Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Discrete Mathematics 36 (2010) 423–429
Electronic Notes in DISCRETE MATHEMATICS
www.elsevier.com/locate/endm

MPQ-trees for orthogonal packing problem

Cédric Joncour [2]

University of Bordeaux (LaBRI, INRIA), France

Arnaud Pêcher [1]

University of Toulouse (IRIT, INRIA), France

Petru Valicov [3]

University of Bordeaux (LaBRI), France

Abstract

Finding a feasible solution for a bi-dimensional Orthogonal Packing Problem (OPP-2) consists in deciding whether a set of rectangular boxes (items) can be packed in a "big" rectangular container without overlapping. The rotation of items is not allowed. In this paper we present a new algorithm for solving OPP-2, based on the characterization of solutions using interval graphs proposed by Fekete and Schepers. The algorithm uses MPQ-trees - combinatorial structures introduced by Korte and Möhring.

Keywords: Orthogonal Packing Problem, interval graph, MPQ-tree.

[1] Email:arnaud.pecher@irit.fr
[2] Email:cedric.joncour@math.u-bordeaux1.fr
[3] Email:valicov@labri.fr

1571-0653/$ – see front matter
doi:10.1016/j.endm.2010.05.054

Let V be a set of D-dimensional rectangular shapes. For $d \in \{1, ..., D\}$ and $v \in V$, $w_d(v) \in \mathbb{Q}^+$ describes the length of v in the dimension d. The same notation is used for the container C.

The *D-dimensional orthogonal packing problem* (OPP-D) is to decide if V fits into the container C without overlapping (V is a *feasible* set). Formally speaking, we want to find out whether $\forall d \in \{1, ..., D\}$ there exists a function $f_d : V \to \mathbb{Q}^+$, such that $\forall v \in V, f_d(v) + w_d(v) \leq w_d(C)$ and $\forall v_1, v_2 \in V$, $(v_1 \neq v_2)$, $[f_d(v_1), f_d(v_1) + w_d(v_1)) \cap [f_d(v_2), f_d(v_2) + w_d(v_2)) = \emptyset$.

Let $p_v \in \mathbb{Q}^+$ be the value associated to an item $v \in V$. The *d-dimensional knapsack problem* (OKP-d) consists in computing a feasible set $V' \subseteq V$ such that $\sum_{v \in V'} p_v$ is maximal.

In this paper, we consider the bi-dimensional case, which has been the most studied so far [11,9,6,12,7,3,5,8,1].

1 Fekete & Schepers' model

For each dimension d, let us consider the projections of items on every axis x_d: let $G_d = (V, E_d)$ be the interval graph, with vertex set V and edge set E_d, such that ij is an edge if and only if the projections of items i and j intersect (see Fig. 1).

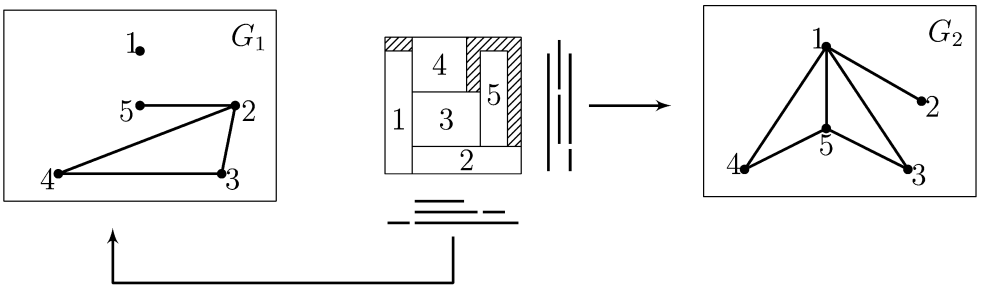

Fig. 1. Example of 2D packing and its associated interval graphs

Fekete and Schepers proved the following characterization of feasible sets:

Theorem 1.1 (Fekete & Schepers [10]) *A set of boxes V is feasible, if and only if there is a set of D graphs $G_d = (V, E_d)$, $d \in \{1, ..., D\}$ such that:*

P1: Every graph G_d is an interval graph.

P2: For every stable set S of G_d, $\sum_{s \in S} w_d(s) \leq w_d(C)$

$$P3: \bigcap_{j=1}^{D} E_d = \emptyset$$

Fekete, Schepers and van der Veen gave an efficient algorithm for solving OKP-D by solving its subproblem OPP-D [12]. Their algorithm is based upon the enumeration of the tuples of graphs satisfying the properties of theorem 1.1. Despite its efficiency, their algorithm may enumerate symmetrical solutions. Our aim is to handle those symmetry issues more efficiently.

2 Our approach

Our algorithm is based upon the characterization of interval graphs involving MPQ-trees, introduced by Korte and Möhring [13]. An MPQ-tree is an extension of the notion of PQ-tree defined by Booth and Lueker [4]. It allows the encoding of an interval graph by exploiting the property of consecutiveness of its maximal cliques. The main idea of our approach is to generate couples of MPQ-trees in order to obtain a couple of interval graphs which satisfies the conditions of theorem 1.1. Contrary to Fekete and Schepers, by generating MPQ-trees, we are able to assert that the search space stays within the set of interval graphs.

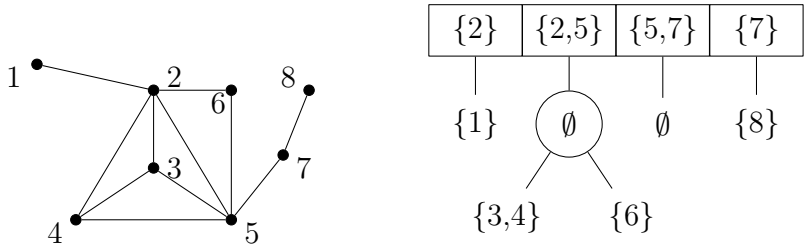

Fig. 2. An interval graph and its associated MPQ-tree.

Let G be an interval graph and T its associated MPQ-tree. Each branch of T represents a maximal clique of G. The internal nodes of T are of type P or Q. All the nodes (internal or leaves) are labeled by sets of vertices of G (potentially empty) as follows:

- a P node is labeled by a set of vertices of G which are *only* contained in *all* cliques represented by the subtree of T rooted in this node.
- a leaf is labeled by a set of vertices of G contained *only* in the clique represented by this leaf.

- a Q node, with m sons $F_1, ..., F_m$, is labeled by a list of sets $S_i, i \in \{1, ..., m\}$ each of them being called a *section*. Each section S_i ($i \in \{1, ..., m\}$) contains the vertices of G contained in *all* cliques represented by the subtree rooted in F_i and also in *all* cliques represented by the subtree rooted in another F_j ($j \in \{1, ..., m\}$ and $j \neq i$).

An interval graph is obtained from its $(M)PQ$-tree by reading its leaves from left to right. Hence, different $(M)PQ$-trees may encode the same interval graph. To avoid treating the same interval graph, the notion of *equivalence* of PQ-trees was introduced by Booth and Lueker [4]. This notion is conserved for MPQ-trees. Two MPQ-trees are equivalent if one can be obtained from the other by carrying out the following operations a finite number of times:

(i) Arbitrarily permute the sons of a node P

(ii) Reverse the order of sons of a node Q

One of the advantages of using MPQ-trees is that they allow us to easily represent a valid packing configuration. Figure 3 shows a possible solution with respect to the tree from figure 2 (the cliques correspond to the intersections of projections of items on the vertical axis):

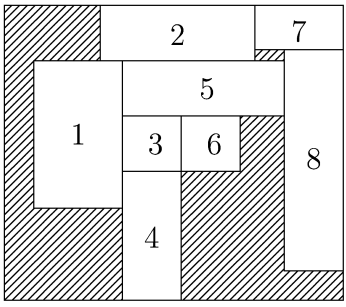

Fig. 3. A valid packing configuration corresponding to the MPQ-tree from figure 2

Using the properties of MPQ-trees described above, we designed an algorithm to generate an MPQ-tree for each dimension in order to check if a set of items is feasible. The algorithm tries to construct the tree recursively by adding vertices one by one, using the templates defined in [13]. The execution is continued as long as the properties P2 and P3 from Theorem 1.1 are satisfied. If for each dimension a valid MPQ-tree is constructed, then the given set of vertices is feasible.

In order to predict unfeasible solutions earlier, we introduced some optimizations using similar ideas from [9,8]. Since we have some information about the relative position of the items already inserted in the MPQ-tree,

we are able to estimate unused spaces, and cut the enumeration procedure rapidly if the remaining area is too small to contain the remaining items.

3 Computational results

We report the performance of our algorithm on 37 classical benchmarks for OKP-2 from [3,2,7,12]. To solve OKP-2 we used a basic branch-and-bound procedure to select the items to test. Once the items have been selected, the algorithm generating the MPQ-trees checks their feasibility.

The program was implemented in Java 6 and was tested on a PC (Pentium 4, 3GHz), which is comparable to the one used by Fekete and Schepers [12].

Table 1 shows the running times of our algorithm along with the ones existing in the literature, as reported in [12]. The first column (JPV) gives our runtimes. The columns FS and BB correspond respectively to the algorithms of Fekete, Schepers & van der Veen [12], and Baldacci & Boschetti [1]. The columns A0, A1, A2 and A3 correspond to the algorithms of Caprara and Monaci, as depicted in [5].

4 Conclusions

The running times of our algorithm are of interest since it is one of the two algorithms to solve all benchmarks within the time limit of 1800 seconds (with the exception of the instance `gcut13` which is still an open benchmark, the optimal value being unknown). Our running times are significantly better for 6 instances (`cgcut2`, `gcut3`, `gcut8`, `gcut11`, `gcut12` and `okp1`), though Fekete & Schepers' algorithm outperforms ours for the 2 instances `okp2` and `okp5`.

Benchmark	**JPV**	**FS**	**BB**	**A0**	**A1**	**A2**	**A3**
ngcut1 - 12	0	0	0				
hccut2 - 5	0	0	0				
cgcut1	0	0	0	0	1	1	1
cgcut2	**135**	>1800	>1800	>1800	>1800	533	531
cgcut3	3	0	95	23	23	4	4
gcut1	0	0	0	0	0	0	0
gcut2	0	0	0	0	0	25	0
gcut3	**0**	4	2	>1800	2	276	3
gcut4	109	195	46	>1800	346	>1800	376
gcut5	0	0	0	0	0	0	0
gcut6	0	0	1	0	0	9	0
gcut7	0	2	3	1	0	354	1
gcut8	**48**	253	186	1202	136	>1800	168
gcut9	0	0	0	0	0	0	0
gcut10	0	0	0	0	0	6	0
gcut11	**2**	8	3	16	14	>1800	16
gcut12	**6**	109	12	63	16	>1800	25
gcut13	>1800	>1800	>1800	>1800	>1800	>1800	>1800
okp1	**1**	10	779	24	25	72	35
okp2	67	20	288	>1800	>1800	1535	1559
okp3	1	5	0	21	1	465	10
okp4	1	2	14	40	2	0	4
okp5	319	11	190	40	>1800	513	488

Table 1
Running times in seconds

References

[1] Baldacci, R. and M. A. Boschetti, *A cutting plane approach for the two-dimensional orthogonal non-guillotine cutting stock problem*, EJOR **183** (2007), pp. 1136–1149.

[2] Beasley, J. E., *Algorithms for unconstrained two-dimensional guillotine cutting*, JORS **36** (1985), pp. 297–306.

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

[4] Booth, K. S. and G. S. Lueker, *Linear algorithms to recognize interval graphs and test for the consecutive ones property*, in: *Proceedings of the seventh Annual ACM Symposium on Theory of Computing (STOC'75)*, 1975, pp. 255–265.

[5] Caprara, A. and M. Monaci, *On the two-dimensional knapsack problem*, Oper. Res. Lett. **32** (2004), pp. 5–14.

[6] Carlier, J., F. Clautiaux and A. Moukrim, *New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation*, Computers and Operations Research **34** (2007), pp. 2223–2250.

[7] Christofides, N. and E. Hadjiconstantinou, *An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts*, European Journal of Operational Research **83** (1995), pp. 21–38.

[8] Clautiaux, F., J. Carlier and A. Moukrim, *A new exact method for the orthogonal packing problem*, European Journal of Operational Research **183** (2007), pp. 1196–1211.

[9] Clautiaux, F., A. Jouglet, J. Carlier and A. Moukrim, *A new constraint programming approach for the orthogonal packing problem*, Computers and Operations Research **35** (2008), pp. 944–959.

[10] Fekete, S. P. and J. Schepers, *On more-dimensional packing i: Modeling*, Technical report, University of Köln, Germany (1997).

[11] Fekete, S. P. and J. Schepers, *On more-dimensional packing iii: Exact algorithms*, Technical report, University of Köln, Germany (1997).

[12] Fekete, S. P., J. Schepers and J. van der Veen, *An exact algorithm for higher-dimensional orthogonal packing*, Operations Research **55** (2007), pp. 569–587.

[13] Korte, N. and R. H. Möhring, *An incremental linear-time algorithm for recognizing interval graphs*, JOC **18** (1989), pp. 68–81.