Technische Universität Berlin

FAKULTÄT II
MATHEMATIK UND
NATURWISSENSCHAFTEN
Institut für Mathematik

Extending partial suborders and implication classes

by

Sándor P. Fekete   Ekkehard Köhler
Jürgen Teich

No. 697/2000

Extending partial suborders and implication classes*

Sándor P. Fekete† Ekkehard Köhler‡ Jürgen Teich§

Abstract

We consider the following problem called *transitive ordering with precedence constraints* (TOP): Given a partial order P=(V,<) and an (undirected) graph G=(V,E) such that all relations in P are represented by edges in G. Is there a transitive orientation D=(V,A) of G, such that P is contained in D? This problem arises naturally in the context of scheduling, where P describes a set of precedence constraints, and the graph G is the (temporal) comparability graph of jobs. Korte and Möhring (1985) have given a linear-time algorithm for deciding TOP. However, their approach is only useful when the full set of edges in G is known. When running a branch-and-bound algorithm for solving a scheduling problem, these edges are only known partially, but they may already prohibit the existence of a feasible solution. We give a pair of necessary and sufficient conditions on graphs G in terms of forbidden substructures. Thus, our conditions can be used quite effectively in the context of a branch-and-bound framework.

1 Introduction

Multi-dimensional packing and scheduling problems occur in many practical contexts; see [1, 2] for overviews. Even the one-dimensional versions of these problems are NP-hard in the strong sense; additional difficulties in multi-dimensional scenarios prohibit the straightforward formulation as integer programs of tractable size. This requires the development of additional mathematical tools. Fekete and Schepers [4, 5, 6] have developed a branch-and-bound scheme for solving these kinds of problems, based on a graph-theoretic characterization of feasible solutions.

When dealing with precedence constraints of one- or multi-dimensional scheduling problems, we encounter the following natural order-theoretic problem, called *transitive ordering with precedence constraints* (TOP):

*revised April 2001

†Department of Mathematical Optimization, TU Braunschweig, Pockelsstr. 14, 38106 Braunschweig, Germany, `sandor.fekete@tu-bs.de`

‡Department of Mathematics, TU Berlin, Str. des 17. Juni 136, 10623 Berlin, Germany, `ekoehler@math.tu-berlin.de`

§Computer Engineering Laboratory, University of Paderborn, Warburger Str. 100, 33098 Paderborn, Germany, `teich@date.upb.de`

Consider a partial order $P = (V, \prec)$ and a comparability graph $G = (V, E)$, such that all relations in P are represented by edges in G. Is there a transitive orientation $D = (V, A)$ of G, such that P is contained in D?

Korte and Möhring [9] have given a linear-time algorithm for deciding TOP. However, their approach can only be used when the full set of edges in G is known. When running a branch-and-bound algorithm for solving a packing or scheduling problem, these edges of G are only known partially, but they may already prohibit the existence of a feasible solution for a given partial order P. This makes it desirable to come up with structural characterizations that are already useful when only parts of G are known.

In this paper, we give a pair of necessary and sufficient conditions for the existence of a solution for the problem TOP on graphs G in terms of forbidden substructures. Using the concept of packing classes developed in [4], this characterization can be used quite effectively in the context of a branch-and-bound framework, because it can recognize infeasible subtrees at "high" branches of the search tree, where only some of the edges of G are known. The usefulness of these concepts and results has been validated by implementation and computational experiments [2, 3].

2 Motivation: Multi-Dimensional Packing with Precedence Constraints

When considering a feasible packing of d-dimensional boxes into a container, we can describe it by a set of d interval graphs, which arise by projecting the packing onto the d coordinate axes. See Figure 1 for the underlying idea. In [4] a set of easy combinatorial necessary and sufficient conditions is given for characterizing when a set of graphs G_i corresponds to a feasible packing. They also showed that this leads to useful results in practice by using the following enumeration scheme:

For each coordinate, consider a comparability graph G_i and construct its edge set; the complement will be the interval graph $\overline{G_i}$ describing the overlap of projections. This means that we have three basic states for any edge:

(1) edges that have been fixed to be in $\overline{E_i}$, i.e., "non-edges";

(2) edges that have been fixed to be in E_i, i.e., "edges";

(3) edges that have not yet been assigned to $\overline{E_i}$ or E_i, i.e., "unassigned edges".

For more technical details, see [4, 5, 6], and [11] for a practical application in the context of technical computer science.

Now consider a situation where we need to satisfy a partial order $P = (V, <)$ of precedence constraints (e.g., in the time dimension), and let $A_P = \{(u, w) \mid u < w \text{ in } P\}$ be the set of directed arcs describing P. It follows that each arc $a = (u, w) \in A_P$ in this partial order forces the corresponding undirected edge

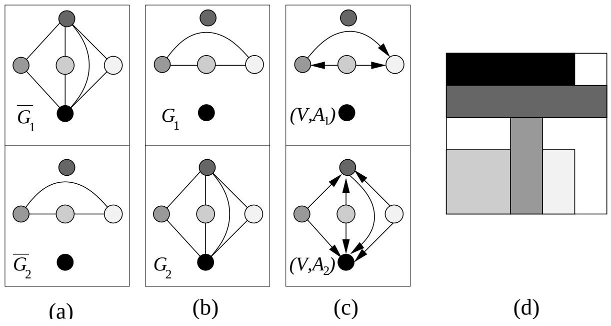

Figure 1: *(a) A pair of interval graphs, characterizing overlap in the coordinate projections. (b) The complements are the corresponding comparability graphs. (c) A transitive orientation, describing the relative positioning in the two directions. (d) A feasible packing corresponding to the orientation.*

$e = \{u, w\}$ to be included in the corresponding comparability graph. Thus, we can simply initialize our algorithm for constructing packing classes by fixing all undirected edges corresponding to A_P to be contained in E. After running the original algorithm, we may get additional comparability edges. As the example in Figure 2 shows, this causes an additional problem: Even if we know that the graph G has a transitive orientation, and all arcs $a = (u, w)$ of the precedence order (V, A_P) are contained in E as $e = \{u, w\}$, it is not clear that there is a transitive orientation that contains all arcs of A_P.

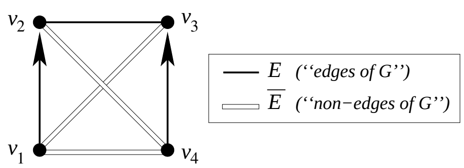

Figure 2: *A comparability graph $G = (V, E)$ with a partial order P contained in E, such that there is no transitive orientation of G that extends P.*

In order to deal with precedence constraints, we also consider orientations of the comparability edges. This means that during the course of our tree search, we can have three different possible states for any edge $e = \{u, w\} \in E$:

(2a) e has orientation (u, w);

(2b) e has orientation (w, u);

(2c) e has no assigned orientation.

The main contribution of this paper is to establish a pair of necessary and sufficient conditions based on small critical subconfigurations of edges. This provides the mathematical basis for a safe and relatively efficient approach for solving constrained multi-dimensional packing and scheduling problems to optimality.

3 Implication Classes and Modular Decomposition

3.1 Implication Classes

Our characterization arises from considering the following two configurations—see Figure 3:

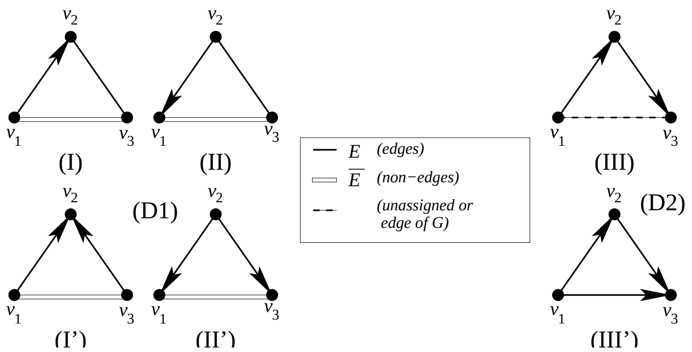

Figure 3: *Implications for edges and their orientations: Above are path implications (D1, left) and transitivity implications (D2, right); below the forced orientations of edges.*

The first configuration consists of two edges $\{v_1, v_2\}, \{v_2, v_3\} \in E$, such that the third edge $\{v_1, v_3\}$ has been fixed to be an non-edge of G. Now any orientation of one of the two edges of G forces the orientation of the other edge of G, as shown in the left part of the figure. Since this configuration corresponds to an induced path in G, we call this arrangement a *path implication*.

The second configuration consists of two directed edges $(v_1, v_2), (v_2, v_3) \in E$. In this case we know that the edge $\{v_1, v_3\}$ must also be an edge of G, with the orientation (v_1, v_3). Since this configuration arises directly from transitivity in G, we call this arrangement a *transitivity implication*.

Clearly, any implication arising from one of the above configurations can induce further implications.

In particular, when considering sequences of only path implications, we get a partition of the edges of G into *path implication classes*: Two edges are in the same implication class, iff there is a sequence of path implications, such that orienting one edge forces the orientation of the other edge. For an example, consider the arrangement in Figure 2. Here, all three edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, and $\{v_3, v_4\}$ are in the same path implication class. Now the orientation of (v_1, v_2) implies the orientation (v_3, v_2), which in turn implies the orientation (v_3, v_4), contradicting the orientation of $\{v_3, v_4\}$ in the given partial order P. It is not hard to see that the implication classes form a partition of the edges, since we are dealing with an equivalence relation. In the following, we refer to path implication classes when we speak of *implication classes*. A violation of a path implication is called a *path conflict*.

As the example in Figure 4 shows, only excluding path conflicts when recursively carrying out path implications does not suffice to guarantee the existence of a feasible orientation: Working through the queue of path implications, we end up with a directed cycle, which violates a transitivity implication.

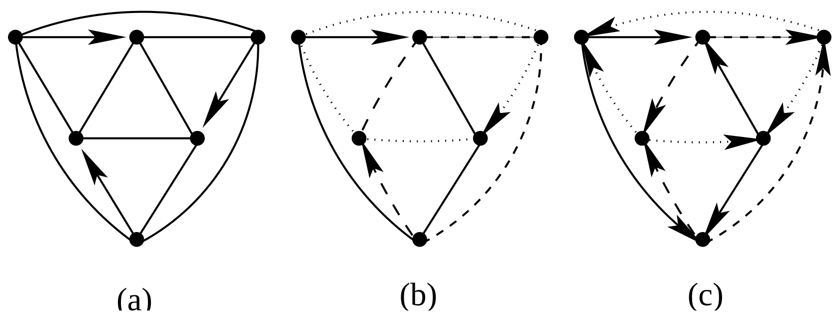

Figure 4: *(a) A graph G with a partial order formed by three directed edges; (b) there are three path implication classes that each have one directed arc; (c) carrying out path implications creates directed cycles, i.e., transitivity conflicts.*

We call a violation of a transitivity implication a *transitivity conflict*.

Summarizing, we have the following necessary conditions for the existence of a transitive orientation that extends a given partial order P:

D1: Any path implication can be carried out without a conflict.

D2: Any transitivity implication can be carried out without a conflict.

As we will see later, these necessary conditions are also sufficient. For proving this result we need another tool: the modular decomposition of a graph.

3.2 Modular Decomposition

The concept of *modular decomposition* of a graph was first introduced by Gallai [7] for studying comparability graphs. This powerful decomposition scheme has a variety of applications in algorithmic graph theory—for further material on this concept and its application the interested reader is referred to [8, 10].

A *module* of a graph $G = (V, E)$ is a vertex set $M \subseteq V$ such that each vertex $v \in V \setminus M$ is either adjacent to all vertices or to no vertex of M in G. (Intuitively speaking, all vertices of a module "look the same" to the other vertices of the graph.) A module is called trivial if $|M| \leq 1$ or $M = V$. A graph G is called *prime* if it contains only trivial modules. Using the concept of modules one can define a decomposition scheme for general graphs by decomposing it recursively into subsets, each of which is a module of G, stopping when all sets are singletons.

For an example, refer to Figure 6. Four modules of the shown graph G are the $M_1 = \{0,1,2,3,4,5,6,7,8,9\}$, $M_2 = \{20\}$, $M_3 = \{10,11\}$, $M_4 = \{12,13,14,15,16,17,18,19\}$; M_2 is a trivial module.

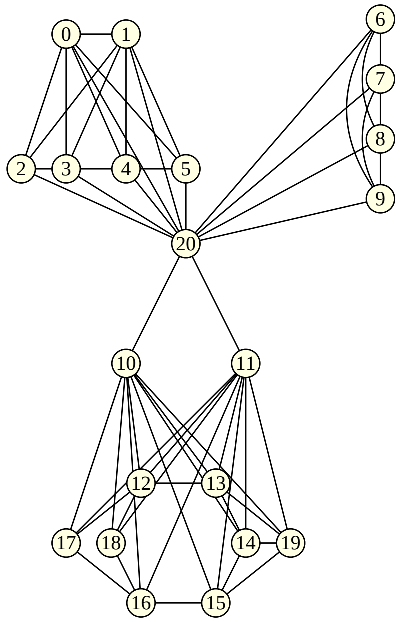

Figure 5: *A graph G.*

Gallai [7] showed that each graph G has a decomposition (the so-called *canonical decomposition*) of its vertex set into a set of modules with a variety of nice properties. He observed that each graph G is either of *parallel type*, i.e., G is not connected and its canonical decomposition is defined by its set of connected components; or G is of *series type*, i.e., $\overline{G}$ is not connected. In the latter case the canonical decomposition is given by the connected components of $\overline{G}$. If both G and $\overline{G}$ are connected, then G is said to be of *prime type*. In this case the canonical decomposition of G is given by decomposing G into its maximal proper submodules. Gallai also showed that this decomposition is unique.

This recursive decomposition defines a *decomposition tree* $T(G)$ for a given graph G in a very natural way: Create a root vertex for the trivial module of

G itself. Label it series, parallel, or prime, depending on the type of G as a module. For each non-singleton module of the canonical decomposition of G create a tree vertex, labeled as series, parallel, or prime type node, depending on the type of the module, and make it a child of the vertex corresponding to G; for each singleton module add a tree-vertex labeled with the corresponding singleton. Now proceed recursively for each subgraph corresponding to a non-trivial module in the decomposition tree until all leaves of the tree are labeled with singletons. Consequently, the leafs of the tree correspond to the vertices of the graph, while the internal vertices all correspond to non-trivial modules of the canonical decomposition of the corresponding parent vertex in $T(G)$. See Figure 6 for the decomposition tree of our example.

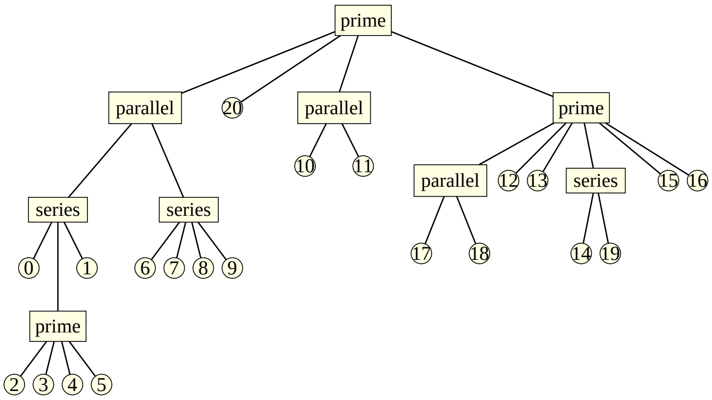

Figure 6: *A modular decomposition tree for the graph G shown in Figure 5.*

The *decomposition graph* $G^{\#}$ of a graph G is the quotient of G by the canonical decomposition into the set of modules $\{A_1, \ldots, A_q\}$, i.e., $V(G^{\#}) = \{A_1, \ldots, A_q\}$, and distinct vertices A_i and A_j are joined by an edge in $G^{\#}$ iff there is an A_iA_j-edge in G. In the following we will look at the decomposition graphs corresponding to internal vertices of $T(G)$ and refer to them as the decomposition graphs of T.

In our example, the decomposition graph $G^{\#}$ of G is given by

$$G^{\#} = (\{M_1, M_2, M_3, M_4\}, \{\{M_1, M_2\}, \{M_2, M_3\}, \{M_3, M_4\}\}).$$

An important property of the modular decomposition is its close relationship to the concept of implication classes. Gallai observed the following properties of implication classes with respect to the modular decomposition:

Observation 1 (Gallai)

1) *If G is not connected and $G_1, \ldots, G_q$ ($q \geq 2$) are the components of G, then the implication classes of $G_1, \ldots, G_q$ are exactly the implication classes of G.*

2) *If $\overline{G}$ is not connected (so that G is connected), $\overline{G_1}, \ldots, \overline{G_q}$ ($q \geq 2$) are the components of $\overline{G}$, and $A_i = V(G_i)$, then A_i and A_j are completely connected to each other whenever $1 \leq i < j \leq q$. Moreover, for all such i and j, the set of A_iA_j-edges form an implication class E_{ij} of G. The implication classes of G that are distinct from any E_{ij} are exactly the implication classes of the graphs $G_i = G[A_i]$ ($i = 1, \ldots, q$).*

3) *If G and $\overline{G}$ are both connected and have more than one vertex, and the canonical decomposition of G is given by $\{A_1, \ldots, A_q\}$, then we have*

   a) *If there is one edge between A_i and A_j ($1 \leq i < j \leq q$), then A_i and A_j are completely joined.*

   b) *The set of all edges of G that join different A_i's forms a single implication class C of G. Every vertex of G is incident with some edge of C, (i.e., $V(C) = V(G)$).*

   c) *The implication classes of G that are distinct from C are exactly the implication classes of the graphs $G_i = G[A_i]$ ($1 \leq i \leq q$).*

This strong relationship between implication classes and the modules in the canonical decomposition of a given graph turns out to be a powerful tool for studying graphs having a transitive orientation. Note that the fastest known algorithms for recognizing comparability graphs and also permutation graphs also make extensively use of this relationship. Gallai used the above properties (among others) for proving the following theorem.

Theorem 2 (Gallai)
Let G be a non-empty graph, let $T = T(G)$ be the tree decomposition of G, and let H be a vertex set corresponding to a node of T.

1) *If G is transitively oriented, and A and B are successors of H in T, then every A, B-edge of G is oriented in the same direction (to or from A). Therefore, $H^{\#}$ receives an induced transitive orientation.*

2) *Conversely, assuming that $H^{\#}$ is transitively orientable for each $H \in T$, one can choose an arbitrary transitive orientation of each $H^{\#}$ and induce a transitive orientation of G by orienting all A, B-edges (for A and B successors of H in T) in the same direction that $\{A, B\}$ is oriented in $H^{\#}$.*

From this theorem it is straightforward to draw the following helpful corollaries:

Corollary 3
A graph G is a comparability graph if and only if every decomposition graph in the tree decomposition of G is a comparability graph.

Corollary 4
Let G be a comparability graph and T its tree decomposition. Assigning to each of the decomposition graphs of T a transitive orientation independently results in a transitive orientation of G.

Furthermore, if only a partial orientation of G is given and we are interested in extending this orientation to a transitive orientation of G, we can formulate the following lemma.

Lemma 5
Let G be a comparability graph and T its tree decomposition. Furthermore, let P be a partial orientation of G, assigning orientations to some, but not all implication classes of G. P is extendible to a transitive orientation of G if and only if for each decomposition graph $H^{\#}$ of T the orientation induced on $H^{\#}$ by P is extendible to a transitive orientation on $H^{\#}$.

Proof. Follows immediately from Theorem 2 (2). $\square$

As we have seen earlier, there are two necessary condition for a partial orientation of a comparability graph to be extendible to a transitive orientation of G:

D1: Any path implication can be carried out without a conflict.

D2: Any transitivity implication can be carried out without a conflict.

Now we are ready to prove the main theorem of this paper: these two conditions are also sufficient.

Theorem 6
Let $P = (V, <)$ be a partial order with directed arc set A_P that is contained in the edge set E of a given comparability graph $G = (V, E)$. A_P can be extended to a transitive orientation of G, iff all arising path implications and transitivity implications can be carried out without creating a path conflict or a transitivity conflict.

Proof. Suppose there is a transitive orientation F of G that contains P. Because F is a transitive orientation, all arcs implied by path or transitivity implications are contained in F. Furthermore, there cannot be any path or transitivity conflict in F, again because F is a transitive orientation. Thus F shows that all arising path and transitivity implications can be carried out without creating a path or transitivity conflict.

Suppose now that **D1** and **D2** are satisfied, i.e., there is a directed graph F consisting of all arcs of P together with all orientations of edges of G that are implied by a sequence of path and transitivity implications of arcs of P. In other words, F contains all arcs, forced by path or transitivity implications together with all their implied arcs; i.e. all arcs that are forced by arcs of F are contained in F as well. We show that F can be extended to a transitive orientation of G.

First observe that, by assumption, there cannot be a path or transitivity conflict in F. In particular, F is an orientation of edges of G and for each implication class C of G that has at least one edge that is oriented in F, all edges of C are oriented in F and this orientation is conflict-free. By Corollary 4, every single conflict-free oriented implication class of G by itself is extendible to a transitive orientation of G.

Now let T be the decomposition tree of G and consider the decomposition graphs corresponding to T. By the above observation, the orientation of an implication class C in F implies an orientation of the edge(s) corresponding to this implication class in the decomposition graphs of T. More precisely, by Observation 1 (2), for every series type node H of T each edge $e = \{AB\}$ of $H^{\#}$ corresponds exactly to one implication class C_e of G. If C_e is oriented conflict-free in F, this orientation directly induces an orientation of e (see Theorem 2).

For a **prime type node** H the set of edges joining different A_i's forms exactly one implication class C_E of G (see Observation 1 (3)). Again, if C_E is oriented conflict-free in F, this orientation immediately implies an orientation on $H^{\#}$.

All we have to show now is that for each decomposition graph $H^{\#}$ of T, the partial orientation implied by F can be extended to a transitive orientation of $H^{\#}$. Then, by Corollary 4, the implied orientation of G is transitive.

By Corollary 4, a **parallel type node** of T cannot create a contradiction to transitivity—it does not contain any edges. Also a prime type node of T cannot create a contradiction: All its edges are contained in only one implication class and, since all implication classes of G contained in F are oriented conflict-free, the corresponding orientation induced by F on this single implication class has to be transitive.

This leaves the case of **series type nodes**. Suppose there is a series type node H of T with decomposition graph $H^{\#}$, where the partial orientation implied by F cannot be extended to a transitive orientation of $H^{\#}$. Then we claim that this partial orientation has to be cyclic: By definition for each series type node H of T the decomposition graph $H^{\#}$ is a complete graph and every acyclic partial orientation of a complete graph can be extended to a transitive orientation of this complete graph by taking any topological ordering of the vertices that agrees with the partial orientation. Hence, the partial orientation on $H^{\#}$ has to contain a directed cycle.

However, by the definition of T and the implied orientation of $H^{\#}$ by F, a directed cycle in $H^{\#}$ immediately implies a cyclically oriented cycle in F. Furthermore, with every consecutive pair of oriented edges (x, y), (y, z) of this cycle also the oriented edge (x, z), which is implied by transitivity, has to be contained in F. Iterating this argument results in an cyclically oriented triangle in F, which is a transitivity conflict. This contradicts our assumption that there are no transitivity conflicts. □

4 Application: Solving Problems with Precedence Constraints

We conclude this paper by giving a brief description how Theorem 6 can be used for solving problems to optimality. As described in Section 2, the basic idea is to construct the comparability graphs G_i by virtue of a combination of branch-and-bound with underlying graph-theoretic charcterizations.

We start by fixing for all arcs $(u, v) \in A$ the edge $\{u, v\}$ as an edge in the comparability graph G_i, and we also fix its orientation to be (u, v). In addition to the tests for enforcing the conditions for unoriented packing classes described in [4], we employ the implications suggested by conditions D1 and D2. For this purpose, we check directed edges in G_t for being part of a triangle that gives rise to either implication. Any newly oriented edge in G_t gets added to a queue of unprocessed edges. Like for packing classes, we can again get cascades of fixed edge orientations. If we get an orientation conflict or a cycle conflict, we can abandon the search on this tree node. The correctness of the overall algorithm follows from Theorem 6; in particular, the theorem guarantees that we can carry out implications in an arbitrary order.

Results of this application are reported in our paper [2].

References

[1] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European J. Oper. Res.*, 112:3–41, 1999.

[2] S. P. Fekete, E. Köhler, and J. Teich. Multi-dimensional packing with order constraints. Technical Report 698-2000, Technische Universität Berlin, 2000.

[3] S. P. Fekete, E. Köhler, and J. Teich. Optimal FPGA placement with temporal precedence constraints. In *Proc. DATE 2001, Design, Automation and Test in Europe*, 2001.

[4] S. P. Fekete and J. Schepers. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In *Algorithms – ESA '97*, volume 1284, pages 144–156, Springer Lecture Notes in Computer Science, 1997.

[5] S. P. Fekete and J. Schepers. On more-dimensional packing I: Modeling. Technical Report 97-288, Universität Köln, Available at `http://www.zpr.uni-koeln.de/ABS/~papers`, 1998.

[6] S. P. Fekete and J. Schepers. On more-dimensional packing III: Exact algorithms. Technical Report 97-290, Universität Köln, 1998.

[7] T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Acd. Sci. Hungar.*, 18:25–66, 1967.

[8] D. Kelly. Comparability graphs. In I. Rival, editor, *Graphs and Order*, pages 3–40. D. Reidel Publishing Company, Dordrecht, 1985.

[9] N. Korte and R. Möhring. Transitive orientation of graphs with side constraints. In H. Noltemeier, editor, *Proceedings of WG'85*, pages 143–160. Trauner Verlag, 1985.

[10] R. H. Möhring. Algorithmic aspects of the substitution decomposition in optimization over relations, set systems, and Boolean functions. *Annals of Oper. Res.*, 4:195–225, 1985.

[11] J. Teich, S. Fekete, and J. Schepers. Optimization of dynamic hardware reconfigurations. *J. of Supercomputing, to appear*, 2001.

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

705/2000 *Ekkehard Köhler:* Recognizing Graphs without Asteroidal Triples

704/2000 *Ekkehard Köhler:* AT-free, coAT-free Graphs and AT-free Posets

702/2000 *Frederik Stork:* Branch-and-Bound Algorithms for Stochastic Resource-Constrained Project Scheduling

700/2000 *Rolf H. Möhring:* Scheduling under uncertainty: Bounding the makespan distribution

698/2000 *Sándor P. Fekete, Ekkehard Köhler, and Jürgen Teich:* More-dimensional packing with order constraints

697/2000 *Sándor P. Fekete, Ekkehard Köhler, and Jürgen Teich:* Extending partial suborders and implication classes

696/2000 *Sándor P. Fekete, Ekkehard Köhler, and Jürgen Teich:* Optimal FPGA module placement with temporal precedence constraints

695/2000 *Sándor P. Fekete, Henk Meijer, André Rohe, and Walter Tietze:* Solving a "hard" problem to approximate an "easy" one: heuristics for maximum matchings and maximum Traveling Salesman Problems

694/2000 *Esther M. Arkin, Sándor P. Fekete, Ferran Hurtado, Joseph S. B. Mitchell, Marc Noy, Vera Sacristánm and Saurabh Sethia:* On the reflexivity of point sets

693/2000 *Frederik Stork and Marc Uetz:* On the representation of resource constraints in project scheduling

691/2000 *Martin Skutella and Marc Uetz:* Scheduling precedence constrained jobs with stochastic processing times on parallel machines

689/2000 *Rolf H. Möhring, Martin Skutella, and Frederik Stork:* Scheduling with AND/OR precedence constraints

685/2000 *Martin Skutella:* Approximating the single source unsplittable min-cost flow problem

684/2000 *Han Hoogeveen, Martin Skutella, and Gerhard J. Woeginger:* Preemptive scheduling with rejection

683/2000 *Martin Skutella:* Convex quadratic and semidefinite programming relaxations in Scheduling

682/2000 *Rolf H. Möhring and Marc Uetz:* Scheduling scarce resources in chemical engineering

681/2000 *Rolf H. Möhring:* Scheduling under uncertainty: optimizing against a randomizing adversary

680/2000 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* Solving project scheduling problems by minimum cut computations (Journal version for the previous Reports 620 and 661)

674/2000 *Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia:* Optimal covering tours with turn costs

669/2000 *Michael Naatz:* A note on a question of C. D. Savage

667/2000 *Sándor P. Fekete and Henk Meijer:* On geometric maximum weight cliques

666/2000 *Sándor P. Fekete, Joseph S. B. Mitchell, and Karin Weinbrecht:* On the continuous Weber and k-median problems

664/2000 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* On project scheduling with irregular starting time costs

661/2000 *Frederik Stork and Marc Uetz:* Resource-constrained project scheduling: from a Lagrangian relaxation to competitive solutions

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`