Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

More-Dimensional Packing with Order Constraints

by

Sándor P. Fekete Ekkehard Köhler
Jürgen Teich

No. 698/2000

More-Dimensional Packing with Order Constraints

Sándor P. Fekete* Ekkehard Köhler* Jürgen Teich†

Abstract

We present a first exact study on more-dimensional packing problems with order constraints. Problems of this type occur naturally in applications such as logistics or computer architecture and can be interpreted as more-dimensional generalizations of scheduling problems. Using graph-theoretic structures to describe feasible solutions, we develop a novel exact branch-and-bound algorithm. This extends previous work by Fekete and Schepers; a key tool is a new order-theoretic characterization of feasible extensions of a partial order to a given complementarity graph that is tailor-made for use in a branch-and-bound environment. The usefulness of our approach is validated by computational results.

1 Introduction

Scheduling and Packing Problems. Scheduling is arguably one of the most important topics in combinatorial optimization. Typically, we are dealing with a one-dimensional set of objects ("jobs") that need to be assigned to a finite set of containers ("machines"). Problems of this type can also be interpreted as (one-dimensional) packing problems, and they are NP-hard in the strong sense, as problems like 3-Partition are special cases.

Starting from this basic scenario, there are different generalizations that have been studied. Many scheduling problems have *precedence constraints* on the sequence of jobs. On the other hand, a great deal of practical packing problems consider *more-dimensional* instances, where objects are axis-aligned boxes instead of intervals. More-dimensional packing problems arise in many industries, where steel, glass, wood, or textile materials are cut. The three-dimensional problem is important for practical applications such as container loading.

In this paper, we give the first study of problems that comprise both generalizations: more-dimensional packing problems with order constraints—or, from a different point of view, more-dimensional scheduling problems. In more-dimensional packing, these problems arise when dealing with precedence constraints that are present in many container-loading problems. Another practical motivation to consider more-dimensional scheduling problems arises from optimizing the reconfiguration of a particular type of computer chips called FPGA's—described below.

Field-Programmable Gate Arrays and More-Dimensional Scheduling. A particularly interesting class of instances of three-dimensional orthogonal packing arises from a new type of reconfigurable computer chips, called *field-programmable gate arrays* (FPGA's). An FPGA typically consists of a regular rectangular grid of equal configurable cells (logic blocks) that allow the prototyping of simple logic functions together with simple registers and with special routing resources (see Figure 1). These chips (see e.g. [1, 30]) may support several independent or interdependent jobs and designs at a time, and parts of the chip can be reconfigured quickly during run-time. (For more technical details on the underlying architecture, see our previous paper [28], and the more recent abstract [4].) Thus, we are faced

*Department of Mathematics, TU Berlin, Berlin, Germany, [fekete,ekoehler]@math.tu-berlin.de

†Computer Engineering Laboratory, University of Paderborn, Paderborn, Germany, teich@date.upb.de

with a general class of problems that can be seen as both scheduling and packing problems. In this paper, we develop a set of mathematical tools to deal with these *more-dimensional scheduling problems*, and we show that our methods are suitable for solving instances of interesting size to optimality.

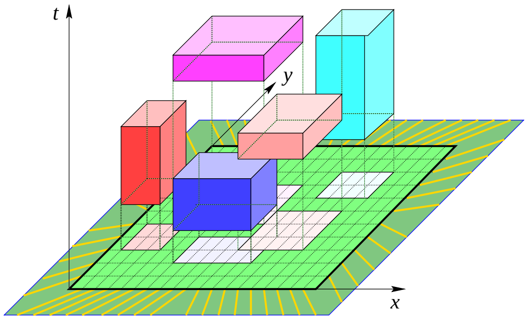

Figure 1: *An FPGA and a set of five jobs, shown in ordinary two-dimensional space and in three-dimensional space-time. Jobs must be placed inside the chip and must not overlap if executed simultaneously on the chip.*

Related Work. We are not aware of any exact study of more-dimensional scheduling problems with order constraints. For a comprehensive survey of classical "one-dimensional" scheduling problems, the reader is referred to [21]. Closest to our problems is the class of so-called *resource-constrained project scheduling problems* (RCSP), which can be interpreted as a step towards more-dimensional packing problems: In addition to a duration t_i and precedence constraints on the temporal order of job, each job may have a number of other "sizes" $x_i^{(j)}$ etc. that indicate the use of some resources. The total amount $\sum_i x_i^{(j)}$ of each resource is limited at any given time. See the book [29] and the references in the article [24] for an extensive survey of recent work in this area. Even though RCSP's can be formulated as integer problems, solving resource-constrained scheduling problems is already quite hard for instances of relatively moderate size: The standard benchmark library used in this area consists of instances with 30, 60, 90 and 120 jobs. Virtually all work deals with lower and upper bounds on these instances, and even for instances with 60 jobs, a considerable number has not yet been solved to optimality.

It is easy to see that any more-dimensional packing problem (possibly with precedence constraints on the temporal order) can be relaxed to a resource-constrained scheduling problem. However, the example in Figure 2 shows that the converse is not true, even for small instances of two-dimensional packing problems without any precedence constraints: An optimal solution for the corresponding resource-constrained scheduling problem may not correspond to a feasible arrangement of rectangles for the original packing problem. (We leave it to the reader to verify the latter claim.)

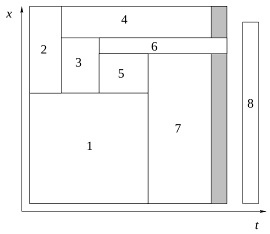

Figure 2: *A set of jobs that is feasible for scheduling with one resource constraint, but infeasible for two-dimensional packing: Job 8 does not violate a resource constraint, but does not fit as a contiguous rectangle.*

More-dimensional packing problems (without order constraints) have been considered by a great number of authors, but only few of them have dealt with the exact solution of general two-dimensional problems. See [6, 8] for an overview. It should be stressed that unlike one-dimensional packing problems, more-dimensional packing problems allow no straightforward formulation as integer programs: After placing one box in a container, the remaining feasible space will in general not be convex. Moreover, checking whether a given set of boxes fits into a particular container (the so-called *orthogonal packing problem*, OPP) is trivial in one-dimensional space, but NP-hard in higher dimensions.

Nevertheless, attempts have been made to use standard approaches of mathematical programming. Beasley [2] and Hadjiconstantinou and Christofides [15] have used a discretization of the available positions to an underlying grid to get a 0-1 program with a pseudopolynomial number of variables and constraints. Not surprisingly, this approach becomes impractical beyond instances of rather moderate size. More recently, Padberg [25] gave a *mixed integer programming* formulation for three-dimensional packing problems, similar to the one anticipated by Schepers [26] in his thesis. Padberg expressed the hope that using a number of techniques from branch-and-cut will be useful; however, he did not provide any practical results to support this hope.

In [6, 8, 9, 10, 28], a different approach to characterizing feasible packings and constructing optimal solutions is described. A graph-theoretic characterization of the relative position of the boxes in a feasible packing (by so-called *packing classes*) is used, which represent d-dimensional packings by a d-tuple of interval graphs (called *component graphs*) that satisfy two extra conditions. This factors out a great deal of symmetries between different feasible packings, it allows to make use of a number of elegant graph-theoretic tools, and it reduces the geometric problem to a purely combinatorial one without using brute-force methods like introducing an underlying coordinate grid. Combined with good heuristics for dismissing infeasible sets of boxes [7], a tree search for constructing feasible packings was developed. This exact algorithm has been implemented; it outperforms previous methods by a clear margin.

See our previous papers for details. For the benefit of the reader, a concise description is contained in Appendix A.

The Graph Theory of Order Constraints. In the context of scheduling with precedence constraints, a natural problem is the following, called *transitive ordering with precedence constraints* (TOP): Consider a partial order $P = (V, \prec)$ of precedence constraints and a (temporal) 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 [18] 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 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.

Results of this paper. In this paper, we give the first exact study of more-dimensional packing with order constraints, which can also be interpreted as *more-dimensional scheduling problems*. We develop a general framework for problems of this type by giving 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, our conditions can be used quite effectively in the context of a branch-and-bound framework, since it can recognize infeasible subtrees at "high" branches of the search tree. In particular, we describe how to find an exact solution to the problem of minimizing the height of a container of given base area. If this third dimension represents time, this amounts to minimizing the makespan of a more-dimensional scheduling problem. We validate the usefulness of these concepts and results by providing computational results. Other problem versions (like more-dimensional knapsack or bin packing problems with order constraints) can be treated similarly.

The rest of this paper is organized as follows. In Section 2, we describe basic assumptions and some terminology. In Section 3, we introduce precedence constraints, describe the mathematical foundations for incorporating them into the search, and explain how to implement the resulting algorithms. Finally, we present computational results for a number of different benchmarks in Section 4.

2 Preliminaries

Problem instances. We assume that a problem instance is given by a *set of jobs* V. Each job has a *spatial requirement* in the x- and y-direction, denoted by $w_x(v)$ and $w_y(v)$, and a *duration*, denoted by a size $w_t(v)$ along the time axis. The available space H consists of an area of size $h_x \times h_y$. In addition, there may be an overall allowable time h_t for all jobs to be completed. A *schedule* is given by a start time $p_t(v)$ for each job. A schedule is *feasible*, if all jobs can be carried out without overlap of computation jobs in time or space, such that all jobs are within spatial and temporal bounds.

Graphs. Some of our descriptions make use of a number of different graph classes. An (undirected) graph $G = (V, E)$ is given by a set of vertices V, and a set of edges E; each edge describes the adjacency of a pair of vertices, and we write $\{u, w\}$ for an edge between vertices u and w. For a graph G, we obtain the *complement graph* $\overline{G}$ by exchanging the set E of edges with the set $\overline{E}$ of non-edges. In a directed graph $D = (V, A)$, edges are oriented, and we write (u, w) to denote an edge directed from u to w. A graph $G = (V, E)$ is a *comparability graph* if the edges E can be oriented to a set of directed arcs A, such that we get the transitive closure of a partial order, i.e., a cycle-free digraph for which the existence of edges $(u, v) \in A$ and $(v, w) \in A$ for any $u, v, w \in V$ implies the existence of $(u, w) \in A$.

Precedence constraints. Mathematically, a set of precedence constraints is given by a partial order $P = (V, \prec)$ on V. The relations in $\prec$ form a directed acyclic graph $D_P = (V, A_P)$, where A_P is the set of directed arcs. In the presence of such a partial order, a feasible schedule is assumed to satisfy the capacity constraints of the container, as well as these additional constraints.

Packing problems. In the following, we treat jobs as axis-aligned three-dimensional boxes with given orientation, and feasible schedules as arrangements of boxes that satisfy all side constraints. This is implied by the term of a *feasible packing*. There may be different types of objective functions, corresponding to different types of packing problems. The *Orthogonal Packing Problem* (OPP) is to decide whether a given set of boxes can be placed within a given "container" of size $h_x \times h_y \times h_t$. For the Constrained OPP (COPP), we also have to satisfy a partial order $P = (V, \prec)$ of precedence constraints in the t-dimension. (To emphasize the motivation of temporal precedence constraints, we write t to suggest that the time coordinate is constrained, and x and y to imply that the space coordinates are unrestricted. Clearly, our approach works the same way when dealing with spatial restrictions.)

There are various optimization problems that have the OPP or COPP as their underlying decision problem. Since our main motivation arises from dynamic chip reconfigurations, where we want to minimize the overall running time, we focus on the *Constrained Strip Packing Problem* (CSPP), which is to minimize the size h_t for a given base size $h_x \times h_y$, such that all boxes fit into the container $h_x \times h_y \times h_t$. Clearly, we can use a similar approach for other objective functions.

3 Problems with Precedence Constraints

As mentioned in the introduction, a key advantage of considering packing classes is that it allows to deal with packing problems independent of precise geometric placement, and that it allows arbitrary feasible interchanges of placement. However, for most practical instances, we have to satisfy additional constraints for the temporal placement, i.e., for the start times of jobs. For our approach, the nature of the data structures may simplify these problems from three-dimensional to purely two-dimensional ones: If the whole schedule is given, all edges E_t in one of the graphs are determined, so we only need to construct the edge sets E_x and E_y of the other graphs. As worked out in detail in [27, 28], this allows it to solve the resulting problems quite efficiently if the arrangement in time is already given.

A more realistic, but also more involved situation arises if only a set of precedence constraints is given, but not the full schedule. We describe in the following how further mathematical tools in addition to packing classes allow useful algorithms.

3.1 Packing Classes and Interval Orders

Any edge (v_1, v_2) in a component graph G_i corresponds to an overlap between the projections of boxes 1 and 2 onto the x_i-axis. (See Appendix A for a brief introduction of the underlying concepts.) This means that the complement graph $\overline{G_i}$ given by the complement $\overline{E_i}$ of the edge set E_i consists of all pairs of coordinate intervals that are "comparable": Either the first interval is "to the left" of the second, or vice versa. Any (undirected) graph of this type is a so-called *comparability graph* (see [14] for further details). By orienting edges to point from "left" to "right" intervals, we get a partial order of the set V of vertices, a so-called *interval order* [22]. Obviously, this order relation is transitive, i.e., $e \prec f$ and $f \prec g$ imply $e \prec g$, which is the reason why we also speak of a *transitive orientation* of the undirected comparability graph G_i. See Figure 3 for a (two-dimensional) example of a packing class, the corresponding comparability graph, a transitive orientation, and the packing corresponding to the transitive orientation.

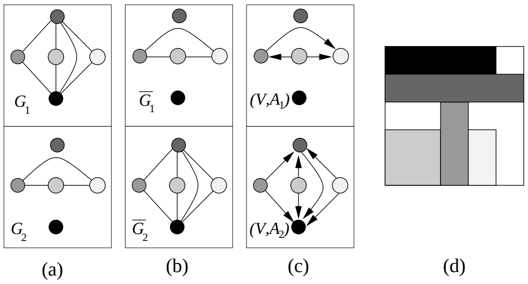

Figure 3: *(a) A two-dimensional packing class. (b) The corresponding comparability graphs. (c) A transitive orientation. (d) A feasible packing corresponding to the orientation.*

Now consider a situation where we need to satisfy a partial order $P = (V, A_P)$ of precedence constraints in the time dimension. It follows that each arc $a = (u, w) \in A_P$ in this partial order forces the corresponding undirected edge $e = \{u, w\}$ to be excluded from E_t. Thus, we can simply initialize our algorithm for constructing packing classes by fixing all undirected edges corresponding to A_P to be contained in $\overline{E_t}$. After running the original algorithm, we may get additional comparability edges. As the example in Figure 4 shows, this causes an additional problem: Even if we know that the graph $\overline{G_t}$ has a transitive orientation, and all arcs $a = (u, w)$ of the precedence order (V, A_P) are contained in $\overline{E_t}$ as $e = \{u, w\}$, it is not clear that there is a transitive orientation that contains all arcs of A_P.

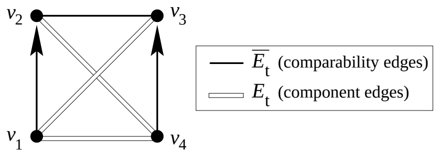

Figure 4: *A comparability graph* $\boldsymbol{\overline{G_t} = (V, \overline{E_t})}$ *with a partial order* $\boldsymbol{P}$ *contained in* $\boldsymbol{\overline{E_t}}$*, such that there is no transitive orientation of* $\boldsymbol{\overline{G_t}}$ *that extends* $\boldsymbol{P}$*.*

3.2 Finding Feasible Transitive Orientations

Consider a comparability graph $\overline{G}$ that is the complement of an interval graph G. The problem TOP of deciding whether $\overline{G}$ has a transitive orientation that extends a given partial order P has been studied in the context of scheduling. Korte and Möhring [18] give a linear-time algorithm for determining a solution, or deciding that none exists. Their approach is based on a very special data structure called *modified PQ-trees.*

In principle, it is possible to solve our more-dimensional packing problems with precedence constraints by adding this algorithm as a black box to test the leaves of our search tree for packing classes:

In case of failure, backtrack in the tree. However, the resulting method cannot be expected to be reasonably efficient: During the course of our tree search, we are not dealing with one fixed comparability graph, but only build it while exploring the search tree. This means that we have to expect spending a considerable amount of time testing similar leaves in the search tree, i.e., comparability graphs that share most of their graph structure. It may be that already a very small part of this structure that is fixed very "high" in the search tree constitutes an obstruction that prevents a feasible orientation of all graphs constructed below it. So a "deep" search may take a long time to get rid of this obstruction. This makes it desirable to use more structural properties of comparability graphs and their orientations to make use of obstructions already "high" in the search tree.

3.3 Implied Orientations

As in the basic packing class approach, we consider the component graphs G_i and their complements, the comparability graphs $\overline{G_i}$. This means that we continue to have three basic states for any edge: (1) edges that have been fixed to be in E_i, i.e., *component edges*; (2) edges that have been fixed to be in $\overline{E_i}$, i.e., *comparability edges*; (3) *unassigned edges.*

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 each comparability edge: (2a) one possible orientation; (2b) the opposite possible orientation; (2c) no assigned orientation.

A stepping stone for this approach arises from considering the following two configurations – see Figure 5:

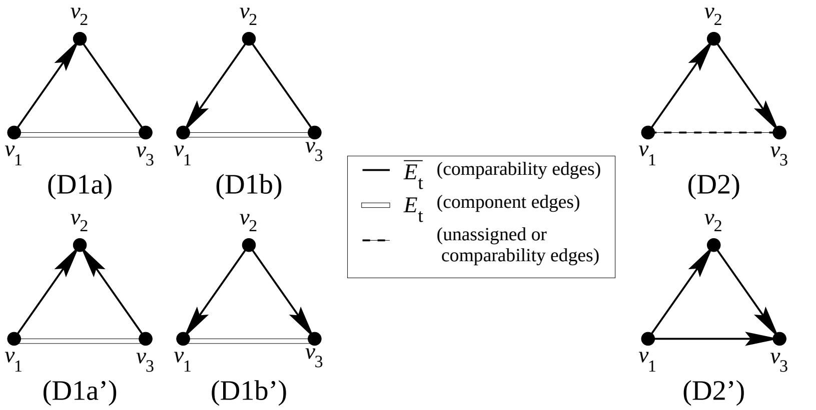

Figure 5: *Implications for edges and their orientations: Above are path implications (D1, left) and transitivity implications (D2, right); below the forced orientations of edges.*

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

The second configuration consists of two directed comparability edges $(v_1, v_2), (v_2, v_3)$. In this case we know that the edge $\{v_1, v_3\}$ must also be a comparability edge, with an orientation of (v_1, v_3). Since this configuration arises directly from transitivity in $\overline{G_i}$, 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 only sequences of path implications, we get a partition of comparability edges into *path implication classes* that will be used in more detail in Appendix B: Two comparability 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 4. Here, all three comparability 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 comparability edges, since we are dealing with an equivalence relation.

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

As the example in Figure 6 shows, only excluding path conflicts when recursively carrying out path implications does not suffice to guarantee the existence a feasible orientation: Working through the queue of path implications, we end up with a directed cycle, which violates a transitivity implication.

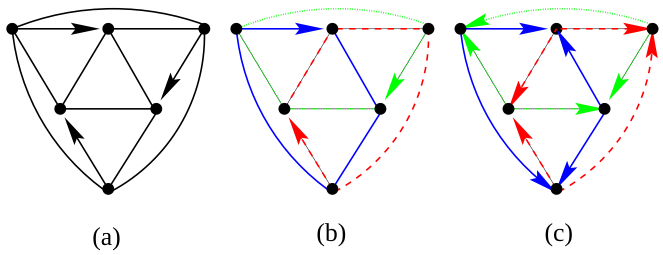

Figure 6: *(a) A graph $\overline{G_t}$ with a partial order formed by three directed edges; (b) there are three path implication classes that each have one directed arc; (c) carrying out path implications creates directed cycles, i.e., transitivity conflicts.*

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.

These necessary conditions are also sufficient:

Theorem 1 (Fekete, Köhler, Teich) *Consider a partial order $P = (V, A_P)$ with arc set A_P 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 can be carried out in any order without creating a path conflict or a transitivity conflict.*

A proof and further mathematical details are described in our report [5]. A sketch is contained in Appendix B. The interested reader may take note that we are extending previous work by Gallai [11], who extensively studied implication classes of comparability graphs. See Kelly [17], Möhring [22] for informative surveys on this topic, and Krämer [20] for an application in scheduling theory.

3.4 Solving OPP's with Precedence Constraints

We start by fixing for all arcs $(u, v) \in A$ the edge $\{u, v\}$ as an edge in the comparability graph $\overline{G_t}$, and we also fix its orientation to be (u, v). In addition to the tests for enforcing the conditions for unoriented packing classes (C1, C2, C3), we employ the implications suggested by conditions D1 and D2. For this purpose, we check directed edges in $\overline{G_t}$ for being part of a triangle that gives rise to either implication.

Any newly oriented edge in $\overline{G_t}$ 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 1; in particular, the theorem guarantees that we can carry out implications in an arbitrary order.

4 Computational Experiments

In the following we present our results for different types of instances: The video-codec benchmark described in Section 4.1 arises from an actual application to FPGA's. In Section 4.2 we give a number of results arising from different geometric packing problems. More details (and a number of figures showing the optimal packings for various order constraints) are contained in Appendix C.

Our code was implemented in C++ and run on a SUN Ultra 2.

4.1 Video-Codec Benchmark

Figure 7 shows a block diagram of the operation of a hybrid image sequence coder/decoder that arises from the FPGA application. The purpose of the coder is to compress video images using the H.261 standard. In this device, transformative and predictive coding techniques are unified. The compression factor can be increased by a predictive method for motion estimates: blocks inside a frame are predicted from blocks of previous images.

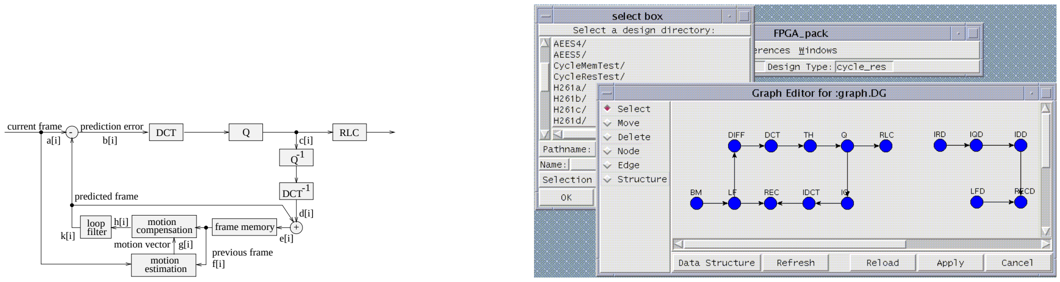

Figure 7: *Block diagram of a video-codec (H.261) (left); problem graph of the video-codec (right).*

The blocks of the operational description shown in the figure possess the granularity of more complex functions. However, this description contains no information corresponding to timing, architecture, and mapping of blocks onto an architecture. Figure 7 (right) shows a problem graph G of the video-codec The problem graph contains a subgraph for the coder and one subgraph for the decoder. For realizing the device, we have a library of three different modules. One is a simple processor core with a (normalized) area requirement of 625 units (25 x 25 cells, normalized to other modules in order to obtain a coarser grid) called PUM. Secondly, there are two dedicated special-purpose modules: a block matching module (BMM) that is used for motion estimation and requires 64 x 64 = 4096 cells; and a module DCTM for computing DCT/IDCT-computations, requiring 16 x 16 = 256 cells. The makespan was minimized for different latency constraints. The result is shown in Table 1.

Table 1: Optimizing reconfigurations for the Video-Codec

test	container sizes			CPU-time (s)
	h_t	h_x	h_y	
1	59	64	64	24.87 s

4.2 Two-dimensional packing problems

Here we describe computational results for two types of two-dimensional objects. The first was constructed from a particularly difficult random instance of 2-dimensional knapsack (see [6]). Results are given for order constraints of increasing size. In order to give a better idea of the computational difficulty, we give separate running times for finding an optimal feasible solution, and for proving that this solution is best possible. (See the appendix for the exact sizes of the 17 rectangles involved, and for figures of optimal packings.)

The second class of instances arises from the well-known tiling of a 112x112 square by 21 squares of different sizes. Again, we have added order constraints of various sizes. For the instance square21-2mat (with order constraints in two dimensions), we could not close the gap between upper and lower bound. For this instance, we report the running times for achieving the best known bounds.

Table 2: Optimal packing with order constraints

instance	optimal h_t	h_x	upper bound	lower bound
okp17-0	169	100	7.29 s	179 s
okp17-1	172	100	6.73 s	1102 s
okp17-2	182	100	5.39 s	330 s
okp17-3	184	100	236 s	553 s
okp17-4	245	100	0.17 s	0.01 s
square21-no	112	112	84.28 s	0.01 s
square21-mat	117	112	15.12 s	277 s
square21-tri	125	112	107 s	571 s
square21-2mat	[118,120]	[118,120]	346 s	476 s

Acknowledgments

We are extremely grateful to Jörg Schepers for letting us continue the work with the packing code that he started as part of his thesis, and for several helpful hints, despite of his departure to industry. We also thank Marc Uetz for a useful discussion on resource-constrained scheduling.

References

[1] Atmel. *AT6000 FPGA configuration guide.* Atmel Inc.

[2] J. E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, **33**, 1985, pp. 49–64.

[3] J. E. Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the Operations Research Society*, **41**, 1990, pp. 1069–1072.

[4] S. P. Fekete, E. Köhler, and J. Teich. Optimal FPGA Module Placement with Temporal Precedence Constraints. Technical Report 696-2000, TU Berlin, available at `http://www.math.tu-berlin.de/coga/publications`.

[5] S. P. Fekete, E. Köhler, and J. Teich. Extending partial suborders and implication classes. Technical Report 697-2000, TU Berlin.

[6] 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.

[7] S. P. Fekete and J. Schepers. New classes of lower bounds for bin packing problems. In *Proc. Integer Programming and Combinatorial Optimization (IPCO'98)*, volume 1412, pages 257–270, Springer Lecture Notes in Computer Science, 1998.

[8] S. P. Fekete and J. Schepers. On more-dimensional packing I: Modeling. Technical Report 97-288, Center for Applied Computer Science, Universität zu Köln, Available at `http://www.zpr.uni-koeln.de/ABS/~papers`, 1997.

[9] S. P. Fekete and J. Schepers. On more-dimensional packing II: Bounds. Technical Report 97-289, Universität zu Köln, 1997.

[10] S. P. Fekete and J. Schepers. On more-dimensional packing III: Exact algorithms. Technical Report 97-290, Universität zu Köln, 1997.

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

[12] A. Ghouilà-Houri. Caractérization des graphes non orientés dont on peut orienter les arrêtes de manière à obtenir le graphe d'une relation d'ordre. *C.R. Acad. Sci. Paris*, **254** 1962, 1370–1371.

[13] P.C. Gilmore and A.J. Hoffmann. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, **16** 1964, 539–548.

[14] M. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.

[15] E. Hadjiconstantinou and N. Christofides. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operations Research*, **83** (1995), 39–56.

[16] C.-H. Huang and J.-Y. Juang. A partial compaction scheme for processor allocation in hypercube multiprocessors. In *Proc. of 1990 Int. Conf. on Parallel Proc.*, pages 211–217, 1990.

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

[18] 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.

[19] N. Korte and R. H. Möhring. An incremental linear–time algorithm for recognizing interval graphs. *Siam Journal of Computing*, **18**, 1989, pp. 68-81.

[20] A. Krämer. *Scheduling Multiprocessor Tasks on Dedicated Processors.* Doctoral thesis, Fachbereich Mathematik und Informatik, Universität Osnabrück, 1995.

[21] E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys. *Sequencing and Scheduling: Algorithms and Complexity.* in: S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin. Logistics of Production and Inventory, vol. 4, Handbooks in Operations Research and Management, North–Holland, Amsterdam, 1993, 445–522.

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

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

[24] R. H. Möhring, A. S. Schulz, F. Stork, and M. Uetz. Solving Project Scheduling Problems by Minimum Cut Computations. Technical Report 680-2000, TU Berlin.

[25] M. Padberg. Packing small boxes into a big box. *Math. Meth. of Op. Res.*, **52** (2000), 1–21.

[26] J. Schepers. Exakte Algorithmen für orthogonale Packungsprobleme. Doctoral thesis, Universität Köln, 1997, available as Technical Report 97-302.

[27] J. Teich, S. Fekete, and J. Schepers. Compile-time optimization of dynamic hardware reconfigurations. In *Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 1097–1103, Las Vegas, U.S.A., June 1999.

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

[29] J. Weglarz. *Project Scheduling. Recent Models, Algorithms and Applications.* Kluwers Academic Publishers, Norwell, MA, USA, 1999.

[30] Xilinx. XC6200 field programmable gate arrays. Technical report, Xilinx, Inc., October 1996.

Appendix A: Solving Unconstrained Packing Problems

A.1 A General Framework

If we have an efficient method for solving OPP's, we can also solve BMP's and SPP's by using a binary search. However, deciding the existence of a feasible more-dimensional packing is a hard problem in higher dimensions, and proposed methods suggested by other authors [2, 15] have been of limited success.

Our framework uses a combination of different approaches to overcome these problems:

1. Try to disprove the existence of a packing by fast and good classes of lower bounds on the necessary size.
2. In case of failure, try to find a feasible packing by using fast heuristics.
3. If the existence of a packing is still unsettled, start an enumeration scheme in form of a branch-and-bound tree search.

By developing good new bounds for the first stage, we have been able to achieve a considerable reduction of the number of cases where a tree search needs to be performed. (Mathematical details for this step are described in [7, 9].) However, it is clear that the efficiency of the third stage is crucial for the overall running time when considering difficult problems. Using a purely geometric enumeration scheme for this step by trying to build a partial arrangement of boxes is easily seen to be immensely time-consuming. In the following, we describe a purely combinatorial characterization of feasible packings that allows to perform this step more efficiently.

A.2 Packing Classes

Consider a feasible packing in d-dimensional space, and project the boxes onto the three coordinate axes. This converts the one d-dimensional arrangement into d one-dimensional ones. (See Figure 8 for an example in $d = 2$.) By disregarding the exact coordinates of the resulting intervals in direction i and only considering their overlap, we get the *component graph* $G_i = (V, E_i)$: Two boxes u and v are connected by an edge in G_i, iff they have overlapping x_i-coordinates. By definition, these graphs are *interval graphs* that have been studied intensively in graph theory (see [14, 22]), and that have a number of very useful algorithmic properties.

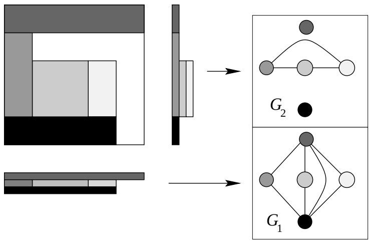

Figure 8: The projections of the boxes onto the coordinate axes define interval graphs (here in 2D: $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$).

Considering sets of d component graphs G_i instead of complicated geometric arrangements has some clear advantages. (Algorithmic implications for our specific purposes are discussed further down.) It is not hard to check that the following three conditions must be satisfied by all d-tuples of graphs G_i that are constructed from a feasible packing:

C1: G_i is an interval graph, $\forall i \in \{1, \cdots, d\}$.

C2: Any independent set S of G_i is i-admissible, $\forall i \in \{1, \cdots, d\}$, i.e., $w_i(S) = \sum_{v \in S} w_i(v) \leq h_i$, since all boxes in S must fit into the container in the ith dimension.

C3: $\cap_{i=1}^{d} E_i = \emptyset$. In other words, there must be at least one dimension in which the corresponding boxes do not overlap.

A d-tuple of component graphs satisfying these necessary conditions is called a *packing class*. The remarkable property (proven in [26, 8]) is that these three conditions are also sufficient for the existence of a feasible packing.

Theorem 2 *A d-tuple of graphs $G_i = (V, E_i)$ corresponds to a feasible packing, iff it is a packing class, i. e., if it satisfies the conditions* **C1, C2, C3**.

This allows it to consider only packing classes in order to decide the existence of a feasible packing, and disregard most of the geometric information.

A.3 Solving OPP's

Our search procedure works on packing classes, i.e., triples of component graphs with the properties C1, C2, C3. Since each packing class represents not only a single packing but a whole family of equivalent packings, we are effectively dealing with more than one possible candidate for an optimal packing at a time. (The reader may check for the example in Figure 8 that there are 36 different feasible packings that correspond to the same packing class.)

The search tree is traversed by Depth First Search, see [10, 26] for details. Branching is done by fixing an edge $\{b, c\} \in E_i$ or $\{b, c\} \notin E_i$. After each branching step, it is checked whether one of the three conditions C1, C2, C3 is violated, or whether a violation can only be avoided by fixing further edges. This is easy for two of the conditions: enforcing C3 is obvious; property C2 is hereditary, so adding edges to E_i later will keep it satisfied. (Note that computing maximum weighted cliques on comparability graphs can be done efficiently, see [14].) In order to ensure that property C1 is not violated, we use a number of graph-theoretic characterizations of interval graphs and comparability graphs. These characterizations are based on two forbidden substructures (again, see [14] for details; the first condition is based on the classical characterizations by [12, 13]: a graph is an interval graph *and* its complement has a transitive orientation, iff it does not contain any induced chordless cycle of length 4.) In particular, the following configurations have to be avoided:

1. induced chordless cycles of length 4 in E_i;
2. so-called 2-chordless odd cycles in the set $\overline{E_i}$ of edges excluded from E_i (see [10, 14] for details);
3. infeasible stable sets in E_i.

Each time we detect such a fixed subgraph, we can abandon the search on this node. Furthermore, if we detect a fixed subgraph, except for one unfixed edge, we can fix this edge, such that the forbidden subgraph is avoided.

Our experience shows that these conditions are already useful when only small subsets of edges have been fixed, since by excluding small sub-configurations, like induced chordless cycles of length 4, each branching step triggers a cascade of more fixed edges.

Appendix B: Proving Theorem 1

For the proof of Theorem 1 we make use of so-called *modular decompositions* of graphs. This concept was introduced by Gallai [11] for studying comparability graphs. Due to space restrictions we cannot give a complete introduction of the theoretical background. However, we give a brief overview of the main ideas that are necessary to understand the proof of Theorem 1. The interested reader is referred to [17, 23] and [5] for more details.

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. Gallai showed that each graph G has a unique decomposition (the so-called *canonical decomposition*) of its vertex set into a set of modules (so-called *quasi-maximal 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*; Gallai showed that in this case there is a unique maximal decomposition of the vertex set of G into non-trivial modules.

The *decomposition graph* $G^{\#}$ of a graph G is the quotient of G by the canonical decomposition $\{A_1, \ldots, A_q\}$, 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.

The *tree decomposition* of a graph G is obtained by iterating the canonical decomposition on the graphs induced by the modules of the canonical decomposition of the corresponding higher level.

An important property of this decomposition scheme is its close relationship to the concept of implication classes (described in Section 3). Gallai observed the following properties of implication classes with respect to the modular decomposition:

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 $\{A_1, \ldots, A_q\}$ is the canonical decomposition of G, 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 also the fastest known algorithms for recognizing comparability graphs and also permutation graphs make extensively use of this relationship. Gallai used the above properties (among others) for proving the following theorem.

Theorem B.1 (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 AB-edges (for A and B successors of H in T) in the same direction that A and B are oriented in $H^{\#}$.*

From this theorem one can draw the following helpful corollaries.

Corollary B.2 *A graph G is a comparability graph if and only if every decomposition graph in the tree-decomposition is a comparability graph.*

Corollary B.3 *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 lemmas.

Lemma B.4 *Given an orientation of a comparability graph that is consistent with the forcing for each implication class. This orientation is transitive if and only if the induced orientation of each of the decomposition graphs in the tree decomposition is transitive.*

Lemma B.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 $H^{\#}$.*

Let $P = (V, A_P)$ be a partial order with arc set A_P contained in the edge set E of a comparability graph G. We call a directed graph F the ***pt**-closure* of P if F consists of all arcs of P together with all orientations of edges of G that are implied by a sequence of path and triangle implications of arcs of P.

Now we are ready to sketch the proof of Theorem 1.

Theorem B.6 *Let $G = (V, E)$ be a comparability graph and P a partial orientation on the edges of G. P can be extended to a transitive orientation on G if and only if conditions* **D1** *and* **D2** *are satisfied.*

Proof. If there is a transitive orientation F of G that contains P, then F trivially satisfies **D1** and **D2**.

Suppose now that **D1** and **D2** are satisfied, let F be the **pt**-closure of P, and let T be the tree decomposition of G. We show that F can be extended to a transitive orientation of G.

First observe that in F an orientation conflict either stems from an orientation conflict on a single edge class containing arcs of P, contradicting **D1**, or otherwise implies a cycle conflict in F, contradicting **D2**. Hence, F is an orientation of edges of G and there is no orientation or cycle conflict in F (i.e. all implication classes in F are oriented *conflict-free*). By Corollary B.3, every single conflict-free oriented implication class of G is extendible to a transitive orientation of G. By this observation, the orientation of an implication class C in a **pt**-closure F implies an orientation of the edge(s) corresponding to this implication class in the decomposition graphs of T. More precisely, 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 B.1). For a prime type node H the set of edges joining different A_i's (see above) forms exactly one implication class C_E of G. 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 B.3, the implied orientation of G is transitive.

By Corollary B.3 neither a parallel type node of T can create a contradiction to transitivity (it does not contain any edges) nor a prime type node of T can create a contradiction, since all its edges are contained in only one implication class and, since all implication classes of F are oriented conflict-free, the corresponding orientation induced by F on this single implication class is transitive.

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 this partial orientation has to be cyclic, because $H^{\#}$ is a complete graph and every acyclic partial orientation of a complete graph can trivially be extended to a transitive orientation of this complete graph. However, by the definition of T and the implied orientation of $H^{\#}$ by F, an directed cycle in $H^{\#}$ immediately implies an oriented cycle in F, contradicting **D2**. □

Appendix C: Benchmark Problems and Their Solutions

Here we provide details of our benchmark instances and show the best solutions we found. The instances consist of the boxes listed in Table 3. For easier reference, the boxes in the `okp17` instances are labeled 1-17 in the given order, while the squares in the `square21` instances are labeled by their edge lengths. Note that instance square21-2mat has constraints in 2 dimensions.

Table 3: The problem instances `okp17` and `square21`.

`okp17`:	base width of container = 100, number of boxes = 17
sizes =	[(8,81),(5,76),(42,19),(6,80),(41,48),(6,86),(58,20),(99,3),(9,52),(100,14),(7,53),(24,54),(23,77),(42,32),(17,30),(11,90),(26,65)]
okp17-0:	no order constraints
okp17-1:	11→8, 11→16
okp17-2:	11→8, 11→16, 8→16
okp17-3:	11→8, 11→16, 8→16, 8→17, 11→7, 16→7
okp17-4:	11→8, 11→16, 8→16, 8→17, 11→7, 16→7, 17→16

`square21`:	base width of container = 112, number of boxes = 21
sizes =	[(50,50),(42,42),(37,37),(35,35),(33,33),(29,29),(27,27),(25,25),(24,24),(19,19),(18,18),(17,17),(16,16),(15,15),(11,11),(9,9),(8,8),(7,7),(6,6),(4,4),(2,2)]
square21-0:	no order constraints
square21-mat:	2→4, 6→7, 8→9, 11→15, 16→17, 18→19, 24→25, 27→29, 33→35, 37→42, 2→50, 50→4
square21-tri:	2→15, 15→17, 2→27, 4→16, 16→29, 4→29, 6→17, 17→33, 6→33, 7→18, 18→35, 7→35, 8→19, 19→37, 8→37, 9→24, 24→42, 9→42, 11→25, 25→50, 11→50
square21-2mat:	x-constraints: 2→19, 6→25, 8→29, 11→35, 16→42, 18→4, 24→7, 27→9, 33→15, 37→17, 50→4, 18→50 y-constraints: 2→4, 6→7, 8→9, 11→15, 16→17, 18→19, 24→25, 27→29, 33→35, 37→42, 2→50, 50→4

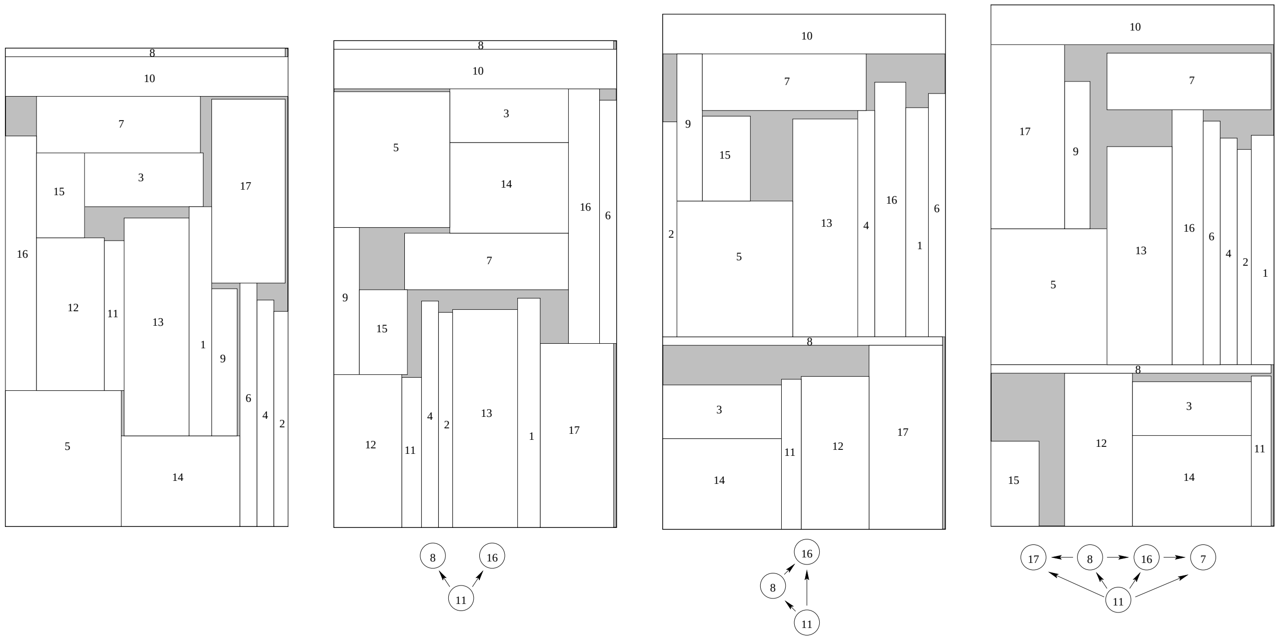

Figure 9: *(a) An optimal packing of okp17-0 of height 169; (b) an optimal packing of okp17-1 of height 172; (c) an optimal packing of okp17-2 of height 182; (d) An optimal packing of okp17-3 of height 184.*

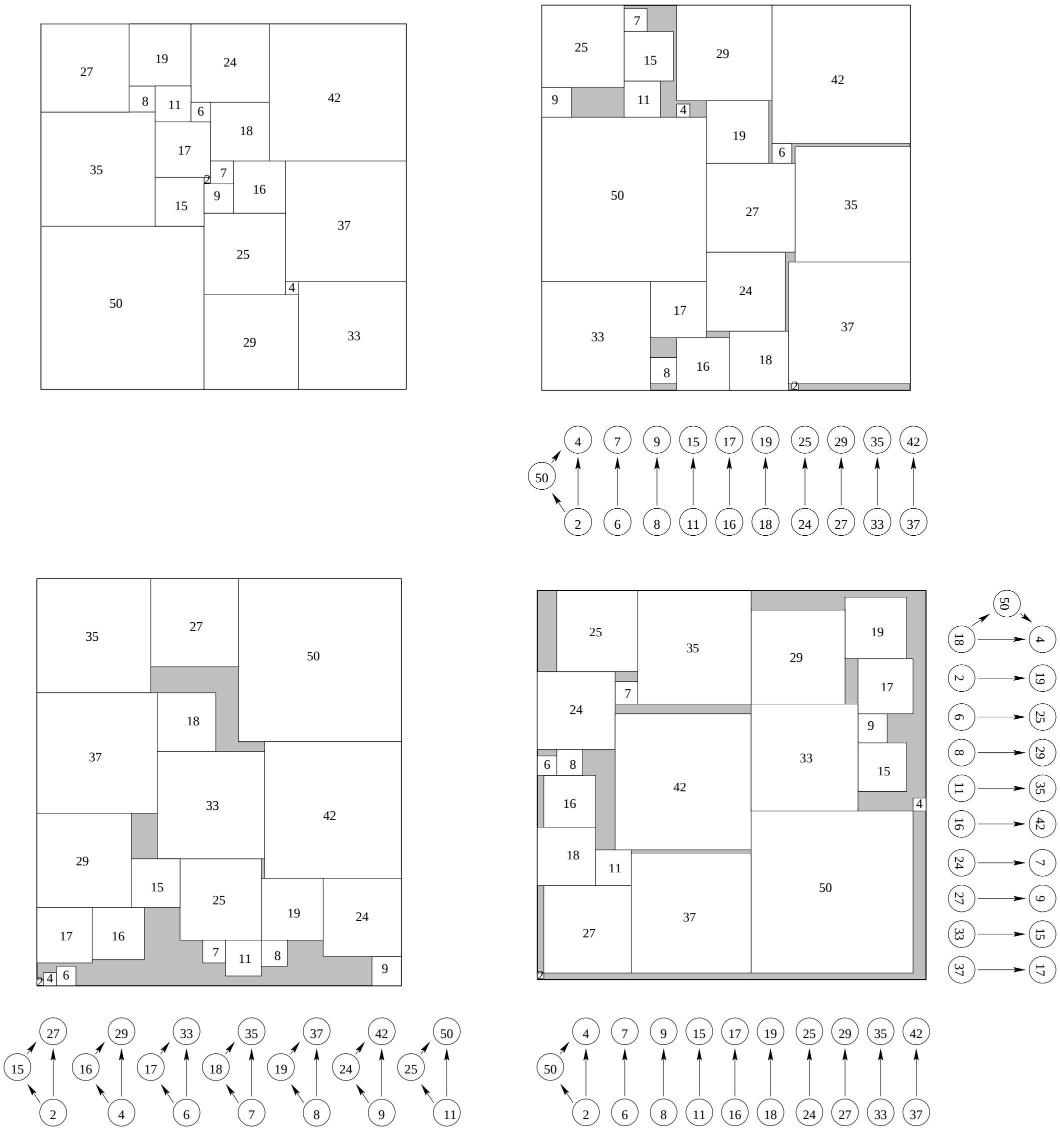

Figure 10: *(a) An optimal packing of square21-0 of height 112; (b) an optimal packing of square21-mat of height 117; (c) an optimal packing of square21-tri of height 125; (d) a packing of square21-2mat of size 120x120.*

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

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

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:* A note on scheduling problems with irregular starting time costs

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

624/1999 *Rolf H. Möhring:* Verteilte Verbindungssuche im öffentlichen Personenverkehr: Graphentheoretische Modelle und Algorithmen

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

611/1998 *Rolf H. Möhring and Markus W. Schäffter:* Scheduling series-parallel orders subject to 0/1-communication delays

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`