Project Scheduling Under Constrained Resources

Fall 1980

Mohammed Benameur

University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation

https://stars.library.ucf.edu/rtd/465

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
PROJECT SCHEDULING
UNDER CONSTRAINED RESOURCES

BY

MOHAMMED BENAMEUR
B.S.E., University of Central Florida, 1978

RESEARCH REPORT

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering at the University of Central Florida; Orlando, Florida

Fall Quarter
1980
ABSTRACT

This report examines the widely acceptable Heuristic and Exact procedures for solving the problem of project scheduling and control under constrained resources. Heuristic approaches are more practical, however they depend on the type of the project as well as the resources involved.

Exact procedures are illustrated using an Integer Linear Programming formulation of the problem, and also solving it using the Branch and Bound Technique. Impracticality of the exact methods stews from the fact that the computations expand to an unmanageable amount.

Director of Research Report
Dr. Y. A. Hosni
ACKNOWLEDGEMENT

I would like to acknowledge the guidance of Dr. Yasser Hosni during the preparation of this report.
# TABLE OF CONTENTS

ACKNOWLEDGEMENT ........................................ iii

Chapter
I. INTRODUCTION ........................................ 1

II. THE CONTEXT ......................................... 4

III. HEURISTIC APPROACH ................................. 7

   Serial Scheduling .................................... 8
   Parallel Scheduling ................................ 8
   Priority Rules ...................................... 11
   Comparison of Rules ................................ 12
   Example of a Heuristic Model ...................... 13
   Brooks' Algorithm .................................. 16

IV. EXACT METHODS ..................................... 24

   Integer Linear Programming (ILP) ................. 24
   Branch & Bound Technique .......................... 31

V. CONCLUSION .......................................... 36

APPENDIX A ............................................. 38

LIST OF REFERENCES ................................... 40
CHAPTER I

INTRODUCTION

The popularity of the Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT) has proven that network models are useful means of formulating a wide variety of activity/scheduling problems. It has long been recognized, however, that these basic procedures are naive models of most real-life situations because they assume unlimited resource availabilities.

The procedures of project scheduling under resource considerations can be divided in three categories:

1 - Time/cost trade-off procedures
2 - Resource leveling procedures
3 - Constrained resource procedures

Time/cost trade-off procedures are directed at determining the least cost schedule for any given project duration. The premise of these methods is that the performance of some or all project activities can be accelerated by the allocation of more resources, at the expense of higher activity direct cost. These procedures are usually under the assumption of unlimited resources. The traditional CPM method is such a time/cost trade-off technique. Resource leveling procedures are used whenever it is desired to
utilize resources at a relatively constant rate, given that there are sufficient resources to schedule all concurrent jobs competing for the same resource types. The objective of the leveling process is to "smooth" as much as possible the profile(s) of resource usage over time, within the given project duration. The acceptable profile is judged accordingly to some predetermined criteria such as maximum utilization of resources, and the project duration is normally determined by critical path procedures and is not allowed to increase.

The third type of procedures deal with the constrained resource problem. This problem arises when the amount of a given resource available during a project is not sufficient to satisfy simultaneously the demands of the concurrent activities. This usually leads to sequencing decisions which often cause an increase in the critical path duration. The procedures available can be grouped into two categories. First, there are the heuristic procedures which use some rule of thumb or "heuristic" in order to produce a good schedule. The second category includes procedures which aim at producing the best possible, or optimal, schedule using rigorous mathematical analysis.

This research will concentrate on these procedures for solving the constrained resource problem. It is divided into three main parts. First, the problem is defined and the criteria that needs to be optimized are identified. The second part introduces the heuristic procedures and their applications. The third part examines the two main optimal procedures, namely the integer linear programming
and the branch and bound technique. Finally conclusions will be drawn.
CHAPTER II

THE CONTEXT

The problem of project scheduling under constrained resources stems from the fact that in industrial organizations, management, usually, has fixed amounts of each resource that it either cannot or does not desire to exceed. Resources in this case are manpower (i.e., labor, engineering, management), equipment (i.e., machinery, facilities, etc.) and capital in its different forms.

Using the activity network representation (see Fig. 1), the problem is completely defined when each activity has associated with it a) a duration, b) the quantity of the resource required, and c) the total available resource quantity.

Fig. 1. Network representation of a project

Fig. 1 is a network representation of a small one resource
type project. Each activity is represented by a directed arc which starts with a node and ends with a node. The first number on the arrow represents the duration of the activity. The number between parentheses is the amount of resource required to perform the activity. For example, activity (1, 2) takes 4 days for completion and uses 2 units of the resource. There are 4 units of resource available for the project. The example represents a small project, however real life networks easily reach the order of several hundred activities.

Once the project is represented as a network the next step for developing a scheduling procedure under limited resources is that of selecting a meaningful criterion to be optimized. Three such criteria are:

1 - Minimize the project slippage
2 - Maximize resource utilization
3 - Minimize in-process inventory

"Project slippage" of an individual project is that number of time units past a project due date or delivery date at which the project is completed. If a project is completed on or before its due date, no project slippage results. Minimizing project slippage is the most desirable objective of analysis. In fact, it is the most widely used objective among the majority of heuristic (and other) procedures. It is advantageous for the following reasons. First, project slippage which results in late deliveries incurs penalty costs which reduce profit. Second, the organization probably accepts new projects with their respective due dates based upon the expected completion times of projects already
in progress; therefore, slippage on one project may cause slippage on other projects. Then there are industries which are heavily customer oriented and where the avoidance of project slippage is the most important criterion.

Efficient resource utilization is another big concern for industrial organizations. Most resources imply a cost to a company whether they are in use or idle. Unused capital funds are not drawing interest; idle labor is not productive; machines must be paid out whether in production or not, it is difficult to name a single resource which does not have a cost for idle time. The problem lies in achieving efficiency of resource allocation while insuring that projects are completed on time. This problem arises because activities of each and all projects are competing in multi-project organizations for scarce resources and it is difficult to determine how to allocate the resources efficiently and such that delivery dates are met.

The third objective is to minimize the amount of in-process inventory or the amount of work which cannot be processed immediately due to a resource shortage. Inventory represents a sizable investment to most industrial organizations, and in-process inventory indicates a lack of efficiency if it exists in large amounts.

The next chapter introduces the heuristic techniques for solving the project scheduling under constrained resources.
CHAPTER III

HEURISTIC APPROACH

Heuristic scheduling procedures substitute mathematical analysis by logical decision rules. They lead to consistently good results, however, unlike analytical procedures, they do not guarantee optimality. In heuristic problem solving alternatives are evaluated following one or a set of rules. An example of such a decision rule used in solving a scheduling conflict is used as follows:

situation: Two or more jobs* are initially competing for the same scarce resource.

problem: A decision must be made as to which job will be processed first, thus delaying the other job(s).

rule: The "shortest operation first" (SOF) discipline will schedule first that job whose expected time duration is smallest. That job whose expected time duration is next smallest is scheduled next and so on. There is no mathematical proof that the SOF rule is the best/optimal choice, however, logically it might lead to a good schedule.

Two basic approaches can be used in resolving the scheduling of several resources; they are referred to as serial and parallel/

*job and activity are used concurrently in this part, they represent the same identity.
scheduling.

1-Serial Scheduling.

In this approach each activity is completely scheduled before considering the next. The activities are ranked according to a priority rule (see the third section of this chapter) and respecting the precedence relationships. They are then rescheduled in that order and also according to the availability of resources. This is illustrated in Figure 2 where the scheduling steps (1, 2, 3 and 4) involve complete activities, i.e., step 1-activity 10, step 2-activity 20, etc. The Figure shows that activities 10 and 20 have been scheduled completely. Activity 30 is being scheduled and its resource requirements (Resource 2 and Resource 3) are compared against the resources remaining after scheduling the previous 2 activities.

2-Parallel Scheduling

In this approach, all activity segments falling within a particular time step are considered together, i.e. in parallel. Within that time step, they are ranked according to a priority rule and compared against resource availability. Figure 3 illustrates the parallel scheduling. It shows that scheduling step number 6 has been reached (i.e. we are scheduling the sixth day if the time unit is one day) and the resources (i.e. Resource 1, Resource 2, etc.) will be assigned to activities 10, 20, 30 and 40 respectively and according to availabilities.
Fig. 2. Serial Scheduling

Fig. 3. Parallel Scheduling

When scheduling simple networks, there is a little difference between the serial and the parallel approach concerning the schedules produced (Woodgate 1977). However, significant variations can occur when larger and more complex networks are analyzed. Some particular types of complex resource scheduling problems are best solved by parallel scheduling and others respond more favorably to serial scheduling methods.

In general, large "thin" networks, i.e. those requiring few activities to be scheduled simultaneously, are best handled by the parallel scheduling method, whereas short "fat" networks, i.e. those with more simultaneous activities, are best analyzed by the serial scheduling procedure.

**Priority Rules**

Whether the parallel or the serial scheduling are used, the ranking of competing activities to be presented for scheduling should be performed according to a priority rule. Some of the most commonly used such priority rules are:

1-Shortest Operation First (SOF): the priority is given to the activity whose expected duration time is smallest.

2-Most Available Resources (MAR): the priority is given to the activity which requires the largest amount of available resources.

3-Most Succeeding Activities (MSA): the priority is given to the activity which controls the largest amount of succeeding activities.
4-Minimum Slack First (MSF): the priority is given to the activity with the minimum slack from the due date of the project.

5-First Come, First Served (FCFS): the priority is given to the activity with the earliest start time.

Any project scheduling model will use one of these priority rules in ranking the activities for scheduling. The effectiveness of these rules depends on the objective to be attained.

**Comparison of Rules**

Several studies have been conducted to evaluate the relative effectiveness of the various priority rules or sequencing heuristics.

One of these studies was conducted by T. L. Pascoe (Davis 1973). His approach was to artificially create thirty-two, 20 jobs, 3 resource networks, along with a real-life 90 jobs network. Each project was then scheduled repeatedly with combination of the heuristics, using both a parallel and serial approach. Six of the heuristics were commonly used rules (minimum slack first, etc.), one was a random choice rule and three were special, complex heuristics of his own devising. Five different objective functions were used to evaluate the schedule produced by each procedure. Pascoe concluded that in general:

- Parallel methods were better than serial methods,
- Best results were obtained using the heuristics of increasing LFT (Late Finish Time) or increasing LST (Late Start Time), the tie breaking rule being unimportant.
L. G. Fendley (1968) tested the effectiveness of eight different heuristics for scheduling multiple on-going projects. Two and five project combinations of eight test projects were used in the experiment, each having up to 20 jobs which required up to three different resource types. Eight criteria were used in ranking each heuristic. The minimum-slack-first rule ranked first by four criteria and was judged the best, in general, of the rules tested.

The results of this experiment are summarized below.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Best rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total system occupancy time</td>
<td>MSF</td>
</tr>
<tr>
<td>Total mean project slippage</td>
<td>MSF</td>
</tr>
<tr>
<td>Total maximum project slippage</td>
<td>MSF</td>
</tr>
<tr>
<td>Expected number of idle resources</td>
<td>MSF</td>
</tr>
<tr>
<td>Maximum number of idle resources</td>
<td>FIFO</td>
</tr>
<tr>
<td>Amount of work waiting</td>
<td>MSA</td>
</tr>
<tr>
<td>Expected number of jobs waiting</td>
<td>SOF</td>
</tr>
<tr>
<td>Maximum number of jobs waiting</td>
<td>SOF</td>
</tr>
</tbody>
</table>

The conclusions reached above can only give the potential user a general idea on which priority rule will be best with his own system objectives and performance measures. However, there is no evidence that any priority rule produces the best results for the solution of the project scheduling problem under constrained resources.

A Heuristic Model

The heuristics just discussed and others are "put together" in a comprehensive scheduling model developed by Wiest (1967) and
called Scheduling Program for Allocation of Resources (SPAR).

The program focuses on available resources which it serially allocates, period by period, to jobs listed in order of their early starting times. Activities are scheduled, starting with the first period, by selecting from the list of those currently available, and ordered according to their total float (based on technological constraints only and normal resource assignments). The most critical jobs have the highest probability of being scheduled first, and as many jobs are scheduled as available resources permit. If an available job fails to be scheduled in that period, an attempt is made to schedule it the next period. Eventually all jobs so postponed become critical and move to the top of the priority list of available jobs.

The basic flow diagram for SPAR is shown in Figure 4. The operation of the basic program just described is modified by a number of scheduling heuristics designed to increase the use of the available resources and/or to decrease the length of the schedule. These scheduling heuristics are:

1-Crew Size.

The program selects from three different crew sizes (minimum, normal and maximum) associated with each activity.

2-Augment Critical Jobs.

Crew sizes which are less than maximum and are assigned to critical activities are increased as much as possible.

3-Multiresources Activities.

When different resources are required for one activity,
Fig. 4. SPAR flow diagram*


* men = resource
crew size = amount of resource applied to a job
day = scheduling step
separate activities are created for each resource and are constrained to start at the same time with the same level of resource assignment.

4- Borrow from Active Jobs.

When resources available are insufficient for scheduling some critical activity, an attempt is made to borrow from currently scheduled jobs.

5- Reschedule Active Jobs.

Sometimes a critical job could be scheduled if an already scheduled job using the same resource is postponed.

6- Add-on Unused Resources.

If some resources are left, after scheduling as many jobs as possible, these resources are added to activities with the smallest slack.

SPAR is written in FORTRAN. On a 32k machine, the model may be dimensioned to handle a project with 1200 single-resource jobs.

A sample of the characteristics of some known computer programs are given in appendix A.

**Brooks' Algorithm**

Another heuristic procedure for solving the project scheduling under limited resources problem is the Brooks' Algorithm (BAG) (Bedworth 1973). Even though the original algorithm developed by Dr. Brooks could only handle single resource requirement, an extension for the multiresource case has been developed by Mason (Bedworth 1973).
The steps required to assign the single resource with BAG are as follows: (Table 1 gives the tabular results of these steps for the network in Figure 5 with 3 resources available).

1. Develop the project network as with the critical path procedure, identifying activities, their required time and required resources.

2. Determine for each activity the maximum time it controls through the network on any one path. This would be like calculating the critical path time through the network assuming that the starting time for each activity being analyzed is the network starting time. This activity control time will be designated by ACTIM and will be sealed from 0 to 100.

3. Rank these in decreasing ACTIM. Ties are ranked in any order. Now determine the following as in Table 1:
   - Duration and resources required for each activity as defined in step 1.
   - TEARL: the earliest it is possible, due to precedence and time limitations, to schedule each activity. The actual time will be equal to or later than TEARL. TEARL equals the latest TFIN time for all immediate predecessor activities.
   - TSTART is the actual start time of the activity. IF there are no resource limitations then TSTART would always equal TEARL.
-TFIN is the completion time of each activity. This equals TSTART added to the activity duration time.

-TNOW is the time at which resource assignments are now being considered.

4. Set TNOW at 0. The allowable activities (ACT. ALLOW.) to be considered for scheduling at TNOW of zero are those activities with TEARL of 0, namely activities 1-2, 1-3, 1-5. These are placed in ACT. ALLOW. row in decreasing ACTIM order. Ties are broken by scheduling the activity of longest duration first. In the resource available column the resources initially available, 3, are placed.

5. Determine if the first activity in ACT. ALLOW., 1-5 can be assigned. Activity 1-5 requires only one resource, and three are available, so 1-5 can be assigned. A line is struck through 1-5 to indicate assignment and the number of resources available is decreased by one. TSTART and TFIN are then set for activity 1-5. This same process is repeated for the remainder of the ACT. ALLOW. activities until the resources are depleted.

6. TNOW is raised to the next TFIN time of 5 which occurs at the completion of both activities 1-2 and 1-3. The resources available are now 2. ACT. ALLOW. includes those activities not assigned at the previous TNOW (in this case none) and those new activities whose predecessors have been completed (2-4, 3-4, 3-5).
### TABLE 1

BROOKS' ALGORITHM SOLUTION TO THE NETWORK IN FIGURE 5 WITH TREE UNITS OF RESOURCES

<table>
<thead>
<tr>
<th>Activity</th>
<th>1-5</th>
<th>1-2</th>
<th>1-3</th>
<th>3-4</th>
<th>2-4</th>
<th>3-5</th>
<th>4-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration (days)</td>
<td>16</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>ACTIM</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>ACTIM (scaled)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>69</td>
<td>50</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>Resources required</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TEARL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>TSTART</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>TFIN</td>
<td>16</td>
<td>5</td>
<td>5</td>
<td>12</td>
<td>9</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>TNOW</td>
<td>0</td>
<td>5</td>
<td>9</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource available</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ACT ALLOW</td>
<td>(1-5),(1,2),(1,3)</td>
<td>(3,4),(2,4),(3,5)</td>
<td>(3,5)</td>
<td>(4,5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iteration no.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5. Project network with one resource type

7. Repeat this assignment process until all activities have been scheduled. The latest TFIN gives the duration of the project, which is 17 for this example.

**Brooks' Algorithm for Multiple Resources**

One tabular computer-oriented approach developed as an extension of Brooks' algorithm gives an excellent simple heuristic approach to getting a good schedule, given multiple resources requirements.

It is a four step procedure, and as before, will be explained better using an example (network in Figure 6 with 2 resources required).

1. Test the resources requirements for each activity against resources available to see if any schedule is feasible. In the example problem, there are 3 units of resource A, and 4 units of resource B. The maximum requirement by any activity is 3 of each.

2. Compute ACTIM.

3. Rank the activities in decreasing ACTIM sequence. Ties will be broken using another heuristic such as "longest activity first" or "most resource requirement first".

4. Construct work table as in Table 3, and follow through solution. For example:

   **Time 0:** starting resource values of 3 and 4 are first given. Highest ACTIM activity 1-2 is scheduled which depletes resource A. Now, time is incre-
**TABLE 2**

WORKING TABLE FOR SOLUTION OF THE NETWORK FIGURE 6

<table>
<thead>
<tr>
<th>TIME</th>
<th>ACTIVITY</th>
<th>DURATION</th>
<th>START</th>
<th>FINISH</th>
<th>AVAILABLE RESOURCE A</th>
<th>AVAILABLE RESOURCE B</th>
<th>ACTIM-RANKED ALLOWABLE ACTIVITIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>1-2, 1-3</td>
</tr>
<tr>
<td>0</td>
<td>1-2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1-2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>1-3, 2-4</td>
</tr>
<tr>
<td>2</td>
<td>1-3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2-4</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1-3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
</tr>
<tr>
<td>5</td>
<td>2-4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>3-4</td>
</tr>
<tr>
<td>5</td>
<td>3-4</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3-4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>PROJECT COMPLETE</td>
</tr>
</tbody>
</table>

Duration (Resource A)(Resource B)

Fig. 6. Project network with two resource types.

mented to 2 the next immediate activity completion time.

Time 2: Activity 1-2 resources returned to available pool.
Next activity in ACTIM sequence 1-3 is scheduled, and the resources necessary are taken out of the pool. Time is then incremented to 3, activity 1-3 completion time. This overall process is continued until all activities have been scheduled.

Extension to Brooks' Algorithm

ACTIM, like any other heuristic, does not provide a good solution to the scheduling problem, all the time. This observation led researchers in the field to investigate other criteria to use with the Brooks' Algorithm. Three other possibilities have been proposed: ACTRES, TIMRES and GENRES. ACTRES incorporates both activity time and resource level in the control criteria. ACTRES is computed by taking the value of the activity time multiplied by the number of resource units for an activity plus the maximum of the ACTRES values following this activity. Again, after the ACTRES value is calculated for all of the network's activities, they are appropriately scaled from 0 to 100.

The TIMRES criteria is a combination of ACTIM and ACTRES. It is calculated by adding ACTIM and ACTRES, and again scaling all the TIMRES from 0 to 100.

GENRES is also a combination of ACTIM and ACTRES, however,
the two criteria are given different weights. This led Whitehouse (1980) to developing a search technique where different weighting are tried and the best project schedule is selected. A flow chart of the GENRES search model is presented in Figure 7.

Fig. 7. GENRES search model

CHAPTER IV

EXACT METHODS

Exact procedures, also termed optimal, use some form of mathematical programming or other rigorous analytical method in order to solve the project scheduling under constrained resources problem. In contrast to the tremendous efforts by both researchers and industry, which have gone into the investigation of heuristic methods, the development of optimal procedures has progressed relatively slowly.

These procedures can be classified according to whether they utilize (1) Integer Linear Programming or (2) enumerative techniques such as Branch and Bound.

**Integer Linear Programming (ILP)**

The very nature of the precedence relation between an activity and its successors indicates the "either-or" nature of the problem; either the activity is completed, hence its successors may start, or it is not completed, hence its successors cannot start. This in turn leads to integer programming models, in particular, 0,1 ILP models. Other such 0,1 variables are needed to indicate resource ceilings, to distinguish among resources, and so forth.
There have been several ILP formulations in the literature for solving the project scheduling problem. For example the Bowman (1959) formulation uses $0,1$ variables to indicate for each period over a scheduling horizon whether or not an activity is being processed.

The following formulation has been developed by Pritsker, Watters, and Wolfe(1969), it uses $0,1$ variables to indicate for select periods (depending upon job arrival time, due dates, sequencing relationships, etc.) whether or not a job is completed in those periods. In this description the word job stands for an activity and a project stands for a set of jobs. This formulation also accommodates the scheduling of multiple projects.

The following definitions are used in the formulation

- $i$ = project number, $i = 1, 2, \ldots, I$; $I$ = number of projects
- $j$ = job number, $j = 1, 2, \ldots, N_i$; $N_i$ = number of jobs in project $i$
- $t$ = time period, $t = 1, 2, \ldots, \max G_i$; $G_i$ = absolute due date
  Project $i$ must be completed in or before period $G_i$. IF an absolute due date is not specified, $G_i$ becomes the last period in the scheduling horizon.
- $g_i$ = desired due date. Project $i$ is not late if it is completed in or before period $g_i$.
- $e_i$ = earliest possible period by which project $i$ could be completed.
- $a_{ij}$ = arrival period of job $j$, project $i$. Arrival occurs at beginning of periods.
- $d_{ij}$ = number of periods required to perform job $j$ of project $i$. 
It is assumed to be known with certainty.

\( l_{ij} \) = the earliest possible period in which job j could be completed.

\( u_{ij} \) = the latest possible period in which job j could be completed (Absolute job due date).

\( k \) = resource number, \( k = 1, 2, \ldots, k \); \( k \) = number of different resource types.

\( r_{ijk} \) = amount of type k resource required on job j of project i.

\( R_{kt} \) = amount of type k resource required available in period t.

\( x_{ijt} \) = a variable which is 1 if job j of project i is completed in period t; 0 otherwise. \( x_{ijt} \) need not be treated as a variable in all periods, since it equals 0 for \( t < l_{ij} \) and for \( t > u_{ij} \).

\( x_{it} \) = a variable which is 1 in period t if all jobs of project i have been completed by period t (i.e., completed in or before period \( t - 1 \)); 0 otherwise. \( x_{it} \) need not be treated as a variable in all periods, since it equals 0 for \( t < e_{i} \) and 1 for \( t > G_{i} \).

To illustrate the above definitions, Fig. 8 shows the scheduling of five jobs belonging to 2 projects requiring 2 resources. The Figure depicts arrival periods, job duration due dates, precedence requirements and values of \( x_{ijt} \) and \( x_{it} \) variables.
Fig. 8. Hypothetical scheduling situation for two projects

There is one unit of resource available for each of the two types of resources; i.e., $R_{kt} = 1$ for $k = 1, 2$ and for all $t$. The resource requirements, $r_{ijk}$, for each job are assumed to be:

<table>
<thead>
<tr>
<th>Resource requirements $r_{ijk}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>$k$</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Objective Functions Formulations

Three objective functions are formulated they are:

1 - Minimizing total project throughput time

Total project throughput time is defined as the elapsed time between project arrival and project completion.

For project $i$, the total throughput time is

$$G_i = G_i - \sum_{t=e_i}^{G_i} x_{it} + 1 - a_i$$

(For example throughput times for project 1 and 2 in Figure 8 are 13 and 10 respectively). Minimizing throughput time is equivalent to maximizing the number of periods,

$$G_i \sum_{t=e_i}^{G_i} x_{it}, \text{ remaining after the project is completed.}$$
Therefore, the objective function of minimizing the sum of the throughput times for all projects can be written as

\[
\text{Maximize } z = \sum_{i=1}^{I} \sum_{t=e_i}^{G_i} x_{it} - (1/M) \sum_{i=1}^{I} \sum_{j=1}^{N_i} \sum_{t=1}^{L_{ij}} t x_{ijt},
\]

where the negative term is to insure that jobs are started as soon as possible without increasing throughput time. M should be positive and large enough to ensure that the contribution of the additional term is less than that of any \(X_{it}\).

\[
M > \sum_{i=1}^{I} \sum_{j=1}^{N_i} u_{ij}
\]

2 - Minimizing Makespan (time by which all projects are completed)

This can be accomplished by:

\[
\text{Max } z = \sum_{t=\max e_i} x_t - (1/M) \sum_{i=1}^{I} \sum_{j=1}^{N_i} \sum_{t=1}^{L_{ij}} t x_{ijt}
\]

where \(x_t = 1\) if all projects are completed by period \(t\)

\(= 0\) otherwise

The negative term is the same as defined in the previous function.

3 - Minimizing lateness penalty

This is accomplished by:
Max \( z = \sum_{i=1}^{I} \sum_{t=g_i+1}^{G_i} p_{it} x_{it} \)

where \( p_{it} \) = lateness penalty when project is not completed by period \( t \). A project is late if \( x_{it} = 0 \) in those periods where \( g_i < t \leq G_i \).

**Constraints Formulations**

The formulation can accomodate several constraints. Some of these are:

1 - Sequencing

When job \( m \) must precede job \( n \), both belonging to project \( i \), the appropriate constraint is:

\[
\sum_{t=1}^{u_{im}} t x_{int} + d_{in} \leq \sum_{t=1}^{u_{in}} t x_{int}
\]

2 - Resource constraints

A job is being processed in period \( t \) if the job is completed in period \( q \) where \( t \leq q \leq t + d_{ij} - 1 \). Therefore the resource constraints can be written as

\[
\sum_{i=1}^{N_i} \sum_{j=1}^{t+d_{ij}-1} \sum_{q=t}^{R_{ikt}} r_{ijk} x_{ijq} \leq R_{kt}
\]

\((t = \min a_{ij}, \ldots, \max G_i, k = 1, 2, \ldots, k)\)

Implementation of this constraint necessitates recognizing predetermined values of \( x_{ijt} \). (Namely, \( x_{ijt} = 0 \) for \( t < l_{ij} \) and for
Additional constraints such as substitutability of resources, nonconcurrency of jobs, job splitting, job and project completion are also developed to satisfy a wide range of environmental constraints.

As an application to the 0,1 ILP formulation discussed above, a 3-project problem consisting of a total of 8 jobs and 3 resources was formulated and then solved with a 0,1 ILP Code. The objective function considered was the minimization of the total throughput time. The problem was also solved using several heuristic sequencing rules. These rules gave suboptimal solution, 20 percent or more higher than the optimal solution produced by the 0,1 ILP formulation.

Branch & Bound

The second class of procedures for solving optimally the project scheduling under resource constraints consists of models that sift through all the possible forms that a project may assume, searching for an optimal solution. These models typically utilize some implicit enumeration approach and stand out as offering the only bright prospect, at the present time of operational utility (Elmaghraby 1977). The Branch and Bound method (B & B) belongs to this class of procedures.

The approach of B & B is basically a tree search in which the space of feasible solutions is systematically searched for the optimum. To illustrate the procedure we examine the B & B algorithm
developed by Hastings (1972).

The Algorithm is described using the network of Fig. 9 which describes the activities of a small project. A tree is developed in which each node represents a sub-problem similar to the original project but with some activities wholly or partly completed. Each branch of the tree represents a set of activities in progress. Searching over the tree leads to an optimal solution.

![Project network for the branch and bound example](image)

**Fig. 9. Project network for the branch and bound example**

### TABLE 3

**PROJECT DATA**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Start node</th>
<th>End node</th>
<th>Duration (days)</th>
<th>Resources (men)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Resources available = 4 men
The search tree for the example is shown in Fig. 10. The circles are the nodes of the tree and the upper number in each node is the node number. Node 1 of the tree represents the original project. The project might start with any of three sets of activities, namely, activity 1, activity 2, or activities 1 and 2 together. The latter is ruled out by the resource constraint; the other are represented by lines or branches emanating from node 1. Numbers against a branch indicate activities in progress for that branch. Node 2 of the tree corresponds to a subproblem which is the original project less activity 1 and node 3 corresponds to the original project less activity 2. The number $t_j$ in the bottom right of node $j$ indicates the time at which subproblem $j$ can start.

Fig. 10. Search Tree
The number $b_j$ in the bottom left of node $j$ is a lower bound on the completion time of the whole project given that subproblem $j$ remains at time $t_j$.

The lower bounds on the completion times are computed by assuming that there are no resource constraints. Thus for node 1 of the tree the lower bound is the completion time of the original project without resource constraints. This is 7 days, so that $b_1 = 7$, $t_1 = 0$. If activity 1 starts at time 0 then at time 1 subproblem 2 remains. The unconstrained completion time for subproblem 2 is 6 days so $t_2 = 1$, $b_2 = 7$. If activity 2 starts at time 0 then at time 2 subproblem 3 remains. The unconstrained completion time for subproblem 3 is 7 days so $t_3 = 2$, $b_3 = 9$.

Node 2 has a smaller lower bound than node 3 so it is processed next. For subproblem 2 the feasible sets of initial activities are $\{2\}, \{3\}, \{3, 4\}, \{4\}$. These sets are represented by branches which lead to nodes 4, 5, 6 and 7 respectively. Where several activities run together the branch finishes when the shortest activity is complete. Thus subproblem 6, corresponding to node 6, has activities 1 and 3 finished and activity 4 started but with 2 days work remaining. It is assumed that activity 4 must continue when subproblem 6 is started.

Having processed node 2 of the tree there is a choice of nodes to branch on. In practice it is convenient to branch on a node which is a successor to the node just processed if this is possible. We choose the successor node with the least lower bound. It so
happens that nodes 4, 5 and 6 all have lower bounds of 9 days so we resolve the tie by choosing the node whose preceding branch has the largest number of activities in progress. This tactic has the practical advantage of giving preference to schedules in which activities are completed sooner rather than later. Node 6 is selected.

The only feasible initial activity for subproblem 6 is activity 4. Completion of activity 4 leads to subproblem 8 which consists of activities 2 and 5 only. Completion of activity 2 leads to subproblem 9 which consists of activity 5. Completion of activity 5 finishes the project and leads to node 10. This is a terminal node corresponding to a project duration of 11 days.

We now search back up the tree and examine the unprocessed nodes. Node 7 has a lower bound of 12 days and must be suboptimal. Node 5 has a lower bound of 9 days; however, its successor nodes have bounds greater than or equal to 11 days and are therefore ruled out. Processing of node 4 leads to a succession of nodes 11, 12 and 13, the last of which is a terminal node with a project duration of 9 days. No unprocessed node has a lower bound less than 9 days so this solution is optimal. The schedule is to start activity 1 at time 0, activity 2 at time 1, activity 3 and 4 at time 3 and activity 5 at time 6, completing the project in 9 days.

An algorithm based on the principles outlined above has been programmed in Fortran (Hastings 1972) and the author gives optimal solution to a 20-jobs problem.
CHAPTER V

CONCLUSION

Project scheduling techniques under constrained resources are used whenever the resources available for the completion of the project are limited and cannot be exceeded. The primary objective of the procedures is then to minimize the project duration which is bound to increase beyond the duration of the critical path for the same project with unlimited resources.

Heuristic procedures, that use a priority rule to solve the conflict arising between two or more jobs competing for a limited resource, give relatively good schedules. These methods have been the basis for all practical scheduling systems used by industrial organizations. These heuristic based systems are generally in the form of large, often complex computer programs, capable of scheduling the largest projects imaginable under almost any desired conditions of resource usage and availabilities. However, there is no evidence that anyone of the "heuristics" produces the best schedules all the time. The effectiveness of the different heuristics depend on the overall objective to be achieved.

In contrast to the heuristic approach, the exact procedures
aim at producing the optimal schedule. These exact methods use either integer linear programming or enumerative techniques such as branch and bound. The scheduling problem under limited resources can be formulated using either methods and the optimal solution can be reached, but only for small problems (maximum of 50 jobs).

There is no optimal procedure that has been demonstrated as computationally-feasible for the sorts of large, complex projects which occur in practice. At this point most of the researchers in the field conclude that this situation will change, as the computing efficiency of computers increases.
APPENDIX A

COMPUTER PROGRAMS

Literally hundreds of elaborate heuristic-based computer programs have been developed. However, the operating details of the majority of these are not available because they were developed by organizations for their own or outside use on a propriety basis. A sample of the characteristics of some of the programs which are available on a commercial basis are given below.

CPM-RPSM (Resource Planning and Scheduling Method)

This program was developed by CEIR, Inc. It can handle up to 8000 jobs per project, four resource types per project and 26 total variable or constraint resource limits. Job splitting and job start/finish constraints are also allowed. The program uses fixed scheduling heuristic.

MSCS (Management Scheduling and Control System)

This program developed by McDonnell Automation can handle multi-projects (up to 25 projects) with a maximum of 18,000 activities and 12 resource types per activity. The scheduling heuristics are based on complex priority rules and are controllable by the user. Many flexible assumptions of job conditions are available. The program also includes project costing and report generation.
PMS/360 (Project Management System)

This program was developed by IBM. It is a large complex management information system consisting of 4 main modules (of which resource allocation is one). It can handle up to 225 multiple projects with a total of up to 32,000 activities and up to 250 resource types. The program gives the user the choice between several sequencing heuristics, and includes many costing features and report options.

PPS IV (Project Planning System)

This program developed by Control Data Corporation, can handle up to 2000 jobs per project and up to 20 resource types per job. It allows resource costing and progress reporting, and uses only one fixed-heuristic procedure.

Project/2

This program was developed by Project Software Inc. It allows up to 50 multiple networks with up to 32,000 jobs and several hundred resource types. The user has the choice between several sequencing heuristics and the program includes many cost analysis features.

RAMPS (Resource Allocation and Multiproject Scheduling)

This program developed by the DuPont Company is probably the first major system for the constrained resource scheduling problem. It can handle up to 100 separate projects each consisting of up to 2,000 activities and requiring up to 100 different resource types. The program includes many costing features and is still widely used within the DuPont Company.
LIST OF REFERENCES


