*Proceedings of the 1996 ACM Symposium on the Theory of Computing (STOC)*, May 22-24, 1996, pp. 109-118.

# Translational Polygon Containment and Minimal Enclosure using Linear Programming Based Restriction

Victor J. Milenkovic*

### Abstract

We introduce and analyze a new technique *Linear Programming Based Restriction* (LP Restriction) for solving translational *containment* and *enclosure* problems. The containment task is to translate $k$ $m$-gons into a $n$-gon container. The enclosure task is to translate $k$ $m$-gons into a minimum area $n$-gon which is convex with fixed orientation edges. All running times are based on an assumption of fixed $k$ and are asymptotic in $m$ and $n$. Lower bounds are proved for containment in a nonconvex container: $\Omega((mn)^{2k})$ for nonconvex polygons and $\Omega(n^k)$ for convex polygons. LP restriction achieves upper bound $O((m^2+mn)^{2k}\log n)$ for nonconvex polygons and $O((mn)^k \log n)$ for convex polygons. The former almost matches the lower bound, and the latter does also for constant $m$. For arbitrary $m$ and $k > 3$, the latter is faster than any other known algorithm for translational containment of convex polygons. A proof is given that the area function for fixed-angle convex polygons is Lorentzian. Enclosure algorithms based on this result have running times $O((m^2+n)^{2k-2}(n+\log m))$ for nonconvex polygons and $O((m+n)^{2k}(n+\log m))$ or $O(m^{k-1}(n^{2k+1}+\log m))$ for convex polygons. We are not aware of any previous running times for $k$-polygon minimum area enclosure. Other containment and enclosure results are given.

LP restriction demonstrates a useful combination of mathematical programming and computational geometry which may be a paradigm for solving other tasks. We show that the containment and enclosure algorithms are useful in industry. Software based on LP restriction has been licensed to industry, and it is the fastest available solution to translational containment and enclosure for nonconvex polygons.

*University of Miami, Department of Math and Computer Science. Email: vjm@cs.miami.edu. This research was funded by the Textile/Clothing Technology Corporation from funds awarded to them by the Alfred P. Sloan Foundation, by NSF grant CCR-91-157993, and by a subcontract of a National Textile Center grant to Auburn University, Department of Consumer Affairs.

## 1 Introduction

A number of industries generate new parts by cutting them from stock material: cloth, leather (hides), sheet metal, *etc.* These industries require good solutions to *containment* or *minimal enclosure* problems. *Containment* is the question whether a given set of part shapes can be fit without overlapping into a given container. *Minimal enclosure* is the problem of minimizing the size (area, length, or other measure) of the container for a given set of parts. One might think that a few good layouts might suffice for a given industry, but such is not the case. One large apparel manufacturer "solves" about 1500 minimal enclosure problems *per week* (see Figure 1 for a typical layout). Human employees generate these near-optimal solutions by hand, and we estimate that their total compensation is about three million dollars per year for that one company. Large companies are highly automated, and they could easily substitute software for humans. However, no current software (even our own) can reliably find enclosures within one percent[1] of those generated by expert humans for the number of parts they routinely lay out (one to two hundred). (See [3, 9, 10, 17] for a survey of current results.)

Containment is NP-hard [4] even if the parts cannot rotate and if the parts and container are rectangles.[2] This hardness result assumes the problem size is the number $k$ of parts. We can probably never hope to exactly solve $k$-part containment or enclosure problems for $k > 100$, especially for nonconvex parts. However, in her Ph.D. thesis [3], Daniels demonstrates that the ability to solve translational containment or enclosure for modest values of $k$, such as $k \leq 10$, is a very useful tool in finding approximate[3] solutions to problems with $k > 100$. Detecting impossible containment problems is particularly useful. For instance, our software finds layouts of a dozen pairs of pants by first laying out the large front and back panels. Containment is then applied to place the smaller trim pieces in the unused patches of material. For other garments types, our enclosure algorithm could pack part subsets into rectangles and then one could lay out these subsets using known rectangle-packing strategies. In textile applications, fabric has a grain, and each part can have only up to eight orientations (hence

[1] One percent is about the break-even point of labor vs. material costs.

[2] The reader will recognize that containment is the "feasibility question" for minimal enclosure, and thus the appropriate complexity question is the hardness of containment.

[3] *Approximate* here means either placing almost all the parts in the container or finding a near-minimum-area container.

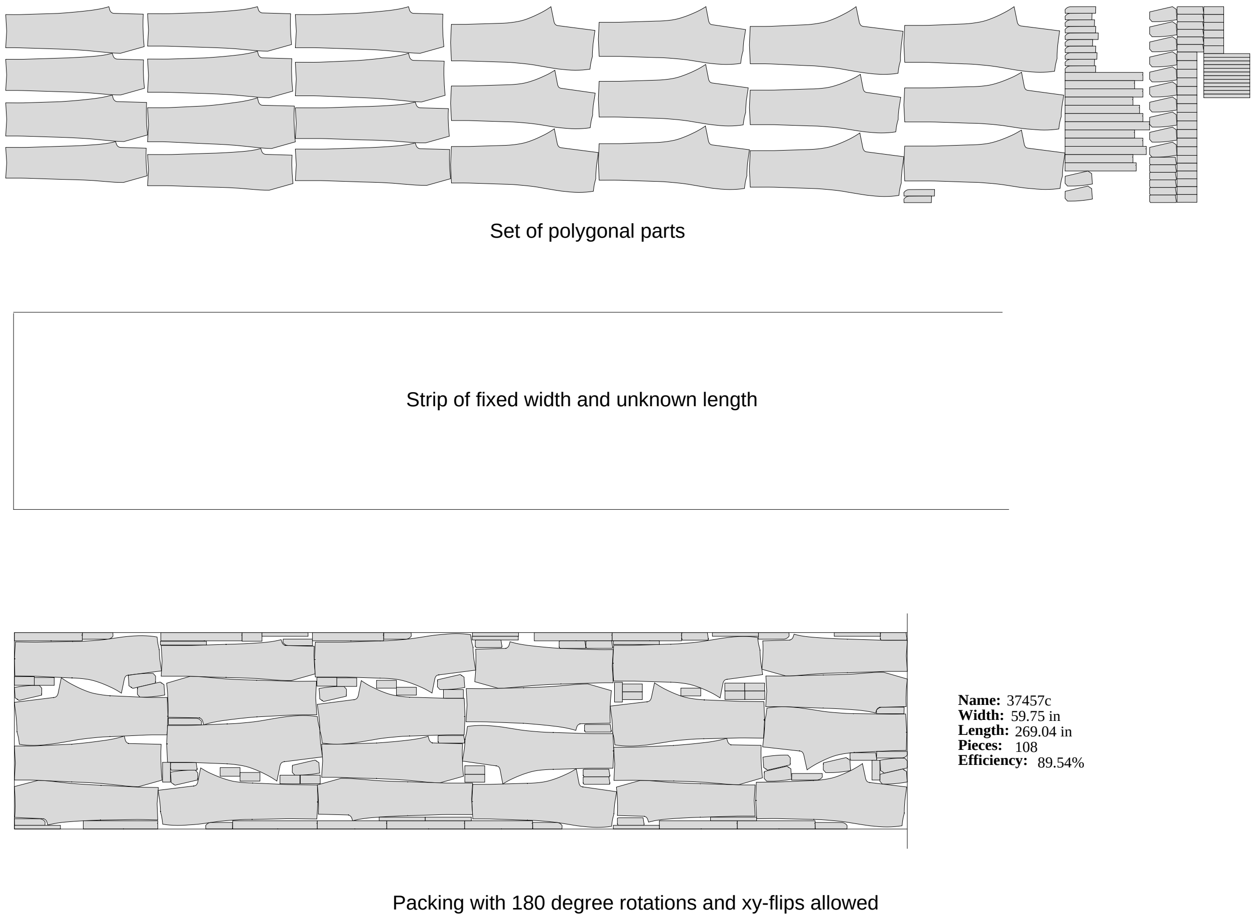


Figure 1: The Marker Making Task in the Apparel Industry (Parts are polygons with 4 to 100 vertices.)

one can iterate over these orientations). In recent work [6], we show how to apply translational containment algorithms to generate industrial-quality layouts of many parts with multiple orientations permitted. We conclude that purely translational containment and enclosure is useful and applicable, even for bounded values of $k$.

Our current software can place (translate) $k$ nonconvex $m$-gons into a nonconvex $n$-gon, or determine that no solution exists, in a few minutes on a desktop workstation for $k \leq 10$, $4 \leq m \leq 100$, and $100 \leq n \leq 400$. This software has been licensed by Gerber Garment Technologies, the largest provider of textile CAD/CAM software in the U.S., and they have incorporated it into their existing CAD/CAM software product.

The fastest version of our software uses an algorithm called *linear programming based restriction* (LP restriction). This paper describes LP restriction and gives a theoretical analysis of its running time for both containment and enclosure problems. This analysis is based on the assumption of *fixed* $k$ and expresses the running time as a function of $m$ and $n$. Because fixed-$k$ algorithms are useful, this is the analysis we really need. The following section describes the new results. Section 1.2 gives related work, and Section 1.3 gives an outline for the rest of the paper.

### 1.1 New Results.

We use the following shorthand: R–rectangle, C–convex, N–nonconvex. Thus the symbol $k$CN denotes the task of translating $k$ convex polygons into a nonconvex container, and "all-$k$CN" denotes finding *all* combinatorially distinct solutions, instead of just one. For enclosure problems, $k$NminR denotes translating $k$ nonconvex polygons into a rectangle of minimum area. For both "minR" and "minC", the edges of the container have fixed orientations. We do not assign a meaning to "minN". The polygons each have up to $m$ vertices and the container has $n$ vertices (unless it is a rectangle). When expressing running times, we assume fixed $k$. (The constant factor depends on $k$.) If the expression involves both $m$ and $n$, then $\log n$ really stands for $\max(\log m, \log n)$.

Two solutions to containment are *equivalent* if there is a nonoverlapping motion that transforms one to the other: they are in the same *connected component* of the set of valid configurations. We prove the following lower bounds on the number of connected components (equivalence classes):

| all-$k$NN | all-$k$NR | all-$k$RN |
|---|---|---|
| $\Omega((mn)^{2k})$ | $\Omega(m^{4k-4})$ | $\Omega(n^k)$ |

We give containment algorithms with the following running time bounds. Bounds are the same for one solution or all solutions:

| $k$CN | $k$CC | $k$CR |
|---|---|---|
| $O((mn)^k \log n)$ | $O((m+n)^k \log n)$ | $O(m^k \log m)$ |
| $k$NN | $k$NC | $k$NR |
| $O((m^2+mn)^{2k} \log n)$ | $O((m^2+n)^{2k} \log n)$ | $O(m^{4k} \log m)$ |

The running time for $k$RN is that of $k$CN with $m = 4$, namely $O(n^k \log n)$. Note that the running times for $k$NN and $k$RN are within a factor of $\log n$ of the lower bounds for these problems.

We also give algorithms to compute a minimal area convex enclosure with $n$ edges of fixed orientation:

$$k\text{NminC} \qquad O((m^2+n)^{2k-2}(n+\log m))$$

$$k\text{CminC} \qquad O((m+n)^{2k-2}(n+\log m)) \text{ or } O(m^{k-1}(n^{2k-1}+\log m))$$

One can set $n = 4$ to obtain the times for minimum rectangular enclosure:

| $k$NminR | $k$CminR |
|---|---|
| $O(m^{4k-4} \log m))$ | $O(m^{k-1} \log m))$ |

The lower bounds are fairly straightforward generalizations of known results. The running time for $k$NN requires an intricate version of LP restriction plus a very careful counting argument. The times for $k$NC and $k$NR are corollaries. It should be mentioned that there is a way to achieve the same bounds by "naive" enumeration, but this naive algorithm is totally impractical and is definitely not output-sensitive: running time is $\Omega((mn)^{2k})$ even if no solution exists. We view the proof for LP restriction as an important first step towards an output-sensitive argument. The LP restriction algorithm for $k$CN is not quite as intricate as the $k$NN algorithm, but it requires a different very careful counting argument. The running times for $k$CC and $k$CR are corollaries. We know of no other way to achieve these running times. To solve the enclosure problems, we must prove that the area function for fixed-orientation convex polygons is Lorentzian. Once we have done this, the algorithms for minimum convex enclosure are straightforward generalizations of the containment algorithms.

### 1.2 Related Work.

In general, industrial systems use heuristics which cannot detect infeasibility or determine if they have found the optimal enclosure. These systems use practical efficiency techniques such as inner and outer approximations of the parts, and thus the running time increases with the tightness of the fit. We have licensed our software including the algorithms described here, and we believe them to be superior to what is currently available.[4]

Fortune [11] gives a $O(mn \log mn)$ time solution to 1CN by computing the Minkowski sum using a generalized Voronoi diagram. Avnaim and Boissonnat [2, 1] use the Minkowski sum and convex decomposition to solve 1NN and 2NN in $O(m^2n^2 \log mn)$ and $O(m^4n^4 \log mn)$ time, respectively, and Avnaim [1] gives a $O(m^{14}n^6 \log mn)$ for 3NN. Devillers [8] gives faster algorithms for 2CN and 3CN with running times $O(m^2n^2 \log m)$ and $O(m^3n^3 \log m)$. Avnaim and Boissonnat also give a solution to the 3NP problem, three nonconvex polygons in a *parallelogram* container (this is equivalent to 3NR), using time in $O(m^{60} \log m)$ [2, 1].

In earlier work, Milenkovic, *et al.* [7] offer three approaches to translational containment. The first is fast algorithms for convex shapes: $O(mn \log n)$ for 2CN and $O(m^3 n \log n)$ for 3CN (see also [5, 15]). The second approach is an MIP (mixed integer programming) model for $k$NN containment (see [14] for full details). The MIP method takes a minute or two on a typical workstation for $k \leq 3$

[4] Companies closely guard their products, and there is no way we can actually compare our algorithms to others short of buying very expensive systems, although we do regularly attend trade shows. Incidentally, although we do not currently do so, we also could apply practical efficiency techniques, such as inner and outer approximations.

polygons, but is slow for $k \geq 4$. The third approach is a *geometric* algorithm for $k$NN which can find a layout with at most $2\epsilon$ overlap using time in $O((\frac{1}{\epsilon})^k \log(\frac{1}{\epsilon}) k^6 s \log s)$, where $s$ is a "practical" complexity measure on the input. There is no theoretical bound on $s$, but it ranges from $n$ to $mn$ in practice. The geometric containment algorithm uses *geometric restriction* which is neither strictly stronger nor strictly weaker than LP restriction. Daniels [3] has shown experimentally that geometric restriction is often stronger than LP restriction. We have not yet implemented an optimal algorithm for Minkowski sums of nonconvex polygons. When we do, the geometric containment algorithm might possibly be faster than the LP restriction based algorithm on some inputs.

More recently, we presented the *naive LP* algorithm [5, 15] with running time

$$O\left(\frac{(2kmn + k^2m^2)^{2k+1}}{k!} \mathrm{LP}(2k, 2kmn + k^2m^2)\right)$$

for $k$NN, where $\mathrm{LP}(a, b)$ is the time to solve a linear program with $a$ variables and $b$ constraints.[5]

A description of the "practical" versions of the algorithms presented here appears in a recent technical report [16]. This report gives the complete detailed LP restriction algorithm and analysis for $k$NN. It also describes a related method that uses an *overlap linear program* (OLP) for overlap minimization. In practice, using LP restriction plus overlap minimization is faster than LP restriction alone, but we have no proof that overlap minimization improves the theoretical bounds. The technical report also gives experimental results on industrial data.[6]

The algorithms presented here match or improve the running times of all previous algorithms except for our $O(mn \log n)$ 2CN and $O(m^3 n \log n)$ 3CN algorithms. In fact, we prove here a lower bound of $\Omega(n^k)$ time for all-$k$RN, yet these algorithms solve 2RN and 3RN in $O(n \log n)$ time! There is no contradiction because these algorithms return only a *single* solution. So far, we have not been able to generalize these algorithms to $k > 3$.

### 1.3 Outline.

Section 2 presents notation and background results on Minkowski sums. It proves the lower bounds for all-$k$NN, all-$k$NR, and all-$k$RN, and it describes the general framework of the LP restriction algorithm. Section 3 gives the algorithms for $k$CN, $k$CC, and $k$CR. Section 4 shows how to apply LP restriction to solve $k$NN, $k$NC, and $k$NR. Finally, Section 5 analyses the area function for fixed-angle convex polygons and gives the algorithms for $k$NminC and $k$CminC.

## 2 Background, Lower Bounds, and Framework

Section 2.1 provides background on the Minkowski sum and notation for the configuration spaces used by LP restriction. Section 2.2 proves lower bounds for containment. Section 2.3 defines LP restriction and gives the framework for a containment algorithm based on this technique.

[5] The latest revision of [15] gives a $O((mn)^{k+1} \log n)$ time algorithm for $k$CN which is similar to the $O((mn)^k \log n)$ algorithm presented here, although it does not use LP restriction. This revision has not yet been accepted for publication, and this algorithm has not appeared in any conference proceedings.

[6] See http://www.cs.miami.edu/~vjm and http://das-www.harvard.edu/users/students/Karen_Daniels/Karen_Daniels.html for recent papers.

### 2.1 Background and Notation.

#### 2.1.1 Minkowski Sum.

The *Minkowski sum* [12] of two point-sets (of $\mathbf{R}^2$ in the case of this paper) is defined

$$A \oplus B = \{a + b \,|\, a \in A,\, b \in B\}.$$

For a point-set $A$, let $\overline{A}$ denote the set complement of A and define $-A = \{-a \,|\, a \in A\}$. For a vector $t$, define $A + t = \{a + t \,|\, a \in A\}$. Note that $A + t = A \oplus \{t\}$. Let $|A|$ denote the number of vertices of $A$. Known complexities of $|A \oplus B|$ are: $\Theta(|A|^2|B|^2)$ for nonconvex $A$ and $B$ [13], $\Theta(|A||B|)$ for convex $A$ [11], $\Theta(|A|)$ for convex $\overline{A}$, $\Theta(|A| + |B|)$ for convex $A$ and $B$ [12].

#### 2.1.2 Configuration Spaces.

A containment algorithm translates $k$ polygonal regions $P_1, P_2, \ldots, P_k$ into a polygonal container $C$ without overlap. If we denote $P_0 = \overline{C}$ to be the complement of the container region, then containment is equivalent to the *placement* of $k+1$ polygons $P_0, P_1, P_2, \ldots, P_k$ in nonoverlapping positions. For translations, $t_i$ and $t_j$, $P_i + t_i$ and $P_j + t_j$ do not overlap if and only if $t_j - t_i \in U_{ij}$, where

$$U_{ij} = \overline{P_i \oplus -P_j}, \qquad 0 \leq i, j \leq k, i \neq j. \qquad (1)$$

The set $U_{ij}$ is the two-dimensional *configuration space* for placing $P_j$ with respect to $P_i$. Clearly, $U_{ij} = -U_{ji}$. Let $\mathcal{P}$ and $\mathcal{U}$ denote the lists of all $P_i$ and $U_{ij}$, respectively. A *configuration* of $\mathcal{P}$ is a list ($(k+1)$-tuple) of translations $\langle t_i \,|\, 0 \leq i \leq k \rangle$ where $t_0$ is arbitrarily set to $(0, 0)$. A *valid configuration* of $\mathcal{P}$ satisfies

$$t_j - t_i \in U_{ij}, \qquad 0 \leq i < j \leq k. \qquad (2)$$

A valid configuration is an exact solution to a translational containment problem. For enclosure problems, we do not represent the container as $\overline{P_0}$, and $i$ and $j$ are in the range $1 \leq i < j \leq k$.

### 2.2 Lower Bounds.

**Lemma 2.1** *For $k$NN, the set of valid configurations has $\Omega((mn)^{2k})$ components.*

**Proof:** In Figure 2, the container is the entire plane minus a set of $n/8$ rectangular "frames" (in black). Each polygon is a set of $m/4$ squares (in grey). The number of ways of putting two squares into two slots is $(m/8)(n/16)(m/8)(n/16)$ which is $\Omega((mn)^2)$. Assuming each polygon has small enough squares, we can make this choice independently for $k$ polygons. ■

**Corollary 2.2** *For $k$NR, there are $\Omega(m^{4k-4})$ components.*

**Proof:** Make the actual container very big. Let $\overline{P_k}$ play the role of the container. Apply previous lemma with $k - 1$ and $n = m$. ■

**Lemma 2.3** *For $k$RN, there are $\Omega(n^k)$ components.*

**Proof:** As in Figure 3, give the container $n' = (n-4)/8$ rooms, each big enough to hold a single rectangle $P_i$. The number of ways of placing rectangles into rooms is

$$n'(n'-1)(n'-2)\cdots(n'-k+1) = \Omega(n^k).$$

■

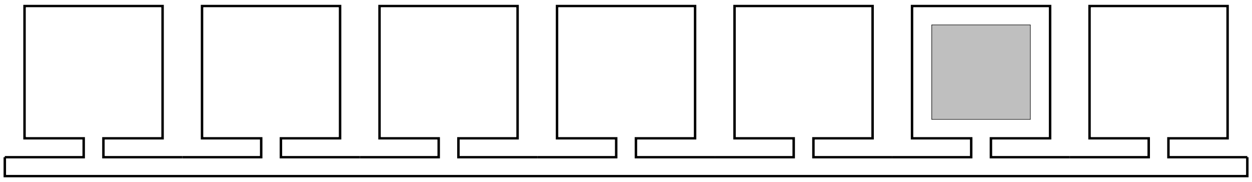

Figure 3: Lower Bound for $k$RN Containment: $n = 60$. (Container is interior of outlined region. Polygon $P_i$ is grey rectangle.)

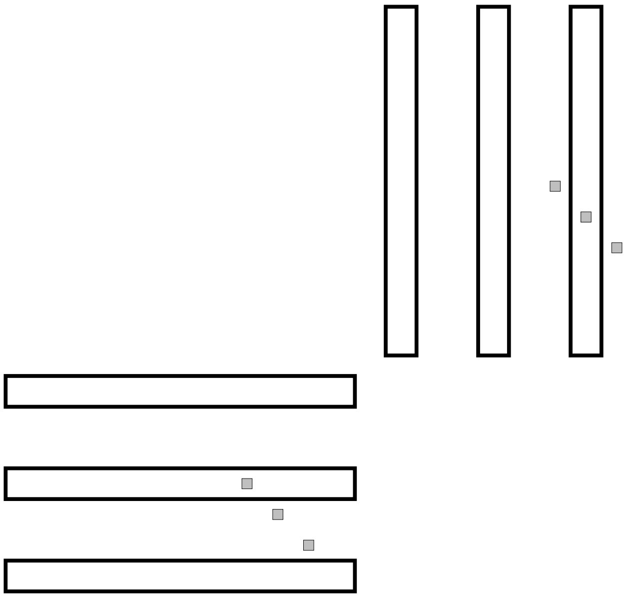

Figure 2: Lower Bound for $k$NN Containment: $n = 48$, $m = 24$. (Container is complement of (thin) black regions. Polygon $P_i$ is set of grey squares.)

### 2.3 Framework for LP Restriction.

This section gives the framework for solving containment using LP Restriction. LP restriction acts on *hypotheses.* A *hypothesis* is a list $\mathcal{C} = \langle C_{ij},\ 0 \le i < j \le k\rangle$ of convex polygons corresponding to the following assertion: there exists a valid configuration $\langle t_0, t_1, t_2, \ldots, t_k\rangle$ such that $t_j - t_i \in C_{ij} \cap U_{ij}$, $0 \le i < j \le k$. *Restriction* of a hypothesis replaces each $C_{ij}$ by a subset. The restriction is *valid* if the hypothesis retains the same truth value. *Evaluation* attempts to find a valid configuration for the hypothesis. *Subdivision* splits a hypothesis $\mathcal{C}$ into $\mathcal{C}^-$ and $\mathcal{C}^+$ such that $\mathcal{C}$ is equivalent to $\mathcal{C}^-$ *or* $\mathcal{C}^+$.

#### 2.3.1 LP Restriction.

The constraint that $t_j - t_i \in C_{ij}$ can be represented by $|C_{ij}|$ linear inequalities, where $|C_{ij}|$ denotes the number of edges of $C_{ij}$. The *constraint linear program* (CLP) is the collection of all these inequalities, $0 \le i < j \le k$. For each $g, h$, $0 \le g < h \le k$, LP restriction replaces $C_{gh}$ by the *range* $C'_{gh}$ of feasible values of $t_h - t_g$ under the constraints of the CLP.

Using the CLP as an oracle, the LP restriction algorithm constructs $C'_{gh}$. Solving the CLP with objective (to maximize) $d \cdot (t_h - t_g)$ yields the vertex $\text{ORACLE}(\mathcal{C}, g, h, d)$ of $C'_{gh}$ which is farthest in the direction $d$. The algorithm first calls the oracle with $d = (1,0)$, $(0,1)$, $(-1,0)$, and $(0,-1)$ to generate a convex polygon $Q$. For each edge $ab$ of $Q$, it calls the oracle with $d_{ab}$, the outward normal vector to this edge. If this generates a new vertex $c$, then it replaces $ab$ by $ac$ and $cb$ and recurses. Otherwise, $ab$ is an edge of $C'_{gh}$. Running time is output-sensitive: $2|C'_{gh}|$ calls to the oracle. The $k$NN algorithm in Section 4 generates the range in a theoretically more efficient manner. Naturally, if the range $C'_{gh}$ is empty (infeasible CLP) or if $C'_{gh} \cap U_{gh} = \emptyset$, then we can discard the hypothesis $\mathcal{C}$.

#### 2.3.2 Solving Containment.

For a containment algorithm, the initial input hypothesis is a list of outer approximations $C_{ij}$, $U_{ij} \subseteq C_{ij}$, $0 \le i < j \le k$: bounding boxes or convex hulls, for instance. The algorithm first applies LP restriction to its input. If for some $g, h$, $C_{gh} \cap U_{gh}$ is not convex, then it chooses a line $L$ to cut $C_{gh}$ into two halves $C^-_{gh}$ and $C^+_{gh}$. The algorithm subdivides $\mathcal{C}$ into $\mathcal{C}^-$ and $\mathcal{C}^+$ by replacing this particular $C_{gh}$ by either $C^-_{gh}$ of $C^+_{gh}$. Then it recurses on each sub-hypothesis $\mathcal{C}^-$ and $\mathcal{C}^+$. Otherwise (if every $C_{ij} \cap U_{ij}$ is convex, $0 \le i < j \le k$), the algorithm *evaluates* the hypothesis: it replaces each $C_{ij}$ by $C_{ij} \cap U_{ij}$ and applies restriction. If the resulting $C_{ij}$ are not empty, the set of configurations $t_0, t_1, \ldots, t_k$ satisfying $t_j - t_i \in C_{ij}$, $0 \le i < j \le k$ will be a convex set of valid configurations.

**Theorem 2.4** *Assuming it terminates, the LP restriction based containment algorithm generates at least one solution in each connected component of the space of valid configurations.*

**Proof:** LP restriction cannot "throw away" any valid solutions. Subdivision also preserves the invariant that every solution satisfying $\mathcal{C}$ will either satisfy $\mathcal{C}^-$ or $\mathcal{C}^+$. Evaluation generates a convex subset of the set of valid configurations. Thus the containment algorithm generates a convex cover of the set of valid configurations. The theorem follows. ■

In practice, we stop when we have found one solution. We use an extension of the CLP called the *overlap linear program* (OLP) to encourage the algorithm to find a solution with as little subdivision as possible [16].

### 3 $k$CN, $k$CC, and $k$CR Containment

The $k$CN containment algorithm partitions each $U_{ij}$ into $y$-monotone subsets. It then applies the LP restriction containment framework of the previous section to solve containment for the $y$-monotone configuration spaces. The following section presents the "$y$-monotone" containment algorithm. Section 3.2 describes the monotone partitions and proves the running of the overall containment algorithm is $O((mn)^k \log n)$.

### 3.1 $y$-Monotone Configuration Spaces.

This section presents an algorithm for $k$CN under the assumption that each $U_{ij}$ is *$y$-monotone*: each horizontal line intersects $U_{ij}$ in a single line segment, a single point, or the empty set. For this version of LP restriction, each $C_{ij}$ is an infinite horizontal strip of the form $a_{ij} \geq y \geq b_{ij}$. Each constraint of the CLP is of the form $y_j - y_i \leq a_{ij}$ or $y_j - y_i \geq b_{ij}$, where $t_i = (x_i, y_i)$, $0 \leq i < j \leq k$.

Initially, the algorithm sets $a_{ij}$ and $b_{ij}$ to be the maximum and minimum $y$-coordinate of $U_{ij}$, $0 \leq i < j \leq k$ (or $+\infty$ and/or $-\infty$ in case of unbounded $U_{ij}$). To accomplish splitting, the algorithm picks $i, j$ such that there exists vertex $v_{ij} = (x_{ij}, y_{ij})$ of $U_{ij}$ in the interior of $C_{ij}$. It uses the splitting line $y = y_{ij}$. If no such $v_{ij}$ exists, then all $C_{ij} \cap U_{ij}$ will be convex trapezoids, and it applies evaluation.

**Lemma 3.1** *Assuming $|U_{ij}| \leq 2m$, $1 \leq i < j \leq k$, the "$y$-monotone" $k$CN algorithm runs in time*

$$\mathrm{O}\left( (\log n) \cdot \prod_{1 \leq i \leq k} (|U_{0i}| + m) \right).$$

**Proof:** (Summary.) When we solve the CLP, the output configuration will have a set of *critical constraints*: a constraint is critical if $t_j - t_i = a_{ij}$ (or $b_{ij}$). Under the assumption of general position, no more than $k$ constraints can be critical. (Recall that $t_0 = (0,0)$, and so there are only $k$ variables $y_i$.)

The hypotheses we visit form a binary tree: each $\mathcal{C}$ has children $\mathcal{C}^-$ and $\mathcal{C}^+$. We claim that we can label each hypothesis with a unique set $\mathrm{CRIT}(\mathcal{C})$ of $k$ constraints: let $\mathrm{CRIT}(\mathcal{C})$ be the critical constraints from one of the calls to ORACLE when we restrict $\mathcal{C}$. We can assume that $\mathrm{CRIT}(\mathcal{C})$ contains a constraint corresponding to its parent's splitting line. If $\mathcal{C}'$ is an ancestor of $\mathcal{C}$, then $\mathrm{CRIT}(\mathcal{C})$ cannot contain this constraint. If $\mathcal{C}'$ is a "cousin" of $\mathcal{C}$, then the only way they can have a common configuration is if $t_j - t_i$ lies on the cutting line $y = y_{ij}$ of their nearest common ancestor. Yet, even in that case one critical set will contain $y \leq y_{ij}$ and the other will contain $y \geq y_{ij}$, so they will not be the same.

The cost of solving a CLP depends only on $k$, which is fixed. Each constraint corresponds to a vertex of some $U_{ij}$. The running time is the number of ways of selecting $k$ vertices from the $U_{ij}$ polygons. The $\log n$ factor comes from binary search to determine the subset of vertices of $U_{ij}$ inside $C_{ij}$. Finally, the cost of evaluation is constant because each $C_{ij} \cap U_{ij}$ is a trapezoid. ■

### 3.2 Nonmonotone Configuration Spaces.

Now we return to the original $k$CN problem. The $k$CN algorithm separates each $U_{ij}$ into connected components and then cuts each component into $y$-monotone components by extending a cut upwards from every reflex local $y$-maximum and downwards from every reflex local $y$-minimum (see Figure 4). Then it calls the "$y$-monotone" $k$CN algorithm with every way of selecting a single $y$-monotone component from each $U_{ij}$.

**Lemma 3.2** *For $1 \leq i < j \leq k$, $U_{ij}$ has two $y$-monotone components with up to $2m$ vertices. For $1 \leq i \leq k$, $U_{0i}$ has $\mathrm{O}(n)$ $y$-monotone components with a total of $\mathrm{O}(mn)$ vertices.*

**Proof:** For $1 \leq i < j \leq k$, $P_i$ and $P_j$ are convex with up to $m$ vertices, and therefore $P_i \oplus -P_j$ is convex with up to

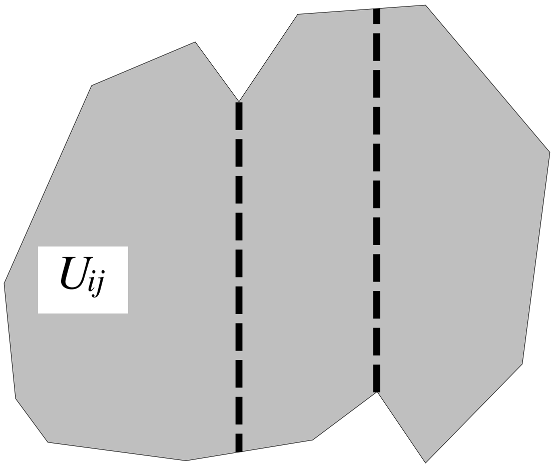


Figure 4: Partitioning a Polygon into $y$-Monotone Components

$2m$ vertices [12]. Its complement $U_{ij} = \overline{P_i \oplus -P_j}$ has two reflex local $y$-extremes.

Fortune [11] proved $P_0 \oplus -P_i$ has $\mathrm{O}(|P_0||P_i|) = \mathrm{O}(mn)$ vertices if $P_i$ is convex. He used a generalized Voronoi diagram. Each component of $U_{0i} = \overline{P_0 \oplus -P_i}$ must contain a Voronoi vertex, and there are only $\mathrm{O}(n)$ of these. Each reflex $y$-maximum of $U_{0i}$ is a sum of a reflex $y$-maximum of $C = \overline{P_0}$ and a $y$-maximum of $P_i$. But $P_i$ has only one $y$-maximum. Thus $U_{ij}$ has only $\mathrm{O}(n)$ reflex $y$-maxima (or minima). ■

**Theorem 3.3** *The $k$CN algorithm has running time in $\mathrm{O}((mn)^k \log n)$.*

**Proof:** Since each $U_{ij}$, $1 \leq i < j \leq k$, has only two $y$-monotone components, there are only a constant number of ways of selecting a component from each.[7] The cost of iterating over all ways of selecting a component $U'_{0i} \subseteq U_{0i}$ is the sum of

$$\mathrm{O}\left( (\log n) \cdot \prod_{1 \leq i \leq k} (|U'_{0i}| + m) \right),$$

over all such selections. By the distributive property, this is

$$\mathrm{O}\left( (\log n) \cdot \prod_{1 \leq i \leq k} (|U_{0i}| + m\mathrm{O}(n)) \right) = \mathrm{O}((mn)^k \log n).$$

■

For $k$CC and $k$CR, each $U_{0i}$ is *already* $y$-monotone and has complexity $\mathrm{O}(n)$ and $\mathrm{O}(1)$, respectively. Our claimed running times follow from Lemma 3.1. The running time for $k\overline{\mathrm{CC}}$ (the complement of the container is convex) is the same as for $k$CC since each $U_{0i}$ can be covered by two $y$-monotone regions with $\mathrm{O}(m+n)$ vertices.

## 4 $k$NN Containment

The $k$NN containment algorithm follows the framework of Section 2.3. In this case, the initial value of $C_{ij}$ is the convex

[7] The constant $2^{k(k-1)/2}$ is odious, but it is possible to reduce it to $k!$ which is reasonable (compared to $m^k$) [15].

hull of $U_{ij}$, $0 \leq i < j \leq k$ (some of these $C_{ij}$ are equal to the entire plane). After restriction, if some $C_{ij} \cap U_{ij}$ is not convex, then some edge $e$ of this $U_{ij}$ intersects the interior of $C_{ij}$. The algorithm splits $C_{ij}$ using the line $L$ containing $e$. It is necessary to modify this algorithm to achieve the $O((m^2 + mn)^{2k} \log n)$ running time bound. Section 4.1 gives the running time without modifications. Section 4.2 shows how to reduce the number of calls to the ORACLE to $O((m^2+mn)^{2k})$. Finally, Section 4.3 shows how to dispense with the expensive ORACLE function and replace it with a $O(\log n)$ time operation.

### 4.1 Counting Range Vertices.

In practice, we call the ORACLE function (Section 2.3.1) to compute the vertices of each range. Each time the oracle generates a new vertex, we can label it with the critical constraints of that solution to that CLP. This time, there are $2k$ critical constraints because we now have $2k$ variables, the $x$ and $y$ coordinates of $t_1, t_2, \ldots, t_k$. For a given hypothesis $\mathcal{C}$, the same set of critical constraints can label at most one vertex in each restricted $C'_{gh}$. The same reasoning as the proof of Lemma 3.1 applies. 1) Each *new* vertex in some restricted $C'_{gh}$ clearly has not appeared in any ancestor hypothesis. 2) If it appears as a new vertex in a "cousin" hypothesis, its label will be different: one of the critical constraints will have a $\leq$ instead of a $\geq$. Hence each label appears $O(k^2) = O(1)$ times.

### 4.2 Reducing Calls to ORACLE.

The LP restriction algorithm of Section 2.3.1 is wasteful because it recalculates many old vertices. This did not matter for $k$CN because the extra cost only depends on $k$. A faster $k$NN algorithm does the following when splitting $C_{ij}$. Let $L$ be the splitting line, let $ab = L \cap C_{ij}$, and let $C^-_{ij}$ and $C^+_{ij}$ be the two "halves". For $1 \leq g < h \leq k$, $g \neq i$ or $h \neq j$, define $C^-_{gh}$ to be the result of replacing $C_{ij}$ by $C^-_{ij}$ and restricting $C_{gh}$. Define $C^+_{gh}$ analogously. Subdividing $\mathcal{C}$ into $\mathcal{C}^-$ and $\mathcal{C}^+$ and restricting both sub-hypotheses is equivalent to calculating all the $C^-_{gh}$ and $C^+_{gh}$ polygons. Instead, to do this more efficiently, the algorithm computes $C^0_{gh}$: the result of replacing $C_{ij}$ by $ab$ and restricting $C_{gh}$. It is clear that $C^0_{gh} = C^-_{gh} \cap C^+_{gh}$ and it only contains *new* vertices. Assuming the vertices of each $C_{ij}$ are stored in the appropriate data structure (such as a balanced binary tree), the algorithm can compute $C^-_{gh}$ and $C^+_{gh}$ from $C_{gh}$ and $C^0_{gh}$ using time $O(|C^0_{gh}| \log |C_{gh}|)$. Details are given in a technical report [16].

### 4.3 Final $k$NN Algorithm.

In fact, it is possible to dispense with the ORACLE and compute each vertex of $C^0_{gh}$ in time $O(\log n)$. Suppose the algorithm is at a vertex $v$ of $C^0_{gh}$. It computes the neighboring vertices of $C^0_{gh}$ as follows. For each of the $2k$ critical constraints of $v$, it removes one constraint and computes the line $\mathcal{L}$ in $\mathbf{R}^{2k}$ corresponding to the remaining $2k - 1$. It projects $\mathcal{L}$ into each $C_{ij}$ and intersects it with $C_{ij}$. It projects the resulting segments back onto $\mathcal{L}$. It projects the intersection of these segments into $C_{gh}$. Out of the $2k$ resulting segments, the two forming the largest angle are the edges of $C^0_{gh}$ neighboring $v$.

**Theorem 4.1** *The LP restriction algorithm for* all-$k$NN *runs in* $O((m^2 + mn)^{2k} \log n)$ *time.*

**Proof:** There are $O(m^2)$ or $O(mn)$ (which are both $O(m^2 + mn)$) ways that a vertex of $P_i$ can touch the extension line of an edge of $P_j$. This implies that the edges of $U_{ij}$ lie on $O(m^2 + mn)$ lines. Each such line can contribute at most two vertices to the convex hull of $U_{ij}$ (the initial $C_{ij}$). Thus the total number of constraints is $O(m^2 + mn)$. The total number of vertex labels (and hence vertices) is $O((m^2 + mn)^{2k})$. The cost of generating each new vertex is logarithmic in the number of previously generated vertices, which is $O(\log(m^2+mn)^{2k}) = O(\log n)$.[8] All other costs are $O(\log n)$ per vertex. Details are given in a technical report [16]. ∎

For $k$NC, each $|U_{0i}|$ has $O(n)$ edges, and for $k$NR, each $U_{0i}$ is a rectangle. The running times, $O((m^2 + n)^{2k} \log n)$ and $O(m^{4k} \log m)$ follow from counting sets of $2k$ critical constraints.

## 5 $k$NminC and $k$CminC Enclosure Problems

This section gives an algorithm for the $k$NminC and $k$CminC enclosure problems: place $k$ polygons into the minimum area convex polygon with edges of fixed orientation. We define the area function and prove that it is Lorentzian. As a consequence, for any convex set of configurations, the minimum area configuration appears at a vertex. Since our containment algorithms generate a convex cover of the set of valid configurations, we can compute the minimum area enclosure by visiting all the vertices.

### 5.1 Convex (Lorentzian) Area Function.

Let $d_1, d_2, d_3, \ldots, d_n$ be a fixed set of unit vectors in counterclockwise order about the origin. Define

$$\mathrm{AREA}(r_1, d_1, r_2, d_2, \ldots, r_n, d_n)$$

to be the area of a convex polygon bounded by lines

$$d_l \cdot (x, y) \leq r_l, \qquad 1 \leq l \leq n.$$

In other words, the edge $e_i$ of the polygon has outward unit normal vector $d_i$, and its distance to the origin is $r_i$. (We allow the case in which the foot of the altitude from the origin to the line of the edge does not lie on the edge.) This section gives a formula for AREA and proves that it is Lorentzian.

#### 5.1.1 Convexity Theorem.

This section gives a formula for AREA and states the theorem that it is a convex function on the domain $r_i > 0$, $i = 1, 2, \ldots, n$.

**Theorem 5.1** *For any fixed area $\alpha > 0$, the set of points of the form $(r_1, r_2, r_3, \ldots, r_n)$ in $\mathbf{R}^n$ such that $r_i > 0$, $i = 1, 2, 3, \ldots, n$ and*

$$\mathrm{AREA}(r_1, d_1, r_2, d_2, \ldots, r_n, d_n) \geq \alpha$$

*is convex.*

More generally, the set of all $(r_1, r_2, \ldots, r_n)$ with AREA $=$ $\alpha$ consists of two convex hyperbolic sheets, only one of which intersects the "positive quadrant": $r_i > 0$, $i = 1, 2, \ldots, n$. To prove Theorem 5.1, we must demonstrate that AREA

[8] Recall from Section 1.1 that we use $\log n$ in place of $\max(\log m, \log n)$.

is a *Lorentzian* quadratic form: it has exactly one positive eigenvalue.[9] Furthermore, the corresponding eigenvector must have all positive components. These facts are enough to establish the theorem.

Actually, this section proves the theorem in a slightly less direct manner. First, it establishes Theorem 5.1 when $d_1, d_2, \ldots, d_n$ are equi-angular about the origin. Second, the result is extended to *rational* angles: the angle between $d_i$ and $d_{i+1}$ is a rational multiple of $2\pi$. A continuity argument proves the theorem for general $d_1, d_2, \ldots, d_n$.

### 5.1.2 Formula for Area.

**Lemma 5.2**

$$\mathrm{AREA}(r_1, d_1, r_2, d_2, \ldots, r_n, d_n) = \sum_{i=1}^{n} \frac{2r_i r_{i+1} - (r_i^2 + r_{i+1}^2)\cos\theta_i}{2\sin\theta_i}, \tag{3}$$

*where $r_{n+1} = r_1$, $d_{n+1} = d_1$, and $\theta_i$ is the angle between $d_i$ and $d_{i+1}$.*

**Proof:** As depicted in Figure 5, suppose edge $e_i$ and $e_{i+1}$ share common vertex $v_i$. Since $v_i$ lies on both edges, it follows that $v_i \cdot d_i = r_i$ and $v_i \cdot d_{i+1} = r_{i+1}$. Setting $v_i = ad_i + bd_{i+1}$ and solving for $a$ and $b$ yields,

$$v_i = \frac{r_i - r_{i+1} d_i \cdot d_{i+1}}{1 - (d_i \cdot d_{i+1})^2} d_i + \frac{r_{i+1} - r_i d_i \cdot d_{i+1}}{1 - (d_i \cdot d_{i+1})^2} d_{i+1} \tag{4}$$

$$= \frac{r_i - r_{i+1}\cos\theta_i}{1 - \sin^2\theta_i} d_i + \frac{r_{i+1} - r_i\cos\theta_i}{1 - \sin^2\theta_i} d_{i+1}. \tag{5}$$

The region of the polygon inside the angle between $d_i$ and $d_{i+1}$ has area,

$$\begin{aligned} A_i &= \frac{(r_i d_i) \times v_i + v_i \times (r_{i+1} d_{i+1})}{2} \\ &= \left( r_i \frac{r_{i+1} - r_i \cos\theta_i}{2(1 - \sin^2\theta_i)} + r_{i+1} \frac{r_i - r_{i+1}\cos\theta_i}{2(1 - \sin^2\theta_i)} \right) d_i \times d_{i+1} \\ &= \frac{2r_i r_{i+1} - (r_i^2 + r_{i+1}^2)\cos\theta_i}{2\sin\theta_i}. \end{aligned}$$

The area of the polygon is the sum over all such regions. ■

### 5.1.3 Equal Angle Case.

**Lemma 5.3** *Theorem 5.1 holds if $\theta_1 = \theta_2 = \theta_3 = \cdots = \theta_n$.*

**Proof:** Let $\theta = 2\pi i/n$. The vector $(r_1, r_2, \ldots, r_n)$ is an eigenvector with eigenvalue $\lambda_j$ of the AREA function (of its symmetric matrix) if

$$\frac{r_{i-1} - 2(\cos\theta) r_i + r_{i+1}}{\sin\theta} = \lambda_j r_i, \qquad i = 1, 2, 3, \ldots, n.$$

(The left hand is just the partial derivative of Equation 3 with respect to $r_i$.) Let $\omega = \cos\theta + i\sin\theta$ be the primitive $n$th root of unity. We claim that for $j = 0, 1, 2, \ldots, n-1$, the complex vector $(w^j, w^{2j}, w^{3j}, \ldots, w^{nj})$ is an eigenvector with eigenvalue

$$\lambda_j = 2\frac{\cos j\theta - \cos\theta}{\sin\theta}.$$

[9] A quadratic form $\sum a_{ij} r_i r_j$ corresponds to a symmetric matrix $[a_{ij}]$. In this paper, the eigenvectors and eigenvalues of the quadratic form are defined to be those of the symmetric matrix.

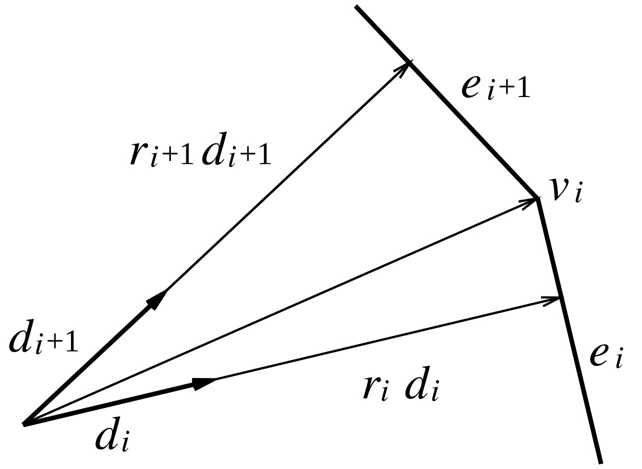


Figure 5: Direction vectors $d_i$, $d_{i+1}$. Edges $e_i$ and $e_{i+1}$ at distance $r_i$ and $r_{i+1}$. Vertex $v_i$.

Note that this eigenvector is positive only for $j = 0$. It is zero for $j = 1$ or $j = n-1$, and it is negative for $j = 2, 3, 4, \ldots, n-2$. One can prove the claim by substitution for $\lambda_j$ and $r_i$ $(= w^{ij})$, $r_{i-1}$, and $r_{i+1}$. Divide out a factor of $w^{ij}$ and apply the identity $w^j + w^{-j} = 2\cos j\theta$.

The claimed eigenvector is complex, but taking the real or imaginary parts yields $n$ real eigenvectors:

$$\begin{array}{ll} (1, 1, 1, \ldots, 1); & \\ (\cos j\theta, \cos 2j\theta, \cos 3j\theta, \ldots, \cos nj\theta), & 0 < j < n/2; \\ (\sin j\theta, \sin 2j\theta, \sin 3j\theta, \ldots, \sin nj\theta), & 0 < j < n/2; \\ (1, -1, 1, -1, \ldots, 1, -1), & \text{if } n \text{ is even.} \end{array}$$

The only eigenvector with positive eigenvalue is $(1, 1, 1, \ldots, 1)$ which clearly points into the "positive quadrant".

Suppose that $(x_1, x_2, \ldots, x_n)$ are the coordinates of $(r_1, r_2, \ldots, r_n)$ with respect to the orthonormal basis of eigenvectors. Under this change of coordinates, the area becomes $\lambda_1 x_1^2 + \lambda_2 x_2^2 + \lambda_3 x_3^2 + \cdots + \lambda_n x_n^2$. The constraint AREA $\geq \alpha$ can be written,

$$\lambda_1 x_1^2 \geq -(\lambda_2 x_2^2 + \lambda_3 x_3^2 + \cdots + \lambda_n x_n^2) + \alpha.$$

Since only $\lambda_1$ is positive, this is indeed a pair of hyperboloids, only one of which lies in the direction of the first eigenvector. This fact implies Theorem 5.1 for the equi-angle case. ■

### 5.1.4 Rational Angle Case.

**Lemma 5.4** *Theorem 5.1 is also true if the angles of $d_1, d_2, d_3, \ldots, d_n$ are rational multiples of $2\pi$.*

**Proof:** Let $n$ be the common denominator of all the rational multiples. Lemma 5.3 implies that the theorem is true if we include *all* rational angles with that denominator. To have only *some* angles with that denominator, we need to selectively eliminate some of the edges.

Suppose we want to keep edges $e_h$ and $e_j$, but we want to eliminate $e_i$, $h < i < j$. Slightly generalizing Equation 5 tells us that the intersection of $e_h$ and $e_j$ is

$$v_{hj} = \frac{r_h - r_j\cos\theta_{hj}}{1 - \sin^2\theta_{hj}} d_h + \frac{r_j - r_h\cos\theta_{hj}}{1 - \sin^2\theta_{hj}} d_j,$$

where $\theta_{hj}$ is the angle between $d_h$ and $d_j$. Edge $e_i$ will have zero length if its line passes through $v_{hj}$: $r_i = v_{hj} \cdot d_i$. This is a homogeneous linear constraint on $r_h$, $r_i$, $r_j$. Adding this constraint is equivalent to intersecting the convex region $\mathrm{AREA}(r_1, d_1, r_2, d_2, \ldots, r_n, d_n) \geq \alpha$ with a hyperplane through the origin–which of course yields another convex region. Projecting into the space of "live" dimensions, all $r_i$ such that $i/n$ is one of the desired fractions, also preserves convexity. ■

#### 5.1.5 Proof of Lorentzian Property.

Finally, we can prove Theorem 5.1.

**Proof:** We can approximate the angles of $d_1, d_2, d_3, \ldots, d_n$ by rational multiples of $2\pi$. The eigenvalues are roots of the characteristic polynomial and therefore vary continuously with angle. Taking the limit with better and better rational approximations preserves the signs of the eigenvalues and the signs of the components of the unique eigenvector with positive eigenvalue. ■

### 5.2 Enclosure Algorithms.

Without increasing the running time of any of the containment algorithms, we can enumerate all vertices of each convex cover of the set of valid configurations. The following section shows that Theorem 5.1 implies that the minimum area enclosure occurs at a vertex, although we might have to add $O(n)$ extra constraints. Using Equation 3, we can compute the area of the enclosure at each vertex in $O(n)$ time. Section 5.2.2 describes how to add the extra constraints and otherwise modify the containment algorithms to solve the minimum enclosure problem.

#### 5.2.1 Applying Theorem 5.1.

In an enclosure problem, there is no container polygon. Instead, we let $P_k$ take on the role of $P_0$ and set $t_k = (0, 0)$. For a given configuration $\langle t_1, t_2, t_3, \ldots, t_k = (0, 0)\rangle$ of $P_1, P_2, P_3, \ldots, P_k$, the value of $r_j$ depends on which polygon $P_i + t_i$ "sticks out" farthest in the direction of $d_j$. Assume for the moment that for each $d_j$, this polygon is unique. In this case, $r_1, r_2, r_3, \ldots, r_n$ depend linearly on $t_1, t_2, \ldots, t_n$.

**Lemma 5.5** *If $r_1, r_2, r_3, \ldots, r_n$ depend linearly on $t_1, t_2, \ldots, t_n$ at the solution to minimum area enclosure, then $\langle t_1, t_2, t_3, \ldots, t_k\rangle$ is a vertex of one of the convex subsets of the space of valid configurations.*

**Proof:** Suppose the minimum area configuration lies in a convex subset $\mathcal{C}$ of the set of valid configurations but it does not lie at a vertex of $\mathcal{C}$. (Assume it lies on the lowest dimensional facet possible.) Let the area be $\alpha$. Then there must be some direction

$$\Delta t = \langle \Delta t_1, \Delta t_2, \Delta t_3, \ldots, \Delta t_k\rangle$$

such that the configuration can move in both of the two directions $\Delta t$ or $-\Delta t$ without leaving $\mathcal{C}$. By Theorem 5.1, the set $\mathrm{AREA} \geq \alpha$ is convex, and therefore one of these two directions will *not* point into the interior of this convex set. Therefore, we can move the configuration in some direction that does not increase the area of the enclosure. We can continue this motion until the configuration is about to leave $\mathcal{C}$, at which point the configuration is on a lower dimensional facet of $\mathcal{C}$. This contradicts the assumption that the starting facet was of the lowest possible dimension. ■

Suppose *two* polygons $P_h + t_h$ and $P_i + t_i$ "stick out" equally far in the direction $d_j$. Then the mapping from $t_1, t_2, \ldots, t_n$ to $r_1, r_2, r_3, \ldots, r_n$ will *not* be linear (it will be only piecewise linear) *unless* we add the constraint that $P_h + t_h$ and $P_i + t_i$ stick out equally in that direction. Let $v_{ij}$ be the vertex of $P_i$ that has maximum value of $v_{ij} \cdot d_j$ over all vertices of $P_i$. The additional constraint on $t_h$ and $t_i$ is

$$(t_h + v_{hj}) \cdot d_j = (t_i + v_{ij}) \cdot d_j. \qquad (6)$$

Since there are $k$ polygons and $n$ directions $d_j$, there are $k(k-1)n/2 = O(n)$ such constraints.

#### 5.2.2 Modifying Containment Algorithms.

To solve $k$NminC enclosure, we simply add the $O(n)$ extra linear constraints of Equation 6. There are $O(m^2)$ constraints arising from possible vertex-edge (actually, edge extension) contacts among the polygons. Also, at each vertex of the configuration space, evaluating the area (Equation 3) uses $O(n)$ time. Since there are only $2(k-1)$ variables (the coordinates of $t_1, t_2, t_3, \ldots, t_{k-1}$), the total time to find the minimum enclosure is $O((m^2+n)^{2k-2}(n + \log m))$.

As far as $k$CminC is concerned, each pair of convex polygons can have at most $O(m)$ vertex-edge contacts. Applying the same reasoning as the previous paragraph yields a running time of $O((m+n)^{2k-2}(n + \log m))$ for $k$CminC enclosure.

If $m$ is big and $n$ is small, for instance if the enclosure is a rectangle ($n = 4$), then we can obtain a better running time by modifying the $k$CN algorithm of Section 3. Since the "container" $\overline{P_k}$ is the complement of a convex polygon, generating a convex cover of the valid configurations takes $O(m^{k-1} \log m)$ time, the same as $(k-1)\mathrm{C}\overline{\mathrm{C}}$ containment with $n = m$. The cover consists of $O(m^{k-1})$ convex regions, each with constant complexity. Redefine $C_{ij}$ to stand for the convex range of $t_j - t_i$ over all valid configurations $\langle t_1, t_2, t_3, \ldots, t_k\rangle$. Each $C_{ij}$ has constant complexity. As the previous section stated, there are $O(n)$ additional constraints (Equation 6) to be considered. At most $2k-2$ of these can be critical. Iterate over all $O(n^{2k-2})$ ways of selecting $2k-2$ additional constraints. Checking a particular selection takes constant time: if one constraint is applied to $t_i - t_h$, then $C_{hi}$ is replaced by a line segment; and if two constraints are applied, $C_{hi}$ is replaced by a point. In all cases, the complexity remains constant. Total time to solve $k$CminC under this method is $O(m^{k-1}(n^{2k-1} + \log m))$.

## 6 Open Problems

The $k$NN algorithm misses the lower time bound by a factor of $\log n$. The $k$CN algorithm also misses by a factor of $\log n$, if $m$ is a constant. However, these results are for finding *all* solutions to containment. We know that we can find a single solution to 3CN in $O(n \log n)$ time for constant $m$, which is much faster than the $O(n^3 \log n)$ running time of the more general $k$CN algorithm applied to $k = 3$.

- **Open Problem I**: Establish upper and lower time bounds for finding a *single* solution to $k$NN and $k$CN. Also prove that the current algorithms are output sensitive or develop output sensitive algorithms.

The solutions to minimum enclosure problems $k$NminC and $k$CminC require that the convex enclosure have fixed orientation edges. We are confident that we can solve the more general problem of finding the minimum area convex hull in time $O((m^2+n)^{2k-2}(m+\log n))$ for $k$NminCH and $O((m+n)^{2k-2}(m+\log n))$ for $k$CminCH. It is not clear how to generalize the $O(m^{k-1}(n^{2k-1}+\log m))$ $k$CminC algorithm.

- **Open Problem II**: Prove, disprove, or improve these bounds.

All of this work only deals with translations, not rotations.

- **Open Problem III**: Generalize these algorithms to translations and rotations.

## 7 Conclusion

A combination of techniques from computational geometry and mathematical programming yields faster solutions to containment and enclosure than applying techniques from just one of these areas.

**Acknowledgements**: Thanks to Karen Daniels, Lisa Ventry Milenkovic, and Veljko Milenkovic for proofreading and suggestions. Thanks also to Shulim Kaliman for suggestions on the proof of the Lorentzian property.

## References

[1] F. Avnaim. *Placement et déplacement de formes rigides ou articulées.* PhD thesis, Université de Franche-Comté, France, 1989.

[2] F. Avnaim and J. Boissonnat. Simultaneous Containment of Several Polygons. In *Proceedings of the 3rd ACM Symposium on Computational Geometry*, pages 242–250, 1987.

[3] K. Daniels. *Containment Algorithms for Nonconvex Polygons with Applications to Layout.* PhD thesis, Harvard University, 1995.

[4] K. Daniels and V. J. Milenkovic. Multiple Translational Containment, Part I: An Approximate Algorithm. *Algorithmica, special issue on Computational Geometry in Manufacturing, accepted, subject to revisions.*

[5] K. Daniels and V. J. Milenkovic. Multiple Translational Containment: Approximate and Exact Algorithms. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[6] K. Daniels and V. J. Milenkovic. Column-Based Strip Packing using Ordered and Compliant Containment. In *Proceedings of the ACM Workshop on Applied Computational Geometry*, Philadelphia, Pennsylvania, May 27-28, 1996.

[7] K. Daniels, V. J. Milenkovic, and Z. Li. Multiple Containment Methods. Technical Report 12–94, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, 1994.

[8] O. Devillers. Simultaneous Containment of Several Polygons: Analysis of the Contact Configurations. Technical Report 1179, INRIA, 1990.

[9] K. A. Dowsland and W. B. Dowsland. Packing Problems. *European Journal of Operational Research*, 56:2 – 14, 1992.

[10] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operations Research*, 44:145–159, 1990.

[11] S. Fortune. A Fast Algorithm for Polygon Containment by Translation. In *Proceedings of the 12th Colloquium on Automata, Languages, and Programming*, pages 189–198. Springer-Verlag, 1985.

[12] L. Guibas, L. Ramshaw, and J. Stolfi. A Kinetic Framework for Computational Geometry. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 100–111, 1983.

[13] A. Kaul, M.A. O'Connor, and V. Srinivasan. Computing Minkowski Sums of Regular Polygons. In *Proceedings of the 3rd Canadian Conference on Computational Geometry*, Vancouver, British Columbia, 1991.

[14] Z. Li. *Compaction Algorithms for Non-Convex Polygons and Their Applications.* PhD thesis, Harvard University, Division of Applied Sciences, 1994.

[15] V. J. Milenkovic. Exact Algorithms for Multiple Containment. *Algorithmica, special issue on Computational Geometry in Manufacturing, accepted, subject to revisions.*

[16] V. J. Milenkovic and K. Daniels. Translational Polygon Containment and Minimal Enclosure using Geometric Algorithms and Mathematical Programming. Technical Report 25–95, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, 1995.

[17] P. E. Sweeney and E. R. Paternoster. Cutting and Packing Problems: A Categorized, Application-Oriented Research Bibliography. *Journal of the Operational Research Society*, 43(7):691–706, 1992.