Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

Resource Constrained Project Scheduling: Computing Lower Bounds by Solving Minimum Cut Problems

by

Rolf H. Möhring Andreas S. Schulz
Frederik Stork Marc Uetz

No. 620/1998

Resource Constrained Project Scheduling: Computing Lower Bounds by Solving Minimum Cut Problems*

Rolf H. Möhring‡ Andreas S. Schulz§ Frederik Stork‡ Marc Uetz‡

Extended Abstract – January 17, 1999

Abstract

We propose a novel approach to compute bounds on the objective function value of a wide class of resource-constrained project scheduling problems. The basis is a polynomial-time algorithm to solve the following scheduling problem: Given a set of activities with start-time dependent costs and temporal constraints in the form of time windows, find a feasible schedule of minimum total cost. Motivated by a known result that this problem can be formulated as a cardinality-constrained stable set problem in comparability graphs, we show how to reduce it to a minimum cut problem in an appropriate directed graph.

We focus on an application of this algorithm to different types of resource-constrained project scheduling problems by using it for the computation of lower bounds on the objective function value via Lagrangian relaxation of these problems. This approach shows to be applicable to various problem settings, and an extensive study based on widely accepted test beds in project scheduling reveals that our algorithm significantly improves upon other fast computable lower bounds at very modest running times. For problems with time windows, we obtain the best known bounds for several instances, and for a class of notoriously hard labor-constrained scheduling problems with time-varying resources, we drastically reduce the time to obtain the lower bounds based on the corresponding LP relaxation. For precedence-constrained scheduling with several resource types, our bounds are again computed very fast, and improve the bounds obtained in reasonable time by all currently known algorithms.

1 Introduction and Problem Formulation

Resource constrained projects usually comprise several activities or jobs which have to be scheduled subject to both temporal and resource constraints in order to meet or minimize a certain objective. Temporal constraints often consist of precedence constraints, that is, certain activities must be completed before others can be processed, but sometimes even arbitrary minimal and maximal time lags, so-called time windows between pairs of activities have to be respected. Moreover, activities require resources while being processed, and the resource availability is limited. Also time-varying resource requirements and resource availabilities are considered. Most frequently, the project makespan is to be minimized, but also other, even non-regular objective functions are considered in the literature. For a detailed account of the various problem settings, most relevant references as well as a classification scheme for resource constrained project scheduling problems we refer to a recent survey by Brucker, Drexl, Möhring, Neumann, and Pesch [2].

*Supported by the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMB+F), grant No. 03-MO7TU1-3.

‡Technische Universität Berlin, Department of Mathematics, Sekr. MA 6–1, Straße des 17. Juni 136, D-10623 Berlin, Germany, Email: {moehring, stork, uetz}@math.tu–berlin.de

§MIT, Sloan School of Management and Operations Research Center, E53-361, Cambridge, MA 02139, Email: schulz@mit.edu

In general, project scheduling problems are NP-hard, and in the case of time windows even the problem of finding a feasible solution is NP-hard [1]. The intractability of these problems asks for good and fast computable lower bounds on the objective value, which may be used for heuristics and exact procedures. Unfortunately, it is very unlikely that provably good lower bounds can be computed within polynomial time, since project scheduling problems contain node coloring in graphs as a special case [19]. Thus, as for node coloring, there is no polynomial approximation algorithm with a performance guarantee less than n^{ε} for any fixed $\varepsilon > 0$, unless $P = NP$. This negative result puts also limits on the computation of good lower bounds.

Problem formulation. We now introduce the model of resource-constrained project scheduling with time windows. Let $V = \{0, \dots, n+1\}$ be a set of activities with integral activity durations p_j. All activities must be scheduled non-preemptively, and by $S = (S_0, \dots, S_{n+1})$ we denote a schedule, where S_j is the start time of activity j. Activities 0 and $n+1$ are assumed to be dummy activities indicating the project start and the project completion, respectively. Temporal constraints in the form of minimal and maximal time lags between pairs of activities are given. By d_{ij} we denote a time lag between two activities $i, j \in V$, and by $L \subseteq V \times V$ we denote the set of all given time lags. We assume that the temporal constraints always refer to the start times, thus every schedule S has to fulfill $S_j \geq S_i + d_{ij}$ for all $(i, j) \in L$. Note that $d_{ij} \geq 0$ ($d_{ji} < 0$) implies a minimal (maximal) positive time lag of S_j relative to S_i, thus so-called *time windows* of the form $S_i + d_{ij} \leq S_j \leq S_i - d_{ji}$ between any two activities can be modeled. Ordinary precedence constraints can be represented by letting $d_{ij} = p_i$ if activity i must precede activity j. In addition to these temporal constraints we will consider a time horizon T as an upper bound on the project makespan. It can be checked in polynomial time by longest path calculations if such a system of temporal constraints has a feasible solution. Throughout the paper we will assume feasibility. We then also obtain for each activity a set of (integral) feasible start times $I_j := \{ES_j, ..., LS_j\}$, $j \in V$, where ES_j and LS_j denote the earliest and latest start time of activity j, respectively.

Activities need resources for their processing. In the model with constant resource requirements, we are given a finite set $\mathcal{R}$ of different, renewable resources, and the availability of resource $k \in \mathcal{R}$ is denoted by R_k, that is, an amount of R_k units of resource k is available throughout the project. Every activity j requires an amount of r_{jk} units of resource k, $k \in \mathcal{R}$, to be performed. But also time-varying resource demands $r_{jk}(t)$ and availabilities $R_k(t)$ are possible. The activities have to be scheduled non-preemptively such as to minimize a given measure of performance, which usually is the project makespan.

Project scheduling problems are often formulated as integer linear programs with time-indexed binary variables x_{jt}, $j \in V$, $t \in \{0, ..., T\}$, which are defined by

$$x_{jt} = \begin{cases} 1 & : \text{ activity } j \text{ starts at time } t \\ 0 & : \text{ otherwise,} \end{cases} \tag{1}$$

This leads to the following, well known integer linear programming formulation.

$$\begin{aligned}
\text{minimize} \quad & \sum_t t \cdot x_{n+1,t} && && (2)\\
\text{subject to} \quad & \sum_t x_{jt} = 1, && j \in V && (3)\\
& \sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \leq 1 && (i,j) \in L,\ t \in \{0, ..., T\} && (4)\\
& \sum_j r_{jk} \Big(\sum_{s=t-p_j+1}^{t} x_{js} \Big) \leq R_k, && k \in \mathcal{R},\ t \in \{0, ..., T\} && (5)\\
& x_{jt} \in \{0, 1\} && j \in V,\ t \in \{0, \dots, T\}. && (6)
\end{aligned}$$

Constraints (3) indicate that each activity is started exactly once, and inequalities (4) represent the temporal constraints given by the time lags L. Inequalities (5) assure that the activities processed simultaneously at time t do not consume more resources than available. Note that by slightly extending the resource inequalities (5), this formulation can easily be generalized to time dependent resource profiles, i.e. $R_k = R_k(t)$ and $r_{jk} = r_{jk}(t)$. In order to simplify notation, we have omitted this generalization in the above formulation.

Previous work. We now briefly review other related work. The above time-indexed formulation for project scheduling problems has been used before by various authors (e.g. [17, 21, 7, 5, 4]), sometimes also with a weaker formulation of temporal constraints as $\sum_t t(x_{jt} - x_{it}) \geq d_{ij}$, $(i, j) \in L$. Most relevant to our work is the paper by Christofides, Alvarez-Valdes, and Tamarit [7]. They have investigated a Lagrangian relaxation of the above integer program in order to obtain lower bounds on the makespan of project scheduling problems. They solve the Lagrangian relaxations by a branch and bound algorithm, not aware that they can be solved in polynomial time by purely combinatorial methods (see Section 2). As a matter of fact, the LP relaxation of (3), (4), and (6) is known to be integral. This structural result has been shown before by Chaudhuri, Walker, and Mitchell [5]. For problems with precedence constraints and time varying resources, the same has also been shown by Cavalcante, De Souza, Savelsbergh, Wang, and Wolsey [4]. The latter authors solve the corresponding linear programming relaxation of (3), (4), (5), and (6) optimally in order to exploit its solution for ordering heuristics to construct good feasible schedules.

Mingozzi, Maniezzo, Ricciardelli, and Bianco [14] have proposed several lower bounds on the project makespan for precedence-constrained project scheduling that rely on a different mathematical formulation. Their approach is based on introducing variables $y_{\ell t}$ which indicate if a (resource feasible) subset of activities $V_\ell \subseteq V$ is in process at a certain time t. Clearly, this formulation is of exponential size, since there are exponentially many such feasible subsets V_ℓ. They derive different lower bounds by considering several relaxations, including a very fast computable lower bound, usually referred to as LB_3, which is based on the idea to sum up the processing times of activities which pairwise cannot be scheduled simultaneously. Their bounds have then been evaluated and modified also by other authors. Here we particularly mention the paper by Brucker and Knust [3]. They solve the following relaxation of the project scheduling problem: Feasible subsets of activities must be scheduled (preemptively) such that every activity receives at least its total processing time. Again, the number of variables is exponential. To solve the problem, they have applied a column generation approach, where the pricing is done by a branch and bound algorithm. They thus obtain the best known bounds on the majority of instances of a well known test bed [13], however, their approach requires very large computation times.

Our approach. In the spirit of Christofides, Alvarez-Valdes, and Tamarit [7], we propose a Lagrangian relaxation of the resource constraints (5) to compute lower bounds for resource-constrained project scheduling problems as defined by (2) – (6). Within a subgradient optimization algorithm we solve a series of project scheduling problems given by (3), (4), and (6), subject to start time dependent costs for each activity. The core of our approach is a direct transformation of the project scheduling problem (3), (4), and (6) to a minimum cut problem in an appropriately defined directed graph. To solve the minimum cut problem, we use a maximum flow code by Cherkassky and Goldberg [6].

The potential of our approach is demonstrated by very promising computational results. We have used widely accepted test beds for makespan minimization in project scheduling, namely problems with ordinary precedence constraints as well as arbitrary minimal and maximal time lags [13, 20], and labor-constrained scheduling problems with a time varying resource profile stemming from chemical production within BASF AG, Germany [12].

The experiments reveal that our approach is capable of computing very good lower bounds at very short computation times. We thus improve previous, fast computable lower bounds, and in the setting with time windows even obtain best known lower bounds for quite a few instances. Compared to other approaches which partially require prohibitive running times, our algorithm offers a good tradeoff between quality and computation time. It also turns out that our algorithm is especially suited for problems with extremely scarce resources, which are the problems that tend to be intractable for other approaches.

For the instances stemming from BASF, Cavalcante et al. [4] report on tremendous computation times for solving corresponding linear programming relaxations. Our experiments show that one can obtain essentially the same value as with the LP relaxation much more efficiently.

Organization of the paper. In Section 2 we present the Lagrangian relaxation of the integer program (2) – (6), and introduce a direct transformation of the resulting subproblems (project scheduling subject to start time dependent weights) to minimum cut problems in an appropriate directed graph. Section 3 is then concerned with an extensive computational study of our approach. We analyze our algorithm in comparison to both the solution of the corresponding LP relaxations, and other lower bounding algorithms. We conclude with some remarks on future research.

2 The Lagrangian Relaxation

Christofides, Alvarez-Valdes, and Tamarit [7] have proposed the following Lagrangian relaxation of the time indexed integer programming formulation of resource-constrained project scheduling given by (2) – (6). They dualize the resource constraints (5), and introduce Lagrangian multipliers $\lambda_{tk} \geq 0$, $t \in \{0,\ldots,T\}$, $k \in \mathcal{R}$, to obtain:

$$\text{minimize} \sum_t t \cdot x_{n+1,t} + \sum_j \sum_t \Big(\sum_{k \in \mathcal{R}} r_{jk} \sum_{s=t}^{t+p_j-1} \lambda_{sk} \Big) x_{jt} - \sum_t \sum_{k \in \mathcal{R}} \lambda_{tk} \cdot R_k \tag{7}$$

subject to (3), (4), and (6).

If we now omit the constant term $\sum_t \sum_{k \in \mathcal{R}} \lambda_{tk} \cdot R_k$ and introduce weights

$$w_{jt} = \begin{cases} \sum_{k \in \mathcal{R}} r_{jk} \sum_{s=t}^{t+p_j-1} \lambda_{sk} & \text{if } j \neq n+1, \\ t + \sum_{k \in \mathcal{R}} r_{jk} \sum_{s=t}^{t+p_j-1} \lambda_{sk} & \text{if } j = n+1, \end{cases}$$

we can reformulate (7) as

$$\text{minimize} \qquad c(x) := \sum_j \sum_t w_{jt} \cdot x_{jt} \qquad \text{subject to (3), (4), and (6).} \tag{8}$$

This formulation specifies a project scheduling problem where the activities have start-time dependent costs, and where the aim is to minimize the overall cost subject to minimal and maximal time lags between activities. We refer to this problem as *project scheduling problem with start-time dependent costs*. It is a basic observation that all weights can without loss of generality be assumed to be positive, since due to (3), any additive transformation of the weights only affects the solution value, but not the solution itself. (In (8) the weights are nonnegative by definition.) Note that the problem can trivially be solved by longest path calculations if the w_{jt} are non-decreasing in t. However, for general weights this is not true.

We finally point out that the above Lagrangian relaxation is not restricted to makespan minimization, but can as well be applied to any other regular, and even non-regular objective function. Thus our procedure is applicable to a variety of project scheduling problems like, for instance, the minimization of the weighted sum of completion times, problems that aim at minimizing lateness, or the so-called *resource investment problem*, where the objective is to minimize the investment in expensive resources such that the production can be accomplished within a given deadline (cf. [15], [8]).

The transformation. We now propose a reduction of the project scheduling problem with start time dependent costs given in (8) to a minimum cut problem in a directed graph $D = (N,A)$ which is defined as follows.

- *Nodes.* The set of nodes N contains for each activity $j \in V$ the nodes $u_{jt}, t \in \{ES_j, ..., LS_j + 1\}$. Furthermore, it contains a dummy source a and a dummy sink b.
  So $N := \{a,b\} \cup \{u_{jt} | j \in V, t \in \{ES_j, ..., LS_j + 1\}\}$.
- *Arcs.* The arc set A can be divided into three disjoint subsets. The set of *assignment arcs* corresponds to the binary variables x_{jt} of the integer program (8) and contains all arcs $(u_{jt}, u_{j,t+1})$ for all $j \in V$ and $t \in I_j$. Then, x_{jt} corresponds to $(u_{jt}, u_{j,t+1})$. Thus, the set of assignment arcs is defined by the set $\{(u_{jt}, u_{j,t+1}) | j \in V, t \in I_j\}$.

  The set of *temporal arcs* guarantee that no temporal constraint is violated, i.e., the set of temporal arcs is defined by the set $\{(u_{is}, u_{j,s+d_{ij}}) | (i,j) \in L, s \in I_i\}$.

  Finally, a set of *dummy arcs* connects the source and the sink nodes a and b with the remaining network. The dummy arcs are given by $\{(a, u_{i,ES_i} | i \in V\} \cup \{(u_{i,LS_i+1}, b) | i \in V\}$.
- *Capacities.* The capacity of an assignment arc $(u_{jt}, u_{j,t+1})$, $j \in V$, $t \in I_j$, equals the weight w_{jt} of its associated binary variable x_{jt}. The capacity of every temporal arc and every dummy arc is infinite. All *lower* capacities are assumed to be 0.

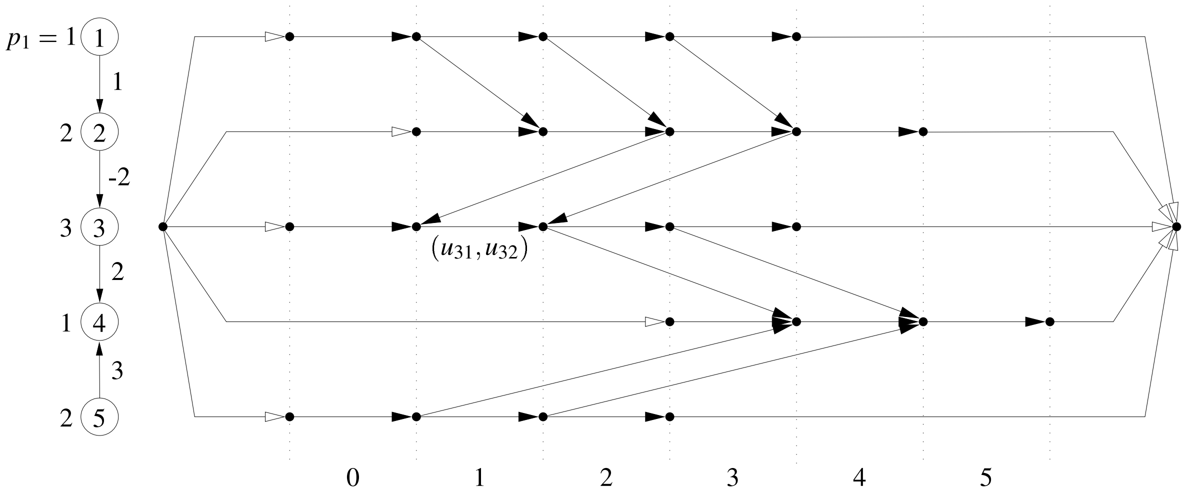

Figure 1: The left digraph represents the relevant data of the underlying example: Each node represents an activity, each arc represents a temporal constraint. The right digraph D is the corresponding graph obtained out of the transformation. Each assignment arc of D corresponds to a binary variable x_{jt} as defined in (1). For instance, the arc (u_{31}, u_{32}) corresponds to x_{31}. Those arcs marked by a white arrow head are dummy arcs that connect the global source a and the global sink b with the remaining network.

Figure 1 shows an example of the graph D which is based on an instance consisting of 5 activities $V = \{1, ..., 5\}$. The set of time lags is defined by $\{d_{12} = 1, d_{23} = -2, d_{34} = 2, d_{54} = 3\}$. The activity durations are $p_1 = p_4 = 1$, $p_2 = p_5 = 2$, and $p_3 = 3$. We assume that $T = 6$ is a given upper bound on the project makespan. Then, the earliest start vector is $ES = (0,1,0,3,0)$ and the latest start vector is $LS = (3,4,3,5,2)$.

Given a directed graph $D = (N,A)$, an *a,b-cut* is a pair $(X,\bar{X})$ of disjoint sets of nodes $X,\bar{X} \subset N$ such that $X \cup \bar{X} = N$, and $a \in X$, $b \in \bar{X}$. The capacity $c(X,\bar{X})$ of such a cut $(X,\bar{X})$ is the sum of the capacities of its forward arcs $c(X,\bar{X}) := \sum_{(u,\bar{u})\in(X,\bar{X})} c(u,\bar{u})$. An arc $(u,\bar{u}) \in A$ is said to be a forward arc of the cut if $u \in X$ and $\bar{u} \in \bar{X}$. A minimum a,b-cut is a cut which has minimal capacity among all a,b-cuts.

Theorem 2.1. *A minimum a,b–cut $(X^*,\bar{X}^*)$ of digraph D corresponds to an optimal solution x^* of integer program* (8) *(the project scheduling problem with start time dependent costs), and $c(X^*,\bar{X}^*) = c(x^*)$.*

Here, x^* is given by

$$x^*_{jt} = \begin{cases} 1 & \text{if } (u_{jt}, u_{j,t+1}) \text{ is a forward arc of the cut } (X,\bar{X}), \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

and $c(x)$ denotes the cost of a solution x of the integer program (8).

The proof crucially uses the fact that each minimum a,b–cut of the digraph D consists of exactly one forward arc $(u_{jt}, u_{j,t+1})$ for every activity j. Note that this only holds since the weights w_{jt} are strictly positive and thus also the capacities of the arcs are strictly positive. Furthermore, it is essential that the given instance has a feasible solution since, otherwise, one of the dummy arcs might be contained in a minimum cut. Theorem 2.1 is proved by the following Lemmas 2.2, 2.3, and 2.4.

Lemma 2.2. *For each feasible solution x of the integer program* (8) *there exists an a,b-cut $(X,\bar{X})$ in D such that $c(x) = c(X,\bar{X})$.*

Proof. Given a feasible solution x of the integer program (8), the construction of the corresponding a,b-cut $(X,\bar{X})$ of D is straightforward. Due to (3), there exists exactly one $x_{j,t_j} = 1$ for some $t_j \in I_j$ for each activity $j \in V$, which corresponds to the activity start time $S_j = t_j$. Let $X := \bigcup_{j\in V}\{u_{jt} | t \le t_j\} \cup \{a\}$, and let $\bar{X} := N \setminus X$. We now show that the capacity $c(X,\bar{X})$ of this cut equals $c(x)$. By definition all arcs (u_{j,t_j}, u_{i,t_j+1}), $j \in V$, are forward arcs of $(X,\bar{X})$, and the sum of the capacities of these arcs is exactly $c(x)$. Now suppose that there exist another forward arc of the cut, which then must be a temporal arc $(u_{is}, u_{jt}), s \le t_i, t > t_j$. By definition of D, we thus have a temporal constraint between jobs i and j which says $S_j - S_i \ge t - s$. But since $S_j - S_i = t_j - t_i < t - s$, we obtain a contradiction to the assumption that x was feasible, and thus $c(x) = c(X,\bar{X})$. □

Lemma 2.3. *Let $(X,\bar{X})$ be a minimum a,b-cut of the digraph D. Then for each activity $j \in V$ exactly one assignment arc $(u_{jt}, u_{j,t+1})$ is contained as forward arc in the cut.*

Proof. Since we have assumed that there always exists a feasible solution x of the integer program (8) it follows by Lemma 2.2 that there always exists a corresponding a,b-cut in D which has finite capacity. Thus, also the minimum cut $(X,\bar{X})$ has finite capacity, i.e., it does not contain any of the dummy or temporal arcs as forward arcs. It follows that for each activity $j \in V$ at least one assignment arc $(u_{jt}, u_{j,t+1})$ is contained as forward arc in $(X,\bar{X})$. Now assume that $(X,\bar{X})$ contains more than one assignment arc $(u_{jt}, u_{j,t+1})$ for some activity $j \in V$. We then construct another cut $(X^*,\bar{X}^*)$ with smaller capacity: For each j let t_j be the smallest time index such that $(u_{j,t_j}, u_{j,t_j+1}) \in (X,\bar{X})$. Let $X^* := \bigcup_{j\in V}\{u_{jt} | t \le t_j\} \cup \{a\}$, and let $\bar{X}^* := N \setminus X^*$. Clearly, $X^* \subset X$ and the set of assignment arcs of $(X^*,\bar{X}^*)$ is a proper subset of the corresponding set of assignment arcs of $(X,\bar{X})$. Now recall that all weights of the scheduling problem (8) are strictly positive, thus all arc capacities are strictly positive as well. To see that $(X^*,\bar{X}^*)$ has smaller capacity than $(X,\bar{X})$, it now suffices to show that $(X^*,\bar{X}^*)$ does not contain any of the temporal

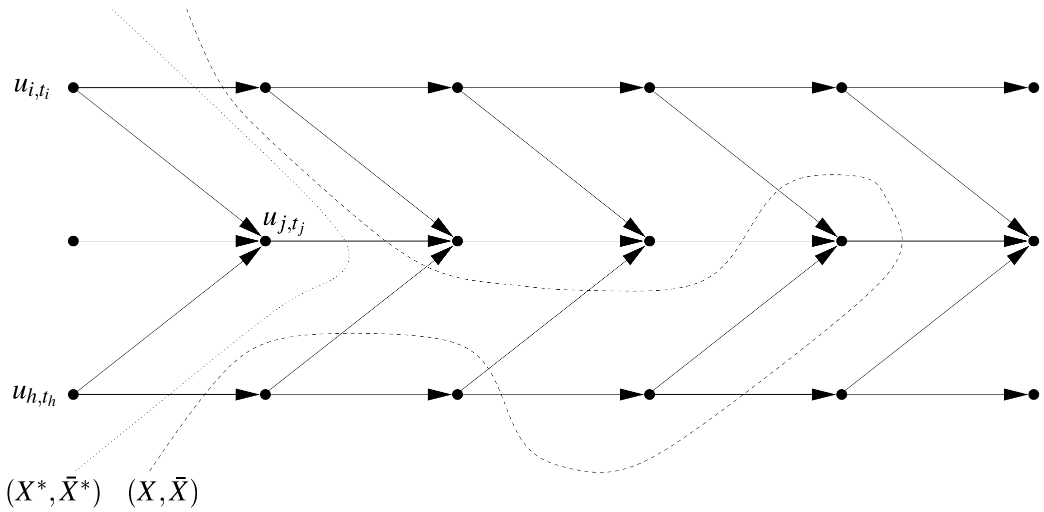

Figure 2: A minimum cut $(X^*,\bar{X}^*)$ of D always contains exactly one forward arc of each j.

arcs as forward arcs. So assume that there exists such a temporal arc $(u_{is},u_{jt}) \in (X^*,\bar{X}^*)$, $s \leq t_i$, $t > t_j$. Since $(u_{is},u_{jt}) \in A$, it follows by construction of D that all arcs $(u_{i,s-z},u_{j,t-z}) \in A$ for all $z \in \mathbb{N}_0$ such that $s-z \in I_i$ and $t-z \in I_j$. Now let $z := t-t_j-1$, then $s-z \in I_i$ and $t-z \in I_j$. Since $s \leq t_i$, we have $u_{i,s-z} \in X^* \subset X$. Moreover, by definition of t_j, $u_{j,t-z} = u_{j,t_j+1} \in \bar{X}$. Thus we have identified a temporal arc $(u_{i,s-z},u_{j,t-z}) \in (X,\bar{X})$, i.e., a forward arc of the cut $(X,\bar{X})$ which is a contradiction to the assumption that it had finite capacity. □

With the use of Lemma 2.3 the following lemma can be derived straightforwardly, which eventually concludes the proof of Theorem 2.1.

Lemma 2.4. *For each minimum cut $(X^*,\bar{X}^*)$ in D there exists a feasible solution x of the integer program* (8) *such that $c(x) = c(X^*,\bar{X}^*)$.*

Proof. Given a minimum cut $(X^*,\bar{X}^*)$ of D, let, according to the assignment (9), $x_{jt} = 1$ for all arcs $(u_{jt},u_{j,t+1}) \in (X^*,\bar{X}^*)$ and $x_{jt} = 0$ otherwise. Clearly, the cost of this assignment is exactly the cost of the corresponding cut, thus $c(X^*,\bar{X}^*) = c(x)$. By Lemma 2.3, it follows that the resulting solution x fulfills equalities (3). Now suppose that the temporal constraints (4) were not fulfilled by the constructed solution x. Then there exists a violated temporal constraint, say $x_{is} = x_{jt} = 1$ where $t-s < d_{ij}$. By Lemma 2.3, it follows that $u_{j,s+d_{ij}} \in \bar{X}$. However, by definition of the digraph D, there exists a temporal arc $(u_{is},u_{j,s+d_{ij}})$, and since $u_{is} \in X$, this temporal arc is a forward arc of the cut. Thus the cut $(X^*,\bar{X}^*)$ has infinite capacity, which contradicts the assumption that its capacity is minimal. (Remember that there is a feasible solution to (8), thus by Lemma 2.2 there is a cut of finite capacity.) Consequently, also inequalities (4) are valid, and x is in fact a feasible solution of the integer program (8). □

Since D has $O(n \cdot T)$ nodes and $O((n+m) \cdot T)$ arcs, a minimum cut in D can be computed in $O(nmT^2 log(T))$ time when applying the classical push-relabel-algorithm for maximum flows [9]. Here, m is the number of given time lags L.

A related transformation has been investigated by Chaudhuri, Walker, and Mitchell [5]. They transform the integer program (8) into a cardinality-constrained stable set problem in comparability graphs, with the objective to identify a stable set of minimum weight among all stable sets of maximum cardinality. The weighted stable set problem in comparability graphs can be transformed in polynomial time to a minimum flow maximum cut problem on a digraph, in which the maximum cut corresponds to the maximum weighted stable set, cf. [10, 16]. However, the resulting digraph is dense while the digraph resulting from our transformation has a very sparse structure, since the set L of temporal constraints is usually sparse. Moreover, the directed graph D as defined above need not be acyclic, and thus cannot be derived from a transitive orientation of the comparability graph defined in [5].

To conclude, the proposed transformation into a minimum cut problem is the key to efficiently solve the integer program (8). In the following section we will also demonstrate its practical relevance and efficiency within a subgradient optimization algorithm in order to solve the multiplier problem of the Lagrangian relaxation (7).

3 Experimental Study

In this section we study several aspects of the proposed lower bounding algorithm. We first compare our approach with the corresponding LP relaxation of the initial integer program. We then empirically analyze how running time and quality of the Lagrangian relaxation algorithm depend on different parameters such as the time horizon, the number of activities, and the scarceness of the available resources. Next, we compare our bounds with those computed by other lower bounding algorithms for resource-constrained project scheduling. We finally briefly investigate the computation of feasible schedules based on the solution of the Lagrangian relaxation.

The Lagrangian multiplier problem which has to be solved within our approach is computed by a standard subgradient optimization procedure which is aborted if the objective value was not improved significantly over five consecutive iterations. If this happens within the first 10 iterations we restart the procedure with another choice of step sizes.

For the computations we have used a slight extension of the resource inequalities (5) which has been proposed by Christofides, Alvarez-Valdes, and Tamarit [7]. They suggest to strengthen the resource constraints by ensuring that no activity $j \in V \setminus \{n+1\}$ is performed simultaneously to the dummy sink $n+1$. To do so, we set the resource requirements $r_{n+1,k} = R_k$ for all resources $k \in \mathcal{R}$. Then, once activity $n+1$ has been started it consumes all available resources. The modified resource constraints can be written as follows.

$$\sum_j r_{jk}\Big(\sum_{s=t-p_j+1}^{t} x_{js}\Big) + r_{n+1,k} \sum_{s=ES_{n+1}}^{t} x_{n+1,s} \le R_k, \qquad k \in \mathcal{R}, t \in \{0,\dots,T\} \tag{10}$$

3.1 Benchmark Instances

We have applied our algorithms to the widely accepted test beds of the ProGen and the ProGen/max library [13, 20]. Furthermore, we have considered a small test bed of problems modeled after chemical production processes with labor constraints [11].

The ProGen library currently provides instances for precedence-constrained scheduling with multiple resource constraints. These instances consist of 30, 60, 90, and 120 activities, respectively. They are generated by modifying three parameters of the instance generator, the *network complexity* which reflects the average number of direct successors of an activity, the *resource factor* which describes the average number of resources required in order to process an activity, and the *resource strength*, which is a measure of the scarcity of the resources. The parameter controlling the resource strength varies between 0.1 and 0.7 where a small value indicates very scarce resources. This variation results into 480 instances of each of the first three instance sizes (30, 60, 90), and 600 instances of 120 activities. The activity durations were chosen randomly between 1 and 10 and the maximum number of different resources is 4 per activity. The library also contains best known upper bounds on these instances, which we have used as time horizon T in order to compute the latest start times of the activities (for the instances of 30 activities these upper bounds represent the optimum). Among the whole set of instances we only took those for which the given upper bound is larger than the trivial lower bound LB_0 which is the earliest start time ES_{n+1} of the dummy activity $n+1$. The number of instances then reduces to 264 (30 activities), 185 (60 activities), 148 (90 activities), and 432 (120 activities). The time horizon of these instances varies between 35 and 306. For further details on this library we refer to [13].

The ProGen/max library provides 1080 instances scheduling problems with time windows and multiple resource requirements, each of which consists of 100 activities. The parameters of the instance generator are similar to those of the ProGen generator but an additional parameter controls the number of cycles in digraph of temporal constraints. Again, the activity durations were chosen randomly between 1 and 10, and the maximum number of different resources is 5. 21 of the 1080 instances are infeasible and for another 693 instances there exists a feasible solution with a project makespan which equals the trivial lower bound LB_0. Thus, the number of instances of interest within this test bed reduces to 366. For these instances, the time horizon is set again to the documented best known upper bounds. It varies between 253 and 905. For further details on the parameters of this library we refer to [20].

Finally, we consider instances which have their origin in a labor-constrained scheduling problem (LCSP) from BASF AG, Germany, which can briefly be summarized as follows: The production process for a set of so-called *orders* has to be scheduled. Every order represents the output of a constant amount of a chemical product, and the aim is to minimize the project *makespan*, i.e., the time to complete all orders. The production process for an order consists of a sequence of identical activities, each of which must be scheduled non-preemptively. *Due dates* for individual orders are given, and due to technical reasons there may be precedence constraints between activities of different orders. Additionally, *resource constraints* have to be respected, which are imposed by a limited number of available workers: An activity usually consists of several consecutive tasks such as blending, heating or other, and these tasks require a certain amount of personnel. Thus, the personnel requirement of any activity is varying over time, and given by a piecewise constant requirement function. We refer to instances of this type as LCSP instances. A more detailed problem description has been published by Kallrath and Wilson [12]. The instances considered here are taken from [11].

3.2 Computing Environment

Our experiments were conducted on a Sun Ultra 2 with 200 MHz clock pulse operating under Solaris 2.6 with 512 MB of memory. Our code is written in C++ using the Standard Template Library (STL) and has been compiled with the GNU g++ compiler version 2.7.2. The Maximum Flow Code which is provided by Cherkassky and Goldberg [6] is written in C and has been compiled with the GNU gcc compiler version 2.7.2. Both compilers used the -O4 optimization option. To solve the linear programs we have used CPLEX v. 4.0.8. All reported CPU times to compute the lower bounds are averaged over three runs.

3.3 Computational Results

LP relaxation versus Lagrangian relaxation Since the LP relaxation of (8) is integral, the lower bounds obtained by the Lagrangian relaxation will never exceed those of the LP relaxation. Since the LPs are usually very large (and notoriously difficult particularly for the LCSP instances) we have compared the running times to solve these LPs with our Lagrangian relaxation. For the test bed with precedence constraints and 30 activities, the LP is solved within 18 seconds on average, while the Lagrangian relaxation plus the subgradient optimization requires only one second on average, with an average number of 104 iterations. The average deviation of the values of the LP relaxation and the subgradient optimization is only 1%. Most instances of the test beds with a larger number of activities require exhaustive computation time and memory, particularly for large time horizons T.

For the LCSP instances Cavalcante, De Souza, Savelsbergh, Wang, and Wolsey [4] have computed

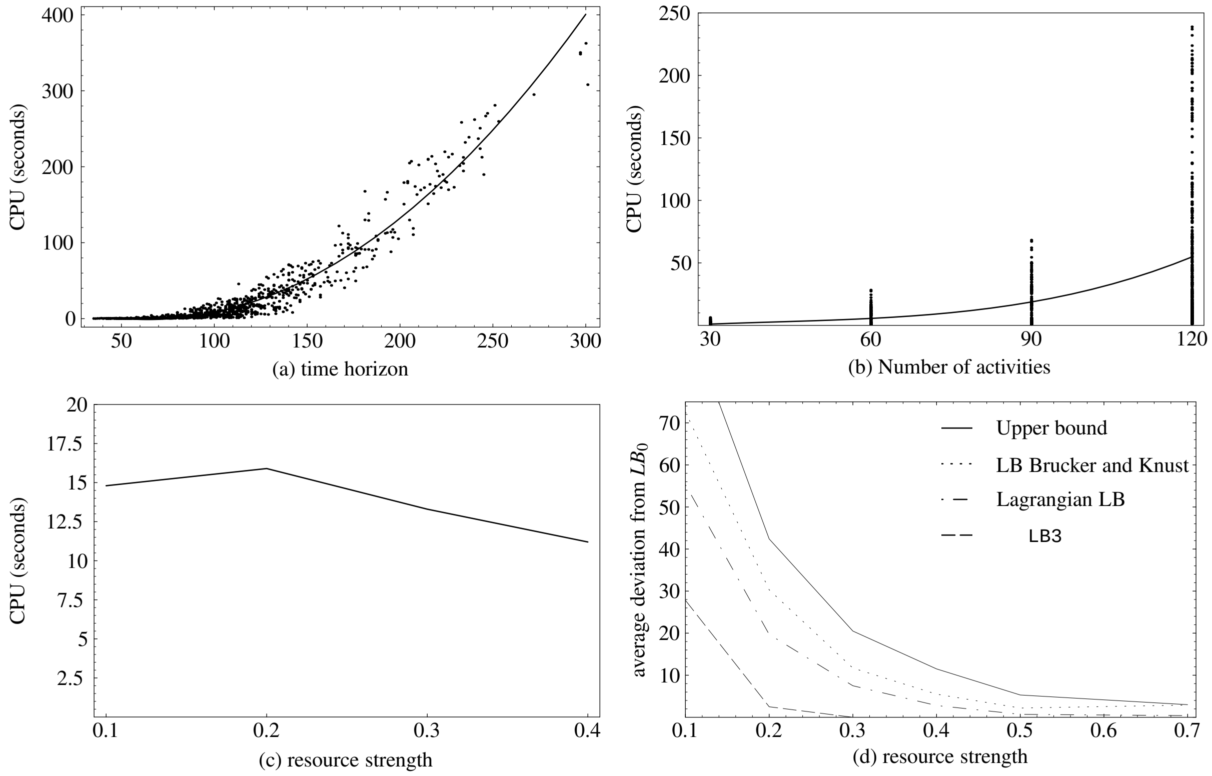

Figure 3: Plots (a) and (b) show the running time depending on the time horizon and the number of activities. Graphic (c) displays the effect of the resource strength on the running time for fixed $n = 120$ and $T \in [100, 120]$. and Graphic (d) visualizes the quality of the different bounds depending on the resource strength. The ordering of the curves is: Best known upper bound, lower bound by Brucker and Knust [3], Lagrangian lower bound, LB_3.

the LP relaxation of the integer programming formulation of the problem in order to deduce suitable orderings among the activities for generating feasible solutions for these instances. They also report on excessive running times required for the computation of such LP relaxations. This can be drastically reduced when alternatively applying our algorithm.

Empirical analysis Next, we empirically analyze the running time and the performance with respect to varying problem parameters. For this part of the study we considered all of the ProGen test beds. We vary the number of activities, the time horizon, and the resource strength. Figures 3 (a) and (b) display plots that show how the running time depends on both the time horizon and the number of activities. Each plot additionally contains of a corresponding regression curve. Obviously, the obtained results coincide with the theoretical running times as stated in Section 2.

Since other algorithms often require large running time when dealing with instances consisting of very hard resource constraints, we investigate the dependency of the running time of our algorithm on the resource strength. Recall that a small value of the resource strength parameter indicates very scarce resources. As depicted in Figure 3 (c), the running time of our algorithm is only slightly affected by the resource strength parameter.

Other lower bounding algorithms We next compare the results of our algorithm with those ones computed by other lower bounding procedures. The aim is to show that our algorithm behaves quite reasonable with respect to the tradeoff between quality and computation time. Besides the trivial lower

								best known	
Type	#act.	#inst.	LB_0	LB	UB	CPU	LB_3	LB	CPU
prec	60	185	71.3	78.8	90.6	6.1	74.2	85.6	13.5
prec	90	148	86.3	99.6	115.8	20.0	86.8	106.1	170.8
prec	120	432	94.6	116.7	137.6	56.9	102.0	124.9	n.a.
temp	100	366	431.4	435.9	499.0	72.1	434.2	452.2	n.a.

Table 1: Lower bounds obtained by the Lagrangian relaxation for the different test beds as described in Section 3.1. *prec* and *temp* indicates whether the instances consist of precedence constraints or of arbitrary time lags.

bound LB_0 only very few other techniques have been proposed, and most of them are tailored to minimize the makespan. For the scenario with precedence constraints, we compare our algorithm with two other approaches which are both based on [14]. First, we consider the lower bounds reported by Brucker and Knust [3] which are the strongest known bounds for the ProGen Testbeds, and second, we have implemented the $O(|V|^2)$ lower bound LB_3 (cf. Section 1). The average results on the running time and the quality of the lower bounds are provided in Table 1. While the computation time for the bound LB_3 is negligible (< 0.5 sec.), the algorithm of Brucker and Knust provides better bounds but in exchange for much larger running times. To obtain the lower bounds for the test bed which consist of 120 activities, their algorithm occasionally requires a couple of days per instance (on a Sun Ultra 2 with 167 MHz clock pulse), as reported to us in private communication. We could solve all of these instances within an average of less than a minute and a maximum of 362 seconds using 12 MB memory. When compared to the bound LB_3, our algorithm produces far better bounds in most of the cases.

For the instances with time windows, the algorithm proposed by Brucker and Knust [3] cannot be applied, since it is developed for the model with precedence constraints only. The best known lower bounds collected in the test bed are computed by different algorithms, mostly by a combination of preprocessing steps and a generalization of the lower bound LB_3. As indicated in Table 1, the results of our algorithm on this test bed are less satisfactory. However, the best known lower bounds for these instances are also of low quality. We were able to improve 38 of these best known lower bounds among the 366 instances. The reason for this less satisfactory behavior may be a weaker average resource strength which leads to bounds of low quality, and also the larger time horizons which result in large running times.

For the LCSP instances, no documented lower bounds are available beside the trivial bound LB_0. Our computational experiences on these instances coincide with the above observation that the quality of our lower bounds increases when the availability of personnel is very low.

Computing feasible schedules Besides the computation of lower bounds for complex combinatorial optimization problems, both the LP relaxation and the Lagrangian relaxation method often allow the construction of good upper bounds by exploiting the structure of the corresponding solution to construct a good feasible solution. For labor-constrained scheduling problems, Cavalcante, De Souza, Savelsbergh, Wang, and Wolsey [4] as well as Savelsbergh, Uma, and Wein [18] have proposed such techniques. So far, we have performed experiments in order to compute feasible schedules for the ProGen instances by extracting an ordering on the activities from the solution of the Lagrangian relaxation and used them as priority rules to generate feasible solutions. We could not beat the best known upper bounds (which have partially been computed by expensive local search and truncated branch and bound algorithms) but the priority rules deliver the best schedules when compared to other priority lists stated in the literature. Among all ProGen instances, the average deviation of the upper bounds computed by this technique from the best known upper bounds is only 18%.

In summary, our approach works very well for instances with low resource availability. Many instances with this property turn out to be very hard to solve to optimality within branch and bound algorithms. Furthermore, although the running time depends crucially on the time horizon, we are able to compute instances consisting of large time horizons within reasonable time.

4 Concluding Remarks

We have presented a lower bounding procedure that may be applied to a wide variety of different resource constrained project scheduling problems. The bounds obtained by this algorithm can be computed fast and are particularly strong for scenarios with very scarce resources. The algorithm is easy to implement since it basically computes a sequence of maximum flow problems.

Future research will mainly be concerned with the integration of other classes of inequalities. When additionally dualizing such inequalities, we may further strengthen the lower bounds at a low computational cost, since the structure of the underlying minimum cut problem remains unchanged. In particular, motivated by the approach described by Mingozzi, Maniezzo, Ricciardelli, and Bianco [14], we plan to identify suitable sets W of activities out of which not more than $\ell, \ell < |W|$, activities can be scheduled simultaneously. Based on the used time-indexed binary variables x_{jt}, the above constraint can be formulated as

$$\sum_{j \in W} \sum_{s=t-p_j+1}^{t} x_{js} \leq \ell \qquad t \in \{0, \ldots, T\}.$$

Furthermore, since we have to compute a sequence of maximum flow computations on very similar graphs, it could be valuable to adapt the maximum flow algorithm for our specific application, and also to recycle the flow data of the previous iteration. We also plan to test alternative techniques to solve the Lagrangian multiplier problem. In particular the so-called *One-Shot method* could be very helpful, since, in contrast to the subgradient optimization, only one Lagrangian parameter is changed. This in turn enables the reuse of the minimum cut of the previous iteration.

Acknowledgments. The authors are grateful to Matthias Müller-Hannemann for numerous helpful suggestions and comments in the field of Maximum Flow Algorithms and to Olaf Jahn for his technical support.

References

[1] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16:201–240, 1988.

[2] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999.

[3] P. Brucker and S. Knust. A linear programming and constraint propagation-based lower bound for the RCPSP. Technical Report 204, Osnabrücker Schriften zur Mathematik, 1998.

[4] C. C. B. Cavalcante, C. C. De Souza, M. W. P. Savelsbergh, Y. Wang, and L. A. Wolsey. Scheduling projects with labor constraints. CORE Discussion Paper 9859, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 1998.

[5] S. Chaudhuri, R. A. Walker, and J. E. Mitchell. Analyzing and exploiting the structure of the constraints in the ILP approach to the scheduling problem. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2:456 – 471, 1994.

[6] B. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. In *Proceedings of the 4th Conference on Integer Programming and Combinatorial Optimization*, pages 157–171, 1995.

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

[8] A. Drexl and A. Kimms. Optimization guided lower and upper bounds for the Resource Investment Problem. Technical Report 481, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, 1998.

[9] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. Assoc. Comp. Mach.*, 35:921–940, 1988.

[10] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[11] `http://www.dcc.unicamp.br/~cris/SPLC.html`.

[12] J. Kallrath and J. M. Wilson. *Business Optimisation using Mathematical Programming*. Macmillan Business, London, U.K., 1997.

[13] R. Kolisch and A. Sprecher. PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, 96:205–216, 1996.

[14] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the multiple resource-constraint project scheduling problem based on a new mathematical formulation. *Management Science, to appear*.

[15] R. H. Möhring. Minimizing costs of resource requirements subject to a fixed completion time in project networks. *Operations Research*, 32:89–120, 1984.

[16] R. H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In I. Rival, editor, *Graphs and Order*, pages 41–101. D. Reidel Publishing Company, Dordrecht, 1985.

[17] A. A. B. Pritsker, L. J. Watters, and P. M. Wolfe. Multi project scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108, 1969.

[18] M. W. P. Savelsbergh, R. N. Uma, and J. Wein. An experimental study of LP-based approximation algorithms for scheduling problems. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 453–462, San Francisco, 1998.

[19] M. Schäffter. Scheduling with respect to forbidden sets. *Discrete Applied Mathematics*, 72:141–154, 1997.

[20] C. Schwindt. Generation of resource constrained project scheduling problems with minimal and maximal time lags. Technical Report 489, WIOR, University of Karlsruhe, Germany, 1996.

[21] F. B. Talbot and J. H. Patterson. An efficient integer programmig algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science*, 24:1163–1174, 1978.

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

634/1999 *Karsten Weihe and Ulrik Brandes and Annegret Liebers and Matthias Müller–Hannemann and Dorothea Wagner and Thomas Willhalm:* Empirical Design of Geometric Algorithms

633/1999 *Matthias Müller–Hannemann and Karsten Weihe:* On the Discrete Core of Quadrilateral Mesh Refinement

632/1999 *Matthias Müller–Hannemann:* Shelling Hexahedral Complexes for Mesh Generation in CAD

631/1999 *Matthias Müller–Hannemann and Alexander Schwartz:* Implementing Weighted b-Matching Algorithms: Insights from a Computational Study

629/1999 *Martin Skutella:* Convex Quadratic Programming Relaxations for Network Scheduling Problems

628/1999 *Martin Skutella and Gerhard J. Woeginger:* A PTAS for minimizing the total weighted completion time on identical parallel machines

627/1998 *Jens Gustedt:* Specifying Characteristics of Digital Filters with FilterPro

620/1998 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* Resource Constrained Project Scheduling: Computing Lower Bounds by Solving Minimum Cut Problems

619/1998 *Rolf H. Möhring, Martin Oellrich, and Andreas S. Schulz:* Efficient Algorithms for the Minimum-Cost Embedding of Reliable Virtual Private Networks into Telecommunication Networks

618/1998 *Friedrich Eisenbrand and Andreas S. Schulz:* Bounds on the Chvátal Rank of Polytopes in the 0/1-Cube

617/1998 *Andreas S. Schulz and Robert Weismantel:* An Oracle-Polynomial Time Augmentation Algorithm for Integer Proramming

616/1998 *Alexander Bockmayr, Friedrich Eisenbrand, Mark Hartmann, and Andreas S. Schulz:* On the Chvátal Rank of Polytopes in the 0/1 Cube

615/1998 *Ekkehard Köhler and Matthias Kriesell:* Edge-Dominating Trails in AT-free Graphs

613/1998 *Frederik Stork:* A branch and bound algorithm for minimizing expected makespan in stochastic project networks with resource constraints

612/1998 *Rolf H. Möhring and Frederik Stork:* Linear preselective policies for stochastic project scheduling

609/1998 *Arfst Ludwig, Rolf H. Möhring, and Frederik Stork:* A computational study on bounding the makespan distribution in stochastic project networks

605/1998 *Friedrich Eisenbrand:* A Note on the Membership Problem for the Elementary Closure of a Polyhedron

596/1998 *Rolf H. Möhring, Frederik Stork, and Marc Uetz:* Resource Constrained Project Scheduling with Time Windows: A Branching Scheme Based on Dynamic Release Dates

595/1998 *Rolf H. Möhring Andreas S. Schulz, and Marc Uetz:* Approximation in Stochastic Scheduling: The Power of LP-based Priority Policies

591/1998 *Matthias Müller–Hannemann and Alexander Schwartz:* Implementing Weighted b-Matching Algorithms: Towards a Flexible Software Design

590/1998 *Stefan Felsner and Jens Gustedt and Michel Morvan:* Interval Reductions and Extensions of Orders: Bijections to Chains in Lattices

584/1998 *Alix Munier, Maurice Queyranne, and Andreas S. Schulz:* Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems

577/1998 *Martin Skutella:* Semidefinite Relaxations for Parallel Machine Scheduling

566/1997 *Jens Gustedt:* Minimum Spanning Trees for Minor-Closed Graph Classes in Parallel

565/1997 *Andreas S. Schulz, David B. Shmoys, and David P. Williamson:* Approximation Algorithms

561/1997 *Matthias Müller–Hannemann:* High Quality Quadrilateral Surface Meshing Without Template Restrictions: A New Approach Based on Network Flow Techniques

559/1997 *Matthias Müller–Hannemann and Karsten Weihe:* Improved Approximations for Minimum Cardinality Quadrangulations of Finite Element Meshes

554/1997 *Rolf H. Möhring and Matthias Müller–Hannemann:* Complexity and Modeling Aspects of Mesh Refinement into Quadrilaterals

551/1997 *Hans Bodlaender, Jens Gustedt and Jan Arne Telle:* Linear-Time Register Allocation for a Fixed Number of Registers and no Stack Variables

550/1997 *Karell Bertet, Jens Gustedt and Michel Morvan:* Weak-Order Extensions of an Order

549/1997 *Andreas S. Schulz and Martin Skutella:* Random-Based Scheduling: New Approximations and LP Lower Bounds

Reports may be requested from:

S. Marcus
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: Marcus@math.TU-Berlin.DE

Reports are available via anonymous ftp from:

ftp.math.tu-berlin.de
cd pub/Preprints/combi
file Report–<number>–<year>.ps.Z