Solving Project Scheduling Problems by Minimum Cut Computations[†]

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

July 2000, revised April 2002 and November 2002

Abstract

In project scheduling, a set of precedence-constrained jobs has to be scheduled so as to minimize a given objective. In resource-constrained project scheduling, the jobs additionally compete for scarce resources. Due to its universality, the latter problem has a variety of applications in manufacturing, production planning, project management, and elsewhere. It is one of the most intractable problems in operations research, and has therefore become a popular playground for the latest optimization techniques, including virtually all local search paradigms. We show that a somewhat more classical mathematical programming approach leads to both competitive feasible solutions and strong lower bounds, within quite reasonable computation times. The basic ingredients of our approach are the Lagrangian relaxation of a time-indexed integer programming formulation and relaxation-based list scheduling, enriched with a useful idea from recent approximation algorithms for machine scheduling problems. The efficiency of the algorithm results from the insight that the relaxed problem can be solved by computing a minimum cut in an appropriately defined directed graph. Our computational study covers different types of resource-constrained project scheduling problems, based on several, notoriously hard test sets, including practical problem instances from chemical production planning.

[†]An extended abstract of this work appeared in the Proceedings of the 7th Annual European Symposium on Algorithms (Möhring, Schulz, Stork, and Uetz 1999).

[*]Technische Universität Berlin, Fakultät II, Institut für Mathematik, Sekr. MA 6-1, Straße des 17. Juni 136, D-10623 Berlin, Germany. Email: moehring@math.tu-berlin.de

[**]Massachusetts Institute of Technology, Sloan School of Management, E53-361, 77 Massachusetts Avenue, Cambridge, MA 02139, USA. Email: schulz@mit.edu

[‡]ILOG Deutschland GmbH, Ober-Eschbacher Straße 109, D-61352 Bad Homburg, Germany. Email: fstork@ilog.de

[§]Universiteit Maastricht, Faculty of Economics and Business Administration, Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands. Email: m.uetz@ke.unimaas.nl

1 Introduction

We consider the following resource-constrained project scheduling problem. The objective is to minimize the project duration (the makespan) subject to both temporal and resource constraints. Temporal constraints usually consist of precedence constraints, that is, certain jobs must be completed before others can be started. Resource constraints specify that every job requires capacity of different, renewable resource types while being processed, and the resource availability is limited. In extension of this basic model, we consider problems where the jobs are subject to minimal and maximal time lags, or so-called time windows, and may have resource requirements that vary over time. Not only is the resource-constrained project scheduling problem strongly NP-hard, but it also is hard to approximate. For instance, it contains graph coloring as a special case.

Following Christofides, Alvarez-Valdés, and Tamarit (1987), we use Lagrangian relaxation in order to compute lower bounds on the minimal project makespan. The relaxation is based on a well-known time-indexed integer linear programming formulation of the problem, due to Pritsker, Watters, and Wolfe (1969). Relaxing the resource-constraints results in a Lagrangian subproblem that is equivalent to project scheduling with start-time dependent costs. The latter problem can be efficiently solved as a minimum cut problem in a directed graph. This insight is the key to the practical efficiency of our algorithms, which will be demonstrated in our computational study. In addition, we compute feasible solutions for resource-constrained project scheduling by using relaxation-based list-scheduling heuristics. The priority lists are derived from so-called α–completion times of the jobs in the solutions of the Lagrangian subproblems. Computational experiments show that this approach is very promising in terms of both computation time and solution quality. Although the paper focuses on the minimization of the project makespan, the proposed algorithmic scheme can handle many other regular, and even non-regular objective functions as well. We evaluate our algorithms on various benchmark test sets for resource-constrained project scheduling. On the one hand, these are different instance sets from the project scheduling library PSPLIB (2000), systematically generated by Kolisch and Sprecher (1996) and Schwindt (1996). On the other hand, we use 25 labor-constrained instances in which the jobs have time-varying resource requirements; these instances are discussed in Heipcke, Colombani, Cavalcante, and de Souza (2000). They have been designed so as to resemble a real-world chemical production process at BASF AG, Ludwigshafen; see also Kallrath and Wilson (1997, Ch. 10.5).

The paper is subdivided into four parts, each of which starts with a brief summary of relevant related work. Section 2 presents the reduction of the project scheduling problem with start-time dependent costs to a minimum cut problem in a directed graph. The connection to resource-constrained project scheduling problems is established in Section 3, where we discuss the Lagrangian relaxation approach for computing lower bounds. The Lagrangian-based list scheduling heuristics are described in Section 4. Finally, Section 5 presents our computational results.

2 Project Scheduling with Start-Time Dependent Costs

We first consider the problem to minimize the total cost of a schedule when the jobs are subject to temporal constraints only (i.e., there are no resource constraints), but cause arbitrary, start-time dependent costs. The generality of this objective function encompasses many other, well-known objective functions like the makespan, the weighted sum of completion times, the net present value, or earliness-tardiness costs. In this section, we show that the general project scheduling problem with start-time-dependent costs and arbitrary precedence constraints, and even minimal and maximal time lags, can be solved as a minimum cut problem in a directed graph.

Gröflin, Liebling, and Prodon (1982) as well as Roundy, Maxwell, Herer, Tayur, and Getzler (1991) discuss project scheduling problems with start-time-dependent costs and special cases of precedence constraints (out-trees and chains, respectively). They solve these problems as minimum cost flow problems. For the special case of unit processing times and ordinary precedence constraints, Chang and Edmonds (1985) present a reduction to the minimum weight closure problem in a directed graph, which is well-known to be equivalent to the minimum cut problem. Chaudhuri, Walker, and Mitchell (1994) interpret the problem with unit processing times and arbitrary minimal and maximal time lags as a stable set problem in a comparability graph. Sankaran, Bricker, and Juang (1999) show that a linear programming formulation of the problem with ordinary precedence constraints has integral basic feasible solutions. One can essentially deduce from all these papers that project scheduling problems with start-time-dependent costs can be solved in polynomial time. For more details we refer to Möhring, Schulz, Stork, and Uetz (2001).

Let us formulate the problem and introduce some notation. Throughout the paper, we let $J = \{0,\dots,n\}$ be a set of jobs with integral, non-negative processing times p_j, $j \in J$. Jobs 0 and n are assumed to be artificial jobs indicating the project start and the project completion, respectively. Their processing time is zero. All jobs must be scheduled non-preemptively, and we represent a schedule by the vector $S = (S_0, S_1, \dots, S_n)$ of its start times. We assume that the start times S_j of jobs are integral, and every job j incurs a cost of w_{jt} if it is started at time t. Here, $t = 0,1,2,\dots,T$, and T is some upper bound on the minimal project makespan. Let d_{ij} be the integral length of a time lag (i, j) between two jobs $i, j \in J$, and let $L \subseteq J \times J$ be the set of all given time lags. We denote the number of time lags by $m := |L|$, and we assume that the temporal constraints always refer to the start times of jobs. In other words, every time-feasible schedule S has to satisfy $S_j \geq S_i + d_{ij}$, for all $(i, j) \in L$. The objective is to find a time-feasible schedule of minimal total cost. Since time lags may be of negative length, time windows of the form $S_i + d_{ij} \leq S_j \leq S_i - d_{ji}$ between any two jobs i and j can be modelled. With this interpretation in mind, a time lag of negative length is also called a maximal time lag. Ordinary precedence constraints can be represented by letting $d_{ij} = p_i$ if job i must precede job j. With Bellman's algorithm (1958), one can check in $O(mn)$ time whether a system of temporal constraints has a feasible solution. We shall assume throughout the paper that a time-feasible schedule exists.

2.1 Integer programming formulation

Before we discuss the transformation of the project scheduling problem with start-time dependent costs to a minimum cut problem, we formulate it as an integer programming problem. This formulation will be extended and reused later when we study the Lagrangian relaxation of resource-constrained project scheduling.

A common way to model scheduling problems as integer linear programs is to use time-indexed variables. Pritsker, Watters, and Wolfe (1969) were presumably the first to give an integer programming formulation in time-indexed $0/1$-variables of the type

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

where $j \in J$, and $t \in \{0, ..., T\}$. This leads to the following integer program.

$$\begin{aligned} &\text{minimize} && w(x) = \sum_j \sum_t w_{jt} x_{jt} && && (1) \\ &\text{subject to} && \sum_t x_{jt} = 1, && j \in J, && (2) \\ & && \sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \leq 1, && (i,j) \in L,\ t = 0, \ldots, T, && (3) \\ & && x_{jt} \geq 0, && j \in J,\ t = 0, \ldots, T, && (4) \\ & && x_{jt} \text{ integer}, && j \in J,\ t = 0, \ldots, T. && (5) \end{aligned}$$

Constraints (2) enforce that each job is started exactly once. Inequalities (3) represent the temporal constraints imposed by the time lags L. Given the temporal constraints and the time horizon T, it is easy to compute earliest and latest start times for each job $j \in J$. For convenience of notation, in the above formulation we assume (without stating it explicitly) that all variables with time indices outside these boundaries are 0 so that no job is started before its earliest or after its latest start time. Because of equations (2), any uniform additive transformation of the weights w_{jt} only affects the solution value, but not the solution itself. We therefore assume without loss of generality that $w_{jt} \geq 0$, for all $j \in J$. Instead of (3), another formulation of the temporal constraints is quite common in the literature, namely

$$\sum_t t\,(x_{jt} - x_{it}) \geq d_{ij}, \qquad (i,j) \in L \ . \tag{6}$$

This formulation of the temporal constraints is weaker in the sense that inequalities (2) and (3) imply (6), even if the variables x_{jt} are allowed to take on fractional values; see, e.g., Sankaran et al. (1999). While the original formulation of Pritsker et al. (1969) uses the weak formulation (6), the strong formulation (3) was proposed by Christofides et al. (1987).

2.2 Transformation to a minimum cut problem

We now present our main theoretical result, a transformation of the scheduling problem with start-time dependent costs (1) – (5) to a minimum cut problem in a directed graph. For the special case of ordinary precedence constraints and unit processing times (that is, $d_{ij} = 1$ for all $(i,j) \in L$), Chang and Edmonds (1985) have previously suggested such a transformation. They show that this problem can be reduced to the minimum-weight closure problem, which is equivalent to the minimum cut problem (Rhys 1970; Balinski 1970; Chang and Edmonds 1985). The reduction we propose is direct, and it typically results in a sparser minimum cut digraph.

For each job j, we denote by $e(j) \geq 0$ its earliest feasible start time, and by $\ell(j) \leq T - p_j$ its latest feasible start time. The minimum cut digraph $D = (V, A)$ is defined as follows.

Nodes. There is one node v_{jt} for every job j and for all its potential start times t plus the succeeding period. In other words, $V := \{v_{jt} | j \in J, t = e(j), \ldots, \ell(j)+1\} \cup \{a, b\}$, where the two additional nodes a and b represent source and sink of D, respectively.

Arcs. The arc set A consists of assignment, temporal, and auxiliary arcs. Assignment arcs $(v_{jt}, v_{j,t+1})$ are defined for all j and t, resulting in directed chains $(v_{j,e(j)}, v_{j,e(j)+1})$, $(v_{j,e(j)+1}, v_{j,e(j)+2}), \ldots, (v_{j,\ell(j)}, v_{j,\ell(j)+1})$ for all $j \in J$. Temporal arcs $(v_{it}, v_{j,t+d_{ij}})$ are defined for all time lags $(i,j) \in L$, and for all t which fulfill both $e(i)+1 \leq t \leq \ell(i)$ and $e(j)+1 \leq t+d_{ij} \leq \ell(j)$. Finally, a set of auxiliary arcs connects the source and the sink nodes a and b with the remaining network. The auxiliary arcs are given by $(a, v_{j,e(j)})$ and $(v_{j,\ell(j)+1}, b)$, for all $j \in J$.

Arc capacities. The capacity of each assignment arc $(v_{jt}, v_{j,t+1})$ is given by $c(v_{jt}, v_{j,t+1}) := w_{jt}$, and the capacities of all temporal and auxiliary arcs are infinite.

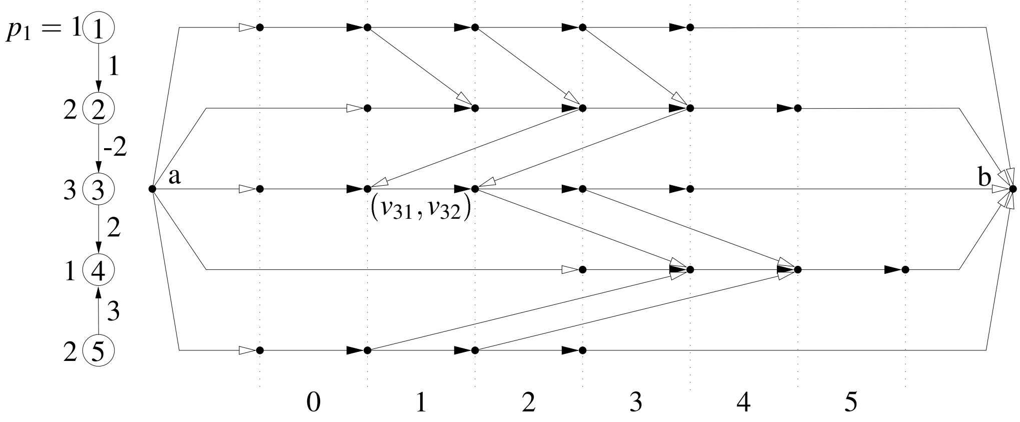

Figure 1: Example for the transformation to a minimum-cut problem.

Figure 1 shows an example of the construction of D. The digraph on the left-hand side represents the relevant data of the underlying example: each node represents a job, and each arc represents a temporal constraint. The values for the time lags are $d_{12} = 1, d_{23} = -2, d_{34} = 2$, and $d_{54} = 3$. The job processing times are $p_1 = p_4 = 1$, $p_2 = p_5 = 2$, and $p_3 = 3$. The time horizon is $T = 6$. Thus the earliest start times are 0, 1, 0, 3, and 0, and the latest start times are 3, 4, 3, 5, and 2, respectively. (Note that we omit the artificial jobs 0 and n in this example.) The digraph on the right-hand side of Figure 1 is obtained by the above described transformation. Arcs marked by a white arrowhead have infinite capacity.

To formulate our main result, we use the following notation. Given $D = (V, A)$, an a-b-cut is an ordered pair $(X, \overline{X})$ of disjoint sets $X, \overline{X} \subset V$ with $X \cup \overline{X} = V$, and $a \in X$, $b \in \overline{X}$. We say that an arc $(u, v) \in A$ is in the cut if $u \in X$ and $v \in \overline{X}$. The capacity $c(X, \overline{X})$ of a cut $(X, \overline{X})$ is the sum of the capacities of the arcs in the cut, $c(X, \overline{X}) := \sum_{(u,\overline{u}) \in (X,\overline{X})} c(u, \overline{u})$. A minimum a-b-cut is an a-b-cut with minimal capacity. Moreover, let us introduce the following notion.

Definition 1. *An a-b cut of the above defined digraph $D = (V, A)$ is called an n-cut if for every job $j \in J$ exactly one assignment arc $(v_{jt}, v_{j,t+1})$ is in the cut.*

The following observation is crucial.

Lemma 1. *Let $(X, \overline{X})$ be a minimum a-b-cut of D. Then there exists an n-cut $(X^*, \overline{X}^*)$ of D with the same capacity. Moreover, given $(X, \overline{X})$, the n-cut $(X^*, \overline{X}^*)$ can be computed in time* $\mathrm{O}(nT)$.

Proof. The proof relies on the fact that all arc capacities of D are non-negative. Let the minimum cut $(X, \overline{X})$ of D be given, and assume that its capacity is finite, otherwise the claim is trivial. Since the auxiliary arcs have infinite capacity, it follows that for each job $j \in J$ at least one assignment arc $(v_{jt}, v_{j,t+1})$ is in the cut $(X, \overline{X})$. Now assume that $(X, \overline{X})$ contains more than one assignment arc for certain jobs $j \in J$. We construct an n-cut $(X^*, \overline{X}^*)$ with the same capacity as follows. For $j \in J$, let t_j be the smallest time index such that $(v_{j,t_j}, v_{j,t_j+1}) \in (X, \overline{X})$. Let $X^* := \bigcup_{j \in J} \{v_{jt} | t \leq t_j\} \cup \{a\}$ and $\overline{X}^* := V \setminus X^*$. Clearly, $X^* \subset X$ and the set of assignment arcs in $(X^*, \overline{X}^*)$ is a proper subset of the corresponding set of assignment arcs of $(X, \overline{X})$. All weights w_{jt} are non-negative. Hence, it suffices to prove that $(X^*, \overline{X}^*)$ does not contain any of the temporal arcs in order to show that $(X^*, \overline{X}^*)$ has the same capacity as $(X, \overline{X})$. Suppose that there exists such a temporal arc $(v_{is}, v_{jt}) \in (X^*, \overline{X}^*)$, $s \leq t_i, t > t_j$. Let $k := t - (t_j + 1) \geq 0$. Then $e(j) + 1 \leq t - k \leq \ell(j)$, since $t - k = t_j + 1$, and all times $t_j, t_j + 1, \ldots, t$ are feasible start times for job j. Moreover, $s - k \leq \ell(i)$, since $s - k \leq s \leq t_i \leq \ell(i)$. Now suppose that $s - k < e(i) + 1$. This implies that $t - k = s - k + (t - s) < e(i) + 1 + (t - s)$. But there is a time lag between i and j of length $t - s$, hence $e(i) + (t - s) \leq e(j)$. This yields $t - k < e(j) + 1$, a contradiction. Thus $s - k \geq e(i) + 1$. In other words, we have $v_{i,s-k} \in X^* \subset X$ and $v_{j,t-k} = v_{j,t_j+1} \in \overline{X}$, and there exists a temporal arc $(v_{i,s-k}, v_{j,t-k}) \in (X, \overline{X})$. Since temporal arcs have infinite capacity, this is a contradiction to the assumption that $(X, \overline{X})$ is a minimum cut of finite capacity. It follows from the definition of $(X^*, \overline{X}^*)$ that it can be computed from $(X, \overline{X})$ in time $\mathrm{O}(nT)$. $\square$

The following theorem establishes the connection between the scheduling problem and the minimum cut problem.

Theorem 1. *There is a one-to-one correspondence between n-cuts $(X, \overline{X})$ of D with finite capacity and feasible solutions x of the project scheduling problem with start-time dependent costs* (1) – (5)*, namely*

$$x_{jt} = \begin{cases} 1 & \text{if } (v_{jt}, v_{j,t+1}) \text{ is in the cut } (X, \overline{X}), \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The capacity $c(X, \overline{X})$ of $(X, \overline{X})$ is equal to the value $w(x)$ of the corresponding scheduling solution x. Moreover, the capacity $c(X, \overline{X})$ of a minimum a-b-cut $(X, \overline{X})$ of D equals the value $w(x)$ of an optimal solution x of (1) – (5).

Theorem 1 is proved with the help of Lemmas 2, 3, and 4 below. Using Lemma 1, any minimum cut of D can be turned into a minimum n-cut in $\mathrm{O}(nT)$ time. By Theorem 1, this yields an optimal solution for the project scheduling problem with start-time dependent costs. Since the digraph D has $\mathrm{O}(nT)$ nodes and $\mathrm{O}(mT)$ arcs, the analysis of the push-relabel-algorithm for maximum flows by Goldberg and Tarjan (1988) implies the following result.

Corollary 1. *The project scheduling problem with start-time dependent costs* (1) – (5) *can be solved in time* $\mathrm{O}(nmT^2 \log(n^2 T/m))$.

If all weights w_{jt} are integer and W is their largest absolute value, Goldberg and Rao's algorithm (1998) leads to a time complexity of $\mathrm{O}(\min\{n^{2/3}mT^{5/3},\ m^{3/2}T^{3/2}\}\log(n^2T/m)\log W)$. Let us next prove Theorem 1. We start with surjectivity.

Lemma 2. *For each feasible solution x of integer program* (1) – (5)*, there exists an n-cut* $(X,\overline{X})$ *of D such that x is the image of* $(X,\overline{X})$ *under the mapping* (7)*. Moreover,* $w(x) = c(X,\overline{X})$.

Proof. Let x be a feasible solution of integer program (1) – (5), and let $w(x)$ be its cost. Due to (2), for each job $j \in J$ there exists exactly one $x_{j,t_j} = 1$, for some t_j. Define a cut in D by setting $X := \bigcup_{j\in J}\{v_{jt} | t \le t_j\} \cup \{a\}$, and $\overline{X} := V \setminus X$. By construction of D, all arcs (v_{j,t_j}, v_{j,t_j+1}), $j \in J$, are arcs of the cut $(X,\overline{X})$, and x is the image of $(X,\overline{X})$ under the mapping (7). Moreover, the sum of the capacities of the assignment arcs in the cut is $w(x)$. Now suppose that there exists another arc in the cut. This must be a temporal arc (v_{is}, v_{jt}), $s \le t_i$, $t > t_j$. Thus, there is a time lag (i, j) with length $d_{ij} = t - s$ between jobs i and j. Since $t_j - t_i < t - s$, we obtain a contradiction to the assumption that x was feasible. Hence, $w(x) = c(X,\overline{X})$. □

The injectivity of the mapping defined by (7) follows by definition. We have to show, though, that its images are in fact feasible solutions of the integer program (1) – (5).

Lemma 3. *For each finite capacity n-cut* $(X,\overline{X})$ *in D, the mapping* (7) *defines a feasible solution x of integer program* (1) – (5) *with* $w(x) = c(X,\overline{X})$.

Proof. Since an n-cut contains exactly one assignment arc for each job $j \in J$, the solution x defined by (7) fulfills equations (2). Given that the n-cut $(X,\overline{X})$ has finite capacity, it can easily be shown that x satisfies the temporal constraints (3) as well. Finally, it is obvious that $w(x) = c(X,\overline{X})$. □

Lemma 4. *The capacity* $c(X,\overline{X})$ *of a minimum a-b-cut* $(X,\overline{X})$ *of D equals the value* $w(x)$ *of an optimal solution x of* (1) – (5).

Proof. The claim immediately follows with the help of Lemmas 1, 2 and 3. □

This concludes the proof of Theorem 1. Note that if all weights w_{jt} are strictly positive, there is a one-to-one correspondence between minimum *a*-*b*-cuts of D and optimal solutions of (1) – (5). This can be inferred from the proof of Lemma 1, since in this case any minimum *a*-*b*-cut is already an *n*-cut. In our computational experiments, arc weights are non-negative. Changing them to strictly positive arc weights is not beneficial.

2.3 Additional remarks

The project scheduling problem with start-time dependent costs can be solved in time $O(nm)$ by longest path calculations (Bellman 1958) if the costs w_{jt} are either monotonically non-decreasing or monotonically non-increasing in t, for all jobs $j \in J$. Other special cases can also be solved more efficiently than the general problem; for instance, the problem with out-tree precedence constraints can be solved in $O(nT)$ time (Gröflin et al. 1982). Notice, however, that with respect to polynomiality results one has to distinguish between problems which require an encoding length of $\Omega(nT)$, which is the case for the problem with general costs w_{jt} discussed here, and problems which allow a more succinct encoding like, e.g., the net present value problem, or problems with piecewise linear, convex cost functions. We refer to Möhring et al. (2001) for a detailed discussion of these issues.

3 Lower Bounds for Resource-Constrained Project Scheduling

We now consider resource-constrained project scheduling problems: In addition to observing temporal constraints, jobs need resources when they are in process. In the model with constant resource requirements, we are given a finite set R of different, renewable resource types, and the capacity of resource $k \in R$ is denoted by R_k. This means that an amount of R_k units of resource k is available throughout the project. During its processing, job j requires an amount of r_{jk} units of resource k (for $j \in J$ and $k \in R$). The jobs have to be scheduled in such a way that the resource consumption does not exceed the resource capacity at any time. The objective considered now is different from the one studied in the previous section. It is the project makespan, which is the completion time of the last job. Resource-constrained project scheduling problems are among the most intractable combinatorial optimization problems. Their hardness is perhaps best underlined by the observation that the node coloring problem in graphs can be formulated as a resource-constrained project scheduling problem with makespan objective; see Schäffter (1997) and Blazewicz, Lenstra, and Rinnooy Kan (1983). Thus, as for node coloring, there is no polynomial-time approximation algorithm with a performance guarantee of $n^{1-\varepsilon}$ for any $\varepsilon > 0$, unless NP = ZPP (Feige and Kilian 1998). This also implies limits on the computation of lower bounds. Yet, real-world instances still call for good and fast computable solutions and lower bounds. For surveys on resource-constrained project scheduling, we refer to Węglarz (1999) and Brucker, Drexl, Möhring, Neumann, and Pesch (1999).

In fact, there are numerous publications on the computation of lower bounds on the minimal makespan for resource-constrained project scheduling problems; let us briefly review those which are most relevant to our work. For problems with ordinary precedence constraints, lower bounds from time-indexed linear programming relaxations were analyzed by Christofides et al. (1987) as well as Cavalcante, de Souza, Savelsbergh, Wang, and Wolsey (2001b). Several other linear programming based lower bounds were proposed by Mingozzi, Maniezzo, Ricciardelli, and Bianco (1998). For problems that also involve maximal time lags between jobs, lower bounds were computed by Heilmann and Schwindt (1997), using a destructive approach. The idea is to reject fictitious upper bounds by proving infeasibility; this concept was also used for problems with ordinary precedence constraints by Klein and Scholl (1999). Based on one of the relaxations by Mingozzi et al. (1998) and using a destructive approach combined with constraint propagation techniques, Brucker and Knust (2000) obtain the strongest lower bounds currently known on one of the benchmark test sets. Recently, Demassey, Artigues, and Michelon (2001a, 2001b) have started further experiments to combine constraint programming and linear programming techniques; while their initial results do not show a clear trend, they remain interesting.

In this section, we revert to the Lagrangian relaxation suggested by Christofides et al. (1987), which is based on the previously mentioned time-indexed integer programming formulation of Pritsker et al. (1969). Appropriate Lagrangian multipliers are calculated with the help of standard subgradient optimization techniques. Since every Lagrangian subproblem can be interpreted as a project scheduling problem with start-time dependent costs, we solve it by using the preflow-push algorithm of Goldberg and Tarjan (1988) in its implementation by Cherkassky and Goldberg (1997). In our computational study, we compare the Lagrangian lower bounds as well as the required computation times to the hitherto best lower bounds, on all benchmark instances. In addition, we report on our experience with two different linear programming relaxations that emerge from the time-indexed problem formulation. It turns out that the Lagrangian approach offers a favorable tradeoff between the quality of the lower bounds and the necessary computational effort. Computational results on Lagrangian lower bounds are presented in Sections 5.2 and 5.4.

3.1 Integer programming formulation

In order to model the resource-constrained project scheduling problem as an integer program, recall that the project makespan is given by the start time S_n of the artificial job n. Hence, we obtain:

$$\text{minimize} \qquad \sum_t t\, x_{nt} \tag{8}$$

subject to (2), (3), (4), (5), and

$$\sum_j r_{jk}\Big(\sum_{s=t-p_j+1}^{t} x_{js}\Big) \le R_k, \qquad k \in R,\ t = 0,\ldots,T. \tag{9}$$

Inequalities (9) ensure that the jobs simultaneously processed at time t do not consume more resources than available. This formulation can easily be generalized to time-dependent, piecewise constant resource profiles, i.e., $R_k = R_k(t)$ and $r_{jk} = r_{jk}(t)$, $t = 0,\dots,T$. In fact, although we discuss in the following the case of time-independent resource profiles only, the presented results carry over to the general case. Computational results for both models are discussed in Section 5.

3.2 Lagrangian relaxation

In order to compute lower bounds on the optimal objective value, Christofides et al. (1987) propose to dualize the resource constraints (9), and introduce non-negative Lagrangian multipliers $\lambda = (\lambda_{tk})$, $t \in \{0,...,T\}$, $k \in R$. One obtains the following Lagrangian subproblem.

$$\begin{array}{ll} \text{minimize} & \sum_t t\, x_{nt} + \sum_j \sum_t \Big(\sum_{k\in R} r_{jk} \sum_{s=t}^{t+p_j-1} \lambda_{sk} \Big) x_{jt} - \sum_t \sum_{k\in R} \lambda_{tk} R_k \\ \text{subject to} & (2), (3), (4), \text{and } (5). \end{array} \tag{10}$$

If one introduces weights

$$w_{jt} := \sum_{k\in R} r_{jk} \sum_{s=t}^{t+p_j-1} \lambda_{sk} \text{ if } j \neq n, \quad \text{and } w_{jt} := t \text{ if } j = n, \tag{11}$$

the Lagrangian subproblem (10) can be rewritten as

$$\text{minimize} \qquad \sum_j \sum_t w_{jt}\, x_{jt} - \sum_t \sum_{k\in R} \lambda_{tk} R_k \qquad \text{subject to (2), (3), (4), and (5).} \tag{12}$$

Evidently, once the constant term $\sum_t \sum_{k\in R} \lambda_{tk} R_k$ is neglected, (12) is a project scheduling problem with start-time dependent costs as discussed in Section 2 before. The weights w_{jt} defined in (11) depend on the vector of Lagrangian multipliers λ; since these multipliers are non-negative, the weights w_{jt} are non-negative as well. Note that the above Lagrangian relaxation of (2), (3), (4), (5), (8), and (9) is not restricted to makespan minimization, but can as well be applied to *any* objective function which can be expressed linearly in the x-variables. Hence, the procedure proposed below is applicable to a variety of scheduling problems, including the minimization of the weighted sum of completion times, problems with earliness-tardiness costs, resource investment problems (Möhring 1984), and net-present-value objectives (Grinold 1972). Inspired by a preliminary version of our paper (Möhring et al. 1999), Drexl and Kimms (2001), Kimms (2001), and Selle (1999) have exploited the transformation of Lagrangian subproblems to minimum cut problems for some of the objective functions mentioned above.

3.3 Lagrangian and linear programming relaxations

For any vector of non-negative Lagrangian multipliers $\lambda = (\lambda_{tk})$, $t = 0,\dots,T$, $k \in R$, the optimal solution value of the Lagrangian subproblem (12) is a lower bound on the value of an optimal solution of the resource-constrained project scheduling problem (2), (3), (4), (5), (8), and (9). If w_λ denotes the value of an optimal solution for the Lagrangian subproblem for a fixed vector of multipliers λ, the *Lagrangian dual* is the maximization problem $\max_{\lambda \geq 0} w_\lambda$. Since the polytope described by inequalities (2), (3), and (4) is integral (see Möhring et al. (2001) for references), the optimal solution for the Lagrangian dual equals the value of an optimal solution for the linear programming relaxation (2), (3), (4), (8), and (9) (Everett III 1963). Hence, the optimal solution value for the linear programming relaxation (2), (3), (4), (8), and (9) is an upper bound for the solution of the Lagrangian subproblem, for any vector of non-negative multipliers λ. Our computational evaluation of the two alternative approaches shows that the Lagrangian dual can typically be solved more efficiently in practice; we refer to Section 5.2 for details. If inequalities (3) are replaced by the weaker inequalities (6), however, it turns out that the corresponding linear programming relaxation is generally much easier to solve. This was also observed by Cavalcante et al. (2001b). The weaker linear programming relaxation may yield worse lower bounds, though, as is demonstrated in the following example.

Observation 1. *For the makespan objective, there exist instances for which the optimal objective function value of the strong LP relaxation* (2), (3), (4), (8), *and* (9) *is arbitrarily close to being* $3/2$ *times the optimal objective function value of the weak LP relaxation* (2), (4), (6), (8), *and* (9).

Proof. We consider a family of instances with five jobs with unit processing times, and one resource of capacity $N \in \mathbb{N}$. There are two jobs which consume N units of the resource; they are forced to start at times 0 and 2, respectively, using corresponding minimal and maximal time lags. There is one job which requires one unit of the resource, predecessor of two more jobs which require $N-1$ units each. The optimal makespan for the weak LP relaxation is 3, whereas the optimal makespan for the strong LP relaxation is arbitrarily close to 4.5 for large N. $\square$

Consequently, the Lagrangian relaxation may also yield stronger bounds than the weak linear programming relaxation. This is indeed confirmed by our computational results presented in Section 5. Finally, it is worth mentioning that simple examples show that the integrality gap, that is, the ratio of the value of an optimal integral solution to the value of an optimal fractional solution is in general unbounded for both linear programming formulations.

3.4 Strengthening lower bounds by feasible cuts

In our computations, we have actually used a strengthened version of the resource constraints, which replaces (9). The stronger inequalities were proposed by Christofides et al. (1987) and reinforce that no job should be scheduled at the same time or after the artificial job n.

$$\sum_{j\neq n} r_{jk}\Big(\sum_{s=t-p_j+1}^{t} x_{js}\Big) \leq R_k\Big(1-\sum_{s=0}^{t} x_{ns}\Big), \quad k\in R, t=0,\dots,T\,. \tag{13}$$

The use of (13) has a nice interpretation in terms of the Lagrangian relaxation (12), since, instead of $w_{nt}=t$, it leads to the weights $w_{nt}=t+\sum_{s=t}^{T}\sum_{k\in R}\lambda_{sk}R_k$ for the artificial job n. Hence, an early start of job n is penalized stronger by using inequalities (13) instead of (9).

The Lagrangian approach is capable of incorporating other families of valid inequalities as well. Once these feasible cuts are dualized, the structure of the minimum-cut digraph D remains the same, only the corresponding arc capacities w_{jt} change. Hence, for some of our experiments we have considered additional inequalities. Let $F\subseteq J$ be a subset of jobs no two of which can be scheduled simultaneously, either because of resource consumption, or because of temporal constraints. This leads to the following family of inequalities, which has also been used by Christofides et al. (1987):

$$\sum_{j\in F}\Big(\sum_{s=t-p_j+1}^{t} x_{js}\Big) \leq 1 \quad t=0,\dots,T\,. \tag{14}$$

In order to compute a collection of such inclusion-maximal sets F, we use a simple greedy heuristic: Consider the jobs in some given order, start with F being the first job in this order, and add job j to F if j cannot be processed simultaneously with any job that has been previously added to F. We have used ten folklore priority lists of jobs to obtain a collection of inequalities of this type. The improvements obtained by additionally using inequalities (14) are documented in Tables 1 and 2 in Section 5.2.

3.5 Lagrangian multiplier computation

We use a standard subgradient method to compute near-optimal vectors of Lagrangian multipliers λ_{tk}, $t=0,\dots,T$, $k\in R$, as described, e.g., in Bertsekas (1999, Ch. 6.3). Given the vector of Lagrangian multipliers λ^i of the i-th iteration, we define $\lambda^{i+1}:=\big[\lambda^i+\delta^i g^i\big]^+$, where, as usual, $[\cdot]^+$ denotes the non-negative part of a vector. Moreover, g^i is a subgradient at λ^i of the function that maps λ to the optimal objective value of (12). If x^i is an optimal solution of (12) for the vector λ^i of Lagrangian multipliers, then $g^i_{k,t}=\sum_j r_{jk}(\sum_{s=t-p_j+1}^{t} x^i_{js})\ -R_k(1-\sum_{s=0}^{t} x_{ns})$. Eventually, the step size δ^i is given by $\delta^i:=\delta(w^*-w_{\lambda^i}(x^i))/||g^i||^2$, where w^* is an upper bound for the optimal value of the Lagrangian dual. The scalar parameter δ is adjusted as a function of the lower bound improvement: If the lower bound does not improve substantially within three iterations, then δ is reduced by a factor of 0.8. If no substantial improvement can be achieved within ten iterations, we stop the algorithm. The corresponding parameter adjustments result from our empirical experiments. In fact, the following refinement proposed by Camerini, Fratta, and Maffioli (1975) leads to improved convergence. Instead of moving into the direction of the subgradient g^i, we use $\lambda^{i+1} := \left[\lambda^i + \delta^i d^i\right]^+$, where $d^i := g^i$, if $i = 0$, and $d^i := g^i + \beta g^{i-1}$, otherwise. Here, $0 \leq \beta \leq 1$ is a scalar that depends on the angle between two successive subgradients.

4 From Minimum Cuts to Feasible Solutions

Relaxation-based list scheduling algorithms have recently led to several approximation results for NP-hard machine scheduling problems. The general idea is to solve some relaxation of the problem at hand, e. g., a linear programming relaxation, and then extract information from this solution to construct provably good schedules. Scheduling jobs in order of their so-called α-completion times in the respective relaxation is one way of doing this. The papers by Phillips, Stein, and Wein (1998), Hall, Schulz, Shmoys, and Wein (1997), Chekuri, Motwani, Natarajan, and Stein (2001), Goemans, Queyranne, Schulz, Skutella, and Wang (2002), and Munier, Queyranne, and Schulz (2002) are just some of the references in this direction. Cavalcante et al. (2001b) as well as Savelsbergh et al. (1998) show that such approaches can also be successfully applied to real-world instances. Likewise, our empirical studies in Sections 5.3 and 5.4 show that Lagrangian relaxation is not only useful to compute lower bounds on the project makespan, but is also suited to compute feasible solutions with relatively small computational effort. The basic idea is motivated by the intuition that in the course of the subgadient optimization, violations of the resource constraints tend to be reduced. Hence, a solution of the Lagrangian relaxation contains valuable information on how resource conflicts can be resolved. We take advantage of this information by using simple list scheduling algorithms that are based on α–completion times of the jobs in the solutions of the Lagrangian subproblems.

To relate the results obtained to other algorithms from the literature, we utilize a survey on heuristic approaches by Kolisch and Hartmann (1999). Therein, some of the recent local search and list scheduling heuristics are evaluated with respect to their performance on the instances from Kolisch and Sprecher (1996). This includes a genetic algorithm by Hartmann (1998, 2002), a simulated annealing algorithm by Bouleimen and Lecocq (2002) as well as sampling-based list-scheduling heuristics by Kolisch (1996). Additionally, we compare our algorithm to an ant colony optimization algorithm by Merkle, Middendorf, and Schmeck (2000), a tabu search algorithm by Nonobe and Ibaraki (2001) as well as a constraint propagation based branch-and-bound algorithm by Dorndorf, Pesch, and Phan Huy (2000a). Our computational results suggest that the Lagrangian-based approach is capable of providing solutions that are comparable to those produced by state-of-the-art algorithms. Moreover, in contrast to purely primal heuristics the Lagrangian approach simultaneously provides strong lower bounds. We also evaluate our algorithm on the labor-constrained instances described by Heipcke et al. (2000) and compare it to previous approaches, including linear programming based list scheduling algorithms by Cavalcante et al. (2001b), a tabu search algorithm by Cavalcante and de Souza (1997), recently improved by Cavalcante, Cavalcante, Ribeiro, and de Souza (2001a), as well as a constraint propagation based branch-and-bound algorithm by Heipcke (1999).

4.1 List scheduling

The literature typically distinguishes two list scheduling rules, which are referred to as the parallel and the serial scheme, respectively. The parallel scheme goes back to Graham (1966); it is therefore also called Graham's list scheduling, whereas the serial scheme is sometimes also called job-based list scheduling. In both cases a priority list is given that determines the order in which the jobs are considered. Parallel list scheduling proceeds over time, starting at time $t = 0$. At any time t, as many available jobs as possible are scheduled, picked in the order given by the list. If no further job can feasibly be scheduled at time t, t is augmented to the time of the next event. (Here, an event is either a completion time $S_i + p_i$, or a date $S_i + d_{ij}$, $(i, j) \in L$). Serial list scheduling proceeds job by job. In the order given by the priority list, each job is scheduled as early as possible with respect to the jobs already scheduled. Both list scheduling algorithms can be implemented to run in $O(|R|n^2)$ time when the resource availability and the resource consumption of every job is constant; see, e.g., Kolisch and Hartmann (1999). List scheduling algorithms can easily be adapted to handle the case when resource consumption varies over time, which results in a higher running time that also accounts for the breakpoints in the resource profiles. It is well-known that both list scheduling rules are incomparable with respect to the quality of the schedules they produce. In order to combine the respective advantages of the parallel and the serial scheme, we have additionally implemented a serial list scheduling algorithm with limited look-ahead: Given a natural number ℓ, we compute the earliest start times of the first ℓ available jobs from the list, and schedule the job with the smallest earliest start time. The time complexity of this algorithm is $O(\ell|R|n^2)$. Within our computations, we have used values of 1, 2, 4, and 8 for ℓ.

4.2 List scheduling by α-completion times

Recall that, for any vector of Lagrangian multipliers, the solution of the Lagrangian relaxation (12) is a time-feasible schedule which, in general, violates some of the resource constraints (9). We generate schedules that are time-feasible and resource-feasible by applying the previously described list scheduling algorithms fed with priority lists derived from the order of the jobs in the solution of the Lagrangian relaxation. Given a time-feasible schedule S and some $0 \leq \alpha \leq 1$, the α-completion time of job j in S is $C_j(\alpha) := S_j + \alpha p_j$. If the temporal constraints are acyclic and if $d_{ij} \geq \max\{0, p_i - p_j\}$ for all $(i, j) \in L$, which is, e.g., the case for ordinary precedence constraints, then the ordering of the jobs according to non-decreasing α-completion times is compatible with the given temporal constraints. It can therefore be used as a priority list for both list scheduling schemes. Furthermore, the number of different priority lists emerging from different values of α is polynomially bounded.

Observation 2. *Let S be a time-feasible schedule, and let q be the maximal number of parallel jobs in S. The number of different orderings of the jobs according to their α-completion times for $\alpha \in [0, 1]$ is at most $nq + 1$.*

4.3 The Lagrangian heuristic

Let us now sketch the combined algorithm that computes both lower bounds and feasible solutions. First, to obtain an initial valid upper bound T on the makespan, we use the parallel list scheduling algorithm described in Section 4.1, on the basis of 10 folklore priority lists, such as shortest/longest processing time first, minimum slack, etc. Then, in each iteration of the subgradient optimization algorithm, a time-feasible, but likely resource-infeasible schedule is computed by solving the Lagrangian subproblem (12) as described in Section 2. The cost of this time-feasible schedule, in terms of the w_{jt} defined in (11), minus the constant term in (12), is a valid lower bound for the makespan of the resource-constrained project scheduling problem (2), (3), (4), (5), (8), and (9). Using orderings according to α-completion times of jobs, we then compute feasible solutions by means of the list scheduling algorithms described above. In fact, we observed that the number of combinatorially different values of α was roughly $n/4$ for the considered instances. For our computations, however, we did not evaluate the priority lists for all relevant values of α, but we only considered 10 different values for α. According to our experiments, this gives a reasonable tradeoff between the required compution times and the solution quality, for the instances considered. The stopping criterion of the algorithm depends on the rate of convergence of the subgradient optimization procedure as described in Section 3. Of course, the algorithm is also aborted as soon as lower and upper bounds match.

5 Computational Study

The computational study is divided into four parts. We start by describing the setup as well as the benchmark instances. In Section 5.2, we analyze the Lagrangian lower bounds, and compare them with the corresponding linear programming relaxations as well as the currently best known lower bounds. Section 5.3 discusses the computation of feasible solutions. Finally, in Section 5.4 we report on our experiments with instances that mimic a typical chemical production process at BASF AG, Ludwigshafen, Germany.

5.1 Setup and benchmark instances

Our experiments were conducted on a Sun Ultra 2 with 200 MHz clock pulse and 512 MB of memory, operating under Solaris 2.7. The code was written in C++ and compiled with the GNU g++ compiler version 2.91.66. For solving minimum cut problems, we used the highest-label maximum flow code by Cherkassky and Goldberg (1997). It was written in C and compiled with the GNU gcc compiler version 2.91.66. Both compilers used the -O3 optimization option. To solve the linear programming relaxations, we used ILOG CPLEX version 6.5.3.

We tested our algorithm using three different types of benchmark instances. First, we considered ProGen instances with 60, 90, and 120 jobs. These are project scheduling problems with ordinary precedence constraints. They are part of the project scheduling problem library PSPLIB (2000). These instances have been generated by Kolisch and Sprecher (1996), systematically modifying three parameters. The *network complexity* reflects the average number of direct successors of a job, the *resource factor* describes the average number of resource types required by a job, and the *resource strength* is a scaling factor which measures the scarcity of the resources. The resource strength varies between 0 and 1. If it is equal to 1, the earliest start schedule with respect to the precedence constraints is already feasible. If it is equal to 0, the resource availability is minimal in the sense that for each resource type there exist jobs which require the full capacity. The library contains 480 instances with 60 and 90 jobs, respectively, and 600 instances with 120 jobs. Each of the instance sets with 60 and 90 jobs contains 120 instances with a resource strength parameter 1. Hence, we only considered the remaining 360 non-trivial instances for our computations. The job processing times for all instances are uniformly distributed between 1 and 10, and the number $|R|$ of different resource types is four. The library also contains best known upper and lower bounds on the optimal makespan for all instances. The makespan of the currently best known solutions for the instances with 60 jobs varies between 44 and 157, for the instances with 90 jobs it is between 60 and 182, and for the instances with 120 jobs it is between 66 and 301.

The ProGen/max instances generated by Schwindt (1996) additionally feature maximal time lags between the jobs. We considered 1059 instances which consist of 100 jobs each. The number $|R|$ of different resource types is five. The control parameters are similar to those of the ProGen instances described above, except that an additional parameter controls the number of cycles in the digraph of temporal constraints. They are also part of the library PSPLIB (2000), and benchmark solutions and lower bounds are maintained at the library PSPLIB/max (2000). The makespan of the best known solutions for these instances varies between 185 and 905. For further details on the test sets we refer to Kolisch and Sprecher (1996), Schwindt (1996) and Kolisch et al. (1999).

We also considered a set of 25 instances which mimic a labor-constrained scheduling problem at BASF AG, Ludwigshafen. Two of these instances stem directly from a chemical production process at BASF AG, Ludwigshafen, and the remaining 23 instances have been generated accordingly (Heipcke et al. 2000). In these instances, each job consists of several consecutive production steps, so-called tasks, which must be executed consecutively without interruption. Each task requires a certain amount of personnel, hence jobs have time-varying resource requirements. The resource constraints are imposed by a limited number of available personnel, which is constant over time. More details can be found in Kallrath and Wilson (1997, Ch. 10.5), Heipcke (1999), and Heipcke et al. (2000). For these instances, the number of jobs varies between 21 and 109, and the number of tasks, that is, the number of breakpoints in the resource profiles, varies between 49 and 2014.

5.2 Lower bounds by Lagrangian relaxation

First, we will relate the lower bounds obtained by Lagrangian relaxation to the currently best known lower bounds, based on the ProGen instances from the PSPLIB (2000). For all results in this section, the upper bound T for the minimal project makespan is the currently best known solution value, taken from the PSPLIB (2000). These currently best known bounds are due to Brucker and Knust (2000); they have been obtained by a composition of two different algorithms, constraint propagation and a linear programming relaxation of Mingozzi et al. (1998). The rows in Table 1 refer to the ProGen instances with 60, 90 and 120 jobs, respectively. The columns show respectively the number of instances considered (#inst.), the average improvement over the critical path lower bound in percent (Dev. CP), the average computation time (CPU), as well as the average number of iterations of the subgradient optimization algorithm (#it). Unless stated differently, all computation times are in seconds. The last two columns show the bounds of Brucker and Knust (2000) in terms of the deviation from the critical path lower bound and computation times. These bounds are also available from the PSPLIB (2000), and the computation times have been taken from Brucker and Knust (2000). All figures are averaged over the respective number of instances, and all figures in parenthesis denote the corresponding maxima. Notice that we included additional

Table 1: Comparison of quality and computation times for lower bounds.

		Lagrangian LB + (14)			*Brucker & Knust*	
#jobs	#inst.	Dev. CP	CPU	#it	Dev. CP	CPU*
60	360	7.46%	2.4	49	10.56%	6.7
		(79%)	(32)	(172)	(86%)	(62)
90	360	7.73%	7.2	45	9.6%	96
		(85%)	(77)	(175)	(90%)	(165)
120	600	19.96%	41	86	23.48%	355**
		(146%)	(537)	(218)	(168%)	(72h)

* On a Sun Ultra 2 with 167 MHz (Brucker and Knust 2000)
** Refers to 481 inst.; CPU time for all remaining 119 inst.: 72h

cuts of the type (14) for these experiments. Obviously, the bounds by Brucker and Knust (2000) are stronger than the ones computed by the Lagrangian approach proposed here. However, even though their computations have not been conducted in exactly the same setting, one can infer that the corresponding computation times are significantly higher, especially for the large instances with 120 jobs. For these instances, the Lagrangian approach requires a maximal computation time of less than 9 minutes (537 sec.), which is in sharp contrast to the 72 hours needed by Brucker and Knust (2000).

Table 2 compares the Lagrangian approach to the corresponding linear programming relaxation (2), (3), (4), (8), and (9), again based on the ProGen instances with 60, 90, and 120 jobs. The computation times for the LP relaxation are given for both primal simplex (ps) and barrier solver (ba). We excluded the additional inequalities (14) for this experiment, since we could not solve the strong LP relaxation within days of computation time for some of the instances. Hence, the quality of the Lagrangian lower bounds in Table 2 is slightly inferior in comparison to Table 1. Again, all figures are averaged over the respective number of instances, and figures in parenthesis denote the corresponding maxima. Unless stated differently, all computation times are in seconds.

It turns out that the primal simplex method solves this linear programming relaxation much faster than the barrier method does. With the barrier method, the instances with 120 jobs could not be solved in reasonable time, hence the data is missing in Table 2. More importantly, these computation times are in fact drastically higher than the computation times required to (approximately) solve the Lagrangian dual. The gap between the respective average improvements over the critical path lower bound is caused by slow convergence of the subgradient optimization algorithm in some of the cases. We additionally experimented with linear programming relaxations which use time-indexed variables $z_{jt} = \sum_{s=0}^{t} x_{jt}$ instead. These formulations have also been used by Cavalcante et al. (2001b), and in some cases they can be solved faster. However, on average the

Table 2: Lagrangian and linear programming relaxations.

#jobs	#inst.	*Lagrangian LB* Dev. CP	CPU	#it	*strong LP* Dev. CP	CPU(ps)	CPU(ba)
60	360	6.91%	2.1	49	7.28%	124	203
		(79%)	(30)	(168)	(82%)	(1h)	(1.5h)
90	360	7.52%	6.8	44	7.96%	245	920
		(85%)	(77)	(174)	(88%)	(1.5h)	(8.5h)
120	600	19.63%	39	85	20.51%	2241	—
		(146%)	(475)	(220)	(155%)	(23h)	—

computation times are of the same order of magnitude as the computation times required to solve the linear programming relaxations in x-variables.

Moreover, a comparison of the figures in Tables 1 and 2 shows that the additional inequalities (14) do not have a substantial effect on the computation times for the Lagrangian approach. This is due to the fact that the addition of these inequalities affects the capacities of the maximum flow networks, but not their topology. Although the capacities clearly have an impact on the performance of the preflow-push algorithm, it turned out that the overall impact on the computation times was marginal. Using the additional feasible cuts (14), the quality of the lower bounds can be improved, though.

Table 3 suggests that almost the same lower bounds are obtained when using the weak instead of the strong linear programming relaxation. While the computation times for the instances with 60

Table 3: Weak LP relaxation and impact of additional inequalities (14).

#jobs	#inst.	*weak LP* Dev. CP	CPU(ps)	CPU(ba)	*weak LP* + (14) Dev. CP	CPU(ps)	CPU(ba)
60	360	6.93%	4.0	3.2	7.38%	8.5	3.7
		(82%)	(279)	(27)	(82%)	(490)	(28)
90	360	7.67%	6.2	6.1	7.82%	14	6.8
		(88%)	(80)	(36)	(88%)	(362)	(37)
120	600	20.10%	30	17	20.50%	79	19
		(155%)	(448)	(134)	(155%)	(2817)	(147)

and 90 jobs are comparable to those of the Lagrangian approach, the barrier method (ba) requires less computation time than the Lagrangian approach for the instances with 120 jobs. The averages shown in Table 3, however, do not tell the full truth about the quality of the lower bounds: For example, the strong linear program provides better bounds than the weak linear program for 208 out of the 600 instances with 120 jobs. For 117 of these 208 instances, the Lagrangian approach achieves a stronger lower bound than the weak linear program as well, with a maximum deviation of 3%. For 187 instances, however, the Lagrangian bound is weaker than the one obtained with the weak linear program.

The results for the ProGen/max instances of the PSPLIB are shown in Table 4, again in terms of deviation from the critical path lower bound (Dev. CP). We compare the Lagrangian approach with and without additional inequalities (14) to the weak LP-relaxation. The strong LP relaxation could not be solved at all for several of these instances. Again, the table shows the average and the maximal CPU time required by the primal simplex method (ps) and the barrier method (ba). The table also contains the currently best known lower bounds, which can be obtained from (PSPLIB/max 2000). These lower bounds have been computed by means of different preprocessing algorithms (Heilmann and Schwindt 1997), and as a by-product from branch-and bound algorithms (Dorndorf et al. 2000b; Schwindt 1998; Fest et al. 1998). Computation times are not available. All figures are averaged over only 403 instances of the test set, since the optimal solution matches the critical path lower bound for the remaining 656 instances. Figures in parenthesis denote the corresponding maxima, and unless stated differently, all computation times are in seconds.

Using the primal or dual simplex algorithm, we could not solve the weak LP relaxation with the additional inequalities (14), due to excessive computation times. The ILOG CPLEX barrier code, however, was able to solve the weak LP relaxation with the additional inequalities (14). The average (maximal) computation time for the 403 instances was 411 seconds (2.1 hours), and the average (maximal) improvement over the critical path lower bound was 6.08% (62%). For 204 out of these 403 instances the Lagrangian approach provides stronger lower bounds than the weak linear program, with a maximum deviation of 6.5%. Moreover, in comparison to the running times of the Lagrangian approach, the computation times to solve the weak LP relaxation are quite large, even with the barrier code. Finally, note that for 91 out of the 403 non-trivial instances, we could improve upon the previously best known lower bounds with the Lagrangian approach.

Let us next analyze the dependence of the running times on the input parameters, based on the ProGen instance set of the PSPLIB. Since our approach is based on a time-indexed formulation,

Table 4: Lagrangian, weak LP, and best known lower bounds for ProGen/max instances.

		Lagr. LB + (14)		*Lagr. LB*		*weak LP*			*best known LB*
#jobs	#inst.	Dev. CP	CPU	Dev. CP	CPU	Dev. CP	CPU(ps)	CPU(ba)	Dev. CP
100	403	6.70%	45	6.07%	29	5.68%	1923	184	8.64%
		(57%)	(538)	(57%)	(424)	(62%)	(61h)	(2.5h)	(65%)

it is not surprising that the time horizon T turns out to be the dominating parameter. A large time horizon T can be due to scarce resources, a high resource factor, and a high network complexity. Figure 2 shows (a) how the computation time per iteration of the subgradient optimization depends on T, and (b) how the convergence rate of the subgradient optimization varies with T. The data is based on the 600 ProGen instances with 120 jobs. The regression curve in Figure 2 (a) corresponds

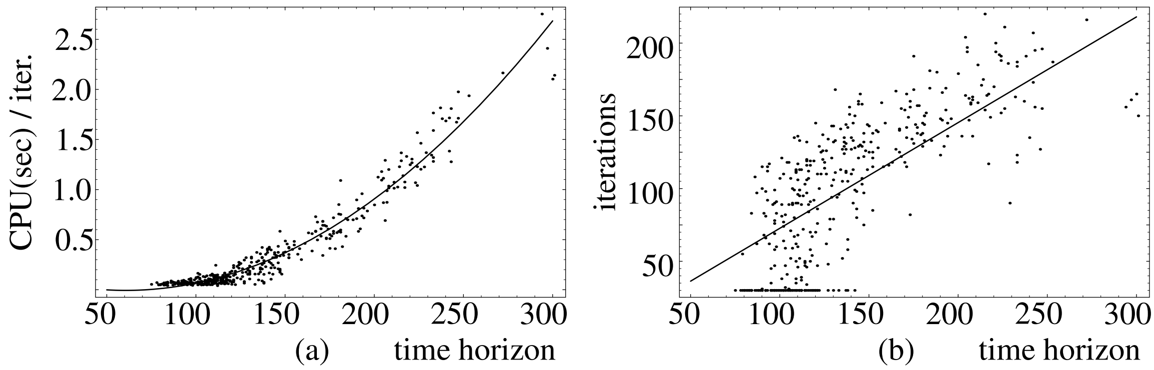

Figure 2: CPU per iteration and rate of convergence depending on T.

to the theoretical bound of $T^2\sqrt{T}$ (we use the highest-label implementation of the preflow-push algorithm). The fact that the number of iterations increases with the time horizon, as suggested by Figure 2 (b), is not because a large time horizon necessarily leads to slow convergence. This is mainly due to the fact that a large time horizon is correlated with scarce resources. For such instances, the deviation between the critical path lower bound and the Lagrangian lower bound is high (see Figure 3). In this case the subgradient optimization tends to converge slower. In fact, we observed a correlation of all three parameters (network complexity, resource factor, resource strength) with both the computation time per iteration and the convergence rate. However, while the time horizon is the dominating parameter for the computation time per iteration, it is the resource strength that is mainly responsible for the number of required iterations in the subgradient optimization. (Both parameters are strongly correlated.)

Figure 3 relates the quality of the Lagrangian lower bounds to the strong LP bounds, the currently best known lower bounds by Brucker and Knust (2000) and the currently best known feasible solutions for the 600 ProGen instances with 120 jobs. The figure depicts the average deviation from the critical path lower bound (in %), depending on the resource strength parameter, and it also contains a plot for the feasible solutions we computed with the Lagrangian approach (see Sections 4 and 5.3 below).

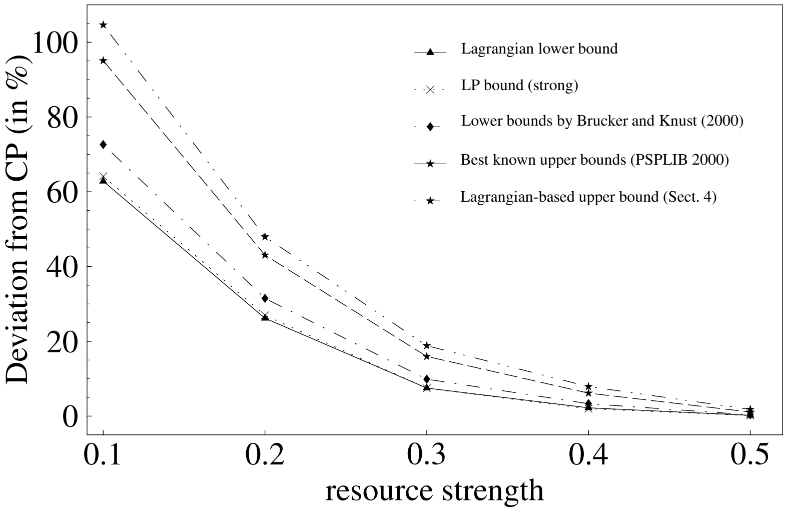

Figure 3: Quality of lower bounds and solutions.

5.3 Lagrangian-based feasible solutions

In this section, we discuss the performance of the combined algorithm which computes both lower bounds and feasible solutions as proposed in Section 4, in comparison to other algorithms from the literature, based on the ProGen instances (Kolisch and Sprecher 1996). For results on the labor-constrained instances from BASF AG, we refer to Section 5.4.

The first two columns of Table 5 display the number of jobs per instance and the number of instances considered. The table further shows the deviation of the time horizon T from the critical

Table 5: Quality and computation times for Lagrangian-based solutions.

		Time Horizon T	*Lagrangian UB*						
#jobs	#inst.	Dev. CP	CPU	#it.	Dev. CP	Dev. LB_{LR}	Dev. UB_{best}	opt.	best
60	360	20.8 %	6.9	53	16.3 %	7.3 %	1.0 %	193	228
		(130 %)	(57)	(195)	(116 %)	(36 %)	(9.0 %)		
90	360	19.6 %	16.2	49	15.6 %	6.4 %	1.0 %	215	234
		(120 %)	(119)	(189)	(110 %)	(36 %)	(8.1 %)		
120	600	42.1 %	72.9	95	36.0 %	11.6 %	2.4 %	156	182
		(226 %)	(654)	(222)	(217 %)	(42 %)	(9.5 %)		

path lower bound (Dev. CP). Here, T has been computed using the parallel list scheduling algorithm with ten different standard priority rules. The next columns display the average computation times per instance in seconds (CPU), and the corresponding average number of iterations in the subgradient optimization (#it). Next, we list the deviations of the solutions from two different lower bounds, the critical path lower bound (Dev. CP) and the Lagrangian lower bound (Dev. LB_{LR}). In addition, the deviation from the currently best known solutions (Dev. UB_{best}) is shown. The latter are maintained at the library PSPLIB (2000) and have been obtained by various authors over the years, using different, partly time-intensive algorithms including branch-and-bound and several local search algorithms. Finally, we display the number of instances that have been solved optimally with our procedure by computing matching lower and upper bounds (opt.) as well as the number of instances where a solution was found which matches the currently best known solution (best). We did not improve upon the currently best known solutions. Numbers in parenthesis denote the corresponding maxima.

In order to relate these figures to other algorithms, Table 6 shows the average deviations from the critical path lower bound for a collection of recent algorithms from the literature. The figures

Table 6: Comparison of a collection of different algorithms from the literature.

Algorithm Type	Reference	Av. dev. from crit. path (in %)
genetic algorithm	Hartmann (2002)	35.4
tabu search	Nonobe and Ibaraki (2001)	35.9
Lagrangian heuristic	this paper	36.0
ant colony optimization	Merkle et al. (2000)	36.6
genetic algorithm	Hartmann (1998)	36.7
constraint propagation	Dorndorf et al. (2000a)	37.1
simulated annealing	Bouleimen and Lecocq (2002)	37.7
sampling	Kolisch (1996)	38.7

are based on 600 ProGen instances with 120 jobs, and have been taken from the survey paper by Kolisch and Hartmann (1999) as well as from the papers of Merkle et al. (2000), Dorndorf et al. (2000a), Hartmann (2002), and Nonobe and Ibaraki (2001). The local search algorithms by Hartmann (1998, 2002), Merkle et al. (2000), Bouleimen and Lecocq (2002), and the sampling heuristic by Kolisch (1996) ran for 5000 iterations each. Computation times are available for the algorithm of Hartmann (2002), which requires 14.1 seconds on average (coded in ANSI C, tested on a Pentium PC at 133MHz under Linux), and for the algorithm of Merkle et al. (2000), which requires 25 seconds on average (tested on a Pentium III with 500MHz). The tabu search algorithm by Nonobe and Ibaraki (2001) ran for 5000 iterations as well, resulting in an average (maximal) computation time of 110 (576) seconds (coded in C, tested on a Sun Ultra 2 with 300 MHz). The constraint propagation based branch-and-bound algorithm of Dorndorf, Pesch, and Phan Huy (2000a) had an average (maximal) computation time of 205 (300) seconds (coded in C++ using ILOG Solver, tested on a Pentium Pro/200 under Windows NT 4.0). The average number of schedules generated within the Lagrangian approach was 4750, at an average (maximal) computation time of 72.9 (654) seconds per instance. Of course, these figures are not directly comparable. Yet, they show that the computational investment is at least roughly of the same order of magnitude for the algorithms mentioned above.

The comparison shows that the Lagrangian based approach is competitive with state-of-the-art algorithms. In particular, the solutions of the Lagrangian relaxation seem to contain valuable information to construct good feasible schedules. We note that the respective local search algorithms shown in Table 6 are capable of computing better solutions if more iterations are allowed, see, e.g., Merkle et al. (2000) and Valls, Ballestín, and Quintanilla (2001) for benchmark results. However, the results obtained with the Lagrangian approach can be improved by straightforward local improvement heuristics. A rudimentary test of such a local improvement heuristic resulted in an average deviation of 35.3% from the critical path lower bound, at an average (maximal) computation time of 88 (678) seconds. More importantly, we like to point out that the Lagrangian approach—in contrast to the local search algorithms—provides both lower and upper bounds at the same time, which leads to improved performance bounds. This can be seen by comparing the deviations from the Lagrangian lower bound (Dev. LB_{LR}) to the deviations from the critical path lower bound (Dev. CP) in Table 5.

5.4 Results for labor-constrained instances

Let us finally report on the experiments with the BASF-type instances described in Heipcke et al. (2000). For these instances, previous work includes linear programming relaxations and corresponding ordering-based heuristics by Cavalcante et al. (2001b), a tabu search algorithm by Cavalcante and de Souza (1997), an extended implementation of which has been recently studied by Cavalcante et al. (2001a), as well as a constraint propagation based branch-and-bound algorithm by Heipcke (1999). Heipcke et al. (2000) give a brief overview and comparison of results up to the year 2000. Due to space limitations, we only report on a sample of six of the 25 instances of Heipcke et al. (2000). Our results on the remaining instances are reported in Uetz (2001).

Table 7 compares the weak linear programming relaxation (2), (4), (6), (8), and (9) to results obtained with the Lagrangian approach. Note that the computation times for solving the strong linear programming relaxation (2), (3), (4), (8), and (9) are prohibitively high for the larger of these benchmark instances, for both primal and dual simplex and the barrier code of ILOG CPLEX. Hence, we only considered the weak formulation. Moreover, we did not consider the additional valid inequalities (14), since they did not lead to better results. The number of iterations of the subgradient optimization procedure was limited to values between 100 and 300. In Table 7, the

Table 7: Comparison of linear programming and Lagrangian lower bounds.

Instance	#jobs	#tasks	*CP*	*weak LP* *LB*	CPU(ps)	CPU(ba)	Lagrangian *LB* *LB*	CPU	#it.	*T*
4o_21j_A	21	126	78	80	<1	< 1	80	< 1	100	82
6o_41j_A	41	295	90	103	5	4	102	4	121	141
8o_63j_A	63	504	174	186	42	18	184	20	148	261
10o_84j_A	84	953	270	379	353	103	378	482	233	636
10o_102j_A	102	1679	550	622	5067	557	616	1447	212	1166
10o_106j_A	106	1653	383	600	4526	709	599	1714	245	1094

columns #jobs and #tasks show the number of jobs and tasks of an instance, respectively. (Remember that each job consists of consecutive tasks, resulting in a piecewise constant requirement of resources.) *CP* is the length of a longest (critical) path in the project network, and the next columns show the respective lower bound values obtained with the weak linear program (2), (4), (6), (8), and (9) and the Lagrangian relaxation. The corresponding computation times are again in seconds. We display the computation times with the primal simplex (ps) as well as the barrier solver (ba); for the Lagrangian approach, the table also shows the number of iterations. The last column (column T) contains the time horizons which have been used for these experiments; they have been taken from Cavalcante et al. (2001b). The results suggests that, using Lagrangian relaxation, one can obtain essentially the same lower bounds as with the weak linear programming relaxation. In comparison to the primal simplex algorithm, the computation times with the Lagrangian approach are significantly smaller. Yet, the barrier solver of ILOG CPLEX performs even better on these instances.

In Table 8, we compare the quality of the feasible solutions that we obtained with the Lagrangian-based heuristic to the results with the Tabu Search algorithm (Tabu S.) of Cavalcante et al. (2001a), the linear programming based heuristics (LP-Heur.) of Cavalcante et al. (2001b), and the constraint propagation approach (Const. Pr.) by Heipcke (1999). For the algorithms from the literature, the table shows the makespan of the best solutions found after several computational experiments. For the Lagrangian approach, we display both first and best found solutions for the better of two experiments (with fixed parameter setting): one with the original instance, and one with an equivalent instance where all temporal restrictions have been reversed. This sometimes leads to better results and has also been exploited in the computational experiments of all papers mentioned above. The last column (column T) again shows the time horizon that has been used. Notice that in this case the time horizon has been obtained by parallel list scheduling using ten standard priority rules, hence the time horizons are larger in comparison to those given in Table 7.

Table 8: Comparison of the Lagrangian-based heuristic to algorithms from the literature.

	Tabu S.	*LP-Heur.*	*Const. Pr.*	*Lagrangian Heuristic*				
Instance	UB_{best}	UB_{best}	UB_{best}	UB_{first}	CPU	UB_{best}	CPU	T
4o_21j_A	82	82	82	82	< 1	82	< 1	82
6o_41j_A	140	145	152	151	< 1	145	5	150
8o_63j_A	259	273	281	283	1	276	34	287
10o_84j_A	634	–	730	731	3	699	668	715
10o_102j_A	1155	–	1239	1228*	13	1206*	2124	1233
10o_106j_A	1087	–	1166	1143*	11	1122*	2203	1146

* The solution has been obtained using an equivalent 'reversed instance'.

The computation times for the algorithms from the literature are not directly comparable, and thus not shown in Table 8. The computation times are moderate for all of the algorithms for small instance sizes (less than 40 jobs). The tabu search algorithm by Cavalcante et al. (2001a) requires up to 2,000 seconds for the medium sized instances (40-80 jobs), and between 2,000 and 14,000 seconds for large sized instances (more than 80 jobs), on a Sun Sparc 1000. The LP-based ordering heuristics by Cavalcante et al. (2001b) require between 43 and 3,575 seconds for the medium sized instances (between 40 and 80 jobs), on a Pentium II with 200 MHz. Results for the large sized instances (more than 80 jobs) are not given by Cavalcante et al. (2001b), due to excessive computation times to solve the corresponding LP-relaxations. The computations by Heipcke (1999) have been conducted on a Sun Ultra 2 with 248 MHz. Her branch-and-bound algorithm has been aborted on reaching 200,000 nodes in the enumeration tree, resulting in computation times between 118 and 985 seconds for medium sized instances (40 to 80 jobs), and between 325 and 1,350 seconds for the larger instances (more than 80 jobs).

Table 8 suggests that the Lagrangian-based heuristic is also capable of finding reasonable good solutions for these notoriously hard instances, in reasonable time. The solutions obtained by list scheduling with standard priority rules (shown in column T) can be substantially improved in most of the cases. For 16 out of the 25 instances from the test set described by Heipcke et al. (2000), our solutions improve upon those obtained with constraint propagation by Heipcke (1999). The tabu search algorithms by Cavalcante et al. (2001a) obviously yield the currently best known solutions for these instances. However, these results are obtained at the expense of relatively large computation times.

6 Conclusions

We have presented a Lagrangian-based approach to compute both lower bounds and feasible solutions for resource-constrained project scheduling problems. Although the ingredients of our approach are classic, they are spiced with some ideas which—we believe—make it attractive. First, there is the insight that scheduling problems with start-time dependent costs can be solved fast by minimum cut computations, which eventually allows one to attack large-scale problem instances of practical relevance. Second, we have demonstrated that the solution of the Lagrangian relaxation, combined with the concept of α-completion times, gives rise to good schedules. The computational results show that our approach offers a fair tradeoff between the quality of the lower bounds and feasible solutions on the one hand and the necessary computational effort on the other hand. The approach could benefit further from a faster method to solve the sequence of minimum cut problems. (We currently solve each minimum cut problem from scratch.) This method should allow an efficient 'warm start' to re-compute a minimum cut after the arc capacities have changed.

Acknowledgments. This work was done while the last two authors were with the Technische Universität Berlin; they received support from the Deutsche Forschungsgemeinschaft (DFG), grant Mo 446/3-3, and from the Bundesministerium für Bildung und Forschung (bmb+f), grant 03-MO7TU1-3, as well as the German-Israeli Foundation for Scientific Research and Development (GIF), grant I 246-304.02/97, respectively. The authors are grateful to Carola Schaad and Lars Stolletz for their help with implementing and testing various parts of the algorithms, and to Matthias Müller-Hannemann for helpful discussions on maximum flow codes.

References

Balinski, M. (1970). On a selection problem. *Management Science 17*, 230–231.

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics 16*, 87–90.

Bertsekas, D. P. (1999). *Nonlinear Programming* (2nd ed.). Belmont (MA): Athena Scientific.

Blazewicz, J., J. K. Lenstra, and A. H. G. Rinnooy Kan (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics 5*, 11–24.

Bouleimen, K. and H. Lecocq (2002). A new efficient simulated annealing algorithm for the resource constrained project scheduling problem. *European Journal of Operational Research*. To appear.

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

Brucker, P. and S. Knust (2000). A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research 127*, 355–362.

Camerini, P. M., L. Fratta, and F. Maffioli (1975). On improving relaxation methods by modified gradient techniques. *Mathematical Programming Studies 3*, 26–34.

Cavalcante, C. B. C., V. C. Cavalcante, C. C. Ribeiro, and C. C. de Souza (2001a). Parallel cooperative approaches for the labor constrained scheduling problem. In C. C. Ribeiro and P. Hansen (Eds.), *Essays and Surveys in Metaheuristics*, Chapter 10, pp. 201–226. Dordrecht: Kluwer.

Cavalcante, C. C. B. and C. C. de Souza (1997). A tabu search approach for scheduling problems under labour constraints. Technical Report IC-97-13, Instituto de Computação, UNICAMP, Campinas, Brazil.

Cavalcante, C. C. B., C. C. de Souza, M. W. P. Savelsbergh, Y. Wang, and L. A. Wolsey (2001b). Scheduling projects with labor constraints. *Discrete Applied Mathematics 112*, 27–52.

Chang, G. and J. Edmonds (1985). The poset scheduling problem. *Order 2*, 113–118.

Chaudhuri, S., R. A. Walker, and J. E. Mitchell (1994). 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.

Chekuri, C., R. Motwani, B. Natarajan, and C. Stein (2001). Approximation techniques for average completion time scheduling. *SIAM Journal on Computing 31*, 146–166.

Cherkassky, B. V. and A. V. Goldberg (1997). On implementing the push-relabel method for the maximum flow problem. *Algorithmica 19*, 390–410.

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

Demassey, S., C. Artigues, and P. Michelon (2001a). Comparing lower bounds for the RCPSP under a hybrid constraint-linear programming approach,. In *Proceedings of the Workshop on Cooperative Solvers in Constraint Programming*, Paphos (Cyprus), pp. 109–123.

Demassey, S., C. Artigues, and P. Michelon (2001b). Constraint propagation based cutting planes: An application to the resource-constrained project scheduling problem. Technical Report 237, Laboratoire d'Informatique d'Avignon, Avignon, France.

Dorndorf, U., E. Pesch, and T. Phan Huy (2000a). A branch-and-bound algorithm for the resource-constrained project scheduling problem. *Mathematical Methods of Operations Research 52*, 413–439.

Dorndorf, U., E. Pesch, and T. Phan Huy (2000b). A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalized precedence constraints. *Management Science 46*, 1365–1384.

Drexl, A. and A. Kimms (2001). Optimization guided lower and upper bounds for the Resource Investment Problem. *Journal of the Operational Research Society 52*, 340–351.

Everett III, H. (1963). Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research 11*, 399–417.

Feige, U. and J. Kilian (1998). Zero-knowledge and the chromatic number. *Journal of Computer and System Sciences 57*, 187–199.

Fest, A., R. H. Möhring, F. Stork, and M. Uetz (1998). Resource constrained project scheduling with time windows: A branching scheme based on dynamic release dates. Technical Report 596/1998 (revised 1999), Institut für Mathematik, Technische Universität Berlin, Berlin, Germany. Submitted.

Goemans, M. X., M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang (2002). Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics 15*, 165–192.

Goldberg, A. and S. Rao (1998). Beyond the flow decomposition barrier. *Journal of the Association for Computing Machinery 45*, 783–797.

Goldberg, A. V. and R. E. Tarjan (1988). A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery 35*, 921–940.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal 45*, 1563–1581.

Grinold, R. C. (1972). The payment scheduling problem. *Naval Research Logistics Quarterly 19*, 123–136.

Gröflin, H., T. M. Liebling, and A. Prodon (1982). Optimal subtrees and extensions. *Annals of Discrete Mathematics 16*, 121–127.

Hall, L. A., A. S. Schulz, D. B. Shmoys, and J. Wein (1997). Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research 22*, 513–544.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics 45*, 733–750.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics 49*, 433–448.

Heilmann, R. and C. Schwindt (1997). Lower bounds for RCPSP/max. Technical Report 511, WIOR, Universität Karlsruhe, Karlsruhe, Germany.

Heipcke, S. (1999). *Combined Modelling and Problem Solving in Mathematical Programming and Constraint Programming*. Ph. D. thesis, School of Business, University of Buckingham, U.K.

Heipcke, S., Y. Colombani, C. C. B. Cavalcante, and C. C. de Souza (2000). Scheduling under labour resource constraints. *Constraints 5*, 415–422.

Kallrath, J. and J. M. Wilson (1997). *Business Optimisation using Mathematical Programming*. London: Macmillan Business.

Kimms, A. (2001). Maximizing the net present value of a project under resource constraints using a lagrangian relaxation based heuristic with tight upper bounds. *Annals of Operations Research 102*, 221–236.

Klein, R. and A. Scholl (1999). Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research 112*, 322–346.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research 90*, 320–333.

Kolisch, R. and S. Hartmann (1999). Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In J. Węglarz (Ed.), *Project Scheduling: Recent Models, Algorithms and Applications*, Chapter 7, pp. 147–178. Dordrecht: Kluwer.

Kolisch, R., C. Schwindt, and A. Sprecher (1999). Benchmark instances for project scheduling problems. In J. Węglarz (Ed.), *Project Scheduling: Recent Models, Algorithms and Applications*, Chapter 9, pp. 197–212. Dordrecht: Kluwer.

Kolisch, R. and A. Sprecher (1996). PSPLIB - A project scheduling problem library. *European Journal of Operational Research 96*, 205–216.

Merkle, D., M. Middendorf, and H. Schmeck (2000). Ant colony optimization for resource-constrained project scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Las Vegas, Nevada, pp. 893–900.

Mingozzi, A., V. Maniezzo, S. Ricciardelli, and L. Bianco (1998). An exact algorithm for the multiple resource-constained project scheduling problem based on a new mathematical formulation. *Management Science 44*, 714–729.

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

Möhring, R. H., A. S. Schulz, F. Stork, and M. Uetz (1999). Resource-constrained project scheduling: Computing lower bounds by solving minimum cut problems. In J. Nešetřil (Ed.), *Proceedings of the 7th Annual European Symposium on Algorithms,* Prague, Czech Republic, Volume 1643 of *Lecture Notes in Computer Science*, pp. 139–150. Berlin: Springer.

Möhring, R. H., A. S. Schulz, F. Stork, and M. Uetz (2001). On project scheduling with irregular starting time costs. *Operations Research Letters 28*, 149–154.

Munier, A., M. Queyranne, and A. S. Schulz (2002). Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. *SIAM Journal on Computing*. To appear.

Nonobe, K. and T. Ibaraki (2001). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In C. C. Ribeiro and P. Hansen (Eds.), *Essays and Surveys in Metaheuristics*, Chapter 25, pp. 557–588. Dordrecht: Kluwer.

Phillips, C. A., C. Stein, and J. Wein (1998). Minimizing average completion time in the presence of release dates. *Mathematical Programming 82*, 199–223.

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

PSPLIB (2000). `ftp://ftp.bwl.uni-kiel.de/pub/operations-research/psplib/HTML/`.

PSPLIB/max (2000). `http://www.wior.uni-karlsruhe.de/RCPSPmax/progenmax/`.

Rhys, J. M. W. (1970). A selection problem of shared fixed costs and network flows. *Management Science 17*, 200–207.

Roundy, R. O., W. L. Maxwell, Y. T. Herer, S. R. Tayur, and A. W. Getzler (1991). A price-directed approach to real-time scheduling of product operations. *IIE Transactions 23*, 149–160.

Sankaran, J. K., D. L. Bricker, and S.-H. Juang (1999). A strong fractional cutting-plane algorithm for resource-constrained project scheduling. *International Journal of Industrial Engineering 6*, 99–111.

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

Schäffter, M. (1997). Scheduling with respect to forbidden sets. *Discrete Applied Mathematics 72*, 141–154.

Schwindt, C. (1996). Generation of resource constrained project scheduling problems with minimal and maximal time lags. Technical Report 489, WIOR, Universität Karlsruhe, Karlsruhe, Germany.

Schwindt, C. (1998). A branch-and-bound algorithm for the resource-constrained project duration problem subject to temporal constraints. Technical Report 544, WIOR, Universität Karlsruhe, Karlsruhe, Germany.

Selle, T. (1999). Lower bounds for project scheduling problems with renewable and cumulative resources. Technical Report 573, WIOR, Universität Karlsruhe, Karlsruhe, Germany.

Uetz, M. (2001). *Algorithms for Deterministic and Stochastic Scheduling*. Ph. D. thesis, Fakultät II – Institut für Mathematik, Technische Universität Berlin, Berlin, Germany. Published: Cuvillier Verlag, Göttingen, Germany.

Valls, V., F. Ballestín, and S. Quintanilla (2001). A population-based approach to the resource-constrained project scheduling problem. Technical Report 10-2001, Departamento de Estadística e Investigación Operativa, Universitat de València, Spain.

Węglarz, J. (Ed.) (1999). *Project Scheduling: Recent Models, Algorithms, and Applications*. Dordrecht: Kluwer.