European Journal of Operational Research 72 (1994) 406–416
North-Holland

Theory and Methodology

A comparative study of computational procedures for the resource constrained project scheduling problem

Osman Oğuz and Hasan Bala
Department of Industrial Engineering, Bilkent University, 06533 Ankara, Turkey

Received September 1991; revised January 1992

Abstract: Performance of two new integer programming based heuristics together with some special purpose algorithms for project scheduling are tested from a computational point of view. The objective of the study is to compare the quality of solutions obtained by using these algorithms and reach conclusions about their relative merits on this specific problem.

Keywords: Project scheduling; Integer programming; Heuristics

1. Introduction

The resource constrained project scheduling problem is an important and challenging problem for both practitioners and mathematicians. It is important because it has a wide variety of application areas such as design of production facilities, installation of computer systems, large scale construction projects, scheduling of radio and television broadcasts, new product development, etc. It is challenging because it is an NP-complete problem [16].

The basic problem is to schedule the activities of a project so that none of the resource constraints, nor any of the predecence relationships, is violated, with the objective of minimizing the total completion time. Holloway et al. [12] provide a classification of resource constrained project scheduling problems using the number of resource types, resource availabilities and resource requirements. The specific problem in this study is the multiple resource constrained, single project scheduling problem which falls into the category of type $n/n/n$, according to Holloway's [12] notation, in which the n's stand for multiple resource types, multiple units of resources, and multiple number of resource types required by an activity, respectively. It is assumed that an activity cannot be interrupted once begun (non-preemptive case). Further, both the resource availabilities and the resource consumption are assumed to be stationary, that is, they remain constant throughout the project duration. The 0–1 formulation of the problem, modified from Pritsker et al. [23], is given in Section 3.1.

Our intention is to give a computational comparison of two new heuristics with some existing ones. Since the new ones are integer programming based, the choice of special purpose heuristic algorithms for

Correspondence to: Dr. O. Oğuz, Department of Industrial Engineering, Bilkent University, 06533 Ankara, Turkey.

SSDI 0377-2217(92)00133-3

comparison adds another dimension to the study by providing a possibility of comparing performance of general versus special purpose algorithms to some extent.

2. Resource constrained scheduling techniques

During the past three decades, since the pioneering work of Kelley [14] and Wiest [30] on the resource constrained project scheduling problem, there have been more than 80 publications and Theses that have investigated various versions of this challenging problem. See Boctor [2] for a classification of these versions according to differing criteria. The techniques aimed at solving this problem, on the other hand, can be categorized into two major groups:

(i) Heuristic, or approximate procedures which are designed to produce good resource-feasible schedules,
(ii) Optimization techniques to obtain the best schedule.

2.1. Heuristic approaches

The inherent difficulty of the problem due to its complexity brings the application of heuristic procedures for obtaining a solution predominantly into the forefront. The earliest studies in this vein have appeared in the 1960s. They were given by Wiest [31] and Fendly [10]. There were many attempts at making comparative studies between existing heuristics and between the heuristics and the optimal solution procedures [1,2,4,5,7,9,10,12,13,15,18,19,29,31]. We use an algorithm given by Davis and Patterson [7] in three slightly differing forms in this study.

More recently, Bell et al. [1] introduced a novel approach to resolve resource conflicts. Khattab et al. [13] developed eight priority rules with the aim of finding the shortest duration. There are also many commercially available computer based heuristic routines. They are often quite elaborate and copyright protected.

2.2. Optimization approaches

Early attempts concentrated on formulation and solution of the problem as a mathematical (usually integer) program. Pritsker et al. [23] gave the first integer programming model. The number of variables of the model increases very rapidly with the problem size. This is a major drawback, considering the capacity of existing integer programming algorithms to obtain a solution. Numerous specialised (usually enumerative) approaches for solving certain versions of this problem optimally were developed [3,6,11,22,24–28].

Patterson [21] presented a comparative study between three approaches [6,25,26] with the reminder that each of the approaches evaluated in his study represents the state of the art in its respective area. The techniques investigated were the Bounded Enumeration Algorithm of Davis [6], Stinson's [25] Branch and Bound Procedure and the Implicit Enumeration Algorithm of Talbot [26]. They differ in such aspects as the order in which candidate problems are considered for evaluation and the methods used to identify and discard inferior partial schedules.

Christofides et al. [3] developed a branch and bound algorithm which uses four different lower bounds to reduce the search tree. Fisher's [11] formulation in 0–1 variables employs Lagrangean relaxation to obtain bounds for a branch and bound algorithm. He relaxes the resource constraints using Lagrange multipliers. Although the relaxed problem is easy to solve, the problem of determining Lagrange multipliers is not.

Recently, Deckro et al. [8] presented a decomposition approach that offers two distinct advantages: (1) the ability to solve large problems realistically, (2) the option of using decomposition approach as a heuristic.

The above and the remaining undiscussed analytical techniques provide optimal solutions for small problems. However, their computational requirements for even moderate size problems are prohibitive.

3. Problem formulation and solution procedures

In this section, we first provide the 0–1 formulation of the problem together with the underlying assumptions. The solution procedures that are tested in this study will be discussed in Section 3.2.

3.1. Problem formulation

The formulation given here is modified from Pritsker et al.'s [23] through the following assumptions:

- single project consisting of a given set of activities,
- no activity can start unless all its predecessors are completed,
- job splitting is not allowed – nonpreemptive case,
- limited multiple resources,
- resource availabilities and resource consumptions are constant over the scheduling horizon,
- no substitution between resources,
- The project due date is set to a predetermined multiple of unconstrained critical path duration, i.e., the length of the critical path when the resource levels are set to infinity.

The definitions related to the formulation are as follows:

Definitions:

Indices:

j: Activity index; $j = 1, 2, \ldots, N$; N = number of activities in project.
k: Resource index; $k = 1, 2, \ldots, K$; K = number of different resource types.
t: Time period; $t = 1, 2, \ldots, T$; T = project due date.

Problem parameters:

d_j: Duration of activity j.
r_{jk}: Amount of resource type k required by activity j.
R_{kt}: Amount of resource type k available in period t.
l_j: Earliest possible period by which activity j could be completed.
u_j: Latest possible period by which activity j could be completed.
H_j: {set of all immediate predecessors for activity j}.

Note that l_j- and u_j-values are computed through the CPM calculations.

Decision variables:

$$x_{jt} = \begin{cases} 1 & \text{if activity } j \text{ is completed in period } t \\ 0 & \text{otherwise} \end{cases}$$

Note that x_{jt} need not be treated as a variable in periods where $t < l_j$ and $t > u_j$. So the problem is formulated as below.

Objective function: The choice of an appropriate performance measure differs for various scheduling environments. In this study, the objective is chosen to minimize the total of activity completion times. That is, the activities in the project are tried to be scheduled as early as possible. So, the objective function is to minimize

$$z = \sum_{j=1}^{N} \sum_{t=l_j}^{u_j} t x_{jt} \tag{1}$$

Constraints:

Activity completion: Each activity has exactly one completion period:

$$\sum_{t=l_j}^{u_j} x_{jt} = 1, \quad j = 1, 2, \ldots, N. \tag{2}$$

Notice that in each constraint, the value of any one x_{jt} can be determined by the values of the others in that constraint. To use this relationship to full advantage, constraint (2) is replaced by

$$\sum_{t=l_j}^{u_j-1} x_{jt} \leq 1 \tag{2'}$$

where $x_{j(u_j)} = 1 - \sum_{t=l_j}^{u_j-1} x_{jt}$. So, by replacing $x_{j(u_j)}$ by its definitional equivalence, the total number of variables in the formulation is reduced by N.

Precedence relationships: Assume that activity n must precede activity n. Let t_m and t_n denote the completion periods of activities m and n, respectively. Then

$$t_m + d_n \leq t_n \quad \text{where } t_m = \sum_{t=l_m}^{u_m} t x_{mt} \text{ and } t_n = \sum_{t=l_n}^{u_n} t x_{nt}.$$

So, the constraint becomes

$$\sum_{t=l_m}^{u_m} t x_{mt} + d_n \leq \sum_{t=l_n}^{u_n} t x_{nt} \quad \forall m \in H_n. \tag{3}$$

Resource constraints: In any period, the amount of resource k used by all activities cannot exceed the available resource k. An activity j is being processed in period t if the activity is completed in period q where $t \leq q \leq t + d_j - 1$. So, the resource constraints are written as

$$\sum_{j=1}^{N} \sum_{q=t}^{t+d_j-1} r_{jk} x_{jq} \leq R_{kt} \quad t = l_j, \ldots, u_j, \quad k = 1, 2, \ldots, K. \tag{4}$$

Implementation of this constraint necessitates recognizing the values of l_j and u_j.

The above formulation has N activity completion, $K \cdot T$ resource and $\sum_{j=1}^{N} o\,|H_j|$ precedence constraints. So, this makes a total of $N + \sum_{j=1}^{N} |H_j| + K \cdot T$ constraints. Due to the advantage of constraints (2'), N variables are dropped from the formulation. Moreover, the variables in periods $t < l_j$ and $t > u_j$ are priorily set to 0, and the number of variables becomes $N \cdot T - N - \sum_{j=1}^{N} (T - u_j + l_j - 1)$. This is equal to $\sum_{j=1}^{N} (u_j - l_j)$.

It is noted that due to replacing $x_{j(u_j)}$ its definitional equivalence, the objective function row takes the form

$$z = \sum_{j=1}^{N} \sum_{t=l_j}^{u_j-1} t x_{jt} + \sum_{j=1}^{N} u_j \left(1 - \sum_{t=l_j}^{u_j-1} x_{jt}\right) = \sum_{j=1}^{N} \sum_{t=l_j}^{u_j-1} (t - u_j) x_{jt} + \sum_{j=1}^{N} u_j.$$

Then it has a constant value of $\sum_{j=1}^{N} u_j$. This constant is not considered during the execution of the heuristics; the objective functions are adjusted accordingly after finding the solutions.

3.2. Solution procedures

The solution procedures tested in this study are the following:

(i) the MIXED-heuristic based on a penalty function method;

(ii) the MINSLK, SDFIRST, and LDFIRST-heuristics which are the applications of the Minimum Job Slack Rule given in [7], with three alternative prioritizing rules;

(iii) SCICONIC [1] Optimization Software which is used to solve a reduced version of the integer programming problem given in the preceding section.

The MIXED heuristic. The MIXED heuristic is a general purpose algorithm designed to solve integer programming problems of the form

$$\text{Max} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \forall i = 1, \ldots, m$$

$$0 \le x_j \le u_j \quad \forall j = 1, \ldots, n \text{ and } x_j \text{ integer.}$$

The problem under consideration is a special case of this from where all upper bounds, u_j, are 1. In the execution of the algorithm, it is first transformed into an unconstrained minimization problem. Three slightly different versions of this transformation are shown below:

Version 1:

$$F(\bar{x}) = \left(z^{u} - \sum_{j=1}^{n} c_j x_j \right)^2 + \sum_{i=1}^{m} p_i \left(\text{Min}\left[0, \, b_i - \sum_{j=1}^{n} a_{ij} x_j \right] \right)^2 + \sum_{j=1}^{n} v_j \left(\text{Min}[0, \, x_j] \right)^2 + \sum_{j=1}^{n} w_j \left(\text{Min}[0, \, u_j - x_j] \right)^2. \tag{1}$$

Version 2:

$$F(\bar{x}) = -\left(\sum_{j=1}^{n} c_j x_j \right)^2 + \sum_{i=1}^{m} p_i \left(\text{Min}\left[0, \, b_i - \sum_{j=1}^{n} a_{ij} x_j \right] \right)^2 + \sum_{j=1}^{n} v_j \left(\text{Min}[0, \, x_j] \right)^2 + \sum_{j=1}^{n} w_j \left(\text{Min}[0, \, u_j - x_j] \right)^2. \tag{2}$$

Version 3:

$$F(\bar{x}) = -\left(\sum_{j=1}^{n} c_j x_j \right) + \sum_{i=1}^{m} p_i \left(\text{Min}\left[0, \, b_i - \sum_{j=1}^{n} a_{ij} x_j \right] \right)^2 + \sum_{j=1}^{n} v_j \left(\text{Min}[0, \, x_j] \right)^2 + \sum_{j=1}^{n} w_j \left(\text{Min}[0, \, u_j - x_j] \right)^2. \tag{3}$$

Here, z^u is the upper bound on the value of the objective function of the original problem, and p_i, v_j and w_j denote penalty weights on violation of constraints, nonnegativity requirements, and upper bounds, respectively. We have used the third version only because the initial tests indicated that it provides solutions in shorter time.

The MIXED heuristic tries to minimize the function given in (3) which is a convex nonsmooth quadratic function. Although the starting point can be chosen randomly, starting from the origin seemed comparatively better. We try to minimize (3) using descent directions based on gradients (subgradients), and n-dimensional search. Penalties are used to attain feasibility. We divide the penalties, p_i's, for constraints into three parts such that:

$p_1, \ldots, p_{jc} = \text{PC}_1$: Penalties corresponding to activity completion constraints;
$p_{jc+1}, \ldots, p_{jc+pc} = \text{PC}_2$: Penalties corresponding to precedence relationships constraints;
$p_{jc+pc+1}, \ldots, p_{jc+pc+rc} = \text{PC}_3$: Penalties corresponding to resource constraints,
where $jc = N$, $pc = \sum_{j=1}^{N} |H_j|$ and $rc = K \cdot T$.

[1] Copyright SCICON Ltd.

The choice of these weights PC_1, PC_2, PC_3, v_j and w_j's together with the strategies to reach a feasible solution will be discussed in Section 4.3.

At any point $\bar{x}^k$, the value of the function $F(\bar{x})$ is calculated at $\bar{x} = \bar{x}^k - \nabla F(\bar{x}^k)$. If it decreases, $\bar{x}^k$ is replaced by $\bar{x}^{k+1}$ and the same process is repeated at the new point. Otherwise, a norm reduction operation on $\nabla F(\bar{x}^k)$ is performed by dividing all components of this vector by its smallest (in absolute value) component which is greater than one, and then rounding down the divisions to the nearest integers. Then this new vector, the subgradient, is used in the same manner as above. In case of failure, the norm reduction process is repeated to find a new subgradient. If all of the components of the resulting subgradient are equal to zero or one, then the MIXED heuristic activates an n-dimensional search. During the n-dimensional search, all $2n$ corners of the n-dimensional cube in which the starting point is the center of the cube are tested. If $F(\bar{x})$ decreases at any of these points, the starting point is moved to this new point and the whole process is repeated at this point. If the MIXED heuristic cannot improve through searching all $2n$ points, it stops and outputs the starting point of the n-dimensional search as the best solution achieved.

For a clearer understanding, a stepwise description of the algorithm is given below:

Step 1. Start with the origin, $\bar{x}^0$.
Step 2. Set $k = 0$ and compute $F(\bar{x}^k)$.
Step 3. Compute the gradient $\nabla F(\bar{x}^k)$.
Step 4. Set $\bar{x}^{k+1} = \bar{x}^k - \nabla F(\bar{x}^k)$.
Step 5. If $F(\bar{x}^{k+1}) < F(\bar{x}^k)$, set $k = k + 1$ and go to Step 3.
Step 6. Determine the smallest, in absolute value, nonzero element of $\nabla F(\bar{x}^k)$ which is greater than one. If there is no such element, go to Step 7. Otherwise, divide all components of $\nabla F(\bar{x}^k)$ by the absolute value of this element and round down the divisions to the nearest integer and go to Step 4.
Step 7. Set $i = 1$ and start n-dimensional search.
Step 8. Set $\bar{x}_i^{k+1} = \bar{x}_i^k + 1$ and all other $\bar{x}_j^{k+1} = \bar{x}_j^k$, $\forall j$, except $j = i$. If $F(\bar{x}^{k+1}) < F(\bar{x}^k)$, set $k = k + 1$ and go to Step 3.
Step 9. Set $\bar{x}_i^{k+1} = \bar{x}_i^k - 1$ and all other $\bar{x}_j^{k+1} = \bar{x}_j^k$, $\forall j$, except $j = i$. If $F(\bar{x}^{k+1}) < F(\bar{x}^k)$, set $k = k + 1$ and go to Step 3.
Step 10. Set $i = i + 1$. If $i > n$, output $\bar{x}^k$ as the best solution and stop. Otherwise go to Step 8.

The rounding down procedure in Step 6 guarantees that the solution is integral if the initial problem data and the starting point are integers.

MINSLK, SDFIRST and LDFIRST heuristics. These heuristics are based essentially on three rules:

(i) Resources are allocated serially in time. That is, start on the first day and schedule all activities possible, then advance to the next day and do the same.
(ii) In case more than one activity compete for the same resources, rank them in the order of minimum activity slack (MINSLK), or shortest duration (SDFIRST), or longest duration (LDFIRST).
(iii) Reschedule noncritical activities, if possible, so as to free resources for scheduling critical activities – those activities which have no slack (not applied in SDFIRST and LDFIRST).

One can refer to references [7,20,31] to get more details on these rules. We omit giving the descriptions of the algorithms here for the sake of brevity. The algorithm design is quite straightforward once these rules are well understood.

The optimizing software: SCICONIC

SCICONIC/VM V1.47 is used on a Data General MV/2000 to solve the integer programming model. This is a commercial package designed to solve linear and nonlinear programming problems. A branch and bound subroutine is embedded in it to find solutions for integer programming problems.

In order to have integer solutions by SCICONIC in a reasonable amount of time, the 0–1 formulation of the problem was reduced in size as mentioned before. The reduction process is based on the fact that the problem size is dependent on activity duration times, d_j. In the reduction process, the time unit is scaled down by a predetermined factor. It is achieved through dividing activity durations by a predetermined scale factor keeping other input parameters constant in the generation process of the test problem. Then, the scaled problem has a smaller number of variables and constraints. In order to keep the integrality of the data, the divisions are truncated to the nearest integer. After finding a feasible integer solution by SCICONIC, the value of the objective function is updated by a rescaling process. We note here that this may result in some information loss which leads to a superoptimal objective function value, as well as the possibility of having no feasible integer solution. However, for low values of the scaling factor this effect seemed insignificant. As is explained in more detail in the next section, we have used a scaling factor of two, and ignored the effect of rescaling completely.

4. Experimentation and results

4.1. Problem generation

In order to test the performance of solution procedures, a problem generator called NGNR is coded in Pascal. A standard random number generator is embedded in the procedure which performs the following steps to generate a problem:

Step 1. Read the following input parameters that are grouped into two:
- (i) Set of parameters related to problem characteristics:
  - * number of nodes, called nnode, in the project where a node represents the start or end of a activity;
  - * upper limit on the number of preceding activities in the project;
  - * upper limit on the number of succeeding activities in the project;
  - * number of different resource types;
  - * lower and upper limits on the resource consumption rates;
  - * range for the duration of activities;
  - * resource tightness ratio, called rt, that is used to calculate the resource avaliabilities;
  - * critical path duration multiplier, called dm, that is used to set the absolute due date, T;
  - * scale parameter, called Scale, that is needed to reduce the problem in size for the solution procedure, SCICONIC.
- (ii) Set of parameters used by NGNR:
  - * initial seed for the random number generator;
  - * upper bound on the number of iterations that is needed to produce incidence matrix, called niter.

Step 2. Generate the incidence matrix in niter number of iterations. The incidence matrix, called NETWORK, is an nnode × nnode upper triangle matrix that is used to define the precedence relationships in the project network. For example, if there is an activity whose starting event is i and ending event j, then NETWORK[i, ju] is set to 1. Activities are represented by arcs whereas events are by nodes in the project network. All arcs are unidirected, with the head showing its starting node and tail representing its ending node. The project has only one starting and ending node.

Step 3. Update the incidence matrix in order to have a complete project network, i.e., append new arcs emanating from unconnected nodes to the ending node.

Step 4. Determine the following for each generated arc:
- its starting node;
- its ending node;

Table 1
The input range for parameters

Parameter name	Input range
# of nodes in project	10– 15
# of ingoing arcs	2– 4
# of outgoing arcs	2– 4
Time duration for activities	2– 12
# of resource types	3– 6
Resource consumption rates	1–180

- activity duration time, d_j. Note that the reduction process discussed in Section 3.2.3 is achieved through dividing the duration time by the scale parameter, Scale;
- each resource requirement, r_{jk};
- the predecessor set, i.e., all the activities that precede the ongoing activity;
- the successor set which is all the activities that succeed the ongoing activity.

Step 5. Perform CPM calculations which results in computed values of:
- early start and finish time;
- latest start and finish time;
- total and free slack-used only by MINSLIK, SDFIRST and LDFIRST heuristics;
- critical path duration;
- absolute due date, T, that is set to critical path duration times dm.

Step 6. Determine the peak resource requirements.

Step 7. Compute the resource availability for each resource type, R_{kt} through multiplying peak resource requirement by the resource tightness ratio, rt.

Step 8. Make necessary arrangements in order to output the problem in the form of formulation that was discussed in Section 3.1.

The input ranges for parameters are tabulated in Table 1.

4.2. Design of analysis

As discussed in Section 4.1, there are two parameters involved in the generation process, namely the resource tightness ratio and the critical path duration multiplier. The experiment is carried out with different combinations of these parameters. Each combination consists of several problems generated using a different seed. The arbitrariness in the number of problems in these combinations is accidental, i.e., no effort was made to make them equal. The nature of the heuristics we are proposing is the reason for this irregularity. When reaching a feasible solution seemed to require an inordinate amount of time,

Table 2
List of different combinations of rt and dm

Combination #	rt	dm	# of problems
1	$\frac{2}{3}$	1.5	13
2	$\frac{2}{3}$	1.7	2
3	0.5	2.0	7
4	0.6	1.8	8
5	0.7	1.6	10
6	0.7	1.7	9

we abandoned the test problem in favor of a newly generated one. To get the statistics for 49 tests, we have run approximately 150 trials. A list of the combinations and related data are give in Table 2.

The feasibility of the problem depends on two factors. As the resource tightness ratio increases, the problem becomes more difficult. Also, the critical path duration multiplier is aimed at relaxing the time constraint. As it increases, the feasibility of the problem becomes more likely.

The test problems have activities ranging from 16 to 36. The number of constraints in the formulated problems changes between 114 and 397, and the number of variables is between 241 and 969. The density of the problems ranges from 0.071 to 0.210. Before running SCICONIC, the reduction process is carried out. The scaling operation reduces the number of variables for each problem by one-half because the scaling factor is equal to 2 for all test problems.

4.3. Results of the study

As stated in Section 3.2, the MIXED heuristic works with a penalty function. It involves choice of penalties associated with job completion, precedence relationships, nonnegativity and upper bound constraints in order to find a feasible solution for the problem. The following strategy in setting penalty weights in executing the MIXED heuristic is implemented:

The penalties, v_j and w_j corresponding to nonnegativity and upper bound constraints are set at least twice greater than the penalties assigned to job completion constraints, PC_1. They are kept equal in all test problems ($v_j = w_j$, $\forall j$). The penalties for precedence relatonships and resource constraints, PC_2 and PC_3 respectively, are always less than the penalty weight associated with job completion constraints. PC_3 is experimentally selected by taking the combination of resource tightness ratio and critical path duration multiplier into account. It is at most equal to PC_2. If the MIXED heuristic solution results in violated constraints with the ongoing set of penalties, the penalty weights corresponding to violated ones are updated and the heuristic is executed again.

In the case of SCICONIC software use, we had to introduce some stopping criteria based on the elapsed time and closeness to the linear programming lower bound to end further branchings. An integer solution that has a objective value within 90% proximity of its corresponding linear optimum solution, excluding the objective constant, $\sum_{j=1}^{N} u_j$, discussed in Section 3.1, is taken to be sufficient. Moreover, maximum CPU time (in seconds) allowed is constrained to 18000 seconds and maximum number of iterations to 200000.

Out of 49 problems, the optimizing software SCICONIC could not find a feasible integer solution for 13 problems in the allowable time and number of iterations. For those problems in combination #3, that are tightly resource constrained, it does not provide any feasible integer solution. It is interrupted in 27 problems in order not to let further branching occur. On the other hand, a feasible integer solution for 14 problems was not achieved by the heuristic MIXED, whereas the other heuristics provided feasible solutions for all test problems. The number of problems that both SCICONIC and MIXED could not yield a solution for is 8. Considering all test problems, the number of times SCICONIC yields the best solution is 21, and for the other heuristics the number are: MINSLK heuristic: 18, SDFIRST: 10, LDFIRST: 2, and MIXED: zero.

On the average, the solutions given by SCICONIC are within 9.23% of their respective linear optimum values covering only those problems where it is ended with feasible integer solutions. On the other hand, the MIXED heuristic provides integer solutions that are deviated 31.59% from their corresponding linear optimum solution for those problems in which the heuristic finds a feasible integer solution. The full list of average (in %) deviations from the linear optimum solutions is:
SCICONIC: 9.23%, MIXED: 31.59%, MINSLK: 15.19%, SDFIRST: 14.03%, LDFIRST: 19.74%.
For the problems provided with feasible integer solutions by MIXED heuristics, average time to spend is found to be 1778 CPU seconds including the number of repetitions. On the other hand, in spite of execution interruption conditions, SCICONIC spends 6055 CPU seconds on the average. It takes an average of 27 seconds to find linear optimum solutions.

5. Conclusions and recommendations

Based on the results obtained from the experiment, we can state the following:

(1) From a computational effort point of view, the heuristics MINSLIK, SDFIRST and LDFIRST are found to be better. This is expected since these heuristics are specially suited for project scheduling problems and they are single-pass type algorithms.

(2) On the basis of both number of times yielding the best solution and deviation from linear optimum, the heuristic LDFIRST is worse than the other two. Although MINSLIK heuristic is better than SDFIRST in view of the number cases where the best solution was achieved, SDFIRST has a better solution quality on the average.

(3) The percent deviation from linear optimum through these single-pass typed heuristics is sensitive to the problem difficulty.

(4) There is no dominating solution procedure if all the cases are taken individually into account, but it is clear that special purpose heuristics are superior are superior to integer programming based heuristics in general. There does not seem to be much likelihood for a general purpose integer programming heuristic to outperform these methods on this specific problem.

(5) The solution quality attained by SCICONIC is the best. It should certainly be considered as a viable alternative to solve small and some medium scale problems. The computational costs can easily be covered by the savings obtained through better solutions.

(6) The MIXED heuristic does not seem fit for these type of problems. However, there may be scope for increasing its efficiency by a more judicious selection of penalty weights and starting solutions.

References

[1] Bell, C.E., and Han, J., "A new heuristic solution method in resource-constrained project scheduling", *Naval Research Logistics* 38 (1991) 315–331.

[2] Boctor, F.F., "Some efficient multi-heuristic procedures for resource constrained project scheduling", *European Journal of Operational Research* 49 (1990) 3–13.

[3] Christofides, N., Valdes, R.A., and Tamarit, J.M., "Project scheduling with resource constraints: A branch and bound approach", *European Journal of Operational Research* 29 (1987) 262–273.

[4] Cooper, D.F., "Heuristics for scheduling resource constrained projects: An experimental investigation", *Management Science* 22/11 (1976) 1186–1184.

[5] Davis, E.W., "Networks: Resource allocation", *Journal of Ind Engineering* 4 (1974) 22–32.

[6] Davis, E.W., and Heidorn, G.E., "An algorithm for optimal scheduling under multiple resource constraints", *Management Science B* 17/12 (1971) 803–816.

[7] Davis, E.W., and Patterson, J.H., "A comparison of heuristic and optimum solutions in resource-constrained project scheduling", *Management Science* 21/8 (1975) 944–955.

[8] Deckro, R.F., Winkofsky, E.P., Hebert, J.E., and Gagnon, R., "A decomposition approach to multi-project scheduling", *European Journal of Operational Research* 51 (1991) 110–18.

[9] Elsayed, E.A., and Nasr, N.Z., "Heuristics for resource constrained scheduling", *International Journal of Production Research* 24/2 (1986) 299–310.

[10] Fendly, L.G., "Toward the development of a complete multiproject scheduling system", *Journal of Ind Engineering* 19/10 (1968) 505–515.

[11] Fisher, M.L., "Optimum solution of problems using Lagrange multipliers: Part I", *Operations Research* **21** (1973) 1115–1127.

[12] Holloway, C.A., Nelson, R.T., and Suraphongschai, V., "Comparison of a multi-pass heuristic decomposition procedure with other resource-constrained project scheduling procedures", *Management Science* 25/9 (1979) 862–872.

[13] Khattab, M.M., and Choobineh, F., "A new approach for project scheduling with a limited resource", *International Journal of Production Research* 29/1 (1991) 185–198.

[14] Kelly, J.E., and Walker, M.R., "Scheduling activities to satisfy resource constraints", in: Muth and Thompson, *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[15] Kurtulus, I., and Davis, E.W., "Multi-project scheduling: Categorization of heuristic rules performance", *Management Science* 28/2 (1982) 161–172.

[16] Lenstra, J.K., and Rinnooy, Kan, A.H.G., "Complexity of scheduling under precedence constraints", *Operations Research* 26 (1978) 22–35.

[17] Levy, F.K., and Wiest, J.D., *A Management Guide to PERT / CPM*, Prentice-Hall, Englewood Cliffs, NJ, 1969.
[18] Mohanty, R.P., and Siddiq, M.K., "Multiple projects-multiple resources constrained scheduling: Some studies", *International Journal of Operation Research* 27/2 (1989) 261–280.
[19] Norbis, M.I., and Smith, M., "Two level heuristic for the resource constrained scheduling problem", *International Journal of Operation Research* 24/2 (1986) 1203–1219.
[20] Patterson, J.H., "Alternate methods of project scheduling with limited resources", *Naval Research Logistics* 20 (1973) 767–84.
[21] Patterson, J.H., "A comparison of exact approaches for solving the multiple constrained resource project scheduling problem", *Management Science* 30/7 (1984) 854–867.
[22] Patterson, J.H., and Huber, W.D., "A horizon varying zero–one approach to project scheduling", *Management Science* 20/6 (1974) 990–998.
[23] Pritsker, A.B., Watters, L.J., and Wolfe, P.M., "Multiproject scheduling with limited resources: A zero–one programming approach", *Management Science* 16/1 (1969) 93–108.
[24] Scrage, L., "Solving resource-constrained network problems by implicit enumeration: Nonpreemptive case", *Operations Research* 10 (1970) 263–278.
[25] Stinson, J.P., Davis, E.W., and Khumawala, B.M., "Multiple resource constrained scheduling using branch and bound", *AIIE Transactions* 10/3 (1978) 1197–1210.
[26] Talbot, F.B., "An integer programming algorithm for resource-constrained project scheduling problem", unpublished Ph.D. Dissertation, Pennsylvania State University, 1976.
[27] Talbot, F.B., "Resource-constrained project scheduling with time resource tradeoffs: The nonpreemptive case", *Management Science* 28/10 (1982) 1197–1210.
[28] Talbot, F.B., and Patterson, J.H., "An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems", *Management Science* 24/11 (1978) 1163–1174.
[29] Thesen, A., "Heuristic scheduling of activities under resource and precedence restrictions", *Management Science* 23/4 (1976) 412–422.
[30] Wiest, J.D., "The scheduling of large projects with limited resources", Ph.D. Dissertation, Carnegie Institute of Technology, 1963.
[31] Wiest, J.D., "A heuristic model for scheduling large projects with limited resources", *Management Science* 13/6 (1967) B359–B377.