University at Raleigh, NC, U.S.A.

FISHER, M. (1973), Optimal Solution of Scheduling Problems Using Lagrange Multipliers - Part I, **Operations Research**, 21(5), 1114-1127.

GORENSTEIN, S. (1972), An Algorithm for Project Sequencing with Resource Constraints, **Operations Research**, 20(4), 835-850.

HERROELEN, W.S. (1972), Resource-constrained Project Scheduling - The State of the Art, **Operational Research Quarterly**, 23, 261-275.

HERROELEN, W.S. (1972), **Heuristische Programmatie - Methodologische benadering en praktische toepassing op een hulpmiddelentoewijzingsprobleem**, Aurelia Books.

HERROELEN, W.S. (1980), Assembly Line Balancing Problems in Perspective, **Tijdschrift voor Economie en Management**, XXV(1), 61-76.

JOHNSON, T.J.R. (1967), An Algorithm for the Resource-Constrained Project Scheduling problem, unpublished Ph.D. Dissertation, MIT.

KURTULUS, I. and E.W. DAVIS (1982), Multiproject Scheduling: Categorization of Heuristic Rules Performance, **Management Science**, 28(2), 161-172.

KURTULUS, I. and S.C. NARULA (1985), Multi-Project Scheduling: Analysis of Project Performance, **IIE Transactions**, 17(1), 58-66.

MODER, J.J., C.R. PHILLIPS and E.W. DAVIS (1983), **Project Management with CPM, PERT and Precedence Diagramming**, Van Nostrand Reinhold Company.

PATTERSON, J. (1984), A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem, **Management Science**, 30(7), 854-867.

PATTERSON, J.H. and W.D. HUBER (1974), A Horizon-Varying Zero-One Approach to Project Scheduling, **Management Science**, 20(6), 990-998.

PATTERSON, J.H. and G. ROTH (1976), Scheduling a Project Under Multiple Resource Constraints: A Zero-One Programming Approach, **Management Science**, 16(1), 93-108.

SCHRAGE, L. (1970), Solving Resource-Constrained Network Problems by Implicit Enumeration - Nonpreemptive Case, **Operations Research**, 18(2), 225-235.

STINSON, J.P., E.W. DAVIS and B.M. KHUMAWALA (1978), Multiple Resource-Constrained Scheduling Using Branch and Bound, **AIIE Transactions**, 10(3), 252-259.

TALBOT, B. and J.H. PATTERSON (1978), An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems, **Management Science**, 24(11), 1163-1174.

ZALOOM, V. (1971), On the Resource-Constrained Project Scheduling Problem, **AIIE Transactions**, 3(4), 302-305.

FIGURE CAPTIONS.

Figure 1. Precedence diagram for the problem example.

Figure 2. Branch-and-bound search tree.

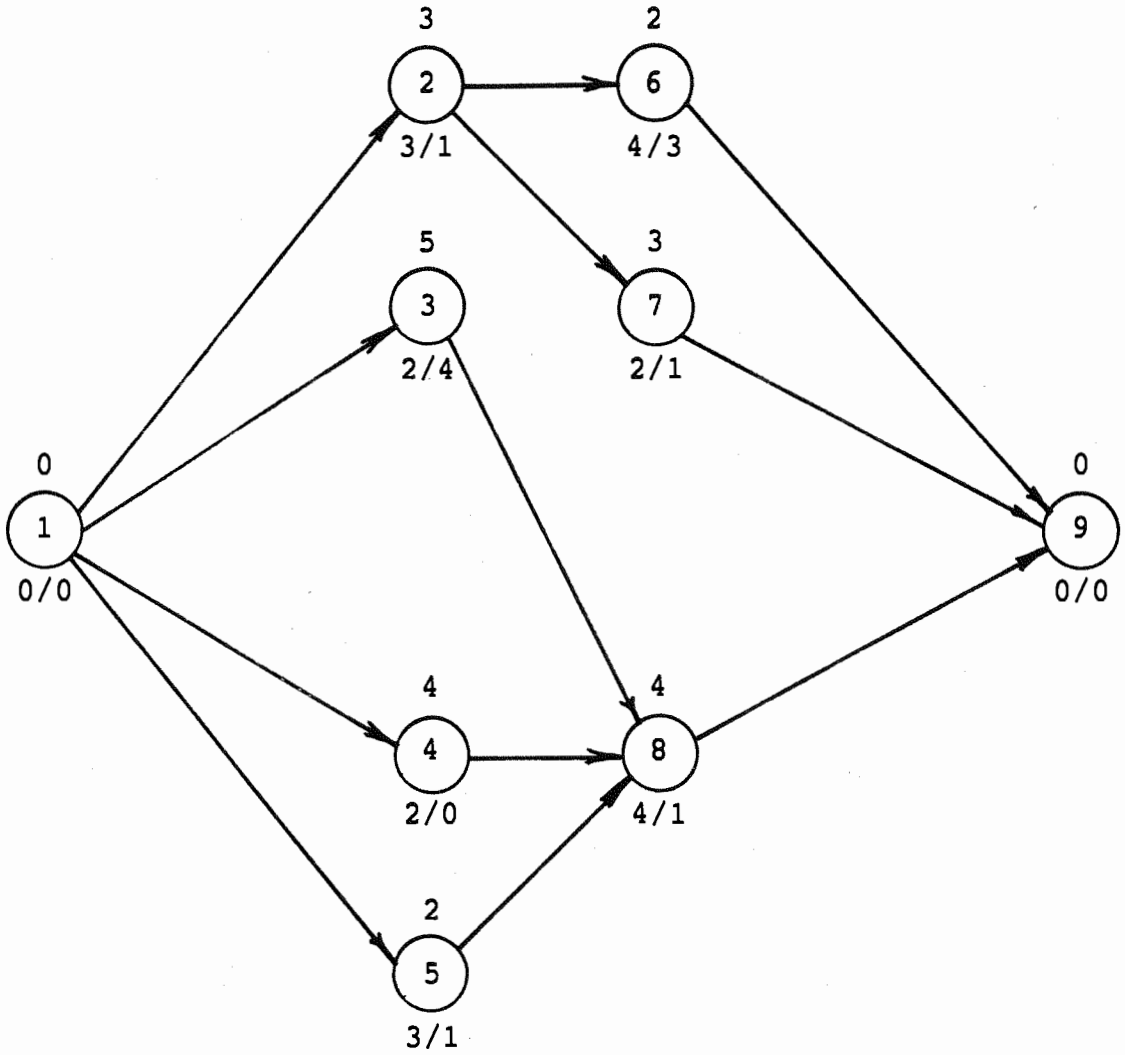

Fig. 1

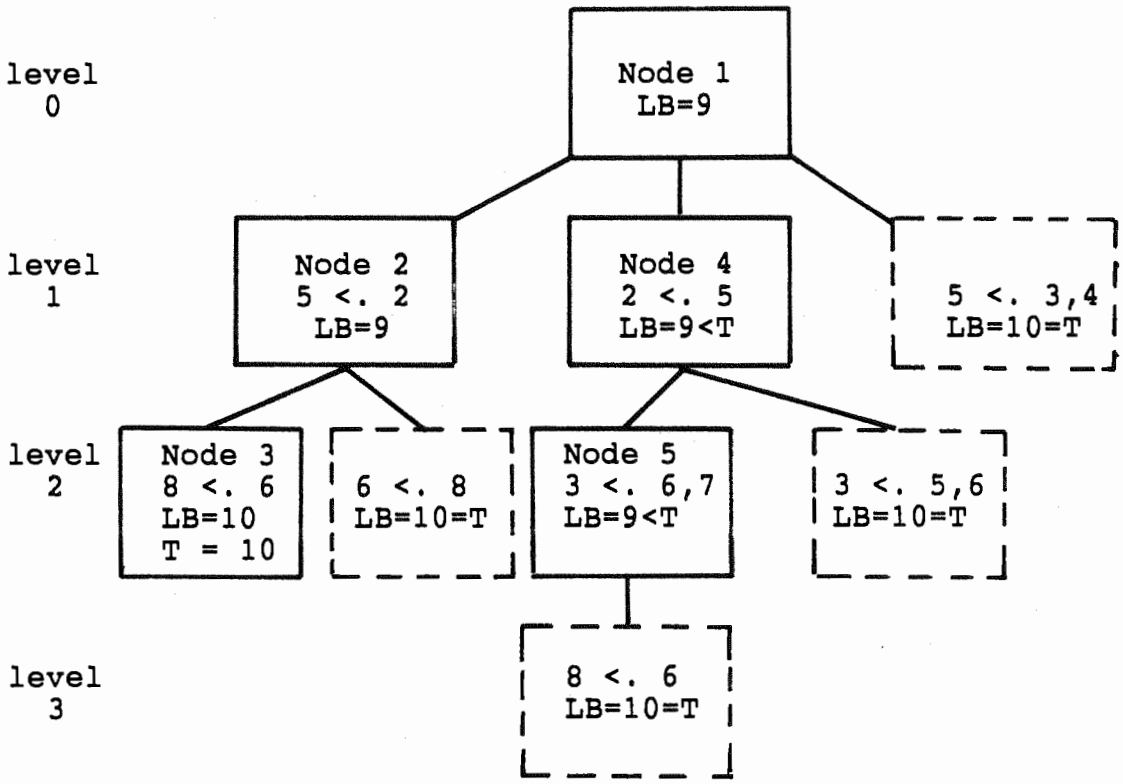
level
0
Node 1
LB=9
level
1
Node 2
5 <. 2
LB=9
Node 4
2 <. 5
LB=9<T
5 <. 3,4
LB=10=T
level
2
Node 3
8 <. 6
LB=10
T = 10
6 <. 8
LB=10=T
Node 5
3 <. 6,7
LB=9<T
3 <. 5,6
LB=10=T
level
3
8 <. 6
LB=10=T

Fig. 2

TABLE CAPTIONS.

Table I. Computational results obtained by the branch-and-bound procedure.

Table II. Summary operating characteristics of each solution procedure.

Problem	Optimal	First time*	Final time**	Maximum p***
1	19	0.00	0.00	5
2	7	0.00	0.00	1
3	20	0.06	0.06	5
4	6	0.06	0.06	0
5	7	0.00	0.00	1
6	8	0.00	0.00	2
7	8	0.00	0.00	2
8	11	0.00	0.00	3
9	19	0.06	0.06	8
10	14	0.00	0.00	0
11	18	0.00	0.00	2
12	13	0.06	0.06	10
13	20	0.44	1.15	11
14	43	0.00	0.06	9
15	43	0.00	0.00	13
16	32	0.05	0.22	13
17	29	1.54	1.71	14
18	41	0.11	0.28	12
19	31	0.00	12.69	16
20	37	0.00	0.00	12
21	48	0.06	0.22	14
22	36	0.11	0.49	12
23	32	0.17	0.39	14
24	40	0.05	0.71	12
25	33	0.00	0.06	8
26	43	0.38	0.38	11
27	36	0.28	0.28	11
28	43	0.28	0.33	13
29	29	0.33	0.88	15
30	32	0.00	0.88	12
31	35	0.06	1.16	14
32	22	0.06	0.06	12
33	31	0.28	2.86	15
34	30	0.44	1.49	18
35	31	0.00	0.66	13
36	33	0.05	0.16	10
37	28	0.05	1.10	12
38	30	0.11	0.55	14
39	31	0.11	0.11	11
40	31	0.06	0.39	13
41	36	0.60	1.04	13
42	28	0.06	0.06	14
43	41	0.06	2.69	15
44	31	0.05	0.05	12
45	39	0.55	1.48	13
46	33	0.22	3.29	15
47	35	1.70	4.39	16
48	23	0.33	0.77	16
49	18	0.00	0.00	7
50	25	0.16	0.33	13

(Table I continued)

Problem	Optimal	First time*	Final time**	Maximum p***
51	25	0.99	5.88	16
52	27	0.16	1.48	13
53	28	0.00	0.72	8
54	50	0.00	0.00	1
55	29	0.00	0.00	6
56	27	0.00	0.00	1
57	21	0.00	0.05	0
58	35	0.05	0.82	12
59	31	0.28	0.39	10
60	39	0.05	0.22	13
61	36	0.22	0.61	15
62	37	0.00	1.42	13
63	40	0.22	0.60	14
64	37	0.60	3.40	17
65	40	0.06	0.39	16
66	38	0.00	0.05	10
67	27	0.88	3.25	14
68	41	0.16	0.22	14
69	30	0.00	2.14	15
70	31	0.06	0.66	14
71	32	1.21	1.49	17
72	41	125.67	162.08	18
73	36	4.88	5.05	19
74	30	0.06	0.06	11
75	34	0.28	3.96	15
76	43	0.06	0.39	11
77	64	0.00	0.88	10
78	53	0.11	7.69	15
79	45	0.11	2.63	17
80	38	0.22	0.66	16
81	36	0.33	1.16	15
82	34	0.06	0.06	11
83	34	0.00	0.06	10
84	33	0.00	0.00	3
85	31	0.00	0.00	5
86	31	0.00	0.00	1
87	29	0.06	2.09	15
88	40	0.05	0.55	14
89	31	0.05	0.16	13
90	39	9.78	12.80	18

(Table I continued)

Problem	Optimal	First time*	Final time**	Maximum p***
91	35	3.74	4.18	16
92	28	0.05	1.48	14
93	26	0.11	1.15	13
94	36	0.11	0.11	16
95	33	0.11	0.33	13
96	26	0.72	1.05	14
97	30	0.11	0.39	12
98	41	3.68	4.34	17
99	37	4.83	7.53	15
100	33	0.28	5.06	15
101	75	0.72	3.90	23
102	83	0.05	0.05	21
103	56	0.06	0.06	6
104	79	0.06	0.06	24
105	76	0.11	0.11	32
106	60	0.55	0.66	14
107	78	0.06	0.06	19
108	61	0.88	1.54	29
109	60	0.99	3.02	22
110	50	0.22	0.22	25

* CPU time in seconds needed to find the first feasible solution with a project length corresponding to the optimal solution. The CPU timer has an accuracy up to 0.05 seconds: smaller times are shown as 0.00.

** Total CPU time in seconds (times below 0.05 are shown as 0.00)

*** Maximum value of p, the level in the search tree.

Characteristics	DAVIS	STINSON	TALBOT	HERROELEN & DEMEULEMEESTER
Number of problems solved (out of 110)*	96	110	97	110
Average time per problem solved **	14.02	0.82	14.98	2.75

* Within time limit imposed of 5 minutes (CPU time) per problem.

** Amdahl 470/V8 CPU time in seconds for DAVIS, STINSON and TALBOT; IBM PS/2 Model 70 A21 (25 MHz processor) CPU time in seconds for HERROELEN & DEMEULEMEESTER.

Table II.