Trunk packing revisited

Ernst Althaus, Tobias Baumann, Elmar Schömer, and Kai Werth

Johannes-Gutenberg-Universität Mainz
Institut für Informatik
{althaus, tba, schoemer, werth}@informatik.uni-mainz.de

Abstract. For trunk packing problems only few approximation schemes are known, mostly designed for the European standard DIN 70020 [10] with equally sized boxes [12, 13, 15, 16]. In this paper two discretized approaches for the US standard SAE J1100 [14] are presented, which make use of different box sizes. An exact branch-and-bound algorithm for weighted independent sets on graphs is given, using the special structure of the SAE standard. Another branch-and-bound packing algorithm using linear programs is presented. With these algorithms axis-oriented packings of different box sizes in an arbitrary trunk geometry can be computed efficiently.

Keywords: approximation algorithms, branch and bound algorithms, branch and cut algorithms, graph algorithms, linear programming, packing problems, weighted independent set

1 Introduction

The cooperation with a German car manufacturer has led to several efficient approximation schemes for the trunk packing problem according to the German standard DIN 70020 [8, 9, 10, 12, 15, 16]. This cooperation has been continued by exploring the baggage volume capacity according to the US standard SAE J1100 [14].

The computation of the baggage capacity has a significant influence on the car design process. According to international regulations, the baggage capacity is not equal to the continuous volume of a trunk which can be easily computed using a standard CAD program. The German standard DIN 70020 uses boxes of size $20 \times 10 \times 5$ cm, which equals 1 l per box. The American standard SAE J1100 uses a more realistic set of boxes: seven different box sizes are used, making an equivalent from 6 up to 67 litres. To create a valid[1] SAE J1100 packing, a few rules must be obeyed: The smallest boxes may not be used until none of the larger boxes fit into the trunk, and each box type has got a fixed number of instances to be used at maximum.

[1] A packing is considered valid if it contains only boxes of allowed sizes and each box size does not occur more often than allowed.

The trunk packing problem has recently been explored for the DIN 70020 case. In [9, 15], a discrete approach is followed using a discretization of the trunk volume into cubes of fixed size. Using discrete coordinates derived from these cubes as representatives for boxes, a conflict graph can be created and the INDEPENDENT SET (IS) problem can be solved on this graph. [9, 15] also provide several algorithms using some structural properties of the conflict graph and optimization techniques for the underlying grid.

Cagan and Ding [7] presented a packing algorithm for SAE J1100 using extended pattern search. Additionally to this approach one could also make use of a conflict graph. The SAE standard differs from the DIN standard in two issues: At first, the goal is to find an independent set with the maximal covered volume. This problem is known as WEIGHTED INDEPENDENT SET (WIS). Second, the maximal WIS might not be a valid solution for the SAE J1100 standard which allows only a fixed number of occurences for each box type. Table 1 shows the allowed box types and their sizes. There also exists an irregular shaped golf bag (type G) which is currently not used.

For the WIS problem itself a greedy algorithm has been explored recently [11, 17]. An approximation ratio of $\min((\bar{d}_w + 1)/2, (\delta_w + 1)/2)$ is derived for the greedy algorithm, where $\bar{d}_w$ denotes the *weighted average degree*, and δ_w is the *weighted inductiveness* of a weighted undirected graph as defined in [17]. The special structure of the conflict graph, however, causes that this approximation ratio can be achieved easily and provides rather poor values for the trunk packing problem.

Some branch-and-bound algorithms for the WIS problem on general graphs have been evaluated [19]. An approximation algorithm using local search has been developed by Berman [6]. This algorithm achieves an approximation ratio of $d/2$ in a d-claw free graph. A d-claw is an induced subgraph containing an independent set of d nodes, and a center node that is connected to all members of the independent set. A graph

box	max.	inch (″)			mm			Volume	
type	occurrences	l	w	h	l	w	h	l	ft^3
A	4	24	19	9	610	483	229	67.47	2.375
B	4	18	13	6.5	457	330	165	24.88	0.880
C	2	26	16	9	660	406	229	61.36	2.167
D	2	21	18	8.5	533	457	216	52.61	1.859
E	2	15	9	8	381	229	203	17.71	0.625
F	2	21	14	7	533	356	178	33.78	1.191
H	20	12.8	6	4.5	325	152	114	5.63	0.200

Table 1. Allowed box types for the SAE J1100 trunk packing problem.

is called d-claw free if it does not contain any d-claws. Unfortunately, the conflict graph as constructed in section 2 contains d-claws for very large d, so this approximation ratio would not provide any gain for the trunk packing problem. Furthermore, these algorithms apply to graphs with arbitrary integer weights. In the trunk packing case, the fact that only seven different weights are present can be exploited. Thus it is necessary to find other efficient algorithms for this special version of the WIS problem.

Additionally, a continuous approach was developed using Simulated Annealing techniques (see [8, 12, 16]). Using moves like *translation*, *rotation*, *creation* and *deletion*, the discretized solutions could be improved in many cases. A new promising approach for continuous packing is the simulation of the physical processes during the motion of rigid bodies. There it is possible, by pushing boxes, to create free space at a designated place and therefore to insert new boxes [5]. The idea to use a contact simulation for a packing problem arose from a programming contest about Circle Packing [21].

This paper is structured as follows: Section 2 deals with the WIS problem itself, the mapping of the container structure to a discretized model and the description of graph algorithms. In section 3 an approach without the use of a grid is presented. There, the packing problem is solved by using linear programs using a convex container structure with convex obstacles. Finally, section 4 presents some quality and runtime results achieved by these combinatorial approaches.

2 Discrete Approach

2.1 Hardness of the discrete trunk packing problem

The Trunk packing problem can be discretized in the following way: First, the rotation of the boxes can be limited to axis-oriented placements. Second, the possible placements are restricted to the cells of a grid. Clearly, these restrictions reduce the solution space and possibly eliminate the optimal solution of the original problem. Reichel [15] shows similar problems (Discrete-Box-Packing for equal boxes) to be $\mathcal{NP}$-complete and the continuous version to be $\mathcal{NP}$-hard. Similar considerations can be applied here.

2.2 Conflict Graph

Reichel presented techniques to compute a grid approximation for the container and uses a graph to describe conflicts between possible box placements on the grid [15]. This can easily be extended to the SAE case.

Now there are seven different box types and therefore up to seven nodes for each anchor cell and orientation[2].

SAE J1100 defines seven box types with 17 different side lengths. The finest grid spacing used for the DIN problem was 12.5 mm, which equals about $0.5''$. This grid size would be too fine for the SAE case since for each box type, there has to be a copy of each node. Since the conflict graph will be very large for small grid spacings, one has to restrict the minimal spacing to $1'' - 1.5''$. Although larger grid spacings lead to smaller conflict graphs, the resulting packings would be larger using small grid spacings such as $1.5''$ or $1''$.

The conflict graph $G = (V, E)$ corresponding to a grid is generated analogue to [9] as follows: For each grid cell (i, j, k), orientation o and box type t, determine whether a box of type t, situated at (i, j, k) in orientation o, would fit into the trunk (i.e. the box would cover only *inside*-cells). If so, then add node $v = (i, j, k, o, t)$ to V. Now, for each pair of nodes $u, v \in V$, check whether the two corresponding boxes would intersect, i.e. whether there is a cell which is covered by both boxes. If so, then add edge $\{u, v\}$ to E.

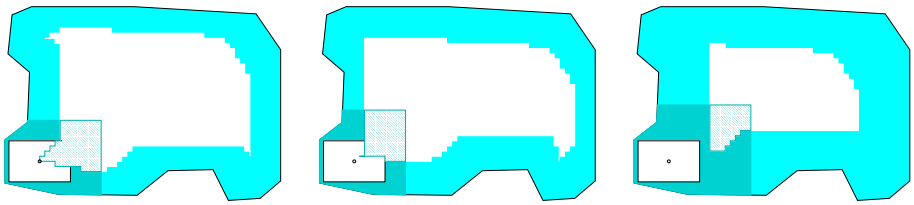
Using instances of the DIN 70020 case, the boxes have a size of at most $16 \times 8 \times 4$ cubes (12.5 mm spacing). With 6 possible orientations, each node representing a box might intersect with 201 other possible boxes, thus the conflict graph will have a maximum degree of 201 for 50 mm grid spacing [15]. A typical conflict graph for the DIN problem would have about 10^4 nodes and 10^6 edges (with 50 mm grid spacing). For the same trunk, a conflict graph using $2''$ spacing for the SAE problem has about 8.000 nodes (due to the larger boxes, where less positions are possible) and about $15 \cdot 10^6$ edges (due to the huge number of conflicts for each box). Including the smallest box-type H, the number of nodes in the conflict graph would actually be twice as much. Since the SAE J1100 standard requires the boxes of type H to be packed after the other boxes have been properly packed, it will be sufficient to exclude the H-boxes temporarily. After having computed a solution without H-boxes, the packing can be extended.

The largest box of the SAE problem consists of $12 \times 9 \times 4$ cubes ($2''$ spacing[3]), hence covering 432 cubes. The box intersects with all boxes

[2] Since SAE boxes are much larger than DIN boxes, there are less grid cells useable for a box, and not all box types might fit at a certain position.

[3] For computational purposes, the SAE box lengths will be rounded down to fit into an integer number of cubes, e.g. the largest box (type A) has $24'' \times 19'' \times 9''$, making a $12 \times 9 \times 4$ grid box. Section 2.8 deals with the problem of rounding the box measures to fit into the grid spacing.

that overlap one of these cubes which means up to 10^5 potential conflicts in a grid of $2''$ spacing for the biggest box of type A.



(a) $6'' \times 4''$ vs $6'' \times 4''$ (b) $6'' \times 4''$ vs $4'' \times 6''$ (c) $6'' \times 4''$ vs $10'' \times 7''$

Fig. 1. Conflict regions

Figure 1 shows regions of forbidden reference points when one box of size $6'' \times 4''$ is already placed. The forbidden area due to the boundary of the trunk is coloured light grey, and the forbidden area due to the placed box is coloured dark grey. The hatched area marks those points that are allowed by the boundary constraints but placing a box there would cause to intersect the previously placed box.

2.3 Reducing the Conflict Graph size

Since most of the following algorithms examine the edges of the conflict graph, it might prove useful to *reduce* its size. To find an IS of maximal size resp. maximal weight, it is necessary to observe some properties of a maximal IS I. The goal is to eliminate some nodes of the graph without reducing the size of the largest IS. In the unweighted case the decision which nodes are to be deleted is quite easy. First, the set of neighbours of a node v has to be defined:

Definition 1 *Let $v \in V$ be a node of G. Then $N(v) = \{u \in V : \{u, v\} \in E\}$ is the set of neighbours of v. The set $N^+(v) = N(v) \cup \{v\}$ denotes the set of neighbours of v including v.*

Now, an elementary property of independent sets can be described.

Theorem 2 *Let $G = (V, E)$ be an undirected graph, and $u, v \in V$ be adjacent nodes ($\{u, v\} \in E$). If $N^+(u) \subsetneq N^+(v)$, then the following is true: For every independent set $I \subseteq V$ containing v, there is also an independent set $I' \subseteq V$ with at least the same cardinality containing u.*

Proof. Obviously, $v \in I \Leftrightarrow u \notin I$ since $\{u, v\} \in E$. $N^+(u) \subsetneq N^+(v)$ states that all neighbours of u are also neighbours of v, so v can be replaced by u. That is, $I' := I \setminus \{v\} \cup \{u\}$.

Theorem 2 can be used in an elegant way to reduce the nodes of a graph. Since the goal is to find an IS of maximum size, it is useful to include

the node u instead of v because v has got a larger neighbourhood and more nodes would be useless for later additions. Since *all* neighbours of u are also neighbours of v, one would rather choose u with its smaller neighbourhood.

What does this mean for the trunk packing problem? If there is a box position near the boundary of the grid, some space between box and boundary would be unusable for other boxes (see figure 1). So it would be better to move the box directly to the boundary, where less space is wasted. This approach is sometimes called a *bottom left justified* packing: All boxes are placed in a way that their bottom, left and front sides touch either other boxes or the surrounding geometry. Any axis-oriented packing can be transformed into a bottom left justified packing [18, 20].

Theorem 2 also applies to the WIS problem:

Corollary 3 *Let $G = (V, E)$ be a weighted undirected graph, and $u, v \in V$ be adjacent nodes ($\{u, v\} \in E$). If $N^+(u) \subsetneq N^+(v)$* and *$w(u) \geq w(v)$, then the following is true:*
For every weighted independent iet $I \subseteq V$ with $v \in I$, there is also a weighted independent set $I' \subseteq V$ with at least the same weight with $u \in I$.

Proof. Simply replace v by u. Then $w(I') = w(I) - w(v) + w(u) \geq w(I)$.

Corollary 3 shows the crucial problem of the SAE case: If two different box types are compared, normally the larger box also gets the larger neighbourhood. So it is not clear which node to eliminate in these cases. Fortunately, there are only seven (a fixed small number) different node weights. So an easy way to reduce the nodes is to do this *typewise*, i.e. to compare only nodes of the same weight. In the general case, one could classify the nodes into a few weight classes and perform the reduction on these classes.

Algorithm 1. Graph reduction
for each type t **do**
  for each edge $\{u, v\}$ with $Type(u) = Type(v) = t$ **do**
    if $N^+(u) \subset N^+(v)$ **then** let $V := V \setminus v$.

Using this procedure it is possible to reduce the number of nodes of a typical conflict graph of the Trunk Packing problem with cell size $2''$ by 75% and the number of edges by up to 90%. Therefore, the performance of the packing algorithms increases significantly because of the smaller conflict graph. The effect of reduceable nodes is directly related to the size of the boxes compared to the trunk. Since the SAE boxes are

relatively large and the DIN boxes are relatively small, the reduction algorithm is only useful for the SAE case. When small boxes (type H) are included, only nodes with relatively close positions to each other could be candidates for Theorem 2. Thus, small boxes reduce the positive effects of graph reduction additionally to their large contribution to the number of nodes and edges. The reduction process itself mainly takes place near the boundary of the grid. Since small trunks have got a large boundary compared to their interior, the conflict graphs of smaller trunks can easily lead to graphs of less than 20% of the original graph size. Additionally, when the grid spacing is reduced, the portion of eliminated nodes tends to increase.

2.4 Brute Force algorithms

The easiest way to find an IS is to successively add a free node to an already existing IS I. A node $v \in V$ is called *free* if there is no edge between v and any node of I. This is done until there are no free nodes left. The SAE case adds the following condition to the definition of free nodes: The number of nodes of the corresponding types must not exceed a pre-defined number, e.g. there may only be four A-boxes and two C-boxes in a valid packing (see Table 1).

This simple approach will hardly lead to the desired optimal solution in the first step. Therefore, a backtracking mechanism has to be added: Test all independent sets of the conflict graph recursively. If there is no free node left, compare the current IS with the best set found so far, and take the better one. Algorithm 2 shows an outline of this procedure. Let therefore $w(v)$ be the weight of node v, and $w(I) = \sum_{v \in I} w(v)$ be the total weight of the set I.

Algorithm 2. Recursive enumeration (G, I)

1. Let F be the set of all free nodes of G
2. **if** $w(I) > w(I_{\max})$ **then** set $I_{\max} := I$.
3. **while** $F \neq \emptyset$ **and** Upper bound $(F, I) > w(I_{\max})$ **do**
   (a) Choose $v \in F$ and set $I' := I \cup \{v\}$, $F := F \setminus \{v\}$
   (b) Recursive enumeration (G, I')
4. **return** $(I_{\max}, w(I_{\max}))$.

Using this approach, the IS of maximum weight will definitely be found since all IS are examined. Unfortunately, the number of independent sets is exponential in the number of nodes. So it is necessary to find an efficient branch-and-bound technique to prune the recursion tree.

2.5 Computing an Upper Bound

Now the main question is: Starting from the current IS, what is the best possible size one could achieve? The exact solution for this problem is equivalent to solving the IS problem on the subgraph induced by all free nodes. Fortunately, the SAE standard gives a hint how to calculate a strong upper bound on the current WIS efficiently: Add the weights of all free nodes but limit the number of the free nodes to the allowed number of boxes for each type. This would mean that the upper bound is calculated at first by using all allowed boxes (≈ 28.7 ft^3), and will only be reduced if there are less free nodes of a certain type left than allowed.

2.6 A Greedy Algorithm

Up to now, only a brute force algorithm has been presented. Although this approach gives the exact result for the current WIS problem, the runtime still is exponential in $|V|$. Maybe a polynomial-time approximation algorithm can be found which is easy to analyze.

For comparison with the optimal solution of the problem, it is necessary to give an additional definition:

Definition 4 *[17] Let $v \in V$ be a node of G. Then the weighted degree $d_w(v)$ is defined as $d_w(v) = \frac{\sum_{u \in N(v)} w(u)}{w(v)}$.*

The weighted degree of a node shows how much weight would become unuseable in relation to its own weight. Now it is possible to sort all free nodes ascending by their weighted degree within the remaining graph. The greedy algorithm for weighted independent sets works as follows:

Algorithm 3. Greedy (G)

1. Let $I := \emptyset$, $U := V$.
2. **while** $U \neq \emptyset$ **do**
   (a) Let $v \in U$ be a node with $d_w(v) = \min_{u \in U} d_w(u)$.
   (b) $I := I \cup \{v\}$.
   (c) $U := U \setminus N(v)$.
   (d) Recompute $d_w(u)$ for all $u \in U$ w.r.t. U (not V).

In [17] an approximation ratio of $\min((\bar{d}_w + 1)/2, (\delta_w + 1)/2)$ for the greedy algorithm is derived. It is obvious that an asymptotic bound is not really useful for the small cases of the Trunk Packing problem. It also comes clear that the greedy algorithm achieves more than the expected asymptotic bound for graphs derived from trunk packing instances.

Observing these facts it will be necessary to extend algorithm 3 by algorithm 2 to enumerate all weighted IS systematically.

Greedy! Line 2d of algorithm 3 needs an efficient update function, especially if it is planned to use this approach for a recursive enumeration of all weighted independent sets. If node v is added to the IS, then all nodes of $N(v)$ would be unuseable. Hence, the weighted degrees of all nodes $w \in N(N(v))$ will be decremented by $w(N(w) \cap N(v))$. In the present implementation, all pairs u, w of nodes of the remaining graph are examined, where $u \in N(v), w \notin N(v)$ and $\{u, w\} \in E$.

Node orders For an enumeration algorithm, the node order is crucial. If the right sequence is chosen, a very good result can be achieved at an early stage, and in combination with a good upper bound, the enumeration process can be cut early. In the previous sections, no specific order of the vertices is assumed. Normally, they are sorted by their time of creation in the conflict graph. In the current implementation the nodes are first sorted by type, then by orientation and position. For instance, a random permutation might be chosen and the recursive algorithm executed like a multi-start approach, increasing the probability to find the optimal solution earlier. Again, this approach is difficult to analyze. The experiments showed a large average runtime for the randomized node order. Another possibility would be taking nodes with small weighted degree first. In the trunk packing problem, the weighted degrees are directly correlated to the sizes of the boxes, hence a node of small weighted degree represents a small box. Therefore, this strategy means packing small boxes first.

For the SAE Trunk Packing problem the following observation can be made: If a large box is taken first, it will cause a large amount of nodes to be unuseable and therefore the remaining conflict graph will be significantly smaller. Also, a large box has only few possibilities to be placed, making the recursion tree narrower. But if a small box is chosen, the number of possibilities will be much larger and the algorithm will be stuck at an early stage by examining many almost equal sets. This approach is very alike a human's way to pack a trunk: take the big cases first, and then stuff the smaller boxes in between. Now an additional effect comes into account: If the nodes with large weights are used first and the small ones afterwards, it is possible that the last few (small) nodes can not exceed the best found IS, and the algorithm can make effective use of the upper bound discussed in section 2.5. This implementation proved to be the fastest in all experiments.

2.8 Rounding the box sizes

Another discouraging aspect is the need for rounding the box sizes. In a $2''$-grid, only 8 of 21 side lengths are represented correctly (only the even ones), and all other lengths would have to be rounded. Table 1 shows that the largest grid spacing without rounding the box sizes would be $0.1''$, thus far from practicable. Now there are some different possibilities to be discussed:

1. All lengths are rounded up to the next even integer.
2. All lengths are rounded down to the last even integer.
3. All lengths are rounded to the nearest even integer.
4. The lengths are rounded up or down with respect to the available space.

The first strategy will provide a *feasible*[4] solution when the set is converted into a real box set. Unfortunately, this strategy will waste space between the boxes. Strategy 2 causes the opposite problem. Since the grid boxes are smaller than the real boxes the resulting solution will be unfeasible due to intersections between boxes. These intersections can be resolved by using a physical simulation of contacts between the boxes and the trunk [5]. However, the intersections might be too severe to be resolved. To prevent this, the outmost layer of *inside*-cells is deleted. This provides a feasible solution in most cases. Strategy 3 will cause both problems of the first two rounding strategies. A mixed strategy would raise the problem of data representation. Currently, strategy 2 is used. In most cases it suffices to delete one layer of *inside*-cells in each dimension to generate legalizable solutions, i.e. the resulting packings themselves would not fit into the trunk, but the packed boxes could be translated and rotated to fit into the trunk. However, this reduction of the grid causes another deficit in comparison to a manual solution.

3 Omitting the grid

An optimal solution with respect to a grid is limited to fixed orientations and discretized placements. Thus, such a solution is unlikely to be optimal with respect to the original trunk packing problem. If the restriction to discrete placements is dropped, it might be possible to find a better solution. Omitting the grid means that the box sizes are not required to be multiples of a grid cell size and therefore do not have to be rounded. So a legalization step is not necessarily required.

Schepers [18] shows how to compute packings of axis-oriented boxes within a cubic container of "unit" length. This is done by solving several linear programs.

[4] A packing is considered feasible if there are no intersections between the packed boxes as well as between the boxes and the trunk.

For a fixed set of boxes we use enumeration to decide the relative position of the boxes – i.e. for each pair of boxes, we decide whether the first box lies left, right, above (but neither left nor right), below, in front of, or back of the other box. Using the center coordinates of each box, the relative positions between boxes can easily be enforced by linear inequalities. Furthermore we can describe the set of all feasible points for the centers of the boxes by linear inequalities.

Hence the feasibility of such a configuration, called packing pattern, can be tested by verifying the feasibility of a linear program. Schepers additionally shows how to solve these linear programs combinatorially, but this result does not generalize to our approach. The optimal set of boxes is found by enumerating all sets of boxes (such that a subset fits into the container).

Let $T \subset \mathbb{R}^3$ be the interior of the trunk. For a box B with fixed type and orientation, let T_B be the Minkowski sum of the trunk boundary and B. Now the set $M = T \setminus T_B$ describes all reference points where box B can be anchored within the trunk.

To generalize the approach described above to the case of a trunk, the set M has again to be described by linear inequalities. As this set is not convex, we over-approximate the set M by a convex region and allow to place some convex bodies, called obstacles, into the set. Similar to the case of two boxes, we have to know the relative position of every box and every obstacle, i.e. knowing which of the half spaces defining the convex obstacle does not intersect the center of the box.

Notice that the complexity of the approach grows rapidly with the number of half spaces describing the inserted obstacles. Therefore we aim to describe a close approximation of the feasible region for the centers by a small number of obstacles.

Currently this description is done manually. First, the trunk itself is approximated by linear equations, and then cuboid obstacles are inserted. It is quite easy to calculate the Minkowski sum of a set of linear equations and a box. Especially the desired set $M = T \setminus T_B$ can be described as the same linear equations translated by the half diagonal of the box. Since the obstacles are restricted to be convex, they can be described in the same way. At the moment, we develop automated methods to describe the set M in the required way.

To improve the practical efficiency of this approach, we enumerate the region, the inequalities of an obstacle and the relative position only if the appropriate constraints are violated by our solution of the linear program.

As the number of feasible packing patterns and therefore the number of linear programs solved increases rapidly with the size of the trunk, the enumeration of all packing patterns is very time consuming. For the manually created approximations, we are able to enumerate all packing patterns within a few days. Moreover, we experimented with the following heuristic. We first enumerate packings that use only large boxes. Then we take some of the best packings and add smaller boxes if possible.

4 Experimental results

All grid based algorithms discussed so far are exact, so they will find the optimal solution for the given discretization of the trunk. Table 2 shows the best achieved results of some typical instances. As can be seen, the results achieved by the LP approach are comparable to those provided by the grids. Very fine grids cause a too large conflict graph for an efficient computation. Also, the finer grids do not necessarily provide the best grid-based solutions, contrary to the DIN case [9]. This follows from the rounding strategy described in section 2.8. Rounding the box sizes according to a large grid spacing leads to larger intersections between the packed boxes. Anyway, it is possible in most cases to resolve these intersections using a physical contact simulation [5]. The best grid-based results compared to the computational effort could be achieved using the $2''$ and $1.5''$ spacings.

trunk	manually	grid-based (algorithm 2)				LP-based
		$3''$	$2''$	$1.5''$	$1''$	
small	5.337 ft^3	4.767 ft^3	5.071 ft^3	5.648 ft^3	5.392 ft^3	5.792 ft^3
mid-size	9.358 ft^3	8.538 ft^3	9.207 ft^3	9.418 ft^3	9.041 ft^3	9.521 ft^3
large	11.837 ft^3	10.871 ft^3	12.056 ft^3	12.674 ft^3	11.790 ft^3	12.637 ft^3

Table 2. Experimental results for some problem instances without H-boxes

The pre-computation of the node order proved quite useful. If the nodes are sorted by their weighted degrees only, the problem of calculating the upper bound remains: One has to scan all free nodes whether there still are nodes of a certain type left or not. If the nodes are sorted by their type, this examination is obsolete: If the algorithm adds a node of type t, then it is guaranteed that no more nodes of previous types are free. This leads to an additional time saving effect. The runtimes needed for verifying the optimal solution are compared in Table 3. Within this context, verifying an optimal solution implies a complete run of the algorithm.

The algorithms `timestamp`, `randomized` and `sorted` refer to algorithm 2 and use different node orders: the time of creation, a randomized order and sorted by box size and weighted degree, respectively. Algorithm `greedy` refers to the recursive greedy algorithm discussed in section 2.7.

trunk	grid	nodes/edges	timestamp	randomized	sorted	greedy
small	$1.5''$	$470/7.8 \cdot 10^4$	$5sec$	$2.5sec$	$< 1sec$	$< 1sec$
small	$1''$	$850/2.5 \cdot 10^5$	$1m40sec$	$20sec$	$< 1sec$	$2sec$
mid-size	$3''$	$480/6.3 \cdot 10^4$	$45sec$	$3m$	$1.75sec$	$7sec$
mid-size	$2''$	$1200/4 \cdot 10^5$	$40m$	$1h40m$	$10sec$	$1m$
mid-size	$1.5''$	$2200/1.4 \cdot 10^6$	$> 24h$	$> 24h$	$3m20sec$	$13m$
large	$3''$	$660/1.1 \cdot 10^5$	$6m$	$10h$	$6sec$	$30sec$
large	$2''$	$2000/9.6 \cdot 10^5$	$18h$	$> 24h$	$17sec$	$1m30sec$
large	$1.5''$	$4800/5.9 \cdot 10^6$	$> 24h$	$> 24h$	$8m20sec$	$45m$

Table 3. Runtimes needed by various grid based strategies for exact solution (without H-boxes)

All grid algorithms were terminated after 24 hrs due to the given time constraints. It is easy to see that the order of the nodes plays an important role in the search for the best possible IS.

The LP algorithm provided exact solutions for a given approximation only for small instances. Otherwise, the algorithm was stopped after 24 hours runtime. Table 2 shows that both approaches, the grid-based and the LP-based approach, are almost equal with slight quality advantage on the LP side. On the other hand, the grid-based algorithms are easier to operate.

5 Summary

In this paper the trunk packing problem for the US standard SAE J1100 has been investigated. Two combinatorial approaches were presented: First, a discretization of the space to be packed, analogous to [9, 15] for the DIN case. Second, an approximation scheme using linear inequalities. For both approaches enumerative algorithms have been described. As shown in section 4, the brute-force algorithms suffice for the trunks used in car design processes to enumerate all possible solutions for the resulting Weighted Independent Set problem. This could be reached by reducing the conflict graph and applying a property provided by independent sets.

Unfortunately, a discretization of the space leads to insufficient representation of the SAE-boxes. So additional methods have to be used to overcome illegal situations such as box-box intersections. These methods include a contact simulation provided by a physics engine [5].

The LP approach needs a space consisting of few convex regions and only few convex obstacles. So far no automated scheme is known to provide such simplifications for a complicated geometric structure like a trunk. Additionally, the enumerative algorithms are exponential in runtime. This means that large instances (e.g. small boxes into a large irregular shaped container) can only be solved approximately.

Further work includes more efficient graph reduction algorithms or a fast exact algorithm for the WIS problem capable of handling large trunk instances and fine grids. For the LP algorithm it is necessary to find a

good approximation of the trunk to ensure the resulting packing to fit into. Also an improved heuristic for large instances is needed.

References

[1] Tobias Baumann, Elmar Schömer, and Kai Werth. Solving geometric packing problems based on physics simulation. Submitted for ESA'07, 2007.

[2] Piotr Berman. A d/2 approximation for maximum weight independent set in d-claw free graphs. *Nordic J. of Computing*, 7(3):178–184, 2000.

[3] Jonathan Cagan and Quan Ding. Automated trunk packing with extended pattern search. *Virtual Engineering, Simulation & Optimization*, SP-1779:33–41, 2003.

[4] Friedrich Eisenbrand, Stefan Funke, Andreas Karrenbauer, Joachim Reichel, and Elmar Schömer. Packing a trunk: now with a twist! In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 197–206, New York, NY, USA, 2005. ACM Press.

[5] Friedrich Eisenbrand, Stefan Funke, Joachim Reichel, and Elmar Schömer. Packing a trunk. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003: 11th Annual European Symposium*, volume 2832 of *Lecture Notes in Computer Science*, pages 618–629, Budapest, Hungary, September 2003. Springer.

[6] Deutsches Institut für Normung e.V. DIN 70020, Teil 1, Straï¿½nfahrzeuge; Kraftfahrzeugbau; Begriffe von Abmessungen., February 1993.

[7] Akihisa Kako, Takao Ono, Tomio Hirata, and Magnús M. Halldórsson. Approximation algorithms for the weighted independent set problem. In *Graph-Theoretic Concepts in Computer Science, 31st International Workshop, WG*, pages 341–350, 2005.

[8] Andreas Karrenbauer. Packing boxes with arbitrary rotations. Master's thesis, Universität des Saarlandes, Saarbrücken, 2004.

[9] Ulla Neumann. Optimierungsverfahren zur normgerechten Volumenbestimmung von Kofferräumen im europäischen Automobilbau. Master's thesis, Technische Universität Braunschweig, Braunschweig, 2006.

[10] Society of Automotive Engineers. SAE J1100, Motor Vehicle Dimensions, February 2001.

[11] Joachim Reichel. *Combinatorial approaches for the Trunk packing problem.* PhD thesis, Saarbrücken, 2006.

[12] Jens Rieskamp. Automation and Optimization of Monte Carlo Based Trunk Packing. Master's thesis, Universität des Saarlandes, Saarbrücken, 2005.

[13] Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Appl. Math.*, 126(2-3):313–322, 2003.

[14] Jörg Schepers. *Exakte Algorithmen für orthogonale Packungsprobleme.* PhD thesis, Köln, 1997.

[15] Jeffrey S. Warren and Illya V. Hicks. Combinatorial branch-and-bound for the maximum weight independent set problem. http://ie.tamu.edu/people/faculty/Hicks/jeff.rev.pdf, 2006.

[16] Michael Wottawa. *Struktur und algorithmische Behandlung von praxisorientierten dreidimensionalen Packungsproblemen.* PhD thesis, Köln, 1996.

[17] Al Zimmermann. Al Zimmerman's Circle Packing Contest. Al Zimmerman's Programming Contests, http://www.recmath.org/contest/CirclePacking/, 2005.