Packing a Trunk

Friedrich Eisenbrand[1], Stefan Funke[1], Joachim Reichel[1], and Elmar Schömer[2]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany*
{eisen,funke,reichel}@mpi-sb.mpg.de
[2] Universität Mainz, Department of Computer Science, Germany
schoemer@informatik.uni-mainz.de

Abstract. We report on a project with a German car manufacturer. The task is to compute (approximate) solutions to a specific large-scale packing problem. Given a polyhedral model of a car trunk, the aim is to pack as many identical boxes of size $4 \times 2 \times 1$ units as possible into the interior of the trunk. This measure is important for car manufacturers, because it is a standard in the European Union.
First, we prove that a natural formal variant of this problem is NP-complete. Further, we use a combination of integer linear programming techniques and heuristics that exploit the geometric structure to attack this problem. Our experiments show that for all considered instances, we can get very close to the optimal solution in reasonable time.

1 Introduction

Geometric packing problems are fundamental tasks in the field of Computational Geometry and Discrete Optimization. The problem we are considering in this paper is of the following type:

Problem 1. Given a polyhedral domain $P \subseteq \mathbb{R}^3$, which is homeomorphic to a ball, place as many boxes of size $4 \times 2 \times 1$ into P such that no two of them intersect.

We were approached with this problem by a car manufacturer whose problem was to measure the volume of a trunk according to a European standard (DIN 70020). The intention of this standard is that the continuous volume of a trunk does not reflect the actual storage capacity, since the baggage, which has to be stored, is usually discrete. The European standard asks for the number of 200mm $\times$ 100mm $\times$ 50mm = 1 liter boxes, which can be packed into the trunk. Up till now, this problem is solved manually with a lot of effort.

Contributions

We show that Problem 1 is NP-complete by a reduction to 3-SAT. Further, we attack this problem on the basis of an integer linear programming formulation.

* This work was partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

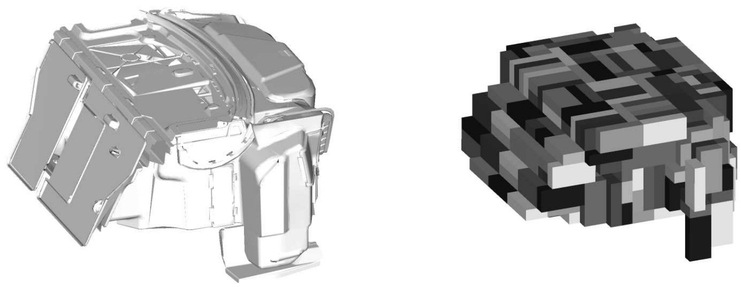

Fig. 1. CAD model of a trunk and a possible packing of boxes

It turns out that the pure ILP approach does not work, even with the use of problem-specific cutting planes in a branch-and-cut framework. We therefore design and evaluate several heuristics based on the LP-relaxation and the geometric structure of the problem. The combination of the exact ILP-approach and the herein proposed heuristics yield nearly optimal solutions in reasonable time.

Related work

Various versions of packing problems have been shown to be NP-complete [1]. We derive our complexity result from an NP-complete packing variant inspected by Fowler, Paterson and Tanimoto [2]. Here the task is to pack unit squares in the plane. In their work, they do not consider rectangular objects that are not squares and the region which has to be packed is not homeomorphic to a disk.

Other theoretical and practical results in the area of industrial packing problems suggest that allowing arbitrary placements and orientations of the objects even within a two-dimensional domain is only viable for extremely small problem instances. For example, Daniels and Milenkovic consider in [3, 4] the problem of minimizing cloth utilization when cutting out a small number of pieces from a roll of stock material. If arbitrary placements and orientations are allowed only very small problem instances (≤ 10 objects) can be handled. For larger problem instances they discretize the space of possible placements and use heuristics to obtain solutions for up to 100 objects.

A survey of the application of generic optimization techniques like simulated annealing, genetic algorithms, gradient methods, etc. to our type of packing problems can be found in [5]. Aardal and Verweij consider in [6] the problem of labelling points on a map with pairwise disjoint rectangles such that the corners of the rectangles always touch the point they are labelling. Similar to our work, this discretization of the possible placements reduces the problem to finding a maximum stable set in the intersection/conflict graph (which for this application is much smaller than in our case, though).

2 Continuous Box Packing is NP-complete

In this section, we prove that Problem 1 is NP-complete. In a first step, we show that a two-dimensional discrete variant of this problem is NP-complete.

Definition 1 ($m \times n$-Rectangle-Packing). *Given integers $k, m, n \in \mathbb{N}, m \geq n$ and sets $H \subseteq \mathbb{Z}^2$, $V \subseteq \mathbb{Z}^2$, decide whether it is possible to pack at least k axis-aligned boxes of size $m \times n$ in such a way that the lower left corner of a horizontal (vertical) box coincides with a point in H respectively V.*

This problem is trivial for $m = n = 1$. For $m = 2, n = 1$, the problem can be formulated as set packing problem with sets of size 2 and solved in polynomial time by matching techniques [1].

Proposition 1. 3×3-Rectangle-Packing *and* 8×4-Rectangle-Packing *are NP-complete.*

Proof. The case $m = n = 3$ has been shown by Fowler et al. [2] using a reduction of 3-SAT to this problem. We use the same technique here for $m = 8, n = 4$ and refer to their article for the details.

Given a formula in conjunctive normal form (CNF) with three literals per clause, Fowler et al. construct a planar graph as follows. For each variable x_n there is a cycle of even length. Such a cycle has two stable sets of maximal cardinality which correspond to the two possible assignments of x_n. Moreover, paths of two cycles can cross each other at so called *crossover regions*, which have the special property that for each maximum stable set both paths do not influence each other. For each clause exists a *clause region*, that increases the value of a maximum stable set iff the clause is satisfied. The number k can be easily deduced from the number of nodes, crossover and clause regions. The proof is completed by explaining how to compute the sets H and V such that the constructed graph is the intersection graph of the packing problem. Thus the formula is satisfiable iff there exists a packing of cardinality $\geq k$.

For the case $m = 8, n = 4$, we need to explain how to construct the crossover and clause region. The crossover region for two cycle paths are realized as shown in Fig. 2. Note that the rectangle in the center has size 9×5 and allows four different placements of a 8×4 rectangle. Clause regions are constructed as shown in Fig. 3. Both constructions maintain their special properties as in the case of 3×3 squares and the remainder of the proof is identical. ◻

The decision variant of Problem 1 is defined as follows:

Definition 2 (Continuous-Box-Packing). *Given a polyhedral domain $P \subseteq \mathbb{R}^3$, which is homeomorphic to a ball, and $k \in N$, decide whether it is possible to pack at least k boxes of size $4 \times 2 \times 1$ into P such that no two of them intersect.*

Theorem 1. Continuous-Box-Packing *is NP-complete.*

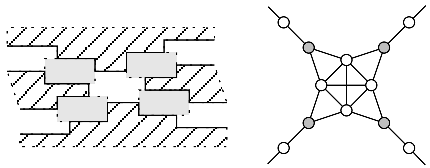

Fig. 2. Crossover region and corresponding intersection graph

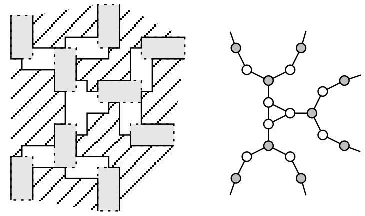

Fig. 3. Clause region and corresponding intersection graph

Proof. We reduce the problem to 8×4-Rectangle-Packing. Let (k, H, V) denote an instance of 8×4-Rectangle-Packing. Intuitively our approach works as follows: We extrude the shape induced by the sets H and V into the third dimension with z-coordinates ranging from 0 to 1. On the bottom of this construction we glue a box of hight $\frac{1}{2}$. The size of the box is chosen such that the construction is homeomorphic to a ball. More formally, $P \subseteq \mathbb{R}^3$ is constructed as follows:

$$P_H := \bigcup_{(x,y)\in H} \left[\frac{1}{2}x, \frac{1}{2}x + 4\right] \times \left[\frac{1}{2}y, \frac{1}{2}y + 2\right] \times [0, 1] \ ,$$

$$P_V := \bigcup_{(x,y)\in V} \left[\frac{1}{2}x, \frac{1}{2}x + 2\right] \times \left[\frac{1}{2}y, \frac{1}{2}y + 4\right] \times [0, 1] \ ,$$

$$P := \ P_H \ \cup \ P_V \ \cup \ X \times Y \times \left[-\frac{1}{2}, 0\right] \ ,$$

where X and Y denote the projection of $P_H \cup P_V$ onto the first respectively second coordinate. This construction can be carried out in polynomial time. It is clear that a projection of a maximal packing to the first two coordinates corresponds to a maximal packing of the two-dimensional problem of the same value and vice versa. $\square$

3 From the CAD Model to the Maximum Stable Set Problem

The data we obtain from our industry partner is a CAD model of the trunk to be packed. Since the manual packings used so far had almost all boxes axis aligned to some coordinate system, we decided to discretize the problem in the following way. We first **Discretize the Space** using a three-dimensional cubic grid. Given the box extensions of 200 mm, 100 mm and 50 mm, a grid width of 50 mm was an obvious choice. In order to improve the approximation of the trunk by this grid, we also work with a refined grid of edge length 25 mm. Even smaller grid widths did not improve the results substantially and for larger CAD models the number of cubes became too big. In the following, numbers depending on the grid granularity refer to a grid of edge length 50 mm and numbers for the refined grid of edge length 25 mm are added in parentheses.

The alignment of the coordinate axes is done such that the number of cubes which are completely contained in the interior of the trunk model is maximized. In practice, the best cubic grids were always aligned with the largest almost planar boundary patch of the trunk model – which most of the time was the bottom of the trunk. For the remaining translational and one-dimensional rotational freedom we use an iterative discrete procedure to find the best placement of the grid. The result of this phase is an approximation of the trunk interior as depicted in Fig. 4.

In the next phase we use the cubic grid to **Discretize the Box Placements**. A box of dimension 200mm $\times$ 100mm $\times$ 50mm can be viewed as a box consisting of $4 \times 2 \times 1$ ($8 \times 4 \times 2$) cubes of the cubic grid. We will only allow placements of boxes such that they are aligned with the cubic grid. So the placement of one box is defined by six parameters (x, y, z, w, h, d), where (x, y, z) denotes the position of a cube in our grid – we will call this the *anchor* of the box – and (w, h, d) denotes how far the box extends to the right, to the top and in depth. (w, h, d) can be any permutation of $\{4, 2, 1\}$ ($\{8, 4, 2\}$), so for a given anchor, there are 6 possible orientations of how to place a box at that position.

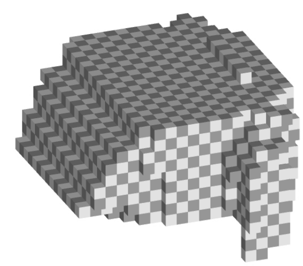

Fig. 4. Interior approximation of the trunk of Fig. 1

Our goal is now to place as many such boxes in the manner described above in our cubic grid such that each box consists only of cubes that approximate the interior of the trunk and no pair of boxes shares a cube.

It is straightforward to formalize this problem using the following construction: The *conflict graph* $G(\mathcal{G}) = (V, E)$ for a cubic grid $\mathcal{G}$ is constructed as follows. There is a node $v_{x,y,z,w,h,d} \in V$ iff the box placed at anchor (x, y, z) with extensions (w, h, d) consists only of cubes located inside the trunk. Two nodes v and w are adjacent iff the boxes associated with v and w intersect.

A *stable* or *independent* set $S \subseteq V$ of a graph $G = (V, E)$ is a subset of the nodes of G which are pairwise nonadjacent. The *stable set problem* is the problem of finding a stable set of a graph G with maximum cardinality. It is NP-hard [1].

There is a one-to-one relationship between the stable sets in $G(\mathcal{G})$ and valid box packings in $\mathcal{G}$, in particular every stable set in $G(\mathcal{G})$ has a corresponding valid box packing in $\mathcal{G}$ of same size and vice versa. We use this one-to-one relationship to reduce the maximum box packing problem to a maximum stable set problem on a graph:

Lemma 1. *The maximum box packing problem for a grid $\mathcal{G}$ can be reduced to a maximum stable set problem in the corresponding conflict graph $G(\mathcal{G})$.*

To give an idea about the sizes of the conflict graphs we are dealing with, we show in Table 1 the sizes of the conflict graphs for our (rather small) trunk model M1 and grid widths 50 mm and 25 mm. We will use the 50 mm discretization of this model as a running example throughout the presentation of all our algorithms in the next section.

Table 1. Grid and Conflict Graph sizes for trunk model M1

grid granularity [mm]	# interior cubes	# nodes in $G(\mathcal{G})$	# edges in $G(\mathcal{G})$
50	2210	8787	649007
25	19651	68548	62736126

4 Solving the stable-set problem

4.1 A branch-and-cut algorithm

In the previous section, we modeled our packing problem as a maximum stable-set problem for the conflict-graph $G(\mathcal{G}) = (V, E)$ for a given grid $\mathcal{G}$. In this Section we describe how we attack this stable-set problem with a branch-and-cut algorithm.

The stable-set problem has the following well known integer programming formulation, see, e.g. [7]:

$$\begin{aligned} \max \sum_{v \in V} x_v & \qquad (1) \\ \{u, v\} \in E : x_u + x_v &\leq 1 \\ u \in V : \quad x_u \quad &\in \{0, 1\} \ . \end{aligned}$$

It is easy to see that the characteristic vectors of stable sets of G are exactly the solutions to this constraint system.

Standard ILP solvers try to solve this problem using techniques like branch-and-bound. These techniques depend heavily on the quality of the *LP relaxation*. Therefore, it is beneficial to have a relaxation which is strong. We pursue this idea via incorporating clique inequalities and (lifted) odd-hole inequalities.

Clique inequalities A *clique* C of G is a subset of the nodes $C \subseteq V$, such that every two nodes in C are connected. If S is a stable set and C is a clique, then there can be at most one element of S which also belongs to C. This observation implies the constraints

$$\sum_{v \in C} x_v \leq 1 \text{ for each } C \in \mathcal{C} \ , \qquad (2)$$

where $\mathcal{C}$ is the set of cliques of G.

If C is a maximal clique, then the corresponding clique inequality (2) defines a facet of the convex hull of the characteristic vectors χ^S of stable sets S of G, see [8]. Thus the clique inequalities are strong in the sense that they cannot be implied by other valid inequalities. The number of maximal cliques can be exponential and furthermore, the *separation problem* for the clique inequalities is NP-hard for general graphs [9]. However, in our application the number of cliques is polynomial and the maximum cliques can be enumerated in polynomial time. This result is established with the following lemma. A proof is straightforward.

Lemma 2. *Every maximal clique in $G(\mathcal{G})$ corresponds to the box placements in $\mathcal{G}$ which overlap one particular cube.*

Therefore we can strengthen the formulation (1) by replacing the edge constraints with the clique constraints (2) and obtain the polynomial *clique formulation.*

Odd hole inequalities An *odd hole* [10] H of G is a cordless cycle of G with an odd number of nodes. If S is a stable set of G, then there can be at most $\lfloor |H|/2 \rfloor$ elements of S belonging to H. This implies the constraints

$$\sum_{v \in H} x_v \leq \lfloor |H|/2 \rfloor \text{ for all } H \in \mathcal{H} \ , \tag{3}$$

where $\mathcal{H}$ denotes the set of odd holes of G. These inequalities can be strengthened with a *sequential lifting* process, suggested in [8, 11], see also [12]. We apply the algorithm of Gerards and Schrijver [13] to identify nearly violated odd hole inequalities and strengthen them using different lifting sequences.

For our running example M1 / 50mm, the LP relaxation yields an upper bound of 268 liters. Running our branch-and-cut approach for 24 hours and taking the best result found, we obtained a packing of 266 liters.

4.2 Heuristics

Unfortunately, the above described branch-and-cut algorithm works only for small packing instances and not for trunks of real-world size. On the other hand, the ILP-approach is an exact algorithm for the discretized problem that we wish to solve. In the following we propose several heuristics for our problem that can be combined with our exact ILP-approach. Depending on the employed heuristic we obtain trade-offs between running time and solution quality.

Partitioned ILP Formulations

This approach partitions the packing problem into independent sub-problems, which are exactly solved with branch-and-cut individually and thereafter combined. We partition a given grid by axis-parallel planes into smaller sections. Tests have shown that one should choose the sizes of the sections ranging from 50 to 100 liters.

But the cutting of the grid into smaller sections leads to waste and obtained solutions can be further improved by local optimization across section boundaries. By moving some of the packed boxes it is possible to bring several uncovered cubes into proximity, such that one more box can be packed. This inspired the following modification.

We slightly change the objective function of (1) such that some packings of a given cardinality are preferred. The old objective function is replaced by $\sum_{v \in V} c_v x_v$, i.e. we solve a *weighted* stable set problem and the coefficients $c_v \in \mathbb{R}$ are computed as follows. Assume the box corresponding to node v is anchored at (x, y, z) and contained in the section $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$. The coefficient c_v is then computed as

$$c_v = 1 + \frac{1}{u} \cdot \frac{x_{max} - x + y_{max} - y + z_{max} - z}{x_{max} - x_{min} + y_{max} - y_{min} + z_{max} - z_{min}} \ , \tag{4}$$

where u is an upper bound for the optimal value of (1) for this section. The new objective function still aims for packings with largest cardinality, but among packings of the same cardinality those with boxes anchored as near as possible to $(x_{min}, y_{min}, z_{min})$ are preferred. Thus uncovered cubes tend to appear near the $x = x_{max}$, $y = y_{max}$ and $z = z_{max}$ boundaries of the section and can be reused when processing the adjacent sections.

Using this approach, we achieved a packing of 267 boxes with 24 hours runtime. Although the quality of the solution has been improved by a small amount, it takes too long to achieve good results.

A greedy Algorithm

The most obvious idea for a heuristic for the stable set problem in a graph is to use a greedy approach. The greedy algorithm selects a vertex with smallest degree and adds it to the stable set S determined so far, then removes this vertex and all its neighbors and repeats.

The algorithm tends to place boxes first close to the boundary and then growing to the inside until the trunk is filled. This is due to the fact that placements close to the boundary 'prohibit' fewer other placements and therefore their degree in the conflict graph is rather low.

There is a bit of ambiguity in the formulation of the Greedy algorithm. If there are several vertices with minimum degree, we can choose the next vertex uniformly at random. This randomized version of the greedy algorithm is repeated several times.

As one might expect, the Greedy algorithm is very fast, but outputs a result of rather low quality. We achieved a solution of 259 liters in less than a minute. The maximum of ten runs of the randomized version was 262 liters.

Geometry-Guided first-level Heuristics

Looking at the model of the trunk, one observes that it is quite easy to tightly pack some boxes in the center of the trunk, whereas difficulties arise when approaching the irregular shape of the boundary. The following two heuristics exploit this fact by filling the center of the trunk with a solid block of boxes. This approach significantly decreases the problem complexity.

Easyfill This algorithm strongly simplifies the problem by restricting the set of allowed placements for boxes. Supposing the first box is packed at (x, y, z, w, h, d), we solely consider boxes with the same orientation anchored at $(x + \mathbb{Z}w, y + \mathbb{Z}h, z + \mathbb{Z}d)$. This leads to a tight packing in the interior of the trunk and the remaining cubes near to the boundary can be packed by one of the other algorithms.

For each of the 6 orientations, there are 8 (64) possibilities to align the first box on the grid. The quality of the results heavily depends on the placement of the first box. Thus we repeat the procedure with all different placements of the first box.

As it turns out, if one exactly uses this algorithm, the remaining space is not sufficient to place many additional boxes, but a lot of cubes are left uncovered. To overcome this problem, we use the following approach. In the first phase we peel off some layers of the cubes representing the interior of the trunk, and then run the Easyfill algorithm. In the second phase we re-attach the peeled-off layers again and fill the remaining part using some other algorithm.

By using Easyfill we were able to improve the results obtained so far. In combination with the Greedy algorithm, we achieved a solution of 263 liters in 1 minute. A better solution of 267 boxes was achieved in combination with the ILP algorithm. Here we also had to terminate the branch-and-cut phase after about 30 minutes for each combination of orientation and alignment of the first box to limit the total running time to 24 hours.

Matching Another interesting idea to get a compact packing of a large part of the trunk is to cluster two boxes to a larger box consisting of $4 \times 2 \times 2$ $(8 \times 4 \times 4)$ cubes and then interpret these boxes as $2 \times 1 \times 1$ cubes on a coarser grid of side length 100 mm (50 mm). As we have seen in Section 2, this special packing problem can be solved in polynomial time using a maximum cardinality matching.

Similar to the Easyfill algorithm, there are 8 (64) possibilities to align the coarse grid with the original grid. Likewise, there is little freedom for packing the remaining cubes and we use the same approach as in the case of the Easyfill algorithm.

The results for the Matching approach combined with Greedy algorithms are comparable to the Easyfill approach. So we obtained a volume of 263 liters with a slightly better running time. In combination with the ILP approach we get a slightly worse result of 265 liters.

LP Rounding

As solving the ILP to optimality is pretty hard, one might wonder how to make use of an optimal solution to the LP relaxation – which can be obtained in reasonable time – to design a heuristic. One way is to solve the LP and round the possibly fractional values of the optimal solution to 0/1-values, of course obeying the stable set constraints. This heuristic is implemented in the ILP-solver, but is not very effective. So we came up with the following iterative procedure:

1. solve the clique LP for G to optimality
2. let B be the box placements corresponding to the 5 % largest LP values
3. use greedily as many placements from B as possible, remove their corresponding vertices and neighbors from G and goto 1

This approach took 45 minutes to compute a solution of 268 boxes. This is the value of the LP relaxation and thus optimal.

5 Experimental Evaluation

In this section, we present experimental results showing the performance of the algorithms. We present results for three models, named M1, M2 and M3 in the following, with grids of granularity of 50 mm and 25 mm.

Table 2 shows some characteristics of both models. Model M2 is about 40% larger than model M1 and model M3 about three times larger than M2. Note that refining the grid granularity quickly increases the size of the conflict graph and enlarges the grid volume, whereas the upper bound obtained by the LP relaxation does not grow by the same factor.

Table 2. Some characteristics of the used models

model grid granularity [mm]	M1 50	M1 25	M2 50	M2 25	M3 50
# nodes in $G(\mathcal{G})$	8787	68548	12857	95380	44183
# edges in $G(\mathcal{G})$	649007	62736126	974037	88697449	3687394
grid volume [l]	276	307	396	429	1214
upper bound (LP relaxation) [l]	268	281	389	398	1202
best solution [l]	268	271	384	379	1184

For model M1 our industrial partner provided a manually achieved solution of 272 liters (which applies to the original trunk model only, not the discretized grid), whereas our best solution has a value of 271 liters.

In Table 3 we present results for the Greedy, Randomized Greedy, LP Rounding and ILP algorithm – standalone as well as combined with the Matching and Easyfill algorithm. The table is completed by the data for the Partitioned ILP algorithm. Each run was stopped after at last 24 hours and in this case, the so far best result is reported.

Table 3. Computed trunk volumes (in liters) and running-times (in minutes). For the results marked with an asterisk (*), we stopped the computation after 24 hours and took the best result found so far.

model	M1				M2				M3	
grid granularity [mm]	50		25		50		25		50	
	vol.	time	vol.	time	vol.	time	vol.	time	vol.	time
Greedy	259	1	262	2	373	1	364	2	1148	1
Easyfill + Greedy	263	1	269	61	378	1	376	79	1169	1
Matching + Greedy	263	1	268	5	375	1	373	7	1167	1
Randomized Greedy	262	1	265	19	377	1	365	27	1147	2
Easyfill + Rand. Gr.	264	5	271	807	381	5	377	1038	1171	26
Matching + Rand. Gr.	263	1	268	67	377	1	373	83	1165	4
LP Rounding	268	45	–	–	384	427	–	–	1184	189
Easyfill + LP Round.	267	29	269	$24h^*$	384	48	376	$24h^*$	1184	95
Matching + LP Round.	265	35	267	453	383	2	379	792	1178	8
ILP	266	$24h^*$	–	–	384	$24h^*$	–	–	1136	$24h^*$
Easyfill + ILP	267	$24h^*$	269	$24h^*$	383	$24h^*$	379	$24h^*$	1180	$24h^*$
Matching + ILP	265	$24h^*$	270	$24h^*$	383	$24h^*$	379	$24h^*$	1176	$24h^*$
Partitioned ILP	267	$24h^*$	260	$24h^*$	384	$24h^*$	378	$24h^*$	1175	$24h^*$

All algorithms using LP- or ILP-based techniques were run on a SunFire 15000 with 900 MHz SPARC III+ CPUs, using the operating system SunOS 5.9. CPLEX 8.0 was used as (I)LP-solver. All other algorithms were run on a Dual Xeon 1.7 GHz under Linux 2.4.18. Our implementation is single-threaded and thus does not make use of multiple CPUs.

For the Matching and Easyfill algorithms, one layer of cubes was peeled off for the first phase. This has turned out as a good compromise between enough freedom and not too large complexity for the second phase. For Randomized Greedy, the best results of ten runs are reported.

One observes that the randomization of the Greedy algorithm leads to better results while the runtime increases according to the number of runs. Both algorithms can be improved by applying Easyfill or Matching first. This imposes a further increase in the running time, due to the many subproblems that have to be solved.

However, all Greedy algorithms are outperformed by (I)LP-based techniques, whereas LP Rounding is significantly faster than the ILP algorithm. Combining both algorithms with Easyfill and Matching leads to worse results on a grid with granularity 50 mm, whereas it is absolutely necessary on the refined grid due to its huge complexity. The results obtained by Partitioned ILP are comparable to LP Rounding and ILP.

6 Conclusion

In this paper we have considered the problem of maximizing the number of boxes of a certain size that can be packed into a car trunk. We have shown that this problem is NP-complete. Our first approach which was based on an ILP formulation of the problem did not turn out to be very practical. Therefore we have designed several heuristics based on the ILP formulation and the geometric structure of the problem. In this way we obtained good trade-offs between running time and quality of the produced solution. In fact, at the end we could compute solutions as good or even better than the best ILP based solutions within a certain time frame.

There are still a number of interesting problems left open. In our problem instances, we could restrict to only axis-aligned placements of the boxes without sacrificing too much of the possible volume, but there might be other instances (maybe not of trunk-type) where this is a too severe restriction.

References

1. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. Freeman (1979)
2. Fowler, R.F., Paterson, M.S., Tanimoto, S.L.: Optimal packing and covering in the plane are NP-complete. Information Processing Letters **12** (1981) 133–137
3. Milenkovic, V.J.: Rotational polygon containment and minimum enclosure using only robust 2d constructions. Computational Geometry **13** (1999) 3–19
4. Daniels, K., Milenkovic, V.J.: Column-based strip packing using ordered and compliant containment. In: 1st ACM Workshop on Applied Computational Geometry (WACG). (1996) 33–38
5. Cagan, J., Shimada, K., Yin, S.: A survey of computational approaches to three-dimensional layout problems. Computer-Aided Design **34** (2002) 597–611
6. Verweij, B., Aardal, K.: An optimisation algorithm for maximum independent set with applications in map labelling. In: Algorithms—ESA '99 (Prague). Volume 1643 of Lecture Notes in Comput. Sci. Springer, Berlin (1999) 426–437
7. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Volume 2 of Algorithms and Combinatorics. Springer (1988)
8. Padberg, M.W.: On the facial structure of set packing polyhedra. Mathematical Programming **5** (1973) 199–215
9. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1** (1981) 169–197
10. Chvátal, V.: On certain polytopes associated with graphs. Journal of Combinatorial Theory Ser. B **18** (1975) 138–154
11. Wolsey, L.: Faces for a linear inequality in 0-1 variables. Mathematical Programming **8** (1975) 165–178
12. Nemhauser, G.L., Wolsey, L.A.: Integer programming. In et al., G.L.N., ed.: Optimization. Volume 1 of Handbooks in Operations Research and Management Science. Elsevier (1989) 447–527
13. Gerards, A.M.H., Schrijver, A.: Matrices with the Edmonds-Johnson property. Combinatorica **6** (1986) 365–379