A Linear Programming and Constraint Propagation-Based Lower Bound for the RCPSP

Peter Brucker
Sigrid Knust *

Universität Osnabrück
Fachbereich Mathematik/Informatik
D-49069 Osnabrück

June 1998

Osnabrücker Schriften zur Mathematik
Reihe P, Heft 204

Abstract

A new destructive lower bound for the resource-constrained project scheduling problem is presented. Given are n activities which have to be processed without preemptions. During the processing period of an activity constant amounts of renewable resources are needed where the available capacity of each resource type is limited. Furthermore, finish-start precedence relations between the activities are given. The objective is to determine a schedule with minimal makespan.

The lower bound calculations are based on two methods for proving infeasibility of a given threshold value T for the makespan. The first uses constraint propagation techniques, while the second is based on a linear programming formulation. A column generation procedure is presented for the linear programming formulation and computational results are reported for an algorithm combining both concepts.

Keywords: resource-constrained project scheduling problem, lower bounds, linear program, column generation, constraint propagation

*Supported by the Deutsche Forschungsgemeinschaft, Project 'Komplexe Maschinen-Schedulingprobleme'

1 Introduction

The **resource-constrained project scheduling problem (RCPSP)** may be formulated as follows: Given are n activities $i = 1, \ldots, n$ and r renewable resources. A constant amount of R_k units of resource k is available at any time. Activity i must be processed for p_i time units, where preemption is not allowed. During this time period a constant amount of r_{ik} units of resource k is occupied. The values R_k, p_i, and r_{ik} are supposed to be non-negative integers. Furthermore, precedence relations $(i, j) \in A$ are defined between the activities. The objective is to determine starting times S_i for the activities $i = 1, \ldots, n$ in such a way that

- at each time t the total resource demand is less than or equal to the resource availability for each resource type,
- the given precedence constraints A are fulfilled, i.e. $S_i + p_i \leq S_j$ for all $(i, j) \in A$, and
- the makespan $C_{\max} = \max_{i=1}^{n} C_i$ is minimized, where $C_i := S_i + p_i$ is the completion time of activity i.

The problem may be presented by an activity-on-the-node network with dummy starting and end nodes 0 and $n + 1$, respectively, and $p_0 = p_{n+1} = 0$. All nodes in $V := \{0, 1, \ldots, n, n + 1\}$ are weighted with their processing times p_i. We assume $S_0 = 0$ and identify S_{n+1} with the $C_{\max}$-value.

Two types of lower bounds can be derived for the RCPSP: **constructive** and **destructive** bounds. Constructive bounds are usually provided by solving relaxations of the RCPSP. A simple constructive bound is the length of a longest path in the network, which is the solution value of the RCPSP when all resource constraints are relaxed. Stinson et al. [9] and Demeulemeester [4] proposed lower bounds based on extensions of critical path analysis. A preemptive linear programming relaxation has been proposed by Mingozzi et al. [8]. Since this LP-formulation contains an exponential number of variables, for larger n they use some "weighted node packing-bound" heuristics based on the dual linear program. Another method to deal with the large number of columns has been suggested by Baar et al. [1] who used the technique of delayed column generation to solve the linear program.

Destructive bounds are derived in the following way (cf. Klein and Scholl [5]). Starting with a hypothetical upper bound T an attempt is made to obtain contradictions to feasibility. If some method provides the result that no feasible schedule with makespan $C_{\max} \leq T$ exists, $T + 1$ is a valid lower bound value. To find the largest lower bound T binary search is applied.

Given a hypothetical upper bound T, for each activity a time window $[r_i, d_i]$ may be defined in the following way. We define release dates (heads) r_i as the length of a

longest path in the network from node 0 to i, which is obviously a lower bound for the earliest starting time of activity i. Symmetrically, tails q_i may be defined as the length of a longest path from i to $n+1$ (weight p_i excluded). Associated with the upper bound T we define deadlines $d_i := T - q_i$. Since q_i is a lower bound for the time we need after finishing activity i, in each schedule with makespan $C_{\max} \leq T$ activity i has to be finished not later than time d_i. Thus, in a feasible schedule each activity i has to be processed completely in its time window, i.e. the starting times S_i have to fulfill the conditions $r_i \leq S_i$ and $S_i + p_i \leq d_i$.

In this paper we will discuss two methods for proving that no schedule exists which satisfies the time window constraints. The first is based on constraint propagation techniques (immediate selection) which will be described in Section 2. Another method based on linear programming will be presented in Section 3. We try to find a preemptive schedule such that all activities are processed within their time windows and all resource constraints are respected. This preemptive problem is formulated as a linear program and solved by column generation. Computational results for an implementation combining both approaches are presented in Section 4.

2 A lower bound based on constraint propagation

In this section we present the main ideas of using constraint propagation (immediate selection) for lower bound calculations. As described in the Introduction, in a destructive approach methods are needed to prove infeasibility of a given threshold value T for the makespan. If infeasibility cannot be detected, we use the result of constraint propagation techniques to get a more restricted instance (e.g. with smaller time windows for the activities or more precedence constraints). Applied to this instance, the linear programming formulation presented in the next section may detect a contradiction more easily.

The immediate selection techniques we use are based on schedule schemes which represent sets of feasible schedules. They were introduced in connection with a branch-and-bound algorithm (Brucker et al. [3]) and have also been used in a tabu-search procedure (Baar et al. [1]).

A **schedule scheme** $\mathcal{S} := (\mathcal{C}, \mathcal{D}, \mathcal{N}, \mathcal{F})$ consisting of four disjoint relations $\mathcal{C}, \mathcal{D}, \mathcal{N}, \mathcal{F}$ (conjunctions, disjunctions, parallelity relations, and flexibility relations) represents the set of all schedules where

- activity j does not start before i is completed for all $i \rightarrow j \in \mathcal{C}$,
- the processing intervals of activities i and j do not overlap for all $i - j \in \mathcal{D}$,
- activities i and j are processed in parallel for at least one time unit for all $i \parallel j \in \mathcal{N}$, and

- there are no timing restrictions on activities i and j for all $i \sim j \in \mathcal{F}$.

Note that a schedule scheme may represent infeasible schedules as well as feasible ones because the resource constraints may be violated. If $\mathcal{C}_0$ is the set of all given precedence relations A, $\mathcal{D}_0$ is the set of all pairs of activities which cannot be processed in parallel due to the resource constraints, and $\mathcal{F}_0$ is the set of all remaining pairs $i \sim j$, then $(\mathcal{C}_0, \mathcal{D}_0, \emptyset, \mathcal{F}_0)$ represents the set of all feasible schedules.

By immediate selection we denote certain constraint propagation techniques which are based on schedule schemes and time windows $[r_i, d_i]$ for the activities. Immediate selection methods are used to

- deduce additional conjunctions, disjunctions, or parallelity relations,
- strengthen the time windows, or
- detect infeasibility.

In the following we will briefly review different immediate selection techniques which are used in our algorithm (for details see Brucker et al. [3]).

For a given schedule scheme $\mathcal{S} = (\mathcal{C}, \mathcal{D}, \mathcal{N}, \mathcal{F})$ and a threshold value T for the makespan we define a **distance matrix** $D_{\mathcal{S}} = (d_{ij})_{i,j \in V}$ by

$$d_{ij} := \begin{cases} 0 & \text{if } i = j \\ p_i & \text{if } i \rightarrow j \in \mathcal{C} \\ -(p_j - 1) & \text{if } i \parallel j \in \mathcal{N} \\ p_i - T & \text{otherwise.} \end{cases} \tag{2.1}$$

It can easily be shown that for each pair $(i, j) \in V \times V$ the value d_{ij} is a lower bound for the difference $S_j - S_i$. Additionally, we may incorporate further temporal restrictions (heads and tails) into $D_{\mathcal{S}}$ by setting $d_{0i} := r_i$ and $d_{i,n+1} := p_i + q_i$ for $i = 1, \ldots, n$ because $r_i \leq S_i = S_i - 0 = S_i - S_0$ and $S_i + p_i + q_i \leq C_{\max} = S_{n+1}$, respectively.

The distance matrix may be transitively adjusted according to the formula

$$d_{ij} := \max\{d_{ij}, d_{ik} + d_{kj}\} \text{ for all } i, j, k \in V$$

by the Floyd-Warshall algorithm. If then $d_{ii} > 0$ for an activity i holds, a positive cycle is indicated and no feasible schedule for $\mathcal{S}$ with makespan $C_{\max} \leq T$ exists. Thus, $T + 1$ is a valid lower bound value. On the other hand, after calculating the transitive closure additional conjunctions or parallelity relations may be deduced.

With the concept of **symmetric triples** several modifications of a schedule scheme $\mathcal{S}$ may be derived. A triple of activities (i, j, k) is called **symmetric** if $k \parallel i$ and $k \parallel j$

hold and i, j and k cannot be processed simultaneously because of resource constraints. Then we may introduce a disjunction $i - j \in \mathcal{D}$. Furthermore, considering a fourth activity l, additional relations in $\mathcal{C}, \mathcal{D}$ and $\mathcal{N}$ may be derived (for details see Brucker et al. [3]).

Provided with heads and tails we apply classical concepts of immediate selection. The basic idea of these concepts is to improve heads and tails of activities belonging to a **clique** in an undirected graph $G_{\mathcal{S}} = (V, E_{\mathcal{S}})$ associated with a schedule scheme $\mathcal{S}$. The set of edges $E_{\mathcal{S}}$ consists of all pairs of incompatible activities, i.e. all pairs of activities between which a conjunction or disjunction exists. After generating some cliques heuristically, heads and tails of activities belonging to a clique are improved by using well-known procedures for the job-shop problem (see, e.g. Brucker [2]). Then the improved heads and tails are retransformed into distances which may lead to additional conjunctions. The whole process is repeated until no more additional conjunctions can be derived.

3 A linear programming-based lower bound

In this section we will describe a lower bound which strengthens a linear programming-based lower bound formulated by Mingozzi et al. [8]. Besides partially relaxing the precedence constraints and allowing preemptions, time windows $[r_i, d_i]$ of the activities i are taken into account. Given a threshold value T, we first formulate the problem of finding a preemptive schedule with $C_{\max} \leq T$ such that all activities are processed within their time windows and all resource constraints are respected.

In the following we assume that the threshold value T is taken into account in the time windows $[r_i, d_i]$ by setting $d_{n+1} := T$ and by modifying the other deadlines appropriately.

Let $z_0 < z_1 < \ldots < z_\tau$ be the ordered sequence of all different r_i- and d_i-values. For $t = 1, \ldots, \tau$ we consider the intervals $I_t := [z_{t-1}, z_t]$ of length $z_t - z_{t-1}$. With each interval I_t we associate a set F_t of all activities i which can be planned in this interval, i.e. $r_i \leq z_{t-1}$ and $z_t \leq d_i$.

A subset $X \subseteq F_t$ is called **feasible** if all activities $i \in X$ may be processed simultaneously, i.e. there are no precedence constraints or disjunctions (induced by immediate selection) between any pair of activities in X and all resource constraints are respected. We denote all feasible subsets for an interval I_t by X_{jt} $(j = 1, \ldots, q_t)$. With each set X_{jt} we associate an incidence vector $a^{jt} \in \{0,1\}^n$ defined by

$$a_i^{jt} = \begin{cases} 1, & \text{if } i \in X_{jt} \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, let x_{jt} be a variable denoting the number of time units where all activities in X_{jt} are processed simultaneously. Then the preemptive feasibility problem may be written as follows:

$$\sum_{t=1}^{\tau}\sum_{j=1}^{q_t} a_i^{jt} x_{jt} \geq p_i \qquad (i = 1, \ldots, n) \tag{3.1}$$

$$\sum_{j=1}^{q_t} x_{jt} \leq z_t - z_{t-1} \qquad (t = 1, \ldots, \tau) \tag{3.2}$$

$$x_{jt} \geq 0 \qquad (t = 1, \ldots, \tau;\ j = 1, \ldots, q_t) \tag{3.3}$$

Due to restrictions (3.1) all activities i are processed for at least p_i time units. Conditions (3.2) ensure that the number of time units scheduled in interval I_t do not exceed the length $z_t - z_{t-1}$ of this interval.

By introducing artificial variables u_t for $t = 1, \ldots, \tau$ in conditions (3.2) the feasibility problem can be formulated as a linear program:

$$\text{MIN} \qquad \sum_{t=1}^{\tau} u_t \tag{3.4}$$

$$\text{s.t.} \qquad \sum_{t=1}^{\tau}\sum_{j=1}^{q_t} a_i^{jt} x_{jt} \geq p_i \qquad (i = 1, \ldots, n) \tag{3.5}$$

$$-\sum_{j=1}^{q_t} x_{jt} + u_t \geq -z_t + z_{t-1} \qquad (t = 1, \ldots, \tau) \tag{3.6}$$

$$x_{jt} \geq 0 \qquad (t = 1, \ldots, \tau;\ j = 1, \ldots, q_t) \tag{3.7}$$

$$u_t \geq 0 \qquad (t = 1, \ldots, \tau) \tag{3.8}$$

A solution exists for the preemptive feasibility problem if and only if the linear program has the optimal solution value zero, i.e. if all values of the artifical variables become zero.

As in the linear program formulated by Mingozzi et al. [8], the number of columns in this LP-formulation grows exponentially with the number n of activities. In order to reduce the number of columns, Mingozzi et al. [8] and Baar et al. [1] restricted the set of feasible subsets to the set of nondominated subsets (these are maximal sets which are not a proper subset of another feasible set). For formulation (3.1) to (3.3) it is irrelevant which sets are used, since both possibilities are equivalent due to the "$\geq$"-relation in constraints (3.1). We tested both possibilities.

In the following we describe the details of our approach which is an extension of our column generation procedure for the situation without time windows (see Baar et al.

[1]). We work iteratively with a restricted working set of columns which contains a basis, the artifical variables u_t, and all slack variables according to (3.5) and (3.6). Initially, we determine for each activity $i = 1, \ldots, n$ an index $t(i)$ of an interval $I_{t(i)}$ in which i can be planned, i.e. $I_{t(i)} \subseteq [r_i, d_i]$. By processing each set $\{i\}$ in the interval $I_{t(i)}$ for p_i time units a feasible basic starting solution for the LP (3.4) to (3.8) can be obtained by choosing the n variables $x_{j(i),t(i)}$ corresponding to the initial sets $\{i\}$ and τ additional variables among the variables u_t $(t = 1, \ldots, \tau)$ and the slack variables according to (3.6) as basic variables.

In the general step of the column generation procedure the linear program is solved to optimality with the current working set of columns. Then we have to calculate new entering columns or show that no improving column exists. If we denote the dual variables corresponding to conditions (3.5) by y_i $(i = 1, \ldots, n)$ and the dual variables belonging to (3.6) by w_t $(t = 1, \ldots, \tau)$, then the constraints in the associated dual problem have the form

$$\sum_{i=1}^{n} a_i^{jt} y_i - w_t \leq 0 \qquad (t = 1, \ldots, \tau;\ j = 1 \ldots, q_t) \tag{3.9}$$

$$w_t \leq 1 \qquad (t = 1, \ldots, \tau) \tag{3.10}$$

$$y_i \geq 0 \qquad (i = 1, \ldots, n) \tag{3.11}$$

$$w_t \geq 0 \qquad (t = 1, \ldots, \tau) \tag{3.12}$$

Since the linear program coresponding to the current set of columns is solved to optimality and all artificial variables and slack variables are kept in the working set, the dual constraints (3.10), (3.11), and (3.12) are satisfied by the current solution. Furthermore, the dual constraints (3.9) may only be violated for columns $a^{jt} \in \{0,1\}^n$ which do not belong to the current working set. Thus, to find an entering column we have to find a column a^{jt} with

$$\sum_{i=1}^{n} a_i^{jt} y_i - w_t > 0. \tag{3.13}$$

Feasible columns satisfying (3.13) are calculated as follows. We scan iteratively through the intervals I_t $(t = 1, \ldots, \tau)$ and search with a branch-and-bound algorithm (which will be described later) for improving columns in I_t. If we find a sufficient number of columns in I_t, they are added to the working set of columns and the linear program is resolved to optimality. Otherwise, we proceed with the next interval $I_{(t+1) \bmod \tau}$. If in no interval an improving column is found, the current solution is optimal and the procedure terminates. If the optimal objective value is greater than zero, infeasibility is detected and $T + 1$ is a valid lower bound value.

The search process in an interval I_t is stopped if one of the following conditions holds:

- we have found a given number of columns,

- we have found at least one column and the quotient of the number of considered columns and the number of improving columns during the search process exceeds a certain limit.

When the LP is resolved to optimality and I_t is the interval where improving columns have been found in the last step, the process for finding new columns is started in the next interval $I_{(t+1) \bmod \tau}$. If the number of columns in the working set exceeds a given size, some arbitrary columns are deleted.

Finally, we describe the branch-and-bound algorithm which is used to determine improving columns in an interval I_t. Since the dual constraints (3.11) are satisfied for the current solution, we have $y_i \geq 0$ for all $i = 1, \ldots, n$. We consider a fixed order of all activities $i \in F_t$ which is obtained by sorting the activities in F_t according to nonincreasing dual values $y_1 \geq y_2 \geq \ldots \geq y_{n_t^0} > 0$, $y_{n_t^0+1} = \ldots = y_{n_t} = 0$ with $n_t := |F_t|$. To calculate incidence vectors a which correspond to feasible sets we define a branching tree in which the vertices represent partial strings $(a_1, \ldots, a_k)$ where $1 \leq k \leq n_t^0$. These strings are extended using a binary branching rule by setting $a_{k+1} := 1$ first and then $a_{k+1} := 0$. For the first possibility we have to check if it is feasible to include activity $k+1$ in the partial string; the second alternative is always feasible.

We undertake a depth-first search in the branching tree where the sets are generated in lexicographic order and leaves correspond to feasible subsets. When we have found a suitable leaf a, we fill the corresponding subset successively with all activities $n_t^0 + 1, \ldots, n_t$ which do not violate feasibility. Subsequently, we backtrack to depth n_t^0. Since all dual values of activities $n_t^0 + 1, \ldots, n_t$ are zero, other extensions of leaf a have the same value $\sum_{i=1}^{n} a_i^{jt} y_i$ and, thus, no columns with better value (3.13) are excluded.

Furthermore, some search tree nodes are cut off by the following simple bounding rule: If we set $a_k := 0$ in step k and

$$\sum_{\lambda=1}^{k-1} a_\lambda \cdot y_\lambda + \sum_{\lambda=k+1}^{n_t^0} y_\lambda \leq w_t$$

holds, the partial string $(a_1, \ldots, a_k)$ cannot be extended to a column satisfying (3.13) and therefore we can backtrack. In the formulation in which only nondominated sets are admissible, dominated sets are excluded by some additional checks as described in Baar et al. [1].

4 Computational results

In this section we present some computational results for an algorithm which combines the concepts of the previous sections. The results were obtained using a Sun Ultra 2

workstation with operating system Solaris 2.5 and 320 MB general storage (167 MHz). The algorithms were coded in C and CPLEX 4.0 was used as an LP-solver.

With a binary search procedure we calculated the largest lower bound LB where infeasibility could be proved. To detect infeasibility we first used the immediate selection techniques presented in Section 2. If this failed to prove infeasibility, then the linear programming formulation of Section 3 was applied to the (possibly more restricted) instance. To get an initial interval for the binary search procedure we used a fast heuristic (based on priority rules and a few local search iterations in the parallelity neighborhood of Baar et al. [1]) to provide an initial upper bound. After applying immediate selection with this upper bound we calculated an initial lower bound LB^0 with the column generation procedure of Baar et al. [1].

We investigated the performance of our algorithm with respect to various benchmark instances generated by the project generator PROGEN published in Kolisch et al. [6] and Kolisch and Sprecher [7]. They specified a large number of parameter constellations and generated problem groups, each group consisting of 10 instances. For each $n = 30, 60, 90$ and $r = 4$ they obtained 48 groups.

At first we tested the two possibilities for generating columns: on the one hand, we allowed all feasible subsets to be generated by the column generation procedure, on the other hand, we restricted the search to nondominated subsets. Since the average computational times for the first formulation were smaller, we will only present the results obtained by the implementation with all feasible subsets.

For all instances with $n = 30$ the optimal values are known. The calculated lower bounds LB are optimal for 318 instances, the average percentage deviation of the value LB from the optimum for all instances is 1.5%, the maximal deviation is 11.1%. The average (maximal) computation times (including the initial calculations of lower and upper bounds and the binary search procedure) were 0.4 (4.3) seconds. The value LB is determined by immediate selection in 409 cases, for the remaining 71 instances the linear programming formulation is superior.

The computational results for the PROGEN-instances with $n = 60$ and $n = 90$ are presented in Table 1. Since the optimal values are not known for all these instances, we used the best known heuristic solutions (collected in the PSPLIB library in Kiel) to obtain bounds for the quality of the calculated lower bounds. The number of problems for which the lower and upper bound values are equal is indicated. Furthermore, we calculated the average percentage deviation $\Delta(UB) := \frac{UB-LB}{UB}$ from the best known upper bound UB. Again, the computational times are presented as well as the number of instances for which immediate selection or the linear programming formulation proves infeasibility.

Table 1 shows that the quality of the lower bound values is very good. Furthermore, the computational times are small. It is noticeable that the linear programming formulation is superior to immediate selection techniques, especially for groups with resource

problem set	$n = 60$	$n = 90$
number of instances	480	480
verified instances	340	343
average $\Delta(UB)$	2.2%	2.6%
maximal $\Delta(UB)$	14.9%	16.1%
average computation time	5 sec	1:12 min
maximal computation time	1:02 min	2:45 h
LB determined by Imm. Sel.	388	380
LB determined by LP	92	100

Table 1: Results for large PROGEN-instances

strength $RS = 0.2$. These are instances with scarce resources in which many resource conflicts occur between the activities. For these groups only very few instances can be verified, which indicates that they may contain the hardest instances. The results for all instances with $RS = 0.2$ are summarized in Table 2, where $\Delta^0(UB) := \frac{UB-LB^0}{UB}$ denotes the deviation of the initial lower bound LB^0 from the best known upper bound UB.

$RS = 0.2$	$n = 60$	$n = 90$
number of instances	120	120
verified instances	19	11
average $\Delta(UB)$	7.5%	9.6%
average $\Delta^0(UB)$	11.4%	13.7%
LB determined by Imm. Sel.	38	26
LB determined by LP	82	94

Table 2: Results for hard PROGEN-instances with $RS = 0.2$

Detailed computational results for all PROGEN-instances can be found under the web site

`http://www.mathematik.uni-osnabrueck.de/research/OR/software.html`.

5 Concluding remarks

A new destructive lower bound for the resource-constrained project scheduling problem based on constraint propagation and linear programming has been developed. The obtained results are very good and confirm the conclusions of Klein and Scholl [5] who reported that destructive methods are often superior to constructive approaches.

References

[1] T. Baar, P. Brucker, S. Knust (1997) Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem, Osnabrücker Schriften zur Mathematik, P 192, to appear in Proceed. of the 2nd International Conference on Metaheuristics - MIC97, Sophia-Antipolis, France.

[2] P. Brucker (1995) Scheduling algorithms, Springer Verlag, Berlin.

[3] P. Brucker, S. Knust, A. Schoo, O. Thiele (1998) A branch & bound algorithm for the resource-constrained project scheduling problem, European J. Oper. Res. 107, 272–288.

[4] E. Demeulemeester (1992) Optimal algorithms for various classes of multiple resource-constrained project scheduling problem, Ph.D. thesis, Department of Applied Economic Sciences, Katholieke Universiteit Leuven.

[5] R. Klein, A. Scholl (1997) Computing lower bounds by destructive improvement - an application to resource-constrained project scheduling, Schriften zur Quant. BWL, TH Darmstadt, to appear in European J. Oper. Res.

[6] R. Kolisch, A. Sprecher, A. Drexl (1995) Characterization and generation of a general class of resource-constrained project scheduling problems, Management Science 41, 1693–1703.

[7] R. Kolisch, A. Sprecher (1997) PSPLIB - A project scheduling library, European J. Oper. Res. 96, 205–216.

[8] A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco (1996) An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation, Technical Report No. 32, Department of Mathematics, Università degli Studi di Bologna, to appear in Management Science.

[9] J.P. Stinson, E.W. Davis, B.H. Khumawala (1978) Multiple resource-constrained scheduling using branch and bound, AIIE Transactions 10, 252–259.