# A SAT Encoding for Multi-dimensional Packing Problems

Stéphane Grandcolas and Cédric Pinto*

LSIS - UMR CNRS 6168,
Avenue Escadrille Normandie-Niemen,
13397 Marseille Cedex 20, France
{stephane.grandcolas,cedric.pinto}@lsis.org

**Abstract.** The Orthogonal Packing Problem (OPP) consists in determining if a set of items can be packed into a given container. This decision problem is NP-complete. Fekete et al. modelled the problem in which the overlaps between the objects in each dimension are represented by interval graphs. In this paper we propose a SAT encoding of Fekete et al. characterization. Some results are presented, and the efficiency of this approach is compared with other SAT encodings.

## 1 Introduction

The multi-dimensional Orthogonal Packing Problem (OPP) consists in determining if a set of items of known sizes can be packed in a given container. Although this problem is NP-complete, efficient algorithms are crucial since they may be used to solve optimization problems like the strip packing problem, the bin-packing problem or the optimization problem with a single container.

S. P. Fekete et al. introduced a new characterization for OPP [1]. For each dimension $i$, a graph $G_i$ represents the items overlaps in the $i^{\text{th}}$ dimension. In these graphs, the vertices represent the items. The authors proved that solving the $d$-dimensional orthogonal packing problem is equivalent to finding $d$ graphs $G_1, \ldots, G_d$ such that **(P1)** each graph $G_i$ is an interval graph , **(P2)** in each graph $G_i$, any stable set is $i$-feasible, that is the sum of the sizes of its vertices is not greater than the size of the container in dimension $i$, and **(P3)** there is no edge which occurs in each of the $d$ graphs. They propose a complete search procedure [1] which consists in enumerating all possible $d$ interval graphs, choosing for each edge in each graph if it belongs to the graph or not. The condition (P3) is always satisfied, forbidding the choice for any edge which occurs in d-1 graphs in the remaining graph. Each time a graph $G_i$ is an interval graph, the $i$-feasibility of its stable sets is verified, computing its maximum weight stable set (the weights are the sizes of the items in the dimension $i$). As soon as the three conditions are satisfied the search stops and the $d$ graphs represent then a class of equivalent solutions to the packing problem. Figure 1 shows an example in two dimensions with two packings among many others corresponding to the same pair of interval graphs.

There are very few SAT approaches for packing. In 2008 T. Soh et al. proposed a SAT encoding for the strip packing problem in two dimensions (SPP) [2]. This problem

* This work is supported by Region Provence-Alpes-Cote-d'Azur and the ICIA Technologies company. We also thank P. Jegou and D. Habet for helpfull discussions.

A. Lodi, M. Milano, and P. Toth (Eds.): CPAIOR 2010, LNCS 6140, pp. 141–146, 2010.

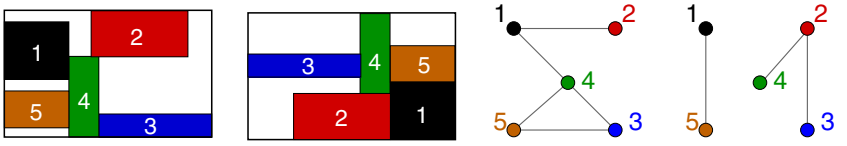


**Fig. 1.** Two packings corresponding to the same interval graphs in a two-dimensional space

consists in finding the minimal height of a fixed width container containing all the items. For that purpose they perform successive searches with different heights (selected with a dichotomy search strategy). Each time, the decision problem is encoded in a SAT formula which is solved with an external SAT solver (Minisat). In their formulation the variables represent the exact positions of the items in the container. Additional variables represent the relative positions of the items one with the others (on the left, on the right, above, under). T. Soh et al. also introduce constraints to avoid reconsidering symmetric equivalent packings. Finally the new clauses that the SAT solver Minisat generates to represent the conflicts are memorised and re-used in further searches. This is possible since successive searches are incremental SAT problems. T. Soh et al. SAT encoding involves $\mathcal{O}(W \times H \times n + n^2)$ Boolean variables for a problem with $n$ items and a container of width $W$ and height $H$.

## 2 A New SAT Encoding

We propose a new SAT encoding based on Fekete et al. characterization for the $d$-dimensional packing problem. Recall that each graph $G_i$ must be an interval graph, and that if this is the case, then there exists a linear ordering of the maximal cliques of $G_i$ such that each vertex occurs in consecutive cliques. This ordering is called *a consecutive linear ordering* and its size, the number of maximal cliques, is less then or equal to the number of items.

Basically, for each dimension $i$, Boolean variables indicate the presence of the edges in the graph $G_i$, that is the overlaps between the objects in dimension $i$. Furthermore, Boolean variables represent a linear clique decomposition of the graph $G_i$, ensuring that the graph is an interval graph if this decomposition is a consecutive linear ordering. The cliques are numbered from 1 to $n$. Then, Boolean variables indicate for each item and for each clique if the item occurs in the clique. Finally additional variables have been introduced to simplify the formulation of the constraints. The variables used in our formulation are defined as follows (note that some of these variables are not necessary in the basic formalisation of the packing problem):

$e^i_{x,y}$ : **true** if the edge $\{x, y\}$ is in $G_i$,
$c^i_{x,a}$ : **true** if item $x$ is in clique $a$,
$p^i_{x,y,a}$ : **true** if items $x$ and $y$ both occur in clique $a$,
$u^i_a$ : **true** if clique $a$ is not empty,

The stable set feasability of the graph $G_i$ is verified with clauses that forbid the unfeasible stable sets. The set of all the unfeasible stable sets in dimension $i$ is denoted $S^i$. Then the packing problem is encoded by the following formulas:

1. **[All objects are packed]**
$$x \in O, 1 \leq i \leq d,$$
$$c^i_{x,1} \vee \ldots \vee c^i_{x,n}$$
2. **[Consecutive linear ordering]**
$$x \in O, 1 \leq i \leq d, 1 \leq a < b - 1 < n,$$
$$(c^i_{x,a} \wedge c^i_{x,b}) \Rightarrow c^i_{x,a+1}$$
3. **[No-overlap Constraint]**
$$x, y \in O,$$
$$\neg e^1_{x,y} \vee \ldots \vee \neg e^d_{x,y}$$
4. **[Stable set feasibility]**
$$1 \leq i \leq d, N \in S^i,$$
$$\bigvee_{x,\ y\ \in N} e^i_{x,y}$$
5. **[No empty cliques]**
$$1 \leq i \leq d, 1 \leq a \leq n,$$
$$(\neg c^i_{1,a} \wedge \ldots \wedge \neg c^i_{n,a}) \Rightarrow (\neg c^i_{1,a+1} \wedge \ldots \wedge \neg c^i_{n,a+1})$$
6. **[Correlations between the variables]**
$$x, y \in O, 1 \leq a \leq n, 1 \leq i \leq d,$$
$$p^i_{x,y,a} \Leftrightarrow (c^i_{x,a} \wedge c^i_{y,a}) \text{ and } (p^i_{x,y,1} \vee \ldots \vee p^i_{x,y,k}) \Leftrightarrow e^i_{x,y}$$

The formulas $(1)$ force each item to occur in at least one clique, while the formulas $(2)$ force each item to occur in consecutive cliques (Fekete et al. property $P1$: the graphs are interval graphs). The formulas $(3)$ state that no two objects may intersect in all the dimensions (Fekete et al. property $P3$). The stable set feasability is enforced by the formulas $(4)$: for each unfeasible stable set $N \in S^i$ in the dimension $i$, a clause ensures that at least two items of the stable set intersect each other. In fact only the minimal unfeasible stable sets are considered. For example, if two items $x$ and $y$ are too large to be packed side by side in the $i^{\text{th}}$ dimension, then $\{x, y\}$ is a stable set of $S^i$ and the unit clause $e^i_{x,y}$ is generated. Then the SAT solver will immediately assign to the variable $e^i_{x,y}$ the value true and propagate it. The formulas $(5)$ forbid empty cliques. Finally the formulas $(6)$ establish the relations between the Boolean variables.

The following constraints are not necessary but they may help during the search:

7. **[Consective linear ordering (bis)]**
$$x \in O, 1 \leq a \leq n, 1 \leq i \leq d,$$
$$(c^i_{x,a} \wedge \neg c^i_{x,a+1}) \Rightarrow (\neg c^i_{x,a+2} \wedge \ldots \wedge \neg c^i_{x,n})$$
$$(c^i_{x,a} \wedge \neg c^i_{x,a-1}) \Rightarrow (\neg c^i_{x,a-2} \wedge \ldots \wedge \neg c^i_{x,1})$$
8. **[Maximal cliques]**
$$1 \leq a \leq n, 1 \leq i \leq d,$$
$$u^i_a \Leftrightarrow (c^i_{1,a} \vee \ldots \vee c^i_{n,a})$$
$$(u^i_a \wedge u^i_{a+1}) \Rightarrow ((c^i_{1,a} \wedge \neg c^i_{1,a+1}) \vee \ldots \vee (c^i_{n,a} \wedge \neg c^i_{n,a+1}))$$
$$(u^i_a \wedge u^i_{a+1}) \Rightarrow ((\neg c^i_{1,a} \wedge c^i_{1,a+1}) \vee \ldots \vee (\neg c^i_{n,a} \wedge c^i_{n,a+1}))$$

9. **[Identical items ordering]**

$$x, y \in O, x \equiv y \text{ and } x \prec y, 1 \leq a < n, a < b \leq n$$
$$(c^{\delta}_{y,a} \wedge c^{\delta}_{x,b}) \Rightarrow c^{\delta}_{x,a}$$

The formulas $(7)$ propagates the consecutive cliques ordering property, the formulas $(8)$ forbid cliques which are not maximal, and the formulas $(9)$ force identical objects to respect a given *a priori* ordering in only one dimension $\delta$, so as to avoid the generation of equivalent permutations of these objects. This SAT encoding involves $\mathcal{O}(n^3)$ and $\mathcal{O}(n^4+2^n)$ clauses. However, since only the minimal unfeasible stable sets are encoded, in the general case there are much less than $2^n$ clauses of type (4).

## 3 Experimental Results

### 3.1 Orthogonal Packing Problem

The problem consists to determine if a given set of items may be packed into a given container. We have compared our approach with that Fekete et al. on a selection of two-dimensional problems, using as reference the results published by Clautiaux et al. [3]. Table 1 shows the characteristics of the instances, the results of Fekete et al. (**FS**), and the results of our approach with two modelisations: the modelisation **M1** corresponds to the formulas from $(1)$ to $(6)$ and $(9)$, while the modelisation **M2** contains, furthermore, the facultative formulas $(7)$ and $(8)$. All of our experimentations were run on Pentium IV 3.2 GHz processors and 1 GB of RAM, using Minisat 2.0.

**Table 1.** Comparison with Fekete et al

| Instance | | | | FS | M1 | | | M2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Name | Space | Fais. | $n$ | Time (s) | Time (s) | #var. | #claus. | Time (s) | #var. | #claus. |
| E02F17 | 02 | $F$ | 17 | 7 | **4.95** | 5474 | 26167 | 13.9 | 6660 | 37243 |
| E02F20 | 02 | $F$ | 20 | - | 5.46 | 8720 | 55707 | **1.69** | 10416 | 73419 |
| E02F22 | 02 | $F$ | 22 | 167 | **7.62** | 11594 | 105910 | 21.7 | 13570 | 129266 |
| E03N16 | 03 | $N$ | 16 | **2** | 39.9 | 4592 | 20955 | 47.3 | 5644 | 30259 |
| E03N17 | 03 | $N$ | 17 | **0** | 4.44 | 5474 | 27401 | 9.32 | 6660 | 38477 |
| E04F17 | 04 | $F$ | 17 | 13 | **0.64** | 5474 | 26779 | 1.35 | 6660 | 37855 |
| E04F19 | 04 | $F$ | 19 | 560 | 3.17 | 7562 | 46257 | **1.43** | 9040 | 61525 |
| E04F20 | 04 | $F$ | 20 | 22 | 5.72 | 8780 | 59857 | **2.22** | 10416 | 77569 |
| E04N18 | 04 | $N$ | 18 | **10** | 161 | 6462 | 32844 | 87.7 | 7790 | 45904 |
| E05F20 | 05 | $F$ | 20 | 491 | 6.28 | 8780 | 59710 | **0.96** | 10416 | 77422 |
| Average | | | | $> 217$ | 23.9 | 7291 | 46159 | **18.8** | 8727 | 60894 |

Our approach outperforms FS on satisfiable instances, and even the instance E02F20 is not solved by Fekete et al. within the timeout (15 minutes). On unsatisfiable instances they have better performances, probably because they compute very relevant bounds (see DFF in [4]) which help them to detect dead ends during the search very early.

### 3.2 Strip Packing Problem

We have also compared our approach with Soh and al. on two-dimensional strip packing problems of the OR-Library available at `http://www.or.deis.unibo.it/`

**Table 2.** Results for OR-Library instances

| Instance | | | | Soh et | M1 | | | | M2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $n$ | Width | $LB$ | al. | Height | #var. | #claus. | Time | Height | #var. | #claus. | Time |
| HT01 | 16 | 20 | 20 | **20** | **20** | 4592 | 22963 | 13.3 | **20** | 5644 | 32267 | 19.4 |
| HT02 | 17 | 20 | 20 | **20** | **20** | 5474 | 28669 | 744 | **20** | 6660 | 39745 | 444 |
| HT03 | 16 | 20 | 20 | **20** | **20** | 4592 | 24222 | 18.5 | **20** | 5644 | 33526 | 25.5 |
| HT04 | 25 | 40 | 15 | **15** | 16 | 16850 | 271500 | 1206 | 19 | 19396 | 305392 | 521 |
| HT05 | 25 | 40 | 15 | **15** | 16 | 16850 | 337395 | 438 | 16 | 19396 | 372287 | 536 |
| HT06 | 25 | 40 | 15 | **15** | 16 | 16850 | 494500 | 146 | 16 | 19396 | 528392 | 295 |
| CGCUT01 | 16 | 10 | 23 | **23** | **23** | 4592 | 26745 | 5.89 | **23** | 5644 | 36049 | 9.71 |
| CGCUT02 | 23 | 70 | 63 | 65 | 66 | 13202 | 115110 | 1043 | 70 | 15360 | 188222 | 1802 |
| GCUT01 | 10 | 250 | 1016 | **1016** | **1016** | 1190 | 4785 | 0.11 | **1016** | 1606 | 7237 | 0.04 |
| GCUT02 | 23 | 250 | 1133 | 1196 | 1259 | 8780 | 105810 | 37.3 | 1196 | 10416 | 123522 | 1241 |
| NGCUT01 | 10 | 10 | 23 | **23** | **23** | 1190 | 5132 | 0.23 | **23** | 1606 | 7584 | 0.09 |
| NGCUT02 | 17 | 10 | 30 | **30** | **30** | 5474 | 29662 | 1.6 | **30** | 6660 | 40738 | 2.74 |
| NGCUT03 | 21 | 10 | 28 | **28** | **28** | 10122 | 108138 | 273 | **28** | 11924 | 128542 | 580 |
| NGCUT04 | 7 | 10 | 20 | **20** | **20** | 434 | 1661 | 0.01 | **20** | 640 | 2577 | 0.01 |
| NGCUT05 | 14 | 10 | 36 | **36** | **36** | 3122 | 15558 | 6.01 | **36** | 3930 | 21906 | 4.44 |
| NGCUT06 | 15 | 10 | 31 | **31** | **31** | 3810 | 18629 | 1.92 | **31** | 4736 | 26361 | 2.91 |
| NGCUT07 | 8 | 20 | 20 | **20** | **20** | 632 | 2535 | 0 | **20** | 900 | 3855 | 0 |
| NGCUT08 | 13 | 20 | 33 | **33** | **33** | 2522 | 11870 | 2.74 | **33** | 3220 | 17010 | 9.73 |
| NGCUT09 | 18 | 20 | 49 | **50** | 50 | 6462 | 33765 | 391 | 50 | 7790 | 46825 | 53.3 |
| NGCUT10 | 13 | 30 | 80 | **80** | **80** | 2522 | 11790 | 0.75 | **80** | 3220 | 16930 | 0.39 |
| NGCUT11 | 15 | 30 | 50 | **52** | **52** | 3810 | 18507 | 19.7 | **52** | 4736 | 26239 | 25.9 |
| NGCUT12 | 22 | 30 | 79 | **87** | **87** | 11594 | 173575 | 886 | **87** | 13570 | 196931 | 24.5 |

`research.html`. The problem is to determine the minimal height of a fixed width container which may contain a given set of items. As Soh et al. we perform a sort of dichotomy search starting with a lower bound given by Martello and Vigo [5] and an upper bound which is calculated using a greedy algorithm. In table 2 we have reported the sizes of the encodings (numbers of variables and clauses) and the minimal height which was found within the timeout of 3600 seconds. Optimal heights are in bold (this occurs when the minimal height is equal to the lower bound or when the solver proves that there is no solution with a smaller height). Instances in which the number of items is large have been discarded, since the number of unfeasible stable sets becomes too important and so the number of corresponding clauses. Note that Soh and al. used also the solver Minisat. For 16 instances among 22 our system discovers the optimal height. Furthermore, among these 16 instances, 14 are solved in less than 30 seconds with one of our two modelisations. The ability of Soh and al. solver to reuse the conflict clauses that Minisat generates during the search is a real advantage since many unsuccessfull searches are then avoided.

## 4 Conclusions and Future Works

We have proposed a SAT encoding which outperforms significantly Fekete et al. method on satisfiable instances. Moreover, we have experimented this encoding on strip-packing problems. In future work we will try to integrate the DFF computation to improve the search on unsolvable problems. We will also try to characterize the situations in which the conflicts clauses which are generated by the SAT solver, may be re-used. This occurs in particular when successive calls to the solver are performed, for example when searching the minimal height in strip-packing problems.

## References

1. Fekete, S.P., Schepers, J., van der Veen, J.: An exact algorithm for higher-dimensional orthogonal packing. Operations Research 55(3), 569–587 (2007)
2. Soh, T., Inoue, K., Tamura, N., Banbara, M., Nabeshima, H.: A SAT-based Method for Solving the Two-dimensional Strip Packing Problem. In: Proceedings of the 15th RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (2008)
3. Clautiaux, F., Carlier, J., Moukrim, A.: A new exact method for the two-dimensional orthogonal packing problem. European Journal of Operational Research 183(3), 1196–1211 (2007)
4. Fekete, S.P., Schepers, J.: A general framework for bounds for higher-dimensional orthogonal packing problems. Mathematical Methods of Operations Research 60(2), 311–329 (2004)
5. Martello, S., Monaci, M., Vigo, D.: An exact approach to the strip-packing problem. Journal on Computing 15(3), 310–319 (2003)