Packing a Trunk – now with a Twist!

Friedrich Eisenbrand
MPI für Informatik
Saarbrücken, Germany
eisen@mpi-sb.mpg.de

Stefan Funke
Computer Science Department
Stanford University
sfunke@stanford.edu

Andreas Karrenbauer
MPI für Informatik
Saarbrücken, Germany
karrenba@mpi-sb.mpg.de

Joachim Reichel
MPI für Informatik
Saarbrücken, Germany
reichel@mpi-sb.mpg.de

Elmar Schömer
Johannes Gutenberg Universität
Mainz, Germany
schoemer@informatik.uni-mainz.de

Abstract

In an industry project with a German car manufacturer we are faced with the challenge of placing a maximum number of uniform rigid rectangular boxes in the interior of a car trunk. The problem is of practical importance due to a European industry norm which requires car manufacturers to state the trunk volume according to this measure.

No really satisfactory automated solution for this problem has been known in the past. In spite of its NP hardness, combinatorial optimization techniques, which consider only grid-aligned placements, produce solutions which are very close to the one achievable by a human expert in several hours of tedious work. The remaining gap is mostly due to the constraints imposed by the chosen grid.

In this paper we present a new approach which combines the grid-based combinatorial method with *Simulated Annealing* on a continuous model. This allows us to explore arbitrary orientations and placements of boxes, hence closing the gap even further, and – in some cases – even surpass the manual expert solution.

The implemented software system allows our industrial partner to incorporate the trunk volume in a very early stage of the car design process without relying on a repeated and cumbersome manual evaluation of the volume.

CR Categories: J.6 [Computer-aided Engineering]: Computer-aided Design; G.1.6 [Numerical Analysis]: Optimization—Simulated Annealing I.6.8 [Simulation and Modeling]: Types of Simulation—Monte Carlo; General Terms: Algorithms, Design, Measurement

Keywords: Trunk Packing, Car Design, Simulated Annealing, Combinatorial Optimization, Computational Geometry

1 Introduction

Geometric packing problems are of great interest to the communities of Computational Geometry (see for example [Baur and Fekete 2001; Chan 2003]) and Combinatorial Optimization (see for example [Erdos and Graham 1975; Nelissen 1993; Dyckhoff 1990; Ikonen et al. 1997]) due to their great importance for industrial applications. The packing problem considered in this paper arose from a joint project with a major German car manufacturer who is interested in measuring the volume of a trunk according to the German standard DIN 70020. The reason for the existence of this standard is that the continuous volume of a trunk does not reflect its actual storage capacity, since the baggage to be stored is usually discrete. DIN 70020 asks for the number of rigid 200mm $\times$ 100mm $\times$ 50mm $=$ 1 liter boxes, that can be packed into the trunk.

Figure 1: Physical measurement according to DIN 70020

So far this task of determining the volume of a trunk required cumbersome manual work by an experienced engineer who packs the trunk by hand as seen in Fig. 1. Since design decisions also depend on their effect to the volume of the trunk, the engineers estimate the volume by manually placing boxes upon visual judgment with a CAD system into a virtual model of the trunk.

An industry-strength automated solution to the problem has to meet the following requirements:

- Boxes are not allowed to overlap with each other.
- Boxes must not pierce the boundary of the trunk by more than a predefined threshold (which models its deformability).
- The system has to deal with any output of the CAD system: input models as exported from the CAD system are just a set of triangles; they might exhibit holes, dangling triangles, and typically do not form a manifold. There is no notion of *inside* or *outside the trunk* in this data.
- The number of packed boxes should never fall short of the expert's solution by more than 1% or 10 liters.
- The solution for a problem instance should be computed within a time-frame of about one day.

Previous Work

Automated solutions known in the past have never been able to get close to the quality of a human expert. Ding and Cagan [2003] published an approach that is suited for the Society of Automotive Engineers (SAE) standard used in the USA. This standard differs significantly from the European norm. It defines a luggage set consisting of objects varying from 6 to 67 liters. Therefore, our software has to handle far more objects than in the SAE case.

NP-Hardness of our packing problem was established in a recent result ([Eisenbrand et al. 2003]), but still, in the same paper a first almost industry-strength solution was presented that using a discretization model and techniques from combinatorial optimization produced solutions very close to the ones achievable by a human expert. It only rarely fell short of the prescribed bounds for the solution quality, which was due to the fact that this algorithm only allowed axis-aligned and discrete placements of the boxes in a grid. For these "bad" problem instances – which for example arise in trunks with side-compartments holding tightly a few boxes (see Fig. 2)–, there was also no hope to obtain a better solution using this approach, as the discretization would have had to adapt locally to the geometry of the trunk[1].

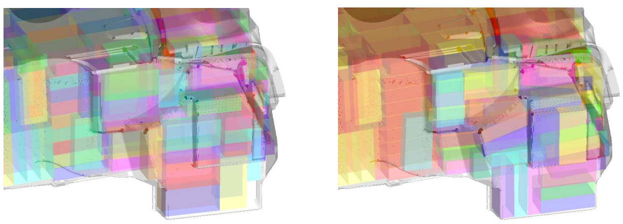

Figure 2: Arbitrary placement necessary in side-compartments next to the wheelhouse

On the theory side, results in this area have been rather discouraging. [Fowler et al. 1981] have shown that already the 2-dimensional problem of packing axis-parallel unit squares into a polygon with holes is NP-complete, even though approximation schemes are available [Hochbaum and Maass 1985]. For polygons without holes, it is conjectured ([Baur and Fekete 2001]) that the problem is polynomially solvable. The more general problem of packing $(a \times b)$ axis-aligned rectangles inside a polygon, is not even known to be in NP, partly because the representation of an optimum solution might be arbitrarily complex (see [Nelissen 1993] for a elaboration on the topic). The 'Open Problem Project' website ([Demaine et al. 2005a; Demaine et al. 2005b], problems 55, 56) keeps track of the current status of these problems.

Our Contribution

In this paper we overcome the weakness of the previous algorithm and present an industry-strength solution to the trunk packing problem which almost always comes very close to the manual solution and in some cases even surpasses it. This breakthrough is achieved by allowing *arbitrary* orientation and placements of the boxes, not restricting to axis-aligned placements on a grid as in the algorithm from [Eisenbrand et al. 2003].

Allowing such a continuous model leads to a very high-dimensional global optimization problem for which standard methods like *Simulated Annealing* [Kirkpatrick et al. 1983] are typically used. Unfortunately, applying these techniques in a straightforward manner yields solutions far worse than the discretization algorithm. Only by combining both techniques we were able to obtain the industry-strength system.

[1] These pathological instances were discovered when evaluating the system from [Eisenbrand et al. 2003] in the industrial production environment.

This paper elaborates on the details of the synthesis of both methods. Roughly speaking, we were able to eliminate the heating process typical for a simulated annealing procedure by using a solution from the discretization algorithm as a starting configuration. Furthermore we devised special procedures for creation of new boxes, a relaxation by a Monte Carlo simulation, and pruning of undesired boxes. Due to physical analogies, we call our method *Specialized Grand Canonical Simulated Annealing*.

The resulting software system meets all requirements of our industrial partner and is currently being installed for use in the actual design process of new cars.

2 Modeling the Problem

We are provided with the digital data of the car trunk by a set of triangles exported from a free-form CAD system. Since the original model in the CAD system often consists of many parts which are not necessarily tightly joined, the resulting set of output triangles neither bound a closed volume nor do they form a manifold. This 'low quality' of the input data requires additional care in the processing of our algorithms. In the following section we will present two ways to model the trunk packing problem such that optimization techniques can be successfully applied.

2.1 Discretizing Space and Box Orientations – a Combinatorial Approach

The approach used in [Eisenbrand et al. 2003] proposes a discretization of space and box orientations by constructing a three-dimensional cubic grid which approximates the interior of the trunk. Boxes can only be placed anchored at a grid cube and in alignment with the grid axes, so the placement of a box is determined by six parameters: the anchor cube with coordinates (x,y,z), and the orientation (w,h,d) which describe the extension of the box in width, height and depth (measured in unit cubes). The spacing $\widetilde{w}$ of the cubic grid is chosen such that $k \cdot \widetilde{w} = 50$mm for some integer k, and hence the orientation (w,h,d) can be any permutation of the set $\{1k, 2k, 4k\}$. As the following stages are very sensitive to a "good" initial placement and orientation of the cubic grid, the latter is chosen such that the number of cubes contained in the interior of the original trunk volume is maximized. See Fig. 3 for an example.

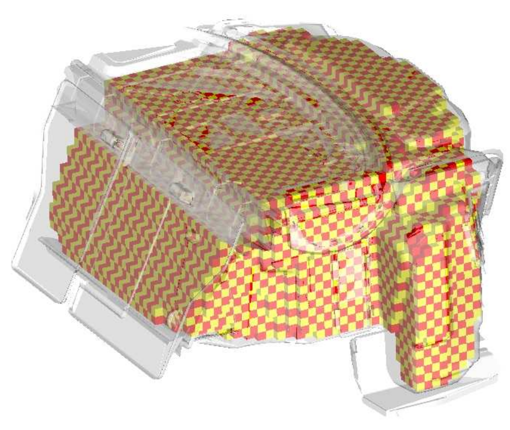

Figure 3: Cubic grid approximating the interior of a trunk

The slightly modified goal is now to place as many boxes in the cubic grid such that each box consists only of cubes approximating the interior of the trunk, and no two boxes share a cube. This problem can be formalized using the following construction: Let $G(V,E)$ be the graph with node set V and edge set E. There is a node $v_{x,y,z,w,h,d} \in V$ iff the box anchored at (x,y,z) and orientation (w,h,d) consists only of cubes in the interior of the trunk. There is an edge $e = (v,w) \in E$ iff the two corresponding box placements of v and w intersect, i.e. share a common cube. G is called the *conflict graph*. Packing the largest number of boxes in the cubic grid is then equivalent to determining the *maximum independent* or *maximum stable* set in the conflict graph G.

In [Eisenbrand et al. 2003], several techniques from integer linear programming and combinatorial optimization were applied to solve this stable set problem. This approach produced packings which most of the time were sufficiently close to the prescribed quality bound of our industrial partner. Though, when evaluating this system in the industrial production environment, some instances showed up, where the results were not satisfactory.

The inherent problem of this approach is that it chooses right at the beginning a discretization of space and orientations which might not accommodate to the local geometry of the trunk. While some choice of the grid axes might be suitable for most regions of the trunk (e.g. typically it is reasonable to have two grid axes parallel to the bottom of the trunk), there are areas where a different orientation is necessary if no space should be wasted. This happens in particular along curved parts of a trunk, see for example in Fig. 4, where the restriction to one cubical grid system wastes a lot of volume along the curved lid (left picture) compared to a solution with arbitrary rotations (right picture).

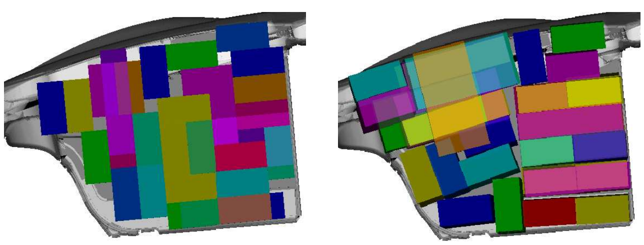

Figure 4: Curved lid

Similar difficulties arise in trunks with side compartments which tightly hold a few boxes. If the grid is not aligned with these places, volume is wasted, see also Fig. 2.

2.2 Arbitrary Placement of Boxes – a Simulated Annealing Approach

To overcome the problems with these pathological cases, we have to extend our model to account for arbitrary positions and orientations of the boxes. In such a model the placement of a box can be characterized by a 6-tuple $(x,y,z,\theta,\varphi,\psi) \in \mathbb{R}^6$ (not $\mathbb{Z}^6$ as in the discrete model). We are interested in a collection of n such 6-tuples such that their corresponding box placements do not overlap and are completely contained in the interior of the trunk. The value n should be as large as possible such that a valid placement still exists.

Let us first focus on the case where n is fixed, and we are only looking for a valid placement of n boxes. This $6n$-dimensional problem is far too complicated to be solved using techniques from convex optimization, hence generic optimization techniques like *Simulated Annealing (SA)* are the method of choice. This technique can be implemented by designing a suitable *potential* or *energy function* $U : \mathbb{R}^{6n} \to \mathbb{R}_0^+$ from possible configurations, i.e. placements of the n boxes, to the real numbers. Valid configurations should result in a low potential or energy value whereas invalid configurations due to overlap or non-containment in the trunk should be assigned high potential values. The goal is to find a global minimum (also called *ground state*) of this potential function U. The approach of SA is to define transitions from one configuration to another and then basically start a random walk on the implicitly defined (potentially infinite) graph. At the beginning, at "high temperature", transitions might happen even to configurations with a higher energy value, but with decreasing temperature, configurations of lower energy are preferred. In our solution we enhance the basic SA approach with a method for growing and shrinking the size of a configuration (i.e. the value n).

Both, a suitable definition of a potential function for our problem as well as some extensions to the basic SA process will be the topic of the rest of this paper. The latter extensions were crucial for obtaining solutions superior to the discretization approach.

3 Simulated Annealing – Potential Function and Basics

In this section we will derive a potential function for the trunk packing problem, give a brief overview of the employed simulated annealing process and provide some details about an efficient evaluation procedure for the potential function.

3.1 The Potential Function

It is natural to split the potential into two parts. On one hand we have the penetration of the exterior that we measure by the so called *wall potential* U_W. On the other hand we have a contribution from pairwise interaction of the boxes. The interaction term consists of the *intersection volume* U_V and the *interpenetration depth* U_I of two boxes. We define our potential as a convex combination of these three parts. Let $x = (x_1,\ldots,x_n)$ be the coordinates of a configuration where x_i is the set of the coordinates for box i. Then, we have for the potential

$$U(x) = \lambda_W \sum_{i=1}^{n} U_W(x_i) + \lambda_V \sum_{i=1}^{n}\sum_{j=1}^{i-1} U_V(x_i,x_j) + \lambda_I \sum_{i=1}^{n}\sum_{j=1}^{i-1} U_I(x_i,x_j),$$

where $\lambda_W, \lambda_V, \lambda_I \geq 0$ are the respective weights for the three contributions. The condition $\lambda_W + \lambda_V + \lambda_I = 1$ for the convexity ensures that we do not effectively change the temperature of the system by reweighting the contributions during the algorithm.

In the next paragraph we develop the contribution of the pairwise box interaction to the potential. For the sake of simplicity, we restrict ourselves to the two-dimensional case. The extension to three dimensions should be straight forward.

Interaction Given two boxes by the open sets B_i, B_j defined by their coordinates x_i, x_j for position and orientation, we define the *intersection volume* U_V as

$$U_V(x_i,x_j) = \int_{B_i \cap B_j} dV.$$

Using only the intersection volume U_V and neglecting the interpenetration depth U_I in the potential would be sufficient in theory. But the overlap is not qualified very intuitively in some exceptional situations. Therefore we do not only consider the intersection volume, but also the penetration depth of two boxes.

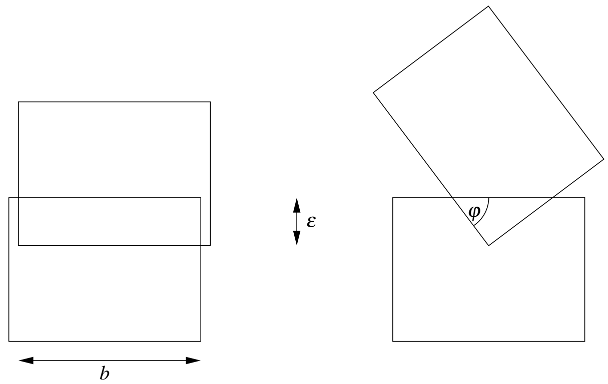

Figure 5: Motivation for penetration depth

Consider two rectangles that touch at an edge with length b. If we now push the rectangles into one another by ε orthogonally to that edge, we get an overlap $U_V = b \cdot \varepsilon$ as depicted in Fig. 5. Now assume that the two rectangles touch by an edge and a vertex. The overlap that results from a penetration of ε is given by $U_V = \frac{\varepsilon^2}{\sin 2\varphi}$. Since $\varepsilon \ll b$ the overlap resulting from the same penetration is much smaller in the second case opposing our intuition. Therefore, we define an additional measure that is more adequate in such situations.

Definition 1. *Given a metric space $(X, |\cdot|)$, the penetration depth of two open sets $A, B \subset X$ is defined as*

$$\min\{|t| : \quad A \cap (t+B) = \emptyset, \quad t \in X\}.$$

Informally speaking, the penetration depth is the distance that one object needs to be translated in order to dissolve the intersection. We set the *interpenetration depth* $U_I(x_i, x_j)$ of two boxes to their penetration depth and get a further contribution to the potential in addition to the intersection volume U_V. We cannot simply replace U_V by U_I because the latter has also its drawbacks. Consider the left situation depicted in Fig. 5. The penetration depth is the small vertical distance ε. If the rectangles are blocked in that direction, i.e. that vertical moves are not permitted by the boundary or other rectangles, then the only way to reduce the overlap is by moving horizontally. But for a great deal of horizontal moves the potential looks the same with respect to the penetration depth, namely ε and thus involving a lot of iterations to get out.

Wall Potential In two-dimensional packing problems one often has a polygon that defines the container. There it is possible to take its complement and treat it like an obstacle. In that case the penetration depth of the boxes with the boundary is well defined.

But this does not hold in our particular case. The data of the trunk is given as a triangular mesh with a penetration threshold p_T for each triangle. We may not assume any further properties on the mesh, e.g. that it is a manifold, watertight or that the normals are oriented consistently.

Though, we can use the principle of the penetration depth similar to the two dimensional case, but we must adjust it to our setting. The penetration depth for each box is defined per triangle instead of the whole body. Thereby, it is also possible to treat certain regions differently, e.g. the bottom of a trunk would hardly give way in contrast to the sides where a few millimeters are always acceptable.

This introduces some problems with respect to the total potential. We cannot just sum up the values for each triangle since then the value would increase if the boundary is triangulated finer. Therefore, we use the maximum of all triangles. It is also possible to use the mean of all positive contributions.

The penetration depth for a triangle with respect to a given box is also defined as the minimum distance to pull it apart from that box.

Thereby, the wall-potential is positive only for those boxes touching the boundary. But this is definitely not what we want for boxes located completely outside.

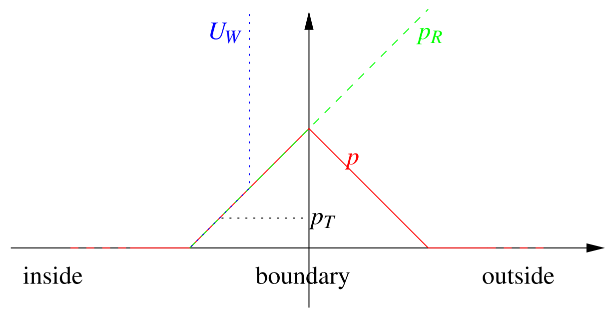

Figure 6: Wall potential of a box moving through the boundary

It is even more counter-intuitive if we think of a box moving from the inside to the outside like illustrated in Fig. 6. Its potential increases to a maximum attained when the center crosses the boundary and then decreases symmetrically to zero again until it does not touch the trunk anymore.

Therefore, we define the *restricted penetration depth* p_R that only considers translations that would move the box back inside the trunk. Hence, we have to define what is inside and outside with respect to the trunk in a way we can use it for computation. But as pointed out earlier, there is an ambiguity in the representation of the trunk since we cannot expect the model to be watertight.

Thus, the restricted penetration p_R is infeasible to compute. But since it coincides with the penetration depth p in certain regions, we just use the latter and ensure that we report "infinity" in case of $p < p_R$.

We define the *wall potential* U_W in terms of a predicate that tells us whether a box should be treated as outside the trunk and use the maximal penetration depth otherwise, i.e.

$$U_W(x_i) = \begin{cases} \infty & \text{if } \texttt{outside}(x_i) \\ \max\{p(x_i, \tau) : \tau \in \mathit{Triangles}\} & \text{otherwise} \end{cases},$$

where $p(x_i, \tau)$ is the minimum distance that the triangle τ has to be translated such that it does not intersect the box. We must design the `outside` predicate very carefully such that it is robust but nevertheless efficient to evaluate. This issue is addressed in Sect. 4.3

Observe that the contribution of a particular box to the potential has a short range, i.e. it is completely determined by objects in the neighborhood. We can benefit from this when evaluating the potential.

3.2 Monte Carlo algorithm

In 1953, Metropolis et al. introduced the Monte Carlo importance-sampling algorithm [Metropolis et al. 1953]. It's a method that is used in statistical physics to predict or check macroscopic properties of complex, i.e. many-body, systems that result from an assumed potential.

In nature, realizations with different energies of such a statistical ensemble do not appear with the same probability. The configurations that lead to a lower potential energy are much more likely to occur. In our case we consider the Boltzmann distribution that suggest a probability for a configuration with potential energy U that is proportional to $e^{-\beta U}$ where β is the inverse temperature. We treat β simply as a parameter and only refer to its physical meaning to get an intuition.

Since it is infeasible to enumerate all configurations, we try to limit the evaluations of the potential to important configurations. Given a starting configuration, we want to "walk" through the configuration space guided by our potential. Therefore, we propose trial moves that are accepted depending on the change in the potential. If the proposed configuration has a lower energy we accept it with probability 1. Otherwise, we accept it with a probability $p = e^{-\beta \cdot \Delta U}$ where ΔU is positive. This scheme is depicted in Fig. 7.

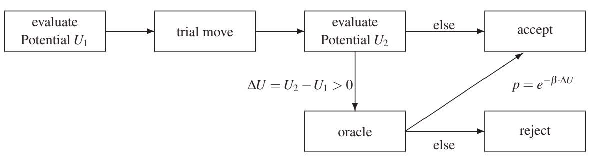

Figure 7: Metropolis Monte Carlo Scheme

It is absolutely necessary to take steps that preliminary lead to a higher energy in order to escape from local minima. This behavior is controlled by the parameter β. The higher its value is the more unlikely is the worsening in terms of the potential energy.

Since the boxes in the middle are much less flexible than the boxes at the boundary, each box has its own range from that we choose our trial moves. We dynamically adapt these ranges by increasing it if a move is accepted and decreasing it otherwise. Thereby, we achieve that the acceptance settles down at 50 %.

Since the potential seen by a box does not look like the same in every direction, we do not adjust the ranges for all coordinates in the same way. We rather distribute the amount of the change to the coordinates with respect to the suggested trial move.

We do not move all boxes simultaneously but one randomly picked in each iteration.

3.3 Evaluating the Potential Function efficiently

This is the most time-consuming task in our algorithm. Profiling experiments have shown that more than 90 percent of the CPU-time is spent in the routines for the computation of the intersection volume and the penetration depth. Again, we consider the two dimensional case to illustrate the idea.

First, we describe a method that decides whether two rectangles overlap. Afterwards we slightly modify that procedure by adding little costs so that it also tells us the penetration depth. At the end of this section, we illustrate how to compute the overlapping area.

It can be easily shown that the definition of the penetration depth is equivalent to

$$\min\{|\vec{t}| : \vec{t} \in \overline{A \oplus (-B)}\},$$

i.e. in the complement of the Minkowski sum of both sets. Actually, we do not compute any Minkowski sums here but use this technique to prove the correctness of our approach.

By the definition of the penetration-depth of two rectangles R_i and R_j, the minimum is attained at the border B of the set $\overline{R_i \oplus (-R_j)}$ which is determined by at most eight constraints that are parallel to the sides of the two rectangles as seen in Fig. 8.

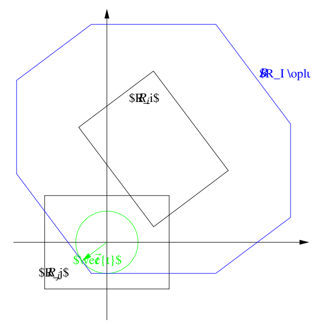

Figure 8: Determining the penetration depth

Since we consider the penetration depth with respect to the Euclidean distance, it coincides with the radius of the maximal circle centered at the origin that fits into B.

By elementary geometry the vector pointing from the origin to the boundary point is perpendicular to the corresponding face. Hence, the direction of a translation that determines the penetration depth is a normal of one of the faces that define B.

By the separating axis theorem for convex polyhedra [Gottschalk et al. 1996], these normals are either the ones of the two polyhedra or are parallel to the cross product of an edge of the first polyhedron and an edge of the second one. Since we consider rectangles in 2D those normals are exactly the directions of the edges of the rectangles.

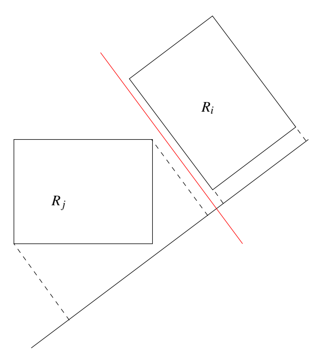

Figure 9: Separating axis

We consider the projections of the objects on one of those directions now. Hence, we have two intervals each corresponding to one of the polyhedra. The separating axis theorem tells us further that the two polyhedra are disjoint iff for at least one of those directions the two intervals are disjoint.

Thus, we have a test that tells us two rectangles apart. In the worst case it requires to test four directions since at least half of the eight edges are parallel. As soon as we get two disjoint intervals we are done and report that there is no overlap or the penetration depth is zero respectively.

Furthermore, we can modify this test slightly in order to get a method that computes the penetration depth directly at not much more cost. Assume we test in direction $k = 1, \dots, 4$ and observe an overlap of t_k of the intervals that result from the projection.

Without loss of generality, we may assume that t_k is positive since otherwise we swap the roles of the rectangles. We construct a translation $\vec{t_k}$ with $|\vec{t_k}| = t_k$ pointing in the direction of the projection.

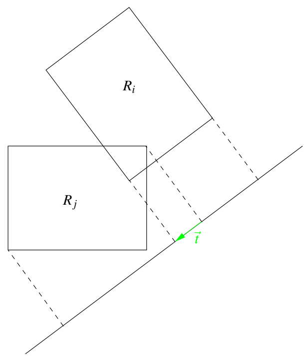

Figure 10: Penetration depth by projection

If we translate the corresponding rectangle by $\vec{t_k}$ the two rectangles become disjoint because then we can place a separating hyperplane with normal $\vec{t_k}$ between them. Recall that the value for the penetration depth is attained by a vector that points into one of the tested direction. Thus, we simply report the minimum absolute value of all those t_k.

Intersection volume of two boxes Since boxes are convex polyhedra their intersection can be simply computed by solving a halfspace intersection problem. In general the intersection body consists of trimmed facets originating from both boxes. In order to find these trimmed facets, we triangulate the boundary of every box and clip its triangles by the three pairs of parallel planes defining the other box. Clipping a triangle by a plane may produce a quadrangle which we decompose into two triangles for further clipping. This basic geometric operation can be implemented very easily and results in a highly specialized and effective routine for determining a set $\mathscr{T}$ of oriented triangles which form the boundary of the intersection body. Connectivity information among these triangles is not required for computing the desired volume: Let $\Delta = (\vec{a}_\Delta, \vec{b}_\Delta, \vec{c}_\Delta)$ denote an oriented triangle of $\mathscr{T}$ then the volume V is given by

$$V = \frac{1}{6} \sum_{\Delta \in \mathscr{T}} \begin{vmatrix} 1 & 1 & 1 & 1 \\ \vec{o} & \vec{a}_\Delta & \vec{b}_\Delta & \vec{c}_\Delta \end{vmatrix},$$

where $\vec{o}$ is an arbitrary fixed reference point.

Because of the short range of interaction, only boxes in the direct neighborhood of a moving box constitute to its intersection volume and penetration depth. That is why we use a uniform grid as a space partitioning scheme to locate all potentially interfering boxes and penetrated boundary triangles. The relatively expensive computation of the intersection volume for two boxes is only executed if the fast disjointness test (based on the separating axes theorem) fails. In this way we succeed in performing thousands of trial moves per second in the Monte Carlo simulation.

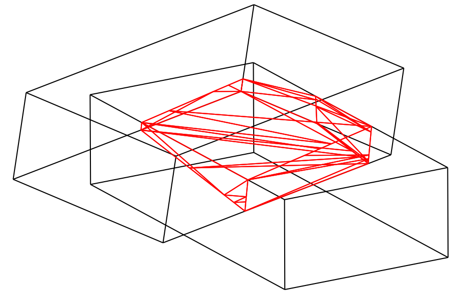

Figure 11: Intersection figure by iterative clipping

4 Simulated Annealing – The Efficient Implementation

This section is dedicated to our extensions and modifications of the simulated annealing approach. At the beginning, we take a combinatorial solution from the discrete model to eliminate the heating process, i.e. we avoid to equilibrate the system at a very high temperature leading to strongly disordered configurations. This issue is described in Sect. 4.5 in more detail.

In our algorithm, we iteratively apply a sequence consisting of

- a special creation procedure for new boxes,
- a relaxation period by a Monte Carlo simulation, and
- a randomly triggered destruction of the "worst" box.

Before we explain the special creation procedure and the destruction of boxes in Sect. 4.4, we give some implementation details that are necessary for an efficient simulation. During the relaxation period, we select with probability of $\frac{1}{2}$ between translational and rotational moves.

4.1 Translational Trial Moves

Let $\Delta_x, \Delta_y, \Delta_z$ be three parameters defining the range

$$R = (-\Delta_x, \Delta_x) \times (-\Delta_y, \Delta_y) \times (-\Delta_z, \Delta_z)$$

from which the displacements $\vec{t} \neq 0$ for the trial moves are picked uniformly at random. We do nothing in the stationary case because it gets trivially accepted and does not give any useful information. The potential change ΔU is given by

$$\Delta U = U(\vec{r} + \vec{t}) - U(\vec{r}),$$

where $\vec{r}$ is the old position and $\vec{t} \in R$ the displacement of the box. Now we consider the two cases "Accepted" and "Rejected" separately.

Accepted In the case of an accepted trial move, we update the position of the box and increase the parameters Δ_x, Δ_y and Δ_z by multiplying by a factor of $1 + |t_x|/|\vec{t}|$, $1 + |t_y|/|\vec{t}|$ and $1 + |t_z|/|\vec{t}|$ respectively.

Rejected If the trial move is rejected, we discard the trial move and decrease the parameters Δ_x, Δ_y and Δ_z by division by a factor of $1 + |t_x|/|\vec{t}|$, $1 + |t_y|/|\vec{t}|$ and $1 + |t_z|/|\vec{t}|$ respectively.

4.2 Rotational Trial Moves

We choose quaternions as representation for orientations and rotations instead of the Eulerian angles θ, φ, ψ because they easily allow choosing rotations uniformly at random. For this purpose, we consider the interpretation

$$q = (\cos\vartheta, \vec{u} \cdot \sin\vartheta)$$

which is a rotation by an angle 2ϑ about an axis represented by the unit vector $\vec{u}$.

We pick a point uniformly distributed in the three dimensional unit ball by choosing three independent uniformly distributed random numbers $p_1, p_2, p_3 \in [0,1)$ and rejecting every triple with $p^2 := p_1^2 + p_2^2 + p_3^2 \geq 1$. Then we generate a quaternion

$$\delta q = \frac{1}{\sqrt{1+p^2}}(1, p_1, p_2, p_3)$$

which defines a rotation about a uniformly distributed axis. Whereas, $\vartheta \in [0, \frac{\pi}{4})$ covers rotations between zero and $2 \cdot 45 = 90$ degrees. By scaling the vector $\vec{p}$ we can control the magnitude of the rotation, too.

Given a trial rotation $\delta q \in \mathbb{R}^4$, the potential change ΔU is given by

$$\Delta U = U(q \cdot \delta q) - U(q),$$

where $q \in \mathbb{R}^4$ is the old orientation. Similar to the case of translational moves we maintain parameters $\Delta_1, \Delta_2, \Delta_3$ to scale the components of $\vec{p}$. Thereby, we effectively choose the rotational axis out of an ellipsoid instead of a ball.

4.3 Prevention of Boxes from Escaping

As we have seen in Sect. 2.2, the penetration depth p does not suffice to model the wall potential. We are interested in the restricted penetration depth p_R, but cannot compute this value due to the deficiencies of the representation of the trunk. Actually, it suffices to distinguish the cases $p = p_R \leq \frac{1}{2} l_{min}$ and $p < \frac{1}{2} l_{min} < p_R$ in Fig. 6 where l_{min} is the shortest side length of a box. In the following we describe an approach to overcome this problem.

The goal is to develop a predicate called `outside`$(\vec{c})$, that reflects the position of a box with center $\vec{c}$ with respect to the trunk. The predicate should return `false` for boxes completely contained in the trunk or for boxes with small restricted penetration depth. If the restricted penetration depth exceeds a certain threshold, the predicate should return `true`. This predicate is used to distinguish both cases in the definition of the wall potential U_W.

We use a three-dimensional grid that segments the bounding box of the trunk into cells. The purpose of the grid is to approximate the space with respect to the trunk. Each grid cell belongs to exactly one of three sets, namely *interior*, *boundary* and *exterior cells*. The spacing d of the grid is discussed later.

The set of boundary cells can be easily determined by computing the intersections between grid cells and the triangular mesh of the trunk. Next we want to identify the interior and exterior cells. We compute connected components of grid cells not yet identified as boundary cells. For this computation, cells are viewed as nodes of a graph and cells next to each other are handled as adjacent nodes.

In order to ensure that regions in the interior and exterior of the trunk do not end up in the same component, we demand that the holes in the triangular mesh do not exceed a rectangle of size $d \times d$. All cells of the connected component(s) that contain(s) the outmost layer of the grid cells clearly belong to the outside (or boundary) and are marked as such.

Since our model is not a manifold we might still have two or more components not yet assigned to one of the tree sets. Therefore we require the user to specify one point in the interior of the trunk. The cells in the corresponding component are marked as inside, all other remaining components (if any) are marked as outside.

Given this data structure, we implement the predicate `outside`$(\vec{c})$ as follows. We return `true` if the center $\vec{c}$ is not contained in the bounding box. Otherwise, we look up the grid cell corresponding to the query point $\vec{c}$ and return `true` iff the cell is marked as boundary or outside.

What remains to be discussed is the choice of the spacing parameter d and its influence on the predicate with respect to the restricted penetration depth of a box. This relation is established by the following proposition.

Proposition 1. *Given a grid with spacing $d > 0$ and a box with center $\vec{c}$ and minimum side length l_{min}. If `outside`$(\vec{c})$ returns `true`, we have for the the restricted penetration depth p_R*

$$p_R \geq \frac{1}{2} l_{min} - d\sqrt{3}.$$

Proof. Consider the case that the cell corresponding to the center $\vec{c}$ is marked as boundary (see Fig. 12). The distance between $\vec{c}$ and the boundary is at most the diagonal of the cell, which is $d\sqrt{3}$. Thus the restricted penetration depth is at least $\frac{1}{2} l_{min} - d\sqrt{3}$.

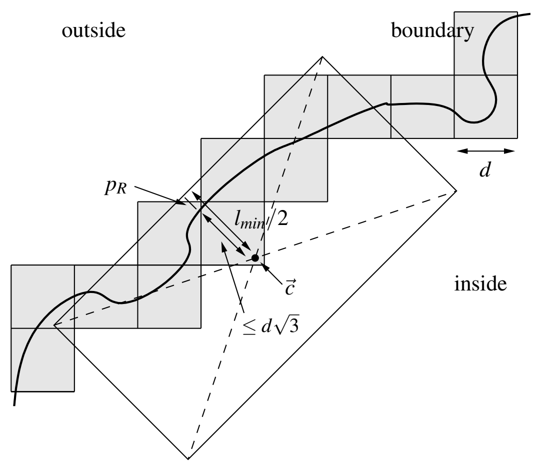

Figure 12: Box center lies in boundary cell

Now consider the case that the cell of the center $\vec{c}$ is marked as outside (see Fig. 13). The restricted penetration depth p_R is greater or equal than the distance of the center $\vec{c}$ to the boundary of the box, hence $p_R \geq \frac{1}{2} l_{min}$.

The case that the center $\vec{c}$ is not contained in the bounding box can be handled as the second case for a grid augmented with one additional outmost layer of cubes marked as outside. □

The contraposition yields that our predicate reports `false` for $p_R < \frac{1}{2} l_{min} - d\sqrt{3}$. Additionally, the predicate returns `true` for $p_R > \frac{1}{2} l_{min}$ (since the center of such a box lies in a cell marked as boundary or outside). Hence the spacing parameter d adjusts the interval of “uncertainty” in which the result of the predicate is not

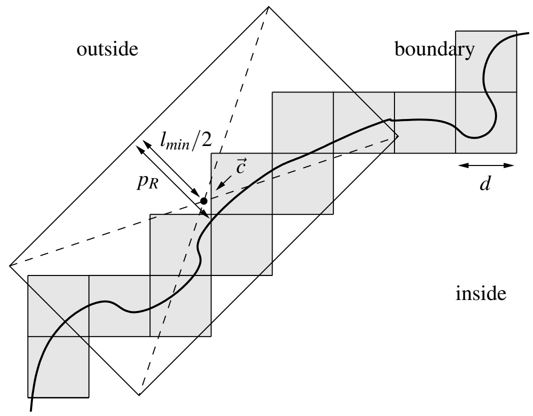

Figure 13: Box center lies in outside cell

solely related to the relative position of the box and the trunk, but also to the alignment and orientation of the grid cells. The smaller the parameter d, the smaller this interval.

On the other hand, one has to take into account the complexity of the grid which scales with d^{-3}. Another reason that prevents arbitrary small values of d is the requirement that the triangular mesh may not contain holes larger than a rectangle of size $d \times d$.

4.4 Creation and Destruction of Boxes

As of yet, we have only described the simulation of ensembles with a constant number of boxes. We have not yet explained how to solve the maximization problem. The missing thing is the creation of boxes and the destruction of prematurely created ones. From a physical point of view, the simulation of a grand canonical ensemble, i.e. an ensemble with creation and destruction of particles, would fix this issue.

In many physical systems, we have particles entering and leaving the region of interest depending on external conditions. The dependency is expressed in the so called chemical potential, that briefly speaking describes the energy that is necessary to add a random particle to the system or the change in the energy if a random particle leaves. Therefore, the higher the chemical potential is, the smaller is the probability of accepting the creation of particle at a random position. The interested reader may have a look at Chapter 5.6 of [Frenkel et al. 2001] for the physical background. Intuitively, the chemical potential increases with the density of the particles in the system.

However, it would be physically nearly impossible that the simulation of such a grand canonical ensemble comes up with a valid optimal solution. Since we deal with close packings, i.e. packings with a very high density, the chemical potential for such a setting would be very large, and hence the probability to create a new box would be vanishing low, or on the other hand, the destruction of boxes would occur too often.

Nevertheless, we can modify the creation procedure so that it finds promising positions and orientations for new boxes and inserts them there. An example for such a promising position can be seen in Fig. 14. Since we may allow temporarily greater penetrations of the trunk than the threshold in the Simulated Annealing approach, positions for new boxes may exist due to missing cubes in the discretization which is constrained by theses thresholds. Furthermore, promising positions may originate during the simulation, because a nearby box has been destructed or rearranged.

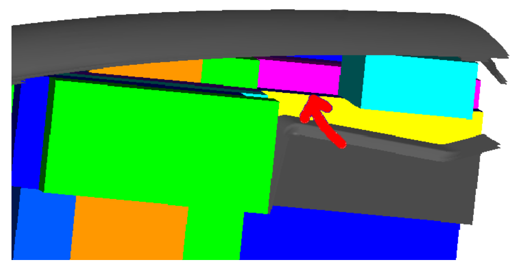

Figure 14: Promising position

Creation In each iteration step we pick a box uniformly at random and investigate its surrounding for promising positions of new boxes. More precisely, we construct six candidate boxes and evaluate their contribution to the potential function.

Each candidate for a new box is obtained by mirroring the selected box at one of the six planes defining its boundary. Candidates outside the trunk as decided by our predicate `outside` are rejected. We also reject candidates that exceed thresholds for intersection volume, interpenetration depth and penetration depth. Remaining candidates (if any) are accepted as new boxes.

Next we describe an additional step modifying our approach above. Consider a candidate that has been obtained by mirroring the selected box at a plane that contains one of its faces of size 200mm $\times$100mm. Imagine an obstacle that interferes with the candidate just slightly below our threshold. Consequently, the candidate would be created. But if the subsequent relaxation step fails to improve the situation, the just created box is likely to be destroyed again. The heuristics would cycle between creation and destruction of a box at that position.

Therefore we rotate the candidate and the original box by 90 degrees such that they cover the same space together, but changed their orientation. By triggering the rotation randomly, we increased the chance that a box may escape out of this situation during the relaxation. A similar rotation is performed for candidates that originate from mirroring the selected box at the planes that contain the faces of size 200mm $\times$ 50mm. No such rotation is possible for the remaining case with respect to the face of size 100mm $\times$ 50mm. Candidates of this case are seldom accepted anyhow.

Destruction In each iteration step we consider the worst box, i.e. the box that contributes most to the potential. We simply remove the box with a probability proportional to its contribution to the potential.

4.5 Eliminating the Heating Process

The standard simulated annealing approach requires that the system is brought into equilibrium with a sufficiently hight temperature at the beginning. Thereby, one can guarantee the convergence of the method assuming a decreasing temperature which is logarithmic in time [Nolte and Schrader 1996].

Since this implies an exponential running time, one has to use faster schedules, e.g. a geometric one. But thereby, we experienced lots of canting of the boxes like depicted in Fig. 15.

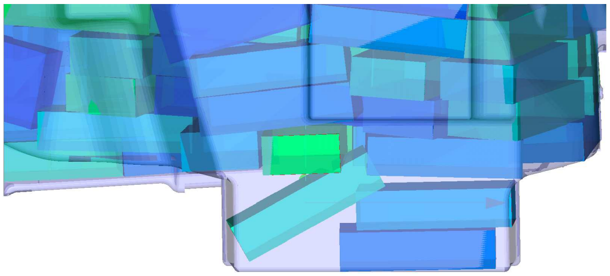

Figure 15: Canting of boxes

Therefore, we came up with the idea to use a combinatorial solution as starting configuration in connection with our creation procedure. In the following, we give a model of the configuration space that explains the success of our method.

Lets assume that we only have uniform objects to pack like in our real world problem. Therefore, the potential does not change by swapping the identity of two boxes. Hence, it is sufficient to consider only one representative configuration $x = (s_1, \ldots, s_n)$ for all its $n!$ permutations where n is the number of boxes and s_i the set of parameters that describe one box.

If we consider the states s_i, s_j of two boxes in a common subspace, then their "distance" cannot be arbitrarily close in a valid packing. Therefore, we may assume that each state occupies a certain volume of the configuration space. Since we are interested in a maximal packing, i.e. a packing with a high density of states, an optimal solution would be a close packing.

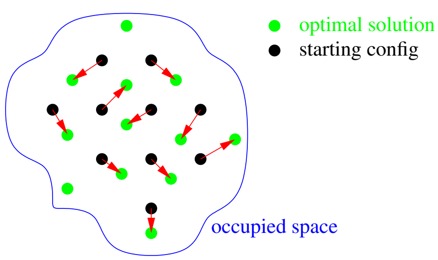

Figure 16: Model of states in the configuration space

Our heuristics consists of choosing a starting configuration in a way that yields a closest packing because thereby we can map each state of the starting configuration to one state of an optimal solution such that each distance is bounded. Therefore, an appropriate starting configuration would be a combinatorial solution to the discrete model with the additional constraint that it is dense in its interior.

Since the starting configuration is "near" an optimal solution but does not match exactly, we have an additional requirement that the combinatorial solution should be "flexible" enough, i.e. rather homogeneous in the orientation of the boxes.

Note that we still follow a cooling schedule in the course of our algorithm. But by using the combinatorial solution, we are allowed to start at a lower temperature that does not lead to a randomization of the whole starting configuration and the related canting problems.

5 Project Overview and Evaluation

The complete project has been implemented in C++ and is portable across today's major system platforms. Several external libraries have been used supporting the implementation of the user interface, the visualization part as well as providing some low-level algorithms and datastructures. Since our system is employed in an industrial production environment, ease-of-use has been a major requirement of our industrial partner.

5.1 Evaluation

Here we briefly report on the results we achieved with our approach of the *Specialized Grand Canonical Simulated Annealing* heuristic.

The plain results are summarized in Tab. 1 where we oppose the values achieved by a human expert, our best combinatoric solutions, and the novel approach which is abbreviated by SGCSA. Just to get an impression, we also added an approximate value for the continuous volume in the last column. This value is the average of an inner and outer approximation by a cubic grid with a spacing of 6.25 mm or 12.5 mm and has been rounded to a precision of 5 liters.

A, B, C, and D denote different trunk types, 'C w/ extras' the same trunk as C but with a reduced volume due to control units for additional equipment like air-conditioning.

	expert	combinatoric	SGCSA	continuous
A	62 l	61 l	64 l	≈95 l
B	80 l	80 l	81 l	≈120 l
C	513 l	507 l	510 l	≈595 l
C w/ extras	480 l	475 l	479 l	≈555 l
D	499 l	491 l	493 l	≈595 l

Table 1: Some test cases

Recall that the quality restrictions of our industrial partner require us to achieve least 99% of their best manual packing and not more than 10 liters less. One can see in Tab. 1 that for models *A* and *B*, our implementation even outperforms the expert's solution. All solutions were computed within a time-frame of one day allowed by our industrial partner; since no user interaction is required after initializing the optimization process, this fits very well into the design process.

6 Conclusion

We have presented an industrial-strength system which enables car manufacturers to estimate reliably the volume of car trunks even at an early stage of the design process. Compared to the previous system presented in [Eisenbrand et al. 2003], the main novelty is the possibility to place boxes in arbitrary orientations and positions. The lack of the latter has shown to be a weakness of the old system when it was evaluated in the actual production environment. The new system has been certified using a large number of different trunk types and proven to be of industrial strength. It is currently being installed for use in the actual design process of the car manufacturer.

References

BAUR, C., AND FEKETE, S. P. 2001. Approximation of geometric dispersion problems. *Algorithmica 30*, 3, 451–470. Approximation algorithms for combinatorial optimization problems.

CHAN, T. M. 2003. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms 46*, 2, 178–189.

DEMAINE, E., MITCHELL, J., AND O'ROURKE, J. 2005. *The Open Problems Project, problem 55* Pallet loading. http://maven.smith.edu/∼orourke/TOPP/P55.html.

DEMAINE, E., MITCHELL, J., AND O'ROURKE, J. 2005. *The Open Problems Project, problem 56* Packing Unit Squares in a Polygon. http://maven.smith.edu/∼orourke/TOPP/P56.html.

DING, Q., AND CAGAN, J. 2003. Trunk packing with extended pattern search. In *Virtual Engineering, Simulation & Optimization - from the SAE 2003 World Congress*.

DYCKHOFF, H. 1990. A typology of cutting and packing problems. *European Journal of Operational Research 44*, 145–159.

EISENBRAND, F., FUNKE, S., REICHEL, J., AND SCHÖMER, E. 2003. Packing a trunk. In *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 2003*, 618–629.

ERDOS, P., AND GRAHAM, R. L. 1975. On packing squares with equal squares. *Journal of Combinatorial Theory 19*, 119–123.

FOWLER, R. F., PATERSON, M. S., AND TANIMOTO, S. L. 1981. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters 12*, 133–137.

FRENKEL, D., FRENKEL, D., AND SMIT, B. 2001. *Understanding Molecular Simulation*. Academic Press, Inc.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. OBBtree: A hierarchical structure for rapid interference detection. *Computer Graphics* (Aug.), 171–180. Proc. SIGGRAPH'96.

HOCHBAUM, D. S., AND MAASS, W. 1985. Approximation schemes for covering and packing problems in image processing and vlsi. *J. ACM 32*, 1, 130–136.

IKONEN, I., BILES, W. E., KUMAR, A., WISSEL, J. C., AND RAGADE, R. 1997. Genetic algorithm for packing three-dimensional non-convex obje cts having cavities and holes. In *Proceedings of the 7th International Conference on Genetic Algo rtithms*, 591–598.

KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. 1983. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983 220, 4598*, 671–680.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equation of state calculation by fast computing machines. *Journal of Chemical Physics 21*, 1087–1092.

NELISSEN, J., 1993. New approaches to the pallet loading problem.

NOLTE, A., AND SCHRADER, R. 1996. A note on the finite time behaviour of simulated annealing. In *Operations Research Proceedings*, Springer, U. Zimmermann, U. Derigs, W. Gaul, R. Möhring, and K.-P. Schuster, Eds., 175–180.