Scandinavian Thins on Top of Cake: New and Improved Algorithms for Stacking and Packing

Helmut Alt* Esther M. Arkin† Alon Efrat‡ George Hart§ Ferran Hurtado¶
Irina Kostitsyna‖ Alexander Kröller** Joseph S. B. Mitchell†
Valentin Polishchuk††

Abstract

We show how to compute the smallest rectangle that can enclose any polygon, from a given set of polygons, in nearly linear time; we also present a PTAS for the problem, as well as a linear-time algorithm for the case when the polygons are rectangles themselves. We prove that finding a smallest convex polygon that encloses any of the given polygons is NP-hard, and give a PTAS for minimizing the perimeter of the convex enclosure. We also give efficient algorithms to find the smallest rectangle simultaneously enclosing a given pair of convex polygons.

It's a piece of cake!

Roald Dahl, The Wonderful Story of Henry Sugar, and Six More

1 Introduction

In the Nordic countries, gingerbread cookies are among traditional Christmas associations – along with singing Jingle Bells, expecting Santa Claus, preparing FUN submissions, and many other classical New-Year's Eve activities. Scandinavian biscuits are especially thin (hence the name – gingerbread *thins*), and are cut into many more different shapes than just the famous shapes of the Gingerbread Man and his family [1, 45]. The thins are often placed as decoration on the top of a cake, and the cake is cut into equal rectangular pieces, with each piece containing exactly one thin. The cake is fatty, so to fulfill dietary restrictions, each piece has to be as small as possible.

The above considerations crystallize into the following algorithmic problem:

Given a set of simple polygons (the thins), find a minimum-area rectangle (a piece of cake) such that each polygon individually can fit into the rectangle after an appropriate translation and rotation. (P0)

*Institut für Informatik, Freie Universität Berlin. alt@inf.fu-berlin.de
†Applied Mathematics & Statistics, Stony Brook University. {estie,jsbm}@ams.stonybrook.edu
‡Computer Science, the University of Arizona. alon@cs.arizona.edu
§george@georgehart.com
¶Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya. Ferran.Hurtado@upc.edu
‖Computer Science, Stony Brook University. ikost@cs.stonybrook.edu
**Computer Science, Technische Universität Braunschweig. a.kroeller@tu-bs.de
††Helsinki Institute for IT, Computer Science, University of Helsinki. polishch@cs.helsinki.fi

Thus, we are looking for a minimum-area rectangle that can enclose any *one* of a given set of polygons. In other words, our problem is to rotate and translate a set of polygons to minimize the area of the minimum enclosing rectangle of their union.

Further motivation Apart from limiting fat intake, one may also be interested in the above problem from the point of view of the caterer who cooks (and cuts) the cake – the caterer has to prepare the ordered number of cake pieces while using as little of the cake area as possible. Another aspect of the thins business where our problem arises is in packaging: if the manufacturer (such as ANNA's [2]) wants to stack different thins in a rectangular box, the problem of minimizing the size of the (cross-section of the) box is exactly our problem.

The problem shows up in many other applications too. Cardboard templates used for patchwork sewing are simple polygons of various shapes; finding the smallest box to store them is our problem. Knowing the (cross-section) of all your guests, you may want to find the dimensions of the smallest door through which all guests may enter your house. (Or knowing all your cats, you, just as Newton [3], may be interested in designing the smallest cat flap.) A photographer may be interested in computing the smallest rectangle that blanks out any given face on a photo. When printing the US atlas (with one state per page, and allowing rotation of states) one may want to minimize the maximum scale of the maps. A capsule hotel owner may want to maximize the number of rooms (equivalently – minimize the area of a room) under the restriction that any traveler fits into the room (assuming the owner knows the measurements/shapes of all travelers). The problem is also interesting in 3D, where one may be interested in questions like what is the smallest-volume capsule into which every astronaut fits, what is the smallest coffin size, how large should a box be so that any one of a set of items that a customer may choose will fit within it, etc.

1.1 Notation

Let $I_1 \dots I_K$ denote the given set of polygons; we will call them *items* to emphasize that they may be of arbitrary nature. Let n_k be the number of vertices of I_k, $k = 1 \dots K$, and let $n = n_1 + \dots + n_K$. We will call the sought enclosing rectangle the *suitcase*. Since a suitcase encloses an item if and only if the suitcase encloses the convex hull of the item, and since the convex hull of a simple polygon can be found in linear time, we will assume that all items are convex. For simplicity we assume that there are no degeneracies: no item has two parallel sides, etc. We also assume that none of the items fits inside another (while the fastest algorithms to test polygon containment take superlinear time [31], our algorithms do not use the assumption; we make it only to facilitate the exposition).

With the above, the problem (P0) can be restated as:

> *Given a set of convex items, find a minimum-area rectangular suitcase such that each item individually can fit into the suitcase after an appropriate translation and rotation.* (P1)

1.2 Convex suitcase

We also study the version of the problem in which the enclosure can be an arbitrary convex polygon (not necessarily a rectangle). Such "shape abstraction" problems can be relevant in machine learning and graphics: imagine that we are given a collection of chairs and a collection of tables, and our task is to summarize how a generic chair looks and how a generic table looks; the minimum enclosure can give an initial such summary (of course, chairs and tables would be more accurately described by non-convex shapes, but a convex polygon may suffice for a first approximation). Formally the problem is:

Given a set of convex items, find a minimum-area convex suitcase such that each item individually can fit into the suitcase after an appropriate translation and rotation. (P2)

1.3 Translations only

The formulation of our basic problem (P1) allowed items to be both translated and rotated in order to find the best fit. (One may also allow items to be flipped; this will not change the problem since an item fits into a suitcase if and only if the item's reflection does.) Equivalently, in problem (P1) the suitcase was allowed to translate and rotate in order to cover different items. We will also consider the variant in which items can be only translated, but rotation of items is forbidden. This translation-only setting is relevant for applications in which the orientation of items must be fixed, e.g., for aesthetic reasons.

Depending on whether the suitcase's orientation must also be fixed or not, we arrive at two versions of the problem:

Given a set of convex items and an orientation for the suitcase, find a minimum-area rectangular suitcase with the given orientation such that each item individually can fit into the suitcase after an appropriate translation. (P3)

Given a set of convex items, find a minimum-area rectangular suitcase such that each item individually can fit into the suitcase after an appropriate translation. That is, the suitcase may have arbitrary orientation, but it is only allowed to translate in order to cover different items; rotation of the suitcase from item to item is forbidden. (P4)

1.4 Packing

The problems we defined above are different from *disjoint packing* problems [4, 32, 48, 63]: we required that the suitcase encloses *any* of the items *one at a time* (or equivalently, encloses all items "stacked" on top of each other); in a packing problem *all* items have to be enclosed by a suitcase *simultaneously* without overlap. Packing is hard when the number of items K is not fixed; even the simplest case of packing translates of axis-aligned rectangular items into an axis-aligned rectangular suitcase is weakly NP-hard, as can be seen by a simple reduction from PARTITION [32, Lemma 2.2].

In this paper we will consider packing two convex items into rectangular suitcases ($K = 2$). As with the "stacking" problems (P1), (P3), (P4), we will consider three versions: (P5) when the items are allowed both to translate and to rotate (the orientation of the suitcase does not matter), (P6) when the items are only allowed to translate and the orientation of the suitcase is specified, and (P7) when the items are only allowed to translate, but we are free to choose orientation of the suitcase:

Given two convex items, find a minimum-area rectangular suitcase such that the items can fit simultaneously, without overlap into the suitcase after an appropriate translation and rotation. (P5)

Given two convex items and an orientation for the suitcase, find a minimum-area rectangular suitcase with the given orientation such that the items can fit simultaneously, without overlap into the suitcase after an appropriate translation. (P6)

Given two convex items, find a minimum-area rectangular suitcase such that the items can simultaneously, without overlap fit into the suitcase after an appropriate translation. The suitcase may have arbitrary orientation. (P7)

1.5 Our Contributions

We give exact and approximation algorithms for the problems (P1) – (P7). To our knowledge, most of the problems we consider have not been studied before by any other researchers.[1] Problem (P2) was studied recently for the special case $K = 2$ [11]; we consider the general case of arbitrary K (Section 3). On the contrary, for problem (P6), algorithms for general K are known [19, 47]; we improve their running times for the special case $K = 2$ (Section 4.2). See Section 1.6 for in-depth survey of earlier work.

Our main results are summarized below. All the results, except for the convex suitcase PTAS, are presented in terms of minimizing the area of the suitcase, but extend straightforwardly to perimeter minimization.

- In Section 2 we consider stacking convex items into a rectangular suitcase when both translation and rotation is allowed (problem (P1)). In Section 2.1 we give an $O(n(2^{\alpha(n)} \log K\ \alpha(K) + \log(n)\ \alpha(n)))$-time algorithm for the problem; here $\alpha(\cdot)$ is the extremely slowly growing inverse Ackermann function. Section 2.2 presents a simpler algorithm, Section 2.3 presents a linear-time algorithm for rectangular items, and Section 2.4 presents a near-linear-time PTAS; the PTAS extends to 3D.

- In Section 2.5 we study the translation-only versions of stacking convex items into rectangular suitcases: problem (P3), with a given suitcase orientation, and problem (P4), when the suitcase orientation can be chosen. We give $O(n)$- and $O(n \log K\alpha(K))$-time, respectively, algorithms for the problems.

- In Section 3 we consider finding a smallest convex (i.e., not necessarily rectangular) suitcase (problem (P2)). We show that the minimum-area problem is NP-hard, and give a PTAS for minimizing the perimeter of the suitcase.

- In Section 4 we study disjoint packing of two convex items into rectangular suitcases (problems (P5), (P6), (P7)). For packing under arbitrary rigid motions (problem (P5) we give a cubic-time algorithm (Section 4.1). When only translations of items are allowed we can find a smallest suitcase with a given orientation (problem (P6)) and the smallest suitcase with arbitrary orientation (problem (P7)) in linear time (Sections 4.2 and 4.3).

1.6 Related work

Computing simple enclosing shapes is a fundamental problem in data simplification. It appears in various domains ranging from computer graphics to pattern recognition to robotics. The enclosure problem has a rich history of research [18, 25, 26, 29, 30, 35, 36, 49, 50, 54, 55, 59].

Our rectangular-suitcase problems are extensions of the *minimum enclosing rectangle* problem. Optimal algorithms for finding a smallest enclosing rectangle for *one* item are due to Toussaint [59]. The "rotating calipers" paradigm that he developed has subsequently been used for many optimal enclosure problems [23, 24, 46].

O'Rourke [53] presented a cubic-time algorithm to find a minimum enclosing box in 3D. Becker et al. [22] gave algorithms for enclosing a set of polygons by two rectangles. Yildirim [64] showed how to find a smallest ellipsoid enclosing a set of ellipsoids. In a series of works [10, 12, 33, 39], optimizing the overlap of *two* objects (both in 2D and 3D) under a rigid motion was studied. Löffler and van Kreveld [43, 60] considered moving points to optimize certain measures of a point set. Martin and Stephenson [44] developed algorithms for *testing* (both in 2D and 3D) whether an object fits into a box (we use their result in one of our algorithms). Chazelle [31] gave an

[1] We reported our results in the conference versions [15, 17].

algorithm to decide whether one convex polygon fits into another. Other papers dealing with polygon containment include [16, 20, 21, 58]; applications in manufacturing, cartography and image processing are described in [41, 61, 62], respectively.

Convex suitcase Finding a minimum convex enclosure for stacked items (our problem (P2)) was studied by Ahn and Cheong [11] for the special case of two items ($K = 2$). They gave exact and approximation algorithms for various versions of the problem; for our problem (P2) (minimizing the area of the convex hull of the union, while allowing overlap), [11] gives a PTAS. In Section 3 we show that for general K the problem is NP-hard and give a PTAS for perimeter minimization.

In higher dimensions, Ahn et al. [8] considered the case of $K = 2$ convex items that are only allowed to translate; they show how to find a minimum-volume convex suitcase in expected time $O(n^{d+1-3/d} \log^{d+1} n)$ for dimension $d \geq 3$. When items are segments in the plane and only translations are allowed, a minimum-area convex suitcase can be found in $O(n \log n)$ time and a minimum-perimeter convex suitcase in $O(n)$ time [7]. Somewhat related to our problem are works on shape matching, where maximizing the overlap (or equivalently, minimizing the symmetric difference) of two items under translations [9, 14, 33, 51] or general rigid motions [12, 13, 27] has been considered.

Packing As mentioned above, disjoint packing problems are NP-hard when the number of items is not fixed [32]. On the positive side, for packing under translations considerable work has been done by Milenkovic (e.g., see [47]), motivated by industrial applications in which one is to minimize waste when cutting parts of specified shapes out of rectangular stock material, such cloth or sheet metal. In particular, for packing K convex items into a minimum-area fixed-orientation rectangle, he gives an $O(n^{K-1} \log n)$ algorithm based on linear programming techniques. The most general problem for disjoint packing under translations was solved by Avnaim and Boissonnat [19], who presented an $O((m^2 + mn)^{2K} \log n)$ algorithm for packing K arbitrary simple n-gons into a simple m-gon. For our special case of $K = 2, m = 4$ (problem (P6)), the algorithms of [19] and [47] run in $O(n^4 \log n)$ and $O(n \log n)$ time, respectively; we present a linear-time algorithm (Section 4.2).

Ahn and Cheong considered packing $K = 2$ convex items into a convex suitcase; Egeblad et al. [37] studied translational packing in higher dimensions. Disk packing was considered in Sugihara et al. [57]; see [40, 52] for combinatorial bounds. For a further packaging problem—packing the cake into a box—see [56]. The packing problem is also related to coordinated motion planning.

2 Stacking Items into Rectangular Suitcases

In this section we consider the problems of stacking items in a rectangular suitcase under general rigid motions (problem (P1)) and under translations (problems (P3), (P4)). Our approach is as follows. Using a pair of orthogonal rotating calipers, build, for every item, its (minimum) width as a function of its length. Overlay the graphs of the functions for all items, and find the upper envelope of the graph. A point of the envelope minimizing the objective function yields an optimal solution. The remainder of the section provides the details.

We start with a lemma on lower envelopes of a set of functions. Recall that the lower envelope of m x-monotone curves that pairwise intersect at most s times has $O(\lambda_{s+2}(m))$ complexity and can be built in $O(\lambda_{s+1}(m) \log m)$ time [42], where $\lambda_s(m)$ is the maximum length of an m-element order-s *Davenport-Schinzel (DS) sequence* (see [6, Sections 1 and 2] for definitions and notation relevant to DS sequences, and their algorithmic and combinatorial properties). Let $f_1 \dots f_K$ be

x-monotone curves composed of a total of m arcs such that any two arcs intersect at most s times.

Lemma 2.1. *The complexity of the lower envelope of $f_1 \dots f_K$ is $O((m/K)\ \lambda_{s+2}(K))$. The envelope can be built in time $O((m/K)\ \lambda_{s+1}(K) \log K)$.*

Proof. Let $x_0, x_1, ..., x_m$ be endpoints of the arcs in ascending x-order. Let $r = \lfloor m/K \rfloor + 1$, and consider $r+1$ intervals $[x_0, x_K], [x_{K+1}, x_{2K}], \dots, [x_{rK+1}, x_m]$. Within each interval there are at most $2K$ arcs (since there are at most K arcs at the left end of an interval, and at most K arcs start within the interval). Therefore the lower envelope of $f_1 \dots f_K$ within each interval has complexity $O(\lambda_{s+2}(2K)) = O(\lambda_{s+2}(K))$ and can be computed in time $O(\lambda_{s+1}(K) \log K)$. □

2.1 Arbitrary rigid motions

Let opt denote an optimal solution to problem (P1). The length and the width of an abstract rectangle will be denoted by l and w, respectively. We call an $l \times w$ rectangle an (l, w)-rectangle or an (l, w)-suitcase. We identify the rectangle with the point (l, w) in the lw-plane.

Width functions: $w(l) \to w_I^{\square}(l) \to w_I^{\min}(l) \to w^*(l)$

Let I be an item (a convex polygon), and let $\vec{v}$ be a direction. The *width* of I in direction $\vec{v}$, denoted $w_I(\vec{v})$, is the length of the projection of I on a line having direction $\vec{v}$. Define the *length* of I in direction $\vec{v}$, $l_I(\vec{v})$, to be the width of I in the direction orthogonal to $\vec{v}$. We say that I is *tight* in a rectangle R if each side of R contains a vertex of I.

Consider, as in [59, Section 2], two pairs of rotating calipers orthogonal to each other (thus, I is tight in the rectangle formed by the calipers). While the pairs are supported by a fixed quadruple of vertices of I,

$$w_I(\varphi) = d_2 \cos(\theta_0 - \varphi) \quad , \quad l_I(\varphi) = d_1 \cos\varphi \tag{1}$$

where φ is the angle of rotation of the calipers, and d_1, d_2, θ_0 are some constants, which can be found in constant time as soon as the vertices on which the calipers rock are given (Fig. 1). The equations (1) are the parametric equations of an elliptic arc in the (l, w)-plane. We call such an arc, obtained by rotating calipers to the maximal extent while maintaining contact with the same vertices of I, an *elementary arc*.

We define $w_I^{\square}(l) = \min_{\vec{v}: l_I(\vec{v}) = l} w_I(\vec{v})$ as the function whose graph is the lower envelope of the elementary arcs (see Fig. 1, right). One can view $w_I^{\square}(l)$ as the minimum width of I provided I has length l *and provided I is tight in an (l, w)-rectangle*. Note that the elementary arcs form a cycle (in the graph-theoretic sense): one can traverse them in the order as they appear as I is rotated. This implies that the domain of $w_I^{\square}(l)$, $\operatorname{Dom} w_I^{\square}$, is a contiguous subset (a segment) of the l-axis. Also note that an elementary arc does not necessarily represent the graph of a function $l(w)$ (as it is shown, for simplicity only, in Fig. 1, right) since an arc may intersect a vertical line in two points; still, the lower envelope of arcs *is* a function.

Next, we define $w_I^{\min}(l) = \min_{l' \le l} w_I^{\square}(l')$ as the minimum width w such that I fits into an (l, w)-suitcase for a given l; for consistency we assume that $w_I^{\min}(l) = +\infty$ for l to the left of $\operatorname{Dom} w_I^{\square}$, and $w_I^{\min}(l) = \min_{l \in \operatorname{Dom} w_I^{\square}} w_I^{\square}(l) = \text{const}$ to the right of $\operatorname{Dom} w_I^{\square}$. The function $w_I^{\min}$ can be obtained by shooting, from every local minimum of $w_I^{\square}$, a horizontal ray in the positive l direction, until intersecting $w_I^{\square}$ again; the rays, together with the monotonically decreasing portions of $w_I^{\square}$, form $w_I^{\min}$ (see Fig. 1, right). Item I fits into an (l, w)-suitcase if and only if (l, w) is on or above the graph of $w_I^{\min}(l)$. In this sense $w_I^{\min}$ are *Pareto-optimal* suitcases for I.

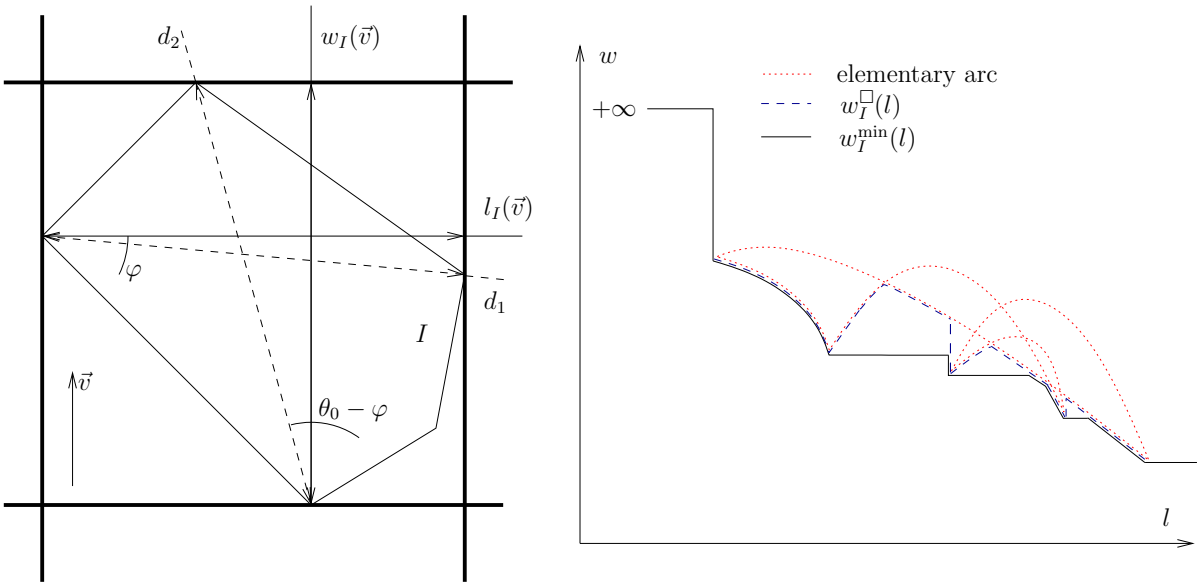


Figure 1: As the pair of calipers (thick lines) rotates around a fixed quadruple of vertices, the width of I is a fixed function of its length; the graph of the function is an elementary arc (the arcs are shown dotted red on the right). $w_I^{\square}(l)$ (dashed blue) is the lower envelope of the arcs; $w_I^{\min}(l)$ (solid black) is the minimum suitcase width necessary to have I fit into a length-l suitcase.

Finally, we define $w^*(l) = \max_{k=1\ldots K} w_{I_k}(l)$ as the maximum of the widths of the items as a function of length. All items fit into an (l, w)-rectangle if and only if (l, w) lies above the graph of $w^*(l)$. That is, the graph of $w^*(l)$ is the set of *Pareto optimal* suitcases; naturally, $w^*(l)$ is non-increasing. A point on the graph of $w^*(l)$ that minimizes wl defines a minimum-area suitcase.

Building $w(l), w_I^{\square}(l), w_I^{\min}(l), w^*(l)$

While in general, two elliptic arcs can intersect 4 times, for our elementary arcs we have:

Lemma 2.2. *Two elementary arcs intersect at most twice.*

Proof. Equations (1) can be rewritten as

$$w_I(\varphi) = d_2 \cos\theta_0 \cos\varphi + d_2 \sin\theta_0 \sin\varphi \quad , \quad l_I(\varphi) = d_1 \cos\varphi \quad .$$

Thus an elementary arc is the image of the arc $\mathcal{A} = \{(\cos(\varphi), \sin(\varphi)) | \varphi \in [0, \varphi_{\max}]\}$, for some $\varphi_{\max}$, under a linear mapping. Since $\mathcal{A}$ is an arc of the unit circle centered at the origin, the elementary arc is an arc of an ellipse centered at the origin. Moreover, $\varphi_{\max} < \pi$ since an item's angles are less than π; thus, the elementary arc is at most half of the ellipse. Two elliptical arcs like that (each at most half of the supporting ellipse, centered at the origin) intersect at most twice, which can be seen from symmetry and the observation that two quarter-arcs of concentric ellipses intersect at most once. □

Let m denote the number of vertices of item I. Every elementary arc obtained when rotating the calipers around I corresponds to a pair of *antipodal* vertices of I [59]. Since there are $O(m)$ antipodal vertices, there are $O(m)$ elementary arcs. By Lemma 2.2, two arcs intersect in at most two points. Thus, the complexity (the number of breakpoints) of $w_I^{\square}(l)$ is $\lambda_4(m)$, and $w_I^{\square}(l)$ can be built in time $O(\lambda_3(m) \log m)$.

By construction of $w_I^{\min}(l)$, its graph consists of pieces of elementary arcs, and horizontal line segments originating at endpoints of elementary arcs; thus, there are $O(m)$ segments. Sweeping the graph of $w_I^{\square}(l)$ with a vertical line, we can build $w_I^{\min}(l)$ from $w_I^{\square}(l)$ in time proportional to the complexity of $w_I^{\square}(l)$. Thus, the complexity of $w_I^{\min}(l)$ is $\lambda_4(m)$, and $w_I^{\min}(l)$ can be built in $O(\lambda_3(m) \log m)$ time.

Finally, $w^*(l)$ is the upper envelope of K functions $w_{I_k}(l)$ (one per item) of total complexity $O(\sum_{k=1}^{K} \lambda_4(n_k)) = O(\lambda_4(n))$. By Lemmas 2.2 and 2.1, the complexity of $w^*(l)$ is $O((\lambda_4(n)/K)\lambda_4(K))$, and $w^*(l)$ can be built in time $O((\lambda_4(n)/K)\lambda_3(K) \log K) = O(n(2^{\alpha(n)} \log K\ \alpha(K) + \log(n)\ \alpha(n)))$.

After $w^*(l)$ is built, we find, for each piece of it, the minimum of $w^*(l) \cdot l$, by computing zeros of the derivative. Overall, we obtain:

Theorem 2.3. *Problem (P1) can be solved in $O(n(2^{\alpha(n)} \log K\ \alpha(K) + \log(n)\ \alpha(n)))$ time.*

2.2 A simpler algorithm

The results in this and the next sections are based on a closer look at the elementary arcs and their connection to "flushness" [59] of the enclosure.

Flushness All known algorithms for enclosure problems (the "standard" enclosure problems, where the goal is to enclose *one* item) employ the fact that an optimal enclosure is *flush* with the enclosed polygon having an edge that contains an edge of the enclosed polygon; the proof can be found, e.g., in [54, Section 3.1.2] or [38]. One way to see this is to observe that a hyperbola $xy = const$ is convex as seen from the origin, while elementary arcs are concave; therefore if one considers the family of hyperbolas $lw = A$ for increasing A, the first time that a hyperbola intersects an elementary arc, the intersection happens at an arc endpoint. We state this fact as a lemma:

Lemma 2.4. *Let A^* be the smallest A for which the hyperbola $lw = A$ intersects an arc; let (l^*, w^*) be a point of intersection. Then (l^*, w^*) is an arc's endpoint.*

We now can give a simpler algorithm for our problem. In contrast to finding the minimum enclosure for one item, in our problem, where we seek to enclose any one of *many* items, the optimum may be attained at an intersection of elementary arcs, i.e., with no item being flush with the optimal suitcase. (If this were not the case, i.e., if opt had to be flush with an item, our problem would have admitted a straightforward solution: guess the flush item and its flush edge, determine the width of the item in the direction perpendicular to the flush edge, and fit the other items into the given width.) Still, after $w^*(l)$ is built, finding a minimum-area suitcase can be seen as finding the smallest number A such that the graph of $lw = A$ intersects the graph of $w^*(l)$. Since $w^*(l)$ is composed of pieces of elementary arcs and horizontal or vertical segments originating at endpoints of the arcs, the intersection can happen either at a breakpoint of $w^*(l)$, or interior to a segment, or interior to an elementary arc. But it is clear that the intersection never happens interior to a segment, and the possibility of an intersection interior to an arc is ruled out by Lemma 2.4. Thus, opt lies at a breakpoint of $w^*(l)$.

Since there are $O(n)$ elementary arcs, there are $O(n^2)$ breakpoints; at each of them we have to test whether all items fit into the suitcase. Testing whether an item I fits into an (l, w)-suitcase can be done by querying $w_I^{\min}(l)$; since $w_I^{\min}(l)$ is monotone, this can be done in time logarithmic in the complexity of $w_I^{\min}(l)$, i.e., in time $O(\log \lambda_4(m)) = O(\log m)$, where m is the number of vertices of I. Testing all K items thus takes $O(\sum_{k=1}^{K} \log n_k) = O(K \log(n/K))$ time.

We thus obtain:

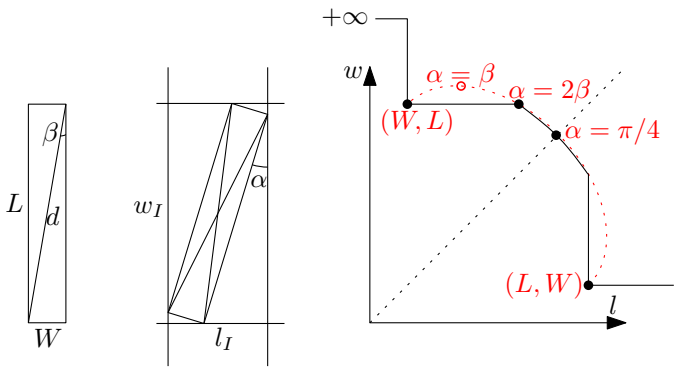


Figure 2: Left: $d = \sqrt{W^2 + L^2}$. Middle: $w_I(\alpha) = d\cos(\alpha - \beta), l_I(\alpha) = d\sin(\alpha + \beta)$. Right: $w(l)$ is dotted red; $w_I^{\min}(l)$ is solid black.

Theorem 2.5. *Problem (P1) can be solved in $O(n^2 K \log(n/K))$ time.*

Although the running time in the above theorem is worse than the running time of the algorithm from the previous section, the solution here has the advantage of being simple and easily implementable.

2.3 Rectangular items: Flushness is Back

As discussed above, in general, an optimal suitcase is not necessarily flush with any of the items. There is one case, however, where the flushness property holds, the case of *rectangular* items.[2] To see this, let us have a closer look at the elementary arcs for a rectangle (Fig. 2).

Because of the symmetry, for a rectangular item I, it is enough to consider only one elementary arc, namely, the one obtained as I is rotated by 90^o (the 3 other arcs, corresponding to the full 360^o rotation, coincide with the arc). In addition, the arc is symmetric with respect to the line $w = l$ in the (l, w)-plane because together with any point $(d\cos(\alpha - \beta), d\sin(\alpha + \beta))$, the arc contains the point $(d\cos(\pi/2 - \alpha - \beta), d\sin(\pi/2 - \alpha + \beta)) = (d\sin(\alpha + \beta), d\cos(\alpha - \beta))$ (the notation is from Fig. 2, where α is the rotation angle, $\beta \le \pi/4$ is the angle between the longer side of I and its diagonal, and d is the diagonal of I). The line is intersected by the arc at $\alpha = \pi/4$ (because $\cos(\pi/4 - \beta) = \sin(\pi/4 + \beta)$).

The starting point of the arc is (W, L), and the endpoint is (L, W), where L and W are the length and the width, respectively, of I (for consistency, we will assume that the item's width is always not larger than its length). Initially, both l and w increase; but when the rotation angle α reaches β, the width starts decreasing and reaches W again when $\alpha = 2\beta$. For larger β, the arc is more "squashed" towards the line $w = l$, and the two limiting cases are as follows: if $\beta = 0$ (so that I is a line segment, $W = 0$), the arc is a quarter-circle from $(0, L)$ to $(L, 0)$ (the width of I monotonically decreases with its length); if $\beta = \pi/4$ (I is a square), the arc is a segment from (W, W) to $(\sqrt{2}W, \sqrt{2}W)$. Refer to Fig. 3, left.

To obtain $w_I^{\min}(l)$, we connect the points (W, L) and $(W, l_I(2\beta))$ with a horizontal segment, and, symmetrically, connect $(w_l(\pi/2 - 2\beta), L)$ to (L, W) with a vertical segment (Fig. 2, right and Fig. 3, right). Note that if $\beta > \pi/8$, the elementary arc hits the line $w = l$ before w starts decreasing, so that the horizontal and the vertical segments intersect; in this case $w_I^{\min}(l)$ consists just of a vertical and a horizontal segment.[3] In any case, $w_I^{\min}(l)$ restricted to $\operatorname{Dom} w_I^{\square}$ is concave;

[2] This special case is very important in practice for those of us who have many laptops, and are wondering what should be the smallest bag into which any of the laptops can be put.

[3] Recall that we also extend $w_I^{\min}(l)$ beyond $\operatorname{Dom} w_I^{\square}$ with the constant values of $+\infty$ and L; hence, we obtain the additional segments in the graph of $w_I^{\min}(l)$ that run outside $\operatorname{Dom} w_I^{\square}$.

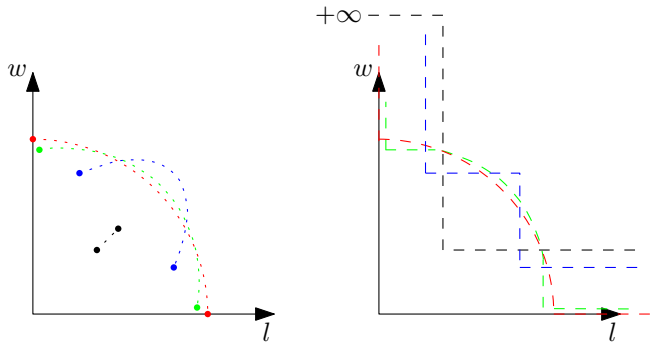


Figure 3: Left: Elementary arcs for a segment ($\beta = 0$, red), a generic rectangle with $0 < \beta < \pi/8$ (green), a rectangle with $\beta = \pi/8$ (blue), and a square ($\beta = \pi/4$, black). Right: $w_I^{\min}(l)$ for the same items.

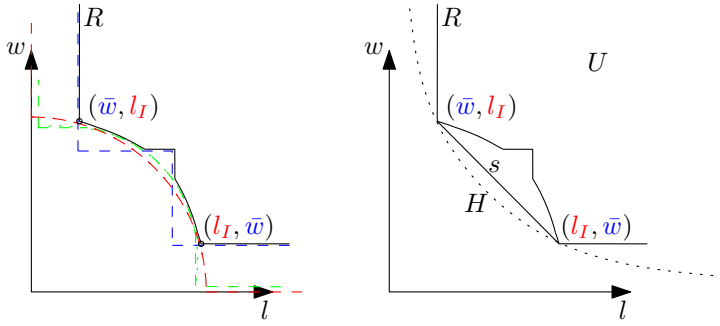


Figure 4: Left: Dashed red, green and blue are $w_I^{\min}(l)$ for three items; solid black is their upper envelope $w^*(l)$. Right: the upper envelope lies in the region U above the segment s above the hyperbola H (dotted).

its graph consists of two segments and an elliptic arc symmetric with respect to the line $w = l$. (The arc may be empty; however, the segments have positive length, for a positive-width item.)

We are now ready to understand how $w^*(l)$—the upper envelope of the functions $w_I^{\min}(l)$—looks for a collection of rectangular items (Fig. 4, left). Let $\bar{w}$ be the width of the widest item (recall that we assume that the width of any item is smaller than its length). Then the graph of $w^*(l)$ contains a vertical ray R going up along the line $l = \bar{w}$ (this means, of course, that it is not possible to fit all items into a box narrower than $\bar{w}$). Let I be an item that has the largest length, l_I, when fit into a strip of width $\bar{w}$. That is, I is the item whose graph $w_I^{\min}(l)$ is the last one intersected by the ray R; the point of the intersection is $(\bar{w}, l_I)$. Because of the symmetry of the function, the point $(l_I, \bar{w})$ also belongs to the graph of $w_I^{\min}(l)$. Since $\bar{w}$ is the maximum item width, $\bar{w} \in \operatorname{Dom} w_I^{\square}$; thus, we also have $l_I \in \operatorname{Dom} w_I^{\square}$. Finally, since $w_I^{\min}(l)$ restricted to $\operatorname{Dom} w_I^{\square}$ is concave, on the interval $[\bar{w}, l_I]$ the graph of $w_I^{\min}(l)$ lies above the segment $s = ((\bar{w}, l_I), (l_I, \bar{w}))$. This implies that the graph of $w^*(l)$ also lies above s (because $w^*(l)$ is the upper envelope of many functions, including $w_I^{\min}(l)$).

Overall, we conclude that the graph of $w^*(l)$ lies inside the shape U bounded by the vertical ray R up from $(\bar{w}, l_I)$, the segment s, and the horizontal ray to the right from $(l_I, \bar{w})$ (Fig. 4, right). Consider now the hyperbola $H = \{l, w | lw = l_I \bar{w}\}$. Any hyperbola $lw = A$ with $A < l_I \bar{w}$ lies below H and hence does not intersect U, and hence does not intersect $w^*(l)$. This means that there can be no feasible (l, w)-suitcase with $lw < l_I \bar{w}$. On the other hand, the $(l_I, \bar{w})$-suitcase *is* feasible and, hence, optimal. In particular, the widest item is flush with the optimal suitcase, yielding the flushness mentioned at the beginning of the section.

To find an optimal suitcase for rectangular items we do not have to build $w^*(l)$. We simply compute the length of each item when fit into a strip of width $\bar{w}$; the longest such item defines

the length of opt. We have:

Theorem 2.6. *For rectangular items problem (P1) can be solved in $O(n)$ time.*

2.4 A PTAS

We return to the case of general convex items (not necessarily rectangles). We give a constant-factor approximation for the problem and then a PTAS, using the standard technique of searching a grid laid out in the solution space.

Recall that an item I is *tight* in a rectangle R if every side of R contains a vertex of I.

Lemma 2.7. *If I is tight in R, then the length of R is not larger than the diameter of I.*

Proof. Let a, b be the vertices of I that lie on the short sides of R; clearly, the length of R is at most $|ab|$, which is at most the diameter of I. □

Let $d_{\max}$ be the maximum diameter of an item, and let $w_{\max}$ be the maximum width of an item; both $d_{\max}$ and $w_{\max}$ can be found in linear time by the rotating calipers technique.

Lemma 2.8. *The area of opt is at most $d_{\max} w_{\max}$.*

Proof. Let the *width-rectangle* of an item I be the rectangle R whose width is equal to width of I, and such that I is tight in R (i.e., the short sides of R are parallel to the width of I). Let I be the item whose width-rectangle has the longest length; let l be this length. The width-rectangles of all items, and hence, all items, fit into an $l \times w_{\max}$ rectangle. By Lemma 2.7, l is not larger than the diameter of I, which is not larger than $d_{\max}$. □

Let l_{opt} and w_{opt} be the length and width of opt, respectively.

Lemma 2.9. $\frac{d_{\max}}{\sqrt{2}} \le l_{opt} \le d_{\max}$.

Proof. Since the maximum-diameter item is inside opt, $d_{\max}$ is smaller than the diameter of opt, which equals to $\sqrt{l_{\mathsf{opt}}^2 + w_{\mathsf{opt}}^2} \le \sqrt{2} l_{\mathsf{opt}}$. On the other hand, there must exist an item I that is tight in opt; by Lemma 2.7, l_{opt} is not larger than the diameter of I, which is not larger than $d_{\max}$. □

Since every item has to fit inside opt, we have:

Lemma 2.10. $w_{opt} \ge w_{\max}$.

By Lemmas 2.8, 2.9 and 2.10, the $(d_{\max}, w_{\max})$-suitcase is a $\sqrt{2}$-approximation to opt. To improve the approximation ratio to $1+\varepsilon$, we lay out a "$1+\varepsilon$ geometric grid" inside the rectangle $\mathcal{R} = [\frac{d_{\max}}{\sqrt{2}}, (1+\varepsilon) d_{\max}] \times [w_{\max}, (1+\varepsilon)\sqrt{2} w_{\max}]$ in the (l, w)-plane. That is, the l-coordinates of the grid points are $\frac{d_{\max}}{\sqrt{2}}, \frac{d_{\max}}{\sqrt{2}}(1+\varepsilon), \frac{d_{\max}}{\sqrt{2}}(1+\varepsilon)^2, \ldots, (1+\varepsilon) d_{\max}$; the w-coordinates are $w_{\max}, w_{\max}(1+\varepsilon), w_{\max}(1+\varepsilon)^2, \ldots, (1+\varepsilon)\sqrt{2} w_{\max}$. By Lemmas 2.8, 2.9 and 2.10, $(l_{\mathsf{opt}}, w_{\mathsf{opt}}) \in \mathcal{R}$; in addition, the grid contains a point (l, w) such that $l \le (1+\varepsilon) l_{\mathsf{opt}}, w \le (1+\varepsilon) w_{\mathsf{opt}}$. So at each grid point (l, w), we test whether each item I_k fits into an (l, w)-suitcase; Martin and Stephenson [44] show how to do the test in $O(n_k)$ time (recall that n_k is the complexity of I_k). Since there are $(\log_{1+\varepsilon} \sqrt{2} + 1)^2 = O(1/\varepsilon^2)$ grid points, we can test all points (and hence find a $(1+\varepsilon)^2$-approximate solution) in overall $O(n/\varepsilon^2)$ time.

We can decrease the running time by doing a binary search for the suitcase area. Specifically, we arrange the grid points onto a family of hyperbolas $H^i = \{l, w | lw = \frac{(1+\varepsilon)^{2i}}{\sqrt{2}} d_{\max} w_{\max}\}$ for $i \in \{0, 1, \ldots, \lceil (\log_{1+\varepsilon} 2)/4 \rceil + 1\}$. For a given query area $A \in [\frac{d_{\max}}{\sqrt{2}} w_{\max}, (1+\varepsilon) d_{\max} w_{\max}]$ let

H_A^-, H_A^+ be the hyperbolas H^i, H^{i+1} such that the hyperbola $lw = A$ lies between H^i and H^{i+1} (i.e., $(1+\varepsilon)^{2i}\frac{d_{\max}}{\sqrt{2}}w_{\max} \le A \le (1+\varepsilon)^{2i+2}\frac{d_{\max}}{\sqrt{2}}w_{\max}$). The query consists of testing all grid points on the hyperbolas H_A^-, H_A^+. If no grid point among those on H_A^+ gives a feasible suitcase, A should be increased. If there is a feasible grid point on H_A^-, A can be decreased. Since there are $O(1/\varepsilon)$ hyperbolas in the family, after $O(\log\frac{1}{\varepsilon})$ steps, we find a value of A such that no point on H_A^- is feasible (which means that the area of opt is at least $A/(1+\varepsilon)^2$), but there is a feasible point (l, w) on H_A^+ (with $lw \le (1+\varepsilon)^2 A$). Since there are $O(1/\varepsilon)$ grid points on each hyperbola, one step of our binary search can be performed in $O(n/\varepsilon)$ time, yielding an overall $O(\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$ running time.

Theorem 2.11. *A $(1+\varepsilon)$-approximate solution to problem (P1) can be computed in $O(\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$ time.*

Finally, we can slightly improve the running time for the case in which the average complexity of an item is high, more specifically, when $K = o(n\sqrt{\varepsilon})$. For that, we replace each item I_k by its *ε-kernel* I'_k, a size-$O(1/\sqrt{\varepsilon})$ subset of the vertices of I_k that can be computed in $O(n_k + 1/\sqrt{\varepsilon}^{3/2})$ time [5, 28]; the kernels for all items can thus be computed in $O(n + K/\sqrt{\varepsilon})$ time. The crucial property of the kernel is that the width of I_k in any direction is at most the width of I'_k in the same direction, divided by $1-\varepsilon$. Thus, if I'_k fits into an (l, w)-suitcase, I_k fits into an $(\frac{l}{1-\varepsilon}, \frac{w}{1-\varepsilon})$-suitcase (obviously, if the kernel does not fit into a suitcase, the original item does not fit either). This implies that replacing the items-inclusion test with the kernels-inclusion test, we only lose an additional $\frac{1}{1-\varepsilon}$ factor in the approximation. Overall, we are testing $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ grid points, and at each point, we test for inclusion each of the K kernels; since each kernel has $O(1/\sqrt{\varepsilon})$ vertices, testing one kernel takes $O(1/\sqrt{\varepsilon})$ time [44].

Corollary 2.12. *A $(1+\varepsilon)$-approximate solution to problem (P1) can be computed in $O(n + \frac{K}{\varepsilon^{3/2}}\log\frac{1}{\varepsilon})$ time.*

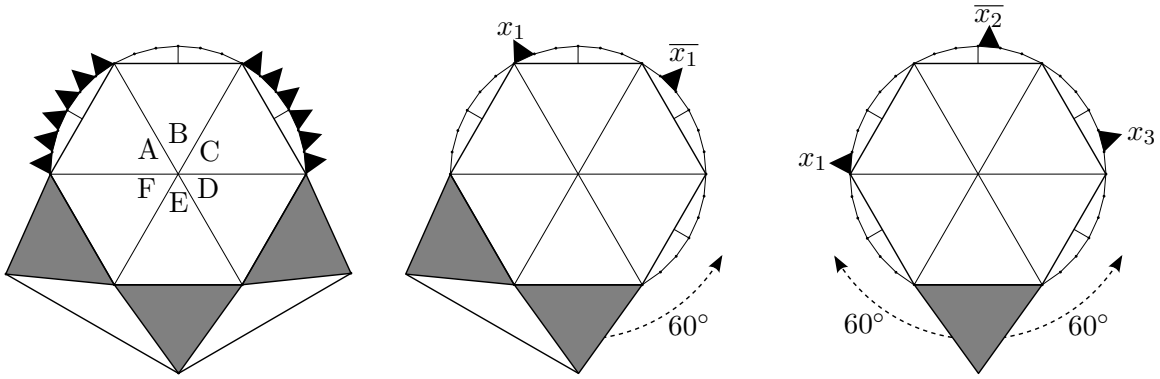
Note that our PTAS extends to 3D: we can similarly search the grid in the (length,width,height)-space for an approximately optimal suitcase; we omit the details. In 3D, apart from finding a minimum-volume suitcase, one may be interested, e.g., in the minimum-surface-area enclosure (in terms of suitcases, it means, e.g., using the minimum amount of leather for the suitcase). We leave open the problem of finding an exact solution in 3D.

2.5 Translation only

When only translation of items is allowed (problems (P3) and (P4)), finding a smallest suitcase becomes simpler. In fact, when the suitcase is required to be axis-aligned (problem (P3)), the solution is trivial: for each item k find the length x_k of its projection onto the x-axis and the length y_k of its projection onto the y-axis; the suitcase must have dimensions $\max_k x_k$ and $\max_k y_k$.

Theorem 2.13. *Problem (P3) can be solved in linear time.*

For an arbitrarily-oriented suitcase (problem (P4)) the solution is similar: The suitcase of a *given* orientation φ must have dimensions $\max_k x_k(\varphi)$ and $\max_k y_k(\varphi)$ where $x_k(\varphi), y_k(\varphi)$ are lengths of the projections of I_k onto directions φ and $\pi/2+\varphi$. Using the first equation in (1), we build the graph of $x_k(\varphi)$; the graph of $y_k(\varphi)$ is obtained by $\pi/2$-shift of the φ-axis. (Note that here we do not treat equations (1) as parametric equations of an arc in the lw-space; instead, each equation is just a function of φ.) We compute the upper envelope $\max_k x_k(\varphi)$; since the graph of $x_k(\varphi)$ consists of $O(n_k)$ pieces, and any two pieces intersect at most twice, by Lemma 2.1 the upper envelope has complexity $O((n/K)\,\lambda_4(K))$ and can be found in time



(a) Ground piece (b) Variable x_1 (c) Clause $(x_1 \vee \overline{x_2} \vee x_3)$

Figure 5: Pieces used in the hardness proof.

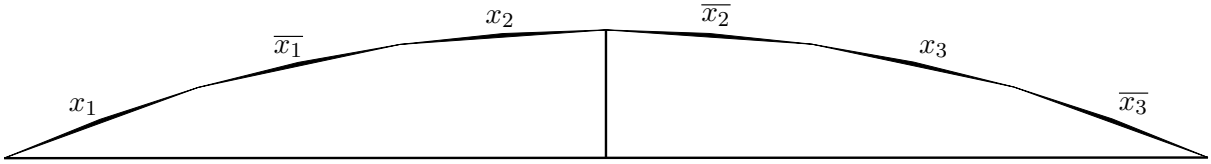


Figure 6: Teeth are much smaller than shown in Figure 5.

$O((n/K)\,\lambda_3(K)\log K) = O(n\alpha(K)\log K)$. We also compute the upper envelope $max_k y_k(\varphi)$. Finally, we scan both envelopes simultaneously in time proportional to their complexity, to find a φ that minimizes the product $\max_k x_k(\varphi) \times \max_k y_k(\varphi)$.

Theorem 2.14. *Problem (P4) can be solved in $O(n\alpha(K)\log K)$ time.*

3 Stacking Items into a Convex Suitcase

In this section we consider the problem of finding a smallest convex suitcase that can enclose any item. We show that finding the minimum-area suitcase (problem (P2)) is NP-hard and give a PTAS for finding a minimum-perimeter suitcase.

Theorem 3.1. *The problem (P2) is NP-hard.*

Proof. The proof is by reduction from 3SAT. Let a 3SAT instance be given using N variables $x_1, \ldots, x_N$ and M clauses. We construct a set of $N + M + 1$ convex items, as shown in Figure 5. The figure shows an inner structure for clarity purposes. Each item consists of a hexagonal inner piece. Between one and three sides have a large *anchor* triangle (shown in grey) attached. The other sides consist of a convex chain of $2N$ segments, each of which may or may not have a small *tooth* triangle attached. Each of the N variables gets mapped to two segments (and teeth), one for the variable, and one for its negation. Note that the chains and teeth in Figure 5 are exaggerated; the chains are flatter (i.e., closer to the hexagon), and the teeth are extremely flat, as shown in Figure 6. We denote by $A_\triangle$ the area of a tooth.

The $1 + N + M$ items we use are the following:

- One *ground item* G (Figure 5(a)). It uses all three anchors. Two of the chains have all teeth present; the middle chain is empty.

- *N variable items* $X_1, \ldots, X_N$ (Figure 5(b)). They use two anchors. In X_i, the two chains opposite to the anchors are equipped with one tooth each, at the positions for x_i, respectively $\overline{x_i}$.
- *M clause items* (Figure 5(c)). They use one anchor. The chain opposed to the anchor and its two adjacent chains bear one tooth each, so that each variable (or negation) in the clause appears as one tooth on the item.

Any suitcase for these items is a superset of G; therefore, we only care about the area needed in addition to G. We observe that every variable item X_i can be rotated and placed on G in a way to be completely covered by it, except for a single tooth sticking out at x_i or $\overline{x_i}$ on the empty chain of G. It is therefore possible to put all variable items onto G with an additional space of $NA_\triangle$, corresponding to every assignment for $x_1, \ldots, x_N$. In every other placement, the variable's anchors will not be covered by G, exposing some additional area. Given that $A_\triangle$ can be made as small as desired (by flattening the teeth), we can make sure that any other orientation causes the additional area to exceed $NA_\triangle$.

Next we consider the clause items: They can be put onto the ground piece in three different orientations such that only one tooth is not covered by G, corresponding to some variable or its negation. If the associated variable item exposes the same tooth, this causes no additional area. Otherwise it causes an additional area of $A_\triangle$. So, the clause items can be added without causing additional area (beyond the area we know to be required) if, and only if, the variable items are arranged according to a variable assignment that fulfills the 3SAT instance.

Therefore, the 3SAT instance is solvable if, and only if, the items can be fit into a suitcase whose area exceeds that of G by $NA_\triangle$. □

Approximation

For a convex suitcase, minimizing the area is equivalent to minimizing the perimeter. Note that a minimum rectangular suitcase is a 2-approximation of the minimum convex suitcase C, both in terms of the area and the perimeter (if C is tight in a rectangle, both the area and the perimeter of the rectangle is at most twice those of C). We now show how to improve the perimeter approximation to $1+\varepsilon$, for any constant $\varepsilon > 0$.

For a convex polygon M, let $|M|$ denote its perimeter. Any convex polygon can be well approximated, in terms of the perimeter, by an *equiangular* polygon:

Lemma 3.2. *There exists an r-gon R, containing C, such that $|R| \leq |C| + O(\frac{1}{r^2})|C|$.*

Proof. Let R be an equiangular r-gon such that C is tight in R, i.e., every edge $q_i q_{i+1}$ $(i = 1 \ldots r)$ of R contains a vertex p_i of C (Fig. 7). $|C|$ is larger than $\sum_{i=1}^{r} |p_{i-1}p_i|$. But since $\angle p_{i-1}q_i p_i = \pi - \frac{2\pi}{r}$, we have that $\forall i, |p_{i-1}q_i| + |q_i p_i| \leq |p_{i-1}p_i| / \cos\frac{\pi}{r} = |p_{i-1}p_i| + |p_{i-1}p_i| O(\frac{1}{r^2})$. Summing over i, we get $|R| \leq \sum_{i=1}^{r} |p_{i-1}p_i|(1 + O(\frac{1}{r^2}))$. □

Let G_δ be the regular square grid with spacing δ. Any r-gon R can be snapped outward onto the grid with an $O(r\delta)$ increase in the perimeter while at most doubling the number of vertices:

Lemma 3.3. *There exists a $2r$-gon $R' \supseteq R$ whose vertices belong to G_δ, and such that $|R'| \leq |R| + O(r\delta)$.*

Proof. Each vertex v of R may be replaced by two gridpoints v', v'' so that the modified polygon increases; the replacement depends on how R goes through the grid pixel to which v belongs. See Fig. 8 for the different cases. □

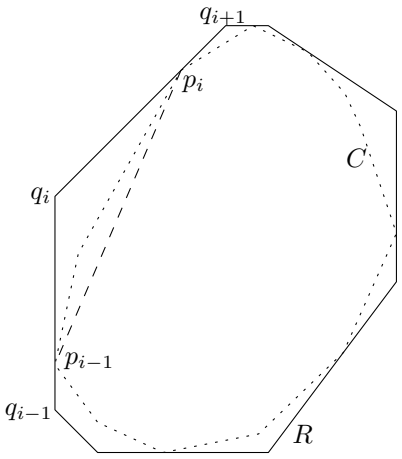


Figure 7: C (dotted) is tight in R (solid).

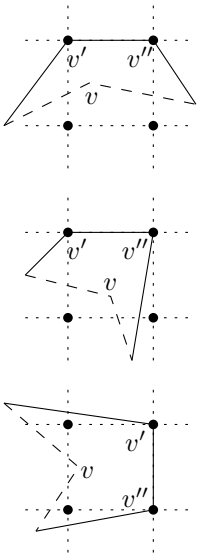


Figure 8: R is dashed, R' is solid.

Remark. In fact, there exists an $(r+4)$-gon with the required properties: Look at the topmost, bottommost, leftmost, rightmost pixels; they each get 2 gridpoints. The monotone chains between them get rounded to a single grid point each. However, for our argument we need a multiplicative growth of the number of vertices, not additive.

Let p be the perimeter of the minimum rectangular suitcase enclosing all items. We lay out the $\frac{1}{\varepsilon} \times \frac{1}{\varepsilon}$ grid $G_{\varepsilon p}$ inside the $p \times p$ square, and, for a constant r, consider all convex $2r$-gons with vertices from the grid (since there are $\frac{1}{\varepsilon^2}$ gridpoints, there are $O(\frac{1}{\varepsilon^{4r}}) = O(1)$ such $2r$-gons); for each $2r$-gon we test whether each item fits inside the $2r$-gon (this can be done in polynomial (in r) time, e.g., with the algorithm of Chazelle [31]), and choose the best feasible $2r$-gon.

Since the perimeter of an optimal convex suitcase C is at most p, C is contained in a $\frac{p}{2} \times \frac{p}{2}$ square. By Lemma 3.2, there exists an r-gon R whose perimeter is not much larger than that of C; hence R is contained in a $p \times p$ square. By Lemma 3.3, there exists a $2r$-gon R' with vertices from $G_{\varepsilon p}$ such that $|R'| \leq |R| + O(r\varepsilon p) = |R| + O(r\varepsilon)|C|$ since $p \leq 2|C|$. Finally, by Lemma 3.2, $|R| + O(r\varepsilon)|C| \leq |C| + O(\frac{1}{r^2} + r\varepsilon)|C| = |C| + O(\varepsilon)|C|$ for $r = O(\frac{1}{\sqrt{\varepsilon}})$.

Theorem 3.4. *For any constant $\varepsilon > 0$, a $(1+\varepsilon)$-approximation to the minimum-perimeter convex suitcase can be computed in polynomial time.*

Remark. We were not able to find a PTAS for minimizing the suitcase area; the grid approach did not work for us because an optimal suitcase may potentially be very thin (essentially 0 area) and snapping its vertices to the grid would increase its area by a large factor.

4 Packing Two Items into Rectangular Suitcases

In this section we give algorithms for the disjoint packing problems (P5) – (P7): given two convex items A and B, find a minimum-area rectangle R into which A and B can be placed simultaneously without overlap. Our algorithms are based on scrolling through all "combinatorially different" placements of the items in the suitcase. Section 4.1 presents a cubic-time algorithm for problem (P5), in which both translations and rotations of items are allowed. For translations only, Section 4.2 presents a linear-time algorithm for the case in which the orientation of R is given

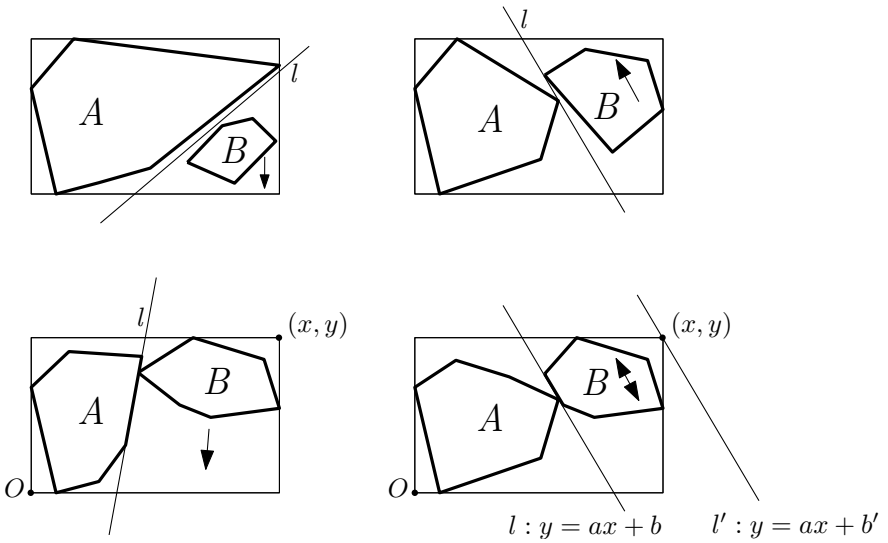


Figure 9: Each side of R is touched by exactly one item. In all cases one can modify the solution by translating one of the items so that either the area of R decreases or so that the area of R stays the same, but the placement of the items satisfies the lemma's claim.

(problem (P6)), and Section 4.3 presents a linear-time algorithm for packing into a suitcase with arbitrary orientation (problem (P7)).

Clearly, in an optimal placement the items touch each other. Also, since items are convex, there is a line l separating them. Based on these simple observations, we derive a useful property that applies to all three problems (P5) – (P7):

Lemma 4.1. *There exists an optimal solution in which at least one side of the rectangle R touches both items A and B.*

Proof. In an optimal placement each side of R is touched by *at least* one item. Consider a packing in which each side of R is touched by *exactly* one item. There are several cases (up to the symmetry) of how the items can touch the sides of R (Fig. 9):
• *One item, say A, touches all four sides of R (Fig. 9, top left)*: B can be translated within R so that it does not cross l and comes into contact with a side of R.
• *A touches three and B touches one side of R (Fig. 9, top right)*: B can be translated parallel to l in one of the directions so that R does not increase.
• *Each of A, B touches two sides of R (Fig. 9, bottom)*: By convexity, the sides touched by A are incident (assume the sides meet at the origin), as are the sides touched by B (assume the sides meet at a point (x, y) with $x, y > 0$). As B translates along l, each of x, y changes linearly, and (x, y) moves along a line l' parallel to l. If the slope a of l is non-negative (Fig. 9, bottom left), then B can be translated so as to decease both x and y. If $a < 0$ (Fig. 9, bottom right), the area of the rectangle R, $xy = x(ax + b') = ax^2 + b'x$ cannot have a local minimum. □

4.1 Arbitrary rigid motions

We present an $O(n^3)$-time algorithm for problem (P5), in which arbitrary isometries of items are allowed. Assume that an optimal suitcase R is axis-aligned and has its lower left corner at the origin. Number the sides of the suitcase 0 to 3 counterclockwise starting from the bottom side, and for $i = 0 \ldots 3$ let a_i (respectively, b_i) denote the vertex of A (respectively B) closest to (or touching) side i of R (Fig. 10). By Lemma 4.1 we may assume that both items A and B

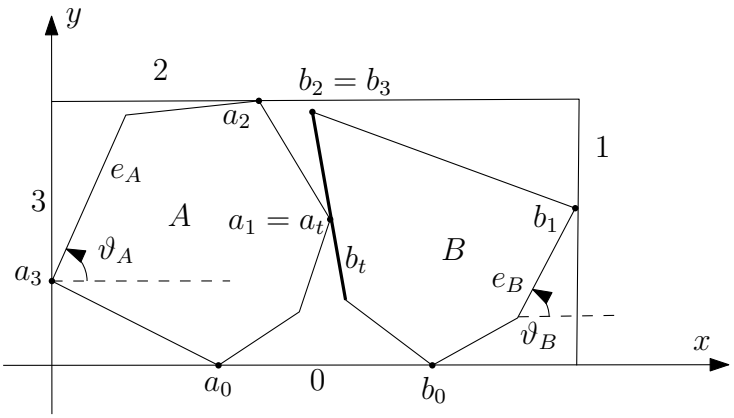


Figure 10: A possible placement of the items.

touch side 0. We furthermore assume that A lies left of B (a_0 is to the left of b_0); we just will have to run our algorithm twice, once for this and once for the opposite assumption.

As mentioned above, in an optimal placement A and B touch each other. Let a_t be the feature (vertex or edge) of A that touches B; let b_t be the edge or vertex of B that touches A. We will call a 10-tuple $(a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3, a_t, b_t)$ that is possible under a certain placement of A and B, a *combinatorial position* of A and B.

Different combinatorial positions can occur through different orientations of A and B. We identify an orientation of A with the angle ϑ_A of some fixed edge e_A of A with the x-axis; similarly, we define the orientation ϑ_B of B (see Fig. 10). A pair $(\vartheta_A, \vartheta_B)$ of orientations uniquely defines (under our assumptions above) the placement of both A and B, and hence also their combinatorial position. We therefore can identify the space of all possible placements, within which we are looking for the optimal solution, with the rectangle $P = [0, 2\pi] \times [0, 2\pi]$ in the $(\vartheta_A, \vartheta_B)$-space. The rectangle P is partitioned into finitely many connected cells corresponding to the combinatorial positions of A and B; let $\mathcal{P}$ be the partition.

Lemma 4.2. *The complexity of $\mathcal{P}$ is $O(n^3)$.*

Proof. There are only $O(n)$ possible 4-tuples (a_0, a_1, a_2, a_3) since, in each tuple the vertices a_0, a_2 are *antipodal* (belonging to parallel supporting lines of A), and there are $O(n)$ antipodal vertices [54, 59]. As ϑ_A varies, such a 4-tuple changes when the orientation ϑ_A causes one of the edges of A to become vertical or horizontal. These events partition P by $O(n)$ vertical lines. Likewise, P is partitioned by $O(n)$ horizontal lines corresponding to the orientations ϑ_B at which the 4-tuple (b_0, b_1, b_2, b_3) changes. Within each of the $O(n^2)$ cells obtained from the horizontal and vertical lines the 8-tuple $(a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3)$ is invariant.

To complete the partition $\mathcal{P}$ it remains to subdivide P further according to different pairs (a_t, b_t). Say that the items are in a *vv-contact, ee-contact*, or *ev-contact* depending on whether both a_t, b_t are vertices, both are edges, or one is a vertex and one is an edge, respectively. Call a set of angles $(\vartheta_A, \vartheta_B)$ corresponding to the same pair (a_t, b_t) of vertices a *vv-arc*; define an *ee-arc* similarly. Any vv- or ee-arc is a curve (a 1-dimensional set) in P (the arcs' equations are given below). If one changes the angles $(\vartheta_A, \vartheta_B)$ continuously, then between any two different ev-contacts one must encounter either a vv- or an ee-contact. That is, the cells of $\mathcal{P}$ corresponding to different ev-contacts are separated by vv- and ee-arcs. Thus, the complexity of $\mathcal{P}$ is determined by the number of intersections between vv-arcs, ee-arcs and the $O(n)$ vertical and horizontal grid lines described above. We now give a cubic bound on this number of intersections.

An ee-arc for edges $e \in A, f \in B$ is defined by equation $\vartheta_A = \vartheta_B + \alpha_{ef}$, where α_{ef} is a constant. Consequently there are $O(n^2)$ ee-arcs, each being a straight-line segment with slope 1.

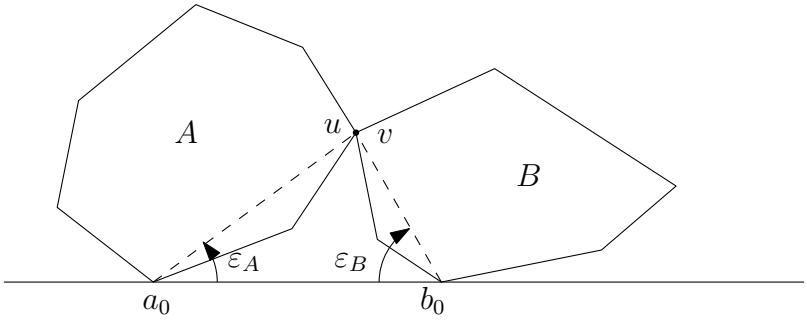


Figure 11: A vv-contact.

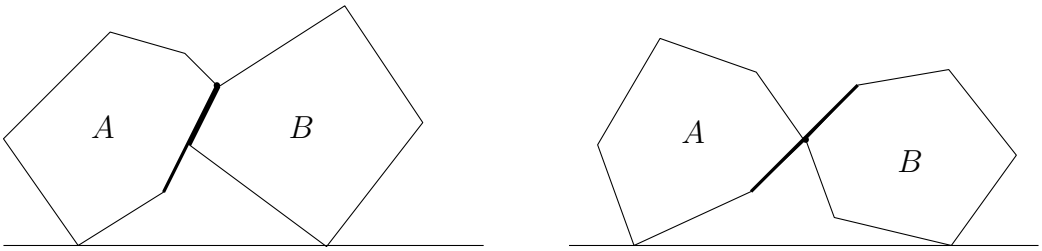


Figure 12: Intersections between an ee- and a vv-arc.

Consider now a vv-arc corresponding to vertices $u \in A, v \in B$. Rotate A and B so that u, v remain in contact and a_0, b_0 slide along the x-axis. Then, in the triangle a_0ub_0 we have for the angles $\epsilon_A = \angle b_0a_0u, \epsilon_B = \angle a_0b_0u$ (Fig. 11): $\epsilon_A = \vartheta_A + \alpha, \epsilon_B = -\vartheta_B + \beta$ for some constants α, β. By the law of sines, $\sin\epsilon_A / \sin\epsilon_B = |b_0u|/|a_0u|$, or $\vartheta_B = \beta - \arcsin(r\sin(\vartheta_A + \alpha))$ for some constants α, β, r. This is the equation for the vv-arc, and overall there are $O(n^2)$ vv-arcs, one per pair of vertices.

Finally, in order to analyze the complexity of the partition $\mathcal{P}$, let us count the number of intersections between the various curves and segments. Clearly, two ee-arcs do not intersect, and neither do two vv-arcs. The $O(n)$ vertical and horizontal straight-line segments intersect $O(n^2)$ ee- and vv-arcs in $O(n^3)$ points. An intersection between an ee-arc and a vv-arc is only possible when one of the v's is adjacent to one of the e's (Fig. 12); thus there are only $O(n^2)$ such intersection points. □

Our algorithm for finding an optimal placement of items builds the partition $\mathcal{P}$ and traverses all of its cells. Within one cell, where all a_i, b_i, a_t, b_t are the same, we compute the optimal placements of two m-gons ($m \le 5$), one with vertices a_0, a_1, a_2, a_3 and feature a_t, and the other with vertices b_0, b_1, b_2, b_3 and feature b_t; this takes constant time. For each of these placements, we check whether it lies in cell C and, if so, keep track of the area of the corresponding rectangle; the smallest rectangle found this way is an optimal solution.

Theorem 4.3. *Problem (P5) can be solved in $O(n^3)$ time.*

Remark. One may expect quadratic complexity in Lemma 4.2; we do not have a lower bound that shows that two convex polygons can have cubically many interesting configurations.

4.2 Translations only: Axis-Aligned Suitcase

In this section we consider problem (P6), packing two items using translations only into a minimum-area rectangular suitcase of a given orientation (without loss of generality, the suitcase

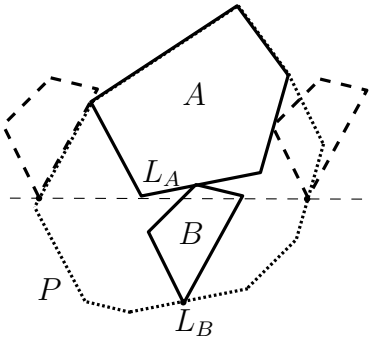


Figure 13: The orbit P of L_B is dotted.

is axis-aligned). We note again that an $O(n \log n)$-time for the problem follows from Milenkovic's $O(n^{K-1} \log n)$ algorithm for packing an arbitrary number K of items [47]; we give an $O(n)$-time algorithm for our special case $K = 2$.

By Lemma 4.1, one of the sides of an optimal suitcase is touched by both items. We present an $O(n)$-time algorithm to find a smallest suitcase whose lower side is touched by both items; one should run the same algorithm for the other three sides, and choose the best solution.

Assume that item A is fixed and consider sliding B around the boundary of A. Let L_A, L_B be the lowest points of the items. Let P denote the *orbit* of L_B, i.e., the locus of positions that L_B assumes as B slides around A (Fig. 13). The orbit P has the following property: B intersects A if and only if L_B is inside P. This means that the region bounded by P is the Minkowski difference of A and $B_{L_B=O}$, where $B_{L_B=O}$ is B translated so that L_B is at the origin, and the Minkowski difference of two sets S, T is $\{s - t | s \in S, t \in T\}$ [34, Chapter 13.3]. Hence, the orbit P can be computed in $O(n)$ time.

Both A and B touch the lower side of the suitcase only when L_B is aligned horizontally with L_A. Such alignment happens only twice, since the orbit is convex. For each of the two alignments, we compute the axis-aligned bounding rectangle of the items, and choose the smaller of the two.

Theorem 4.4. *Problem (P6) can be solved in $O(n)$ time.*

4.3 Translations only: Arbitrarily-Oriented Suitcase

This section gives a linear-time algorithm for problem (P7), packing two items using translations only into the smallest rectangular suitcase R (of arbitrary orientation). We reuse many ideas from the preceding sections.

We reformulate the problem by fixing the orientation of the suitcase and instead allowing the items to translate and rotate *by the same angle* ϑ. As in the preceding sections, without loss of generality, the suitcase is axis-aligned, its lower left corner is at the origin, its sides are numbered 0 to 3, and the combinatorial position of the items is the 10-tuple $(a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3, a_t, b_t)$ indicating the vertices of A and B closest to (or touching) different sides of R, plus the features with which the items touch each other (refer to Fig. 10, with $\vartheta_A = \vartheta_B = \vartheta$). By Lemma 4.1, at least one side of R must be touched by both items. Again, we will present an algorithm to find an optimal positioning of the items under the requirement that they both are anchored at the lower side (side 0) of R, and that $a_0 < b_0$; one should run our algorithm for all four sides and the reverse requirement for the order of touching.

As the common angle of the items rotation ϑ varies, the axis-aligned bounding rectangle R of the items changes continuously, and the combinatorial position changes at discrete *events*.

The events are of two types: *vertex events*, at which a_i, b_i change, and *contact events*, at which a_t, b_t change. As in Section 4.1, there are $O(n)$ vertex events, and for each item the events come to us sorted since the vertices of the item are given in order around the item's boundary; thus, in linear time we can obtain the merged sorted list of vertex events corresponding to both A and B. As in Section 4.2, a contact event corresponds to a vertex of the Minkowski difference of A and $B_{L_B=O}$, and thus there are $O(n)$ contact events.[4] The difference (which can be computed in $O(n)$ time [34, Chapter 13.3]) provides us with the sorted list of contact events; we merge the list with the vertex events list. Finally, also as in Section 4.1, for a fixed combinatorial position, the angle ϑ that minimizes the area of R can be found in constant time.

Theorem 4.5. *Problem (P7) can be solved in $O(n)$ time.*

5 Future work

We considered stacking and packing convex items into different-shape suitcases. Each of our problems has many versions: perimeter vs. area minimization, overlapping vs. disjoint items (stacking vs. packing), rectilinear vs. arbitrary convex suitcases, axis-aligned vs. arbitrary rectangular suitcases, rectangular vs. arbitrary items, general isometries vs. translation only, etc. We did not provide a systematic treatment of all possible variants (some variants are trivial, e.g., stacking rectangular items under translations only into axis-aligned rectangular suitcase), and many open problems remain.

Let us explicitly mention one open problem: stacking rectangular items into a minimum-perimeter convex suitcase. It is shown in [11] (see also [12, Theorem 5]) that the perimeter of the convex hull of the union of rectangles is a convex function with respect to translation of the items. Thus, in an optimal solution the items' centers should be at the same point, and the problem reduces to finding optimal rotations for the items. We did not see how to solve the problem, but found this, rotation-only version quite interesting.

Another interesting direction is using more than one suitcase. For example, for our (admittedly) simplest problem of stacking rectangular items into a rectangular suitcase, one can find two smallest (minsum or minmax) suitcases in $O(n^2 \log n)$ time employing our flushness result: for any partitioning of the items between the two suitcases, the width of each suitcase will be equal to the width of the widest item in the suitcase, and one of the suitcases will have width W_1 equal to the maximum item width; this leaves $n-1$ guesses for the width W_2 of the second suitcases, and for each guess one can find an optimal partitioning of the items by sorting the lengths of the items tightly packed into width-W_1 and width-W_2 rectangles (of course, the W_1-sorting can be done only once). Can the $O(n^2 \log n)$ time be improved, e.g., by adapting matrix searching techniques? (Perhaps the sorted orders for different W_2's do not differ much?) Is the problem hard for a non-constant number of suitcases? What versions of our problems admit polynomial-time solutions for multiple suitcases?

Acknowledgments We thank the FUN'2012 Program Committee member for the suggestion to include a recipe for Scandinavian thins into the full version of the paper. The main ingredient for preparing the thins is honey: to have the thins baked, approach your spouse with "It's been a long time since we had thins; would you mind baking them, honey?". For those who want to prepare the biscuits alone, see for example [45].

[4] A more direct proof of the linear bound on the number of contact events would involve charging an event to the feature of A or B that disappears at the event from the pair (a_t, b_t); once disappeared, the feature never returns to the pair. With such a proof, we would additionally have to argue that the contact events can be discovered one-by-one, "on-the-fly" during the rotation, between the neighboring vertex events; a constant-time implementation of such a discovery is straightforward.

We thank the anonymous referees for their helpful comments. E. Arkin and J. Mitchell are partially supported by the National Science Foundation (CCF-1018388). F. Hurtado is partially supported by projects MICINN MTM2009-07242, Gen. Cat. DGR2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306. V. Polishchuk is supported by the Academy of Finland grant 1138520.

References

[1] `http://en.wikipedia.org/wiki/Ginger_biscuit`.

[2] `http://www.annasthins.ca/`.

[3] `http://en.wikipedia.org/wiki/Cat_flap`.

[4] `http://pack-any-shape.com/`.

[5] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51(4):606–635, 2004.

[6] P. K. Agarwal and M. Sharir. Davenport-Schinzel sequences and their geometric applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 1–47. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[7] H.-K. Ahn, S. W. Bae, O. Cheong, J. Gudmundsson, T. Tokuyama, and A. Vigneron. A generalization of the convex Kakeya problem. In D. Fernández-Baca, editor, *Theoretical Informatics – 10th Latin American Symposium (LATIN 2012), Arequipa, Peru, April 16-20, 2012*, volume 7256 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2012.

[8] H.-K. Ahn, P. Brass, and C.-S. Shin. Maximum overlap and minimum convex hull of two convex polyhedra under translations. *Computational Geometry: Theory and Applications*, 40(2):171–177, 2008.

[9] H.-K. Ahn, S.-W. Cheng, and I. Reinbacher. Maximum overlap of convex polytopes under translation. *Computational Geometry: Theory and Applications*, 46(5):552–565, 2013.

[10] H.-K. Ahn and O. Cheong. Stacking and bundling two convex polygons. In *Algorithms and Computation, 16th International Symposium (ISAAC 2005), Sanya, Hainan, China, December 19-21, 2005*, 2005.

[11] H.-K. Ahn and O. Cheong. Aligning two convex figures to minimize area or perimeter. *Algorithmica*, 62(1-2):464–479, 2012.

[12] H.-K. Ahn, O. Cheong, C.-D. Park, C.-S. Shin, and A. Vigneron. Maximizing the overlap of two planar convex sets under rigid motions. *Computational Geometry: Theory and Applications*, 37(1):3–15, 2007.

[13] H. Alt, J. Blömer, and H. Wagener. Approximation of convex polygons. In M. Paterson, editor, *Proc. Automata, Languages and Programming, 17th International Colloquium (ICALP 1990), Warwick University, England, July 16-20, 1990*, volume 443 of *Lecture Notes in Computer Science*, pages 703–716. Springer, 1990.

[14] H. Alt, U. Fuchs, G. Rote, and G. Weber. Matching convex shapes with respect to the symmetric difference. *Algorithmica*, 21(1):89–103, 1998.

[15] H. Alt and F. Hurtado. Packing convex polygons into rectangular boxes. In J. Akiyama, M. Kano, and M. Urabe, editors, *Discrete and Computational Geometry, Japanese Conference, JCDCG 2000, Tokyo, Japan, November, 22-25, 2000, Revised Papers*, volume 2098 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2000.

[16] H. Aonuma, H. Imai, K. Imai, and T. Tokuyama. Maximin location of convex objects in a polygon and related dynamic voronoi diagrams. In *Proc. 6th Annual Symposium on Computational Geometry*, pages 225–234, New York, NY, USA, 1990. ACM.

[17] E. M. Arkin, A. Efrat, G. Hart, I. Kostitsyna, A. Kröller, J. S. B. Mitchell, and V. Polishchuk. Scandinavian thins on top of cake: On the smallest one-size-fits-all box. In E. Kranakis, D. Krizanc, and F. L. Luccio, editors, *Proc. Fun with Algorithms - 6th International Conference, FUN 2012, Venice, Italy, June 4-6, 2012*, volume 7288 of *Lecture Notes in Computer Science*, pages 16–27. Springer, 2012.

[18] E. M. Arkin, S. Khuller, and J. S. B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10:399–427, 1993.

[19] F. Avnaim and J.-D. Boissonnat. Simultaneous containment of several polygons. In *Proc. 3rd Annual Symposium on Computional Geometry*, pages 242–250, 1987.

[20] B. S. Baker, S. J. Fortune, and S. R. Mahaney. Polygon containment under translation. *J. Algorithms*, 7:532–548, 1986.

[21] G. Barequet and S. Har-Peled. Polygon containment and translational min-Hausdorff-distance between segment sets are 3SUM-hard. *International Journal of Computational Geometry and Applications*, 11:465–474, 2001.

[22] B. Becker, P. G. Franciosa, S. Gschwind, S. Leonardi, T. Ohler, and P. Widmayer. Enclosing a set of objects by two minimum area rectangles. *J. Algorithms*, 21(3):520–541, 1996.

[23] B. K. Bhattacharya and A. Mukhopadhyay. On the minimum perimeter triangle enclosing a convex polygon. In J. Akiyama and M. Kano, editors, *JCDCG'02*, Lecture Notes in Computer Science 2866, pages 84–96. Springer, 2003.

[24] P. Bose, M. Mora, C. Seara, and S. Sethia. On computing enclosing isosceles triangles and related problems. *Int. J. Comput. Geometry Appl.*, 21(1):25–45, 2011.

[25] J. E. Boyce, D. P. Dobkin, R. L. Drysdale, III, and L. J. Guibas. Finding extremal polygons. *SIAM Journal on Computing*, 14:134–147, 1985.

[26] P. Brass, O. Cheong, H. S. Na, C. S. Shin, and A. Vigneron. Approximation algorithms for inscribing or circumscribing an axially symmetric polygon to a convex polygon. In *COCOON 2004, LNCS*, volume 3106, pages 259–267. Springer, 2004.

[27] S. Cabello, M. de Berg, P. Giannopoulos, C. Knauer, R. van Oostrum, and R. C. Veltkamp. Maximizing the area of overlap of two unions of disks under rigid motion. *International Journal of Computational Geometry and Applications*, 19(6):533–556, 2009.

[28] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry: Theory and Applications*, 35(1-2):20–35, 2006.

[29] J. S. Chang. Polygon optimization problems. Report 240, CS, NYU, New York, NY, 1986.

[30] J. S. Chang and C. K. Yap. A polynomial solution for the potato-peeling problem. *Discrete and Computational Geometry*, 1:155–182, 1986.

[31] B. M. Chazelle. The polygon containment problem. *Advances in Computing Research*, 1:1–33, 1983.

[32] K. Daniels and V. J. Milenkovic. Multiple translational containment, Part I: An approximate algorithm. *Algorithmica*, 19(1–2):148–182, Sept. 1997.

[33] M. de Berg, O. Cheong, O. Devillers, M. J. van Kreveld, and M. Teillaud. Computing the maximum overlap of two convex polygons under translations. *Theory of Computing Systems*, 31(5):613–628, 1998.

[34] M. de Berg, O. Cheong, M. Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer, 2008.

[35] N. A. A. DePano. *Polygon approximation with optimized polygonal enclosures: applications and algorithms*. Ph.D. thesis, Department of Computer Science, Johns Hopkins University, 1987.

[36] N. A. A. DePano and A. Aggarwal. Finding restricted k-envelopes for convex polygons. In *Procedings of the 22nd Allerton Conference on Communication, Control, and Computing*, pages 81–90, 1984.

[37] J. Egeblad, B. K. Nielsen, and M. Brazil. Translational packing of arbitrary polytopes. *Computational Geometry: Theory and Applications*, 42(4):269–288, 2009.

[38] H. Freeman and R. Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM*, 18:409–413, 1975.

[39] K. Fukuda and T. Uno. Polynomial time algorithms for maximizing the intersection volume of polytopes. *Pacific Journal of Optimization*, 3(1):37–52, 2007.

[40] R. L. Graham, B. D. Lubachevsky, K. J. Nurmela, and P. R. J. Östergård. Dense packings of congruent circles in a circle. *Discrete Mathematics*, 181(1-3):139–154, 1998.

[41] R. Heckmann and T. Lengauer. Computing upper and lower bounds on textile nesting problems. *European Journal of Operational Research*, 108(3):473–489, 1998.

[42] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inf. Process. Lett.*, 33(4):169–174, 1989.

[43] M. Löffler and M. J. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry: Theory and Applications*, 43(4):419–433, 2010.

[44] R. R. Martin and P. C. Stephenson. Containment algorithms for objects in rectangular boxes. *Theory and Practice of Geometric Modeling*, pages 307–325, 1989.

[45] A.-L. Mattila. *Piparikirja*. Atena, Jyväskylä, 2001. In Finnish.

[46] A. Medvedeva and A. Mukhopadhyay. An implementation of a linear time algorithm for computing the minimum perimeter triangle enclosing a convex polygon. In *Proc. Canadian Conference on Computational Geometry (CCCG 2003)*, pages 25–28, 2003.

[47] V. Milenkovic. Translational polygon containment and minimal enclosure using linear programming based restriction. In *Proc. 28th Annual ACM Symposum on the Theory of Computing*, pages 109–118, 1996.

[48] V. Milenkovic. Rotational polygon containment and minimum enclosure using only robust 2d constructions. *Computational Geometry: Theory and Applications*, 13(1):3–19, 1999.

[49] J. S. B. Mitchell and V. Polishchuk. Minimum-perimeter enclosing k-gon. *Information Processing Letters*, 107(3–4):120–124, 2008.

[50] D. M. Mount and R. Silverman. Minimum enclosures with specified angles. Technical Report CS-3219, University of Maryland, 1994.

[51] D. M. Mount, R. Silverman, and A. Y. Wu. On the area of overlap of translated polygons. *Computer Vision and Image Understanding*, 64(1):53–61, 1996.

[52] K. J. Nurmela and P. R. J. Östergård. More optimal packings of equal circles in a square. *Discrete & Computational Geometry*, 22(3):439–457, 1999.

[53] J. O'Rourke. Finding minimal enclosing boxes. *Internat. J. Comput. Inform. Sci.*, 14:183–199, 1985.

[54] H. Pirzadeh. Computational geometry with the rotating calipers. Master's thesis, McGill U, 1999.

[55] C. Schwarz, J. Teich, A. Vainshtein, E. Welzl, and B. L. Evans. Minimal enclosing parallelogram with application. In *Proc. 11th Annual Symposium on Computational Geometry*, pages C34–C35, 1995.

[56] M. Skopenkov. Packing a cake into a box. *American Mathematical Monthly*, 118(5):424–433, May 2011.

[57] K. Sugihara, M. Sawai, H. Sano, D.-S. Kim, and D. Kim. Disk packing for the estimation of the size of a wire bundle. *Japan Journal of Industrial and Applied Mathematics*, 21:259–278, 2004.

[58] S. Toledo. Extremal polygon containment problems. In *SoCG '91*, pages 176–185, New York, NY, USA, 1991. ACM.

[59] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proceedngs of IEEE MELECON '83*, pages 1–4, 1983.

[60] M. J. van Kreveld and M. Löffler. Approximating largest convex hulls for imprecise points. *J. Discrete Algorithms*, 6(4):583–594, 2008.

[61] M. J. van Kreveld and B. Speckmann. Cutting a country for smallest square fit. In P. Bose and P. Morin, editors, *Algorithms and Computation, 13th International Symposium, (ISAAC 2002), Vancouver, BC, Canada, November 21-23, 2002*, volume 2518 of *Lecture Notes in Computer Science*, pages 91–102. Springer, 2002.

[62] T. van Lankveld, M. J. van Kreveld, and R. C. Veltkamp. Identifying rectangles in laser range data for urban scene reconstruction. *Computers & Graphics*, 35(3):719–725, 2011.

[63] Y.-C. Xu, R.-B. Xiao, and M. Amos. Simulated annealing for weighted polygon packing. `http://arxiv.org/abs/0809.5005`.

[64] E. A. Yildirim. On the minimum volume covering ellipsoid of ellipsoids. *SIAM J. on Optimization*, 17(3):621–641, 2006.