Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

Resource-Constrained Project Scheduling: From a Lagrangian Relaxation to Competitive Solutions

by

Frederik Stork Marc Uetz

No. 661/2000

Resource-Constrained Project Scheduling: From a Lagrangian Relaxation to Competitive Solutions

Frederik Stork* Marc Uetz*

February 7, 2000

Abstract

List scheduling belongs to the classical and widely used algorithms for scheduling problems, but for resource-constrained project scheduling problems most standard priority lists do not capture enough of the problem structure, often resulting in poor performance. We use a well-known Lagrangian relaxation to first compute schedules which do not necessarily respect the resource constraints. We then apply list scheduling in the order of so-called α-completion times of jobs. Embedded into a standard subgradient optimization, our computational results show that the schedules compare to those obtained by state-of-the-art local search algorithms. In contrast to purely primal heuristics, however, the Lagrangian relaxation also provides powerful lower bounds, thus the deviation between lower and upper bounds can be drastically reduced by this approach.

1 Introduction

Resource-constrained project scheduling. Within resource-constrained project scheduling, *jobs* have to be scheduled subject to both temporal and resource constraints in order to minimize a given objective function. Temporal constraints are given by precedence relations between pairs of jobs indicating that the start of a job must not occur before the completion of its predecessors. While in process, every job requires a certain amount of renewable resources (e.g., machines and/or personnel), and the availability of these resources is limited. A schedule is called time- (resource-) feasible if it respects the temporal (resource) constraints. The objective addressed in this abstract is to find a time- and resource-feasible schedule minimizing the *project makespan*, which is the time required to complete all jobs. ($PS|prec|C_{\max}$ in the notation of [2].)

*Technische Universität Berlin, Fachbereich Mathematik, Sekr. MA 6–1, Straße des 17. Juni 136, D-10623 Berlin, Germany. Email: {stork, uetz}@math.tu–berlin.de

From a Lagrangian relaxation to a series of infeasible schedules. In order to compute lower bounds on the optimal objective value for resource-constrained project scheduling problems, Christofides et al. [3] proposed a Lagrangian relaxation where resource constraints are dualized, and the corresponding Lagrangian multiplier problem is solved by subgradient optimization. It has recently been shown by Möhring et al. [9] that each subproblem which has to be solved within the subgradient optimization is equivalent to a minimum-cut problem in a simple auxiliary graph, and can thus be solved efficiently. What is important for the thread of the present paper is that in each iteration of the subgradient optimization procedure, a time-feasible but not necessarily resource-feasible schedule is computed. Moreover, during its course the subgradient optimization tends to reduce these resource infeasibilities. The intuition behind our approach is to exploit the series of these resource-infeasible schedules in order to obtain priority lists which capture a fair amount of the problem structure.

2 From infeasible to feasible schedules

List scheduling. In the literature, two list scheduling algorithms are distinguished, the *parallel* and the *serial* scheme. In both cases a *priority list* L of the jobs is given which determines the order in which the jobs are considered. A job is usually called *available* at a time t if all its predecessors have already been completed by t.

- **Parallel List Scheduling** proceeds over time (starting at time $t = 0$). At any time t, as many available jobs as possible are scheduled according to the order given by L. If no more job can be scheduled at time t, t is augmented to the next completion time of a job.
- **Serial List Scheduling** proceeds job by job. In the order given by L, each job is scheduled as early as possible with respect to the jobs scheduled so far. To make sure that each job is available at the time it is considered, the priority list has to be compatible with the precedence constraints.

List scheduling by α-completion times. One possible way of computing a feasible schedule on the basis of a resource-infeasible one is to apply list scheduling in order of non-decreasing start times of jobs. However, motivated by several approximation results in machine scheduling which make use of so-called α-points of jobs [10, 5, 4], this approach can be refined as follows. Let a time-feasible but resource-infeasible schedule $S = (S_1 \dots, S_n)$ be given, where n is the number of jobs and S_j denotes the start time of job j. Then, for $0 \le \alpha \le 1$, let $S_j + \alpha\, p_j$ be the *α-completion time* of job j, where p_j denotes the processing time of job j. For a given α (and a given schedule S), the ordering according to non-decreasing α-completion times of the jobs can now be used as a priority list for parallel or serial list scheduling. Note that, since the original schedule S was time-feasible, the ordering is compatible with the precedence constraints for all values of α. It is not difficult to see that the number of different priority lists one can obtain using different values of α is not more than $n \cdot m$, where n is the number of jobs and m is the maximum number of jobs processed in parallel in schedule S. We call these values of α *representative*.

The algorithm. In each iteration of the subgradient optimization algorithm, a time-feasible schedule is computed (by solving a minimum cut problem; see [9]). Using the priority lists obtained as orderings according to α-completion times for all representative values of α, we apply both parallel and serial list scheduling. As a folklore trick, we also apply parallel list scheduling backwards, i.e., we schedule the jobs in decreasing time according to the reverse order of α-completion times. We finally output the best schedule found.

3 Computational results

Test set. We have conducted experiments using the well known ProGen test set [8], consisting of instances with 60, 90, and 120 jobs (480, 480, and 600 instances, respectively). In order to exclude "trivial" instances, we first computed feasible schedules using a set of 10 standard priority lists from the literature (see, e.g., [7]). We considered only those instances where the minimal makespan obtained with these priority lists was above the critical path lower bound. The maximal number of iterations in the subgradient optimization was set to 50, which means that we have evaluated maximally 50 infeasible schedules per instance. The number of representative values of α (i.e., the number of different priority lists) for each of these schedules was roughly $n/4$.

Results. The first two columns of Table 1 show the number of jobs per instance (jobs), and the number of instances we considered (inst.). The next columns display the average and maximum computation times per instance in seconds (CPU), and the average number of iterations in the subgradient optimization (it.). Note that the computation time refers to the whole subgradient optimization procedure. We further display the average deviations of the obtained solutions from the critical path lower bound (dv. LB_{cp}), the lower bound obtained with the Lagrangian relaxation (dv. LB_{lg}), and the best known solutions (dv. UB_{best}). The latter are maintained in [1], and have been obtained by different, partly time-intensive algorithms including branch-and-bound as well as various local search procedures. 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).

Conclusions. The quality of the solutions we obtain by using Lagrangian relaxation, resource-infeasible schedules and list scheduling according to α-completion times compares to the most powerful heuristic algorithms that have been recently summarized in [7] and [6]. The solutions, particularly the quality of the lower bounds, can be further improved by allowing more iterations within the subgradient optimization, clearly at the expense of higher computation times. Most important, however, is that the

jobs	inst.	av. CPU	mx. CPU	it.	dv. LB_{cp}	dv. LB_{lg}	dv. UB_{best}	opt.	best
60	266	5.8	18.4	39	22.6 %	12.1 %	2.2 %	88	129
90	253	14.0	39.5	39	22.7 %	11.5 %	2.3 %	102	119
120	563	33.7	147.2	42	38.9 %	16.3 %	3.1 %	107	134

Table 1: Results obtained on a Sun Ultrasparc with 200 MHz clock-pulse. The code was compiled with the EGCS C++ compiler v. 1.1.2 running under Solaris.

approach provides powerful lower and upper bounds at the same time. Compared to purely primal heuristics which rely on the critical path lower bound only, the deviation between lower and upper bounds can thus be drastically reduced.

References

[1] `ftp://ftp.bwl.uni-kiel.de/pub/operations-research/psplib/HTML/index.html`, January 2000.

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

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

[4] M. X. Goemans, *Improved approximation algorithms for scheduling with release dates*, in Proceedings of the 8th ACM–SIAM Symposium on Discrete Algorithms, New Orleans, 1997, pp. 591–598.

[5] L. A. Hall, D. B. Shmoys, and J. Wein, *Scheduling to minimize average completion time: Off-line and on-line algorithms*, in Proceedings of the 7th ACM–SIAM Symposium on Discrete Algorithms, Atlanta, 1996, pp. 142–151.

[6] S. Hartmann, *Self-adapting genetic algorithms with an application to project scheduling*, Tech. Rep. 506/1999, Universität Kiel, 1999.

[7] S. Hartmann and R. Kolisch, *Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis*, in Handbook on Recent Advances in Project Scheduling, J. Węglarz, ed., Kluwer, Amsterdam, 1999, pp. 147–178.

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

[9] R. H. Möhring, A. S. Schulz, F. Stork, and M. Uetz, *Resource-constrained project scheduling: Computing lower bounds by solving minimum cut problems*, in Algorithms – ESA'99, J. Nešetřil, ed., vol. 1643 of Lecture Notes in Computer Science, Springer, 1999, pp. 139–150.

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

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

661/2000 *Frederik Stork and Marc Uetz:* Resource-Constrained Project Scheduling: From a Lagrangian Relaxation to Competitive Solutions

658/1999 *Olaf Jahn, Rolf H. Möhring, and Andreas S. Schulz:* Optimal Routing of Traffic Flows with Length Restrictions in Networks with Congestion

655/1999 *Michel X. Goemans and Martin Skutella:* Cooperative facility location games

654/1999 *Michel X. Goemans, Maurice Queyranne, Andreas S. Schulz, Martin Skutella, and Yaoguang Wang:* Single Machine Scheduling with Release Dates

653/1999 *Andreas S. Schulz and Martin Skutella:* Scheduling unrelated machines by randomized rounding

646/1999 *Rolf H. Möhring, Martin Skutella, and Frederik Stork:* Forcing Relations for AND/OR Precedence Constraints

640/1999 *Foto Afrati, Evripidis Bampis, Chandra Chekuri, David Karger, Claire Kenyon, Sanjeev Khanna, Ioannis Milis, Maurice Queyranne, Martin Skutella, Cliff Stein, and Maxim Sviridenko:* Approximation Schemes for Minimizing Average Weighted Completion Time with Release Dates

639/1999 *Andreas S. Schulz and Martin Skutella:* The Power of α-Points in Preemptive Single Machine Scheduling

634/1999 *Karsten Weihe, Ulrik Brandes, Annegret Liebers, Matthias Müller–Hannemann, 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 *Andreas Fest, 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

Reports may be requested from:

Hannelore Vogt-Möller
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

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

Reports are also available in various formats from

`http://www.math.tu-berlin.de/coga/publications/techreports/`

and via anonymous ftp as

`ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-`*number*`-`*year*`.ps`