Computing Covers of Plane Forests*

Luis Barba†‡ Alexis Beingessner† Prosenjit Bose† Michiel Smid†

Abstract

Let ϕ be a function that maps any non-empty subset A of $\mathbb{R}^2$ to a non-empty subset $\phi(A)$ of $\mathbb{R}^2$. A ϕ-cover of a set $T = \{T_1, T_2, \ldots, T_m\}$ of pairwise non-crossing trees in the plane is a set of pairwise disjoint connected regions such that

1. each tree T_i is contained in some region of the cover,
2. each region of the cover is either
   (a) $\phi(T_i)$ for some i, or
   (b) $\phi(A \cup B)$, where A and B are constructed by either 2a or 2b, and $A \cap B \neq \emptyset$.

We present two properties for the function ϕ that make the ϕ-cover well-defined. Examples for such functions ϕ are the convex hull and the axis-aligned bounding box. For both of these functions ϕ, we show that the ϕ-cover can be computed in $O(n \log^2 n)$ time, where n is the total number of vertices of the trees in T.

1 Introduction

Let a *geometric tree* be a plane straight-line embedding of a tree in $\mathbb{R}^2$. Consider a set $T = \{T_1, T_2, \ldots, T_m\}$ of m pairwise non-crossing geometric trees with a total of n vertices in general position. The *coverage* of these trees is the set of all points p in $\mathbb{R}^2$ such that every line through p intersects at least one of the trees. Beingessner and Smid [1] showed that the coverage can be computed in $O(m^2n^2)$ time. They also presented an example of $m = n/2$ pairwise non-crossing geometric trees (each one being a line segment) whose coverage has size $\Omega(n^4)$. Thus, the worst-case complexity of computing the coverage is $\Theta(n^4)$.

Since the worst-case inputs are rather artificial, we consider the following heuristic for reducing the running time. Let $Conv$ denote the convex hull. We observe that the coverage of the trees in T is equal to the coverage of their convex hulls. Moreover, if two convex hulls $Conv(T_i)$ and $Conv(T_j)$ overlap, then we can replace them by the convex hull of their union without changing the coverage. By repeating this process, we obtain a collection of pairwise disjoint convex polygons whose coverage is equal to the coverage of the input trees. Ideally, the number of these convex polygons and their total number of vertices are much less than m and n, respectively. If this is the case, then running the algorithm of [1] on the convex polygons gives the coverage of the input trees in a time that is much less than $\Theta(n^4)$ time, provided that we are able to quickly compute the collection of pairwise disjoint convex polygons. In this paper, we show that this is possible, by providing an $O(n \log^2 n)$–time algorithm.

We now formally state the above process.

1. Let $\mathcal{C} = \{Conv(T_i) \mid 1 \leq i \leq m\}$.
2. While the elements of $\mathcal{C}$ are not pairwise disjoint:
   (a) Take two arbitrary elements C and C' in $\mathcal{C}$ for which $C \neq C'$ and $C \cap C' \neq \emptyset$.
   (b) Let $C'' = Conv(C \cup C')$.
   (c) Set $\mathcal{C} = (\mathcal{C} \setminus \{C, C'\}) \cup \{C''\}$.
3. Return the set $\mathcal{C}$.

The output $\mathcal{C}$ is a collection of pairwise disjoint convex polygons, which we refer to as the *hull-cover* of T. See Figure 1 for two examples. Since in Step 2(b), the two elements C and C' are chosen *arbitrarily* (as long as they are distinct and overlap), the reader may object to the use of the word "the" in front of "hull-cover". In Section 2 we justify the use of this word by proving that, no matter what choices are made in Step 2(b), the output $\mathcal{C}$ is always the same.

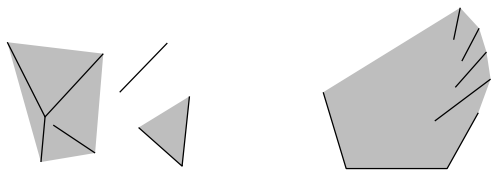

Figure 1: Two examples of hull-covers. Note that the hull-cover on the right demonstrates what is in some sense the worst case for the number of times intersections will need to be re-evaluated.

*The authors were supported by NSERC. A.B. was supported by Carleton University's I-CUREUS program.

†School of Computer Science, Carleton University, Ottawa, Canada.

‡Boursier FRIA du FNRS, Département d'Informatique, Université Libre de Bruxelles

2 ϕ-Covers

Consider a function ϕ that maps any non-empty subset A of $\mathbb{R}^2$ to a non-empty subset $\phi(A)$ of $\mathbb{R}^2$. We assume that this function satisfies the following properties:

Property 1: For any non-empty subset A of $\mathbb{R}^2$,

$$A \subseteq \phi(A).$$

Property 2: For any two non-empty subsets A and B of $\mathbb{R}^2$,

$$\text{if } A \subseteq \phi(B), \text{ then } \phi(A) \subseteq \phi(B).$$

Both the convex hull and axis-aligned bounding box functions satisfy these properties. However, the minimum enclosing circle function does not satisfy Property 2.

We rewrite the algorithm described in Section 1 using the function ϕ instead of $Conv$. We also use a forest $\mathcal{F}$ of binary trees to keep track of the history of the process; each node u in this forest stores a value $\phi(u)$. The forest helps us to prove that the ϕ-cover is well-defined.

1. For each i with $1 \leq i \leq m$, let $\mathcal{T}_i$ be the tree consisting of the single node r_i, whose value $\phi(r_i)$ is equal to $\phi(T_i)$.
2. Initialize the forest $\mathcal{F} = \{\mathcal{T}_i \mid 1 \leq i \leq m\}$.
3. Let $\mathcal{C} = \{\phi(r_i) \mid 1 \leq i \leq m\}$.
4. While the elements of $\mathcal{C}$ are not pairwise disjoint:
   (a) Take two arbitrary roots r and r' in the forest $\mathcal{F}$ for which $r \neq r'$ and $\phi(r) \cap \phi(r') \neq \emptyset$.
   (b) Let $\mathcal{T}$ and $\mathcal{T}'$ be the trees in $\mathcal{F}$ whose roots are r and r', respectively.
   (c) Let r'' be a new node and set its value $\phi(r'')$ to $\phi(\phi(r) \cup \phi(r'))$.
   (d) Create a new tree $\mathcal{T}''$ whose root is r'' and make $\mathcal{T}$ and $\mathcal{T}'$ the two children of r''.
   (e) Set $\mathcal{F} = (\mathcal{F} \setminus \{\mathcal{T}, \mathcal{T}'\}) \cup \{\mathcal{T}''\}$.
   (f) Set $\mathcal{C} = (\mathcal{C} \setminus \{\phi(r), \phi(r')\}) \cup \{\phi(r'')\}$.
5. Return the forest $\mathcal{F}$ and the set $\mathcal{C}$.

We refer to the output set $\mathcal{C}$ as the *ϕ-cover* of T. In Theorem 2 below, we prove that the ϕ-cover is well-defined. Before we prove this theorem, we present a third property of the function ϕ:

Property 3: For any two non-empty subsets A and B of $\mathbb{R}^2$,

$$\phi(A) \subseteq \phi(\phi(A) \cup \phi(B)).$$

Note that this property follows trivially from Property 1, because

$$\phi(A) \subseteq \phi(A) \cup \phi(B) \subseteq \phi(\phi(A) \cup \phi(B)).$$

Lemma 1 *Let $\mathcal{C}$ and $\mathcal{C}'$ be two ϕ-covers with corresponding forests $\mathcal{F}$ and $\mathcal{F}'$, respectively. For each node u in $\mathcal{F}$, there exists a root r' in $\mathcal{F}'$ such that $\phi(u) \subseteq \phi(r')$.*

Proof. We prove the lemma by induction on the height of the subtree rooted at u. First assume that u is a leaf in $\mathcal{F}$. Let i be the index such that $\phi(u) = \phi(T_i)$, let u' be the leaf in $\mathcal{F}'$ for which $\phi(u') = \phi(T_i)$, let $\mathcal{T}'$ be the tree in $\mathcal{F}'$ that has u' as a leaf, and let r' be the root of $\mathcal{T}'$. We prove that $\phi(u) \subseteq \phi(r')$.

Let $u'_1 = u', u'_2, \ldots, u'_k = r'$ be the nodes in $\mathcal{T}'$ on the path from u' to r'. For each i with $1 \leq i < k$, let v'_i be the sibling of u'_i. Since

$$\phi(u'_{i+1}) = \phi(\phi(u'_i) \cup \phi(v'_i)),$$

it follows from Property 3 that $\phi(u'_i) \subseteq \phi(u'_{i+1})$. From this, it follows that

$$\phi(u) = \phi(u') = \phi(u'_1) \subseteq \phi(u'_2) \subseteq \ldots \subseteq \phi(u'_k) = \phi(r').$$

Now assume that u is not a leaf. Let v and w be the children of u. Observe that $\phi(v) \cap \phi(w) \neq \emptyset$. By induction, there exist roots r' and r'' in $\mathcal{F}'$ such that $\phi(v) \subseteq \phi(r')$ and $\phi(w) \subseteq \phi(r'')$. Since $\phi(r') \cap \phi(r'') \neq \emptyset$, we must have $r' = r''$. Thus, since $\phi(v) \cup \phi(w) \subseteq \phi(r')$, Property 2 implies that

$$\phi(u) = \phi(\phi(v) \cup \phi(w)) \subseteq \phi(r').$$

□

Theorem 2 *The ϕ-cover is well-defined.*

Proof. Let $\mathcal{C}$ and $\mathcal{C}'$ be two ϕ-covers with corresponding forests $\mathcal{F}$ and $\mathcal{F}'$, respectively. We have to prove that $\mathcal{C} = \mathcal{C}'$. Observe that

$$\mathcal{C} = \{\phi(r) \mid \ r \text{ is a root in } \mathcal{F}\}$$

and

$$\mathcal{C}' = \{\phi(r') \mid \ r' \text{ is a root in } \mathcal{F}'\}.$$

Let r be a root in $\mathcal{F}$. By Lemma 1, there exists a root r' in $\mathcal{F}'$ such that $\phi(r) \subseteq \phi(r')$. Again by Lemma 1, applied with the roles of $\mathcal{F}$ and $\mathcal{F}'$ interchanged, there exists a root r'' in $\mathcal{F}$ such that $\phi(r') \subseteq \phi(r'')$. Thus, we have

$$\phi(r) \subseteq \phi(r') \subseteq \phi(r'').$$

Since $\phi(r) \cap \phi(r'') \neq \emptyset$, we must have $r = r''$. Therefore, $\phi(r) = \phi(r')$. We conclude that $\mathcal{C} \subseteq \mathcal{C}'$. By a symmetric argument, we can show that $\mathcal{C}' \subseteq \mathcal{C}$. □

Thus, the ϕ-cover is well-defined for both the convex hull and the axis-aligned bounding box. If ϕ is the minimum enclosing circle function, then, in addition to not satisfying Property 2, the ϕ-cover is not well-defined: In Figure 2, an example is given for which the order in which merges are performed can result in different outputs.

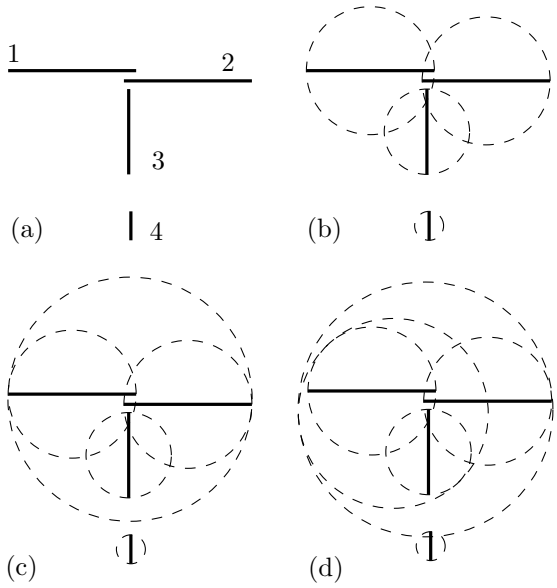

Figure 2: (a) The input forest with trees numbered; (b) The minimum enclosing circle of each tree; (c) Merging 1 and 2 first results in no intersection with 4; (d) Merging 1 and 3 first results in an intersection with 4.

3 Computing the Hull-Cover

In this section, we take for ϕ the convex hull function and show that the ϕ-cover can be computed in $O(n \log^2 n)$ time.

3.1 Weakly Disjoint Polygons

Finding the convex hull of two convex polygons can be a relatively expensive operation due to the fact that their boundaries can cross in $\Omega(n)$ different places. For example, consider a regular n-gon being merged with a copy of itself rotated ϵ degrees. In this section we demonstrate that, because our convex polygons are the convex hulls of disjoint trees, they behave much nicer than general convex polygons.

Let a *weakly disjoint pair* of convex polygons P, Q be a pair of convex polygons such that $P \setminus Q$ and $Q \setminus P$ are both connected sets of points, and P does not share a vertex with Q. Then a *weakly disjoint set* of polygons is a set of polygons such that all pairs of polygons are weakly disjoint. For simplicity, we assume that the convex hull of a line segment is a valid degenerate convex polygon consisting of two edges. We also assume all vertices are in general position. In this section we prove that weakly disjoint polygons are better behaved than general convex polygons, and that the convex hulls of disjoint trees are weakly disjoint.

Lemma 3 *If two convex polygons P, Q are weakly disjoint, then their boundaries intersect at at most two points.*

Proof. Assume the intersection of their boundaries, $\partial P \cap \partial Q$, contains more than two points. Further, assume without loss of generality that P contains points outside of Q. Start at a point on P's boundary ∂P that is outside of Q, and walk along ∂P. Eventually we intersect ∂Q, and now P is separated into two connected regions: points inside of Q, and points outside of Q. If we continue walking along ∂P, we eventually cross ∂Q again. Now there are three regions of P: two outside Q, and one inside Q, but the two outside Q may be the same. Continuing along ∂P we must eventually intersect Q again. Now the second outside region has been completed, and is clearly disconnected from the first. Therefore P and Q aren't weakly disjoint. □

Lemma 4 *If two convex polygons P, Q are weakly disjoint, but not disjoint, then one contains a vertex of the other.*

Proof. If two convex polygons are not disjoint, then they have a non-empty intersection. If this intersection has no area, then they only share part of a boundary. However the vertices are in general position, so this cannot be the case. So their intersection has some non-zero area. Remark that the vertices of $P \cap Q$ are either vertices of P, Q, or points on $\partial P \cap \partial Q$. Since $P \cap Q$ has positive area, it must have at least 3 vertices. However, by Lemma 3, we know that there are at most two points in $\partial P \cap \partial Q$. So it follows that one of these three vertices must be a vertex of P or Q. Therefore a vertex of one is inside the other. □

Lemma 5 *The convex hulls of two disjoint trees are weakly disjoint.*

Proof. Assume there exists two disjoint trees R, S, but their convex hulls are not weakly disjoint. Let $P = Conv(R)$ and $Q = Conv(S)$. If R and S share a vertex, then clearly they are not disjoint, and we have a contradiction. Then either $P \setminus Q$ is disconnected, or $Q \setminus P$ is. Assume without loss of generality that $P \setminus Q$ is disconnected. Then there exists two points $p, p' \in P \setminus Q$ such that there exists no path between p and p' inside of $P \setminus Q$. Since both P and Q are convex and share no vertices, the connected components p and p' are part of must contain a vertex of P. Therefore, without loss of generality, we may assume p and p' are vertices of P. However, that means p and p' are points on R, which has by definition a path that connects them. So either R and S intersect, or there exists a path between p and p'; both of which are contradictions. Therefore, if two trees are disjoint, their convex hulls must be weakly disjoint. □

Since the convex hulls of disjoint trees are weakly disjoint, unlike general convex polygons, finding the convex

hull of their union is simply a matter of finding at most two tangents to join them by. However, in merging two convex hulls it is no longer guaranteed that the new set of convex hulls has this property. Therefore, it would be desirable to merge convex hulls in some way in which we can maintain this property as an invariant.

3.2 Shoot and Insert

If two trees R and S in T have intersecting convex hulls, and we can find an edge to connect R and S without intersecting any other tree in T, then we have effectively merged the two trees, while maintaining the invariant of having a set of pairwise non-crossing trees.

Lemma 6 *Assume R and S are two non-crossing trees whose convex hulls intersect. Then the convex hull of one is strictly inside the other, or there exists a pair of adjacent vertices on the convex hull of one whose visibility is blocked by the other tree.*

Proof. By Lemma 4, we know that one contains a vertex of the other. Assume without loss of generality that a vertex r of $Conv(R)$ is inside of $Conv(S)$. If every other vertex of $Conv(R)$ is inside of $Conv(S)$, then $Conv(R)$ is strictly inside of $Conv(S)$ and we are done. Assume this is not the case. Then there exists some path along R from r to the outside of $Conv(S)$. This path must pass between two vertices of $Conv(S)$, and therefore obstruct their visibility. □

Consider shooting a ray between the two vertices p, q of $Conv(S)$ that are obstructed by one or more other trees. This ray will necessarily intersect some other tree R first at a point q'. By definition, the edge pq' is an edge that joins R and S without intersecting any other tree. If this is the case, then we can stop shooting rays along S, replace S and R with $S \cup R \cup pq'$, and starting shooting rays along the convex hull of that new connected component. Furthermore, if we perform this process for all adjacent pairs of vertices of $Conv(S)$, and every ray reached the target vertex, we can conclude that either S is disjoint from all other convex hulls, or part of a well-nested hierarchy of boundary-disjoint convex hulls. If the former, then S is part of our output. If the latter, then the largest convex hull that contains S is part of our output.

Ishaque et al.[3] provide a ray shooting data structure that supports shooting rays from the boundary of obstacles, that are themselves inserted into the obstacles. Using their structure, a set of n pairwise disjoint polygonal obstacles can be preprocessed in $O(n \log n)$ time and space to support m permanent ray shootings in $O((n+m)\log^2 n + m \log m)$ time. Therefore shooting n rays takes $O(n \log^2 n)$ time. We refer to this data structure as *permashoot*.

3.3 Algorithm

We start by computing the sets

$$\mathcal{C} = \{Conv(T_i) | 1 \leq i \leq m\}$$

and

$$E = \{e |\ e \text{ is an edge of some element of } \mathcal{C}\}.$$

We build a permashoot instance R on T, and a union-find data structure U on T. The latter structure is used for determining what connected component a given edge is part of.

As long as E is non-empty, we do the following: Take an arbitrary edge e in E and remove it from E. If e is not stored in R, search in U for s, the component e is part of. Shoot a ray in R from one endpoint of e along e, and return the component r that was hit. If $s \neq r$, then merge $Conv(s)$ and $Conv(r)$ in $\mathcal{C}$; remove and add edges from E to reflect the new state of $\mathcal{C}$; and union s and r in U.

At this moment, the set E is empty. We perform a plane-sweep on $\mathcal{C}$, and return all the convex hulls that are not contained inside another convex hull.

An example is given in Figure 3.

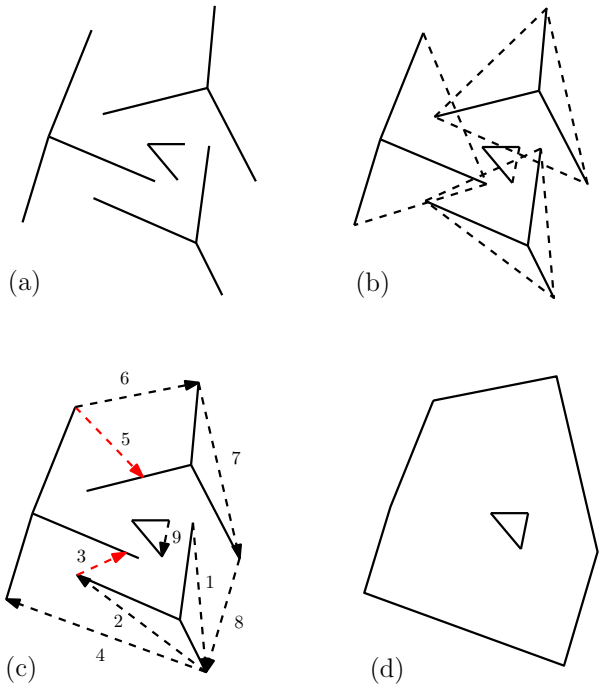

Figure 3: (a) The input; (b) Initial convex hulls of the input; (c) Rays shot by the algorithm (numbered in order they were shot), with rays that caused a merge in red; (d) Well nested hierarchy of hulls that results

Our algorithm shoots a ray for every edge of every convex hull. If any two convex hulls intersect, but are

not well-nested, then they are found during this process, and replaced by the convex hull of their union with an edge that joins them without intersecting any other components. This ensures that the invariant of having a set of pairwise weakly disjoint polygons holds. This continues until no more intersections are found in this way. By Lemma 6, we can conclude that we now have a set of convex hulls that are either disjoint, or part of a well-nested hierarchy. Our plane-sweep then finds all the maximal hulls, and returns only these.

3.4 Analysis

Because we maintain the invariant of having a set of pairwise weakly disjoint polygons, we know that each union adds at most two edges to the set of edges (the tangents between the two hulls). At worst, we perform $O(m) = O(n)$ unions, which adds $O(n)$ edges to check. Initially, there are $O(n)$ edges to check from the starting hulls. Therefore we end up shooting $O(n)$ rays, which takes $O(n \log^2 n)$ time.

For each ray shot we perform a constant amount of union and find operations to our union-find structure, each of which can easily be done in $O(\log n)$ time [2, Chapter 21]. So union-find only takes us $O(n \log n)$ time in total.

Merging two weakly disjoint convex hulls takes $O(\log n)$ time if we maintain them using height balanced binary search trees [5, Section 3.3.7]. Since we merge at most $O(m) = O(n)$ trees, merging the trees takes $O(n \log n)$ time.

Finally, the plane-sweep takes $O(n \log n)$ time to find all the maximal convex hulls.

Therefore our algorithm takes $O(n \log^2 n)$ time. This proves the following theorem.

Theorem 7 *The hull-cover of a set of pairwise non-crossing trees with a total of n vertices can be computed in $O(n \log^2 n)$ time.*

4 Computing the Box-Cover

We now assume that ϕ is the axis-aligned bounding box cover. We refer to the ϕ-cover as the *box-cover*.

Let $Box(S)$ be the axis-aligned bounding box of a tree S. A simple solution to box-cover is as follows. Create a dynamic range searching data structure that stores axis-aligned line segments and supports queries for those line segments in an axis-aligned query box. For each tree S in the input, query the structure for the segments found in the $Box(S)$. For each segment found, remove its parent bounding box from the structure. Then perform a query on the structure with the bounding box of all the boxes found in this way, plus the bounding box we just queried with. Repeat this until no segments are found. Then insert the last box we queried with into the structure. Then run a plane sweep to find all the outermost boxes.

When our algorithm finishes inserting boxes we have a set of boundary-disjoint boxes, as in our hull-cover algorithm. Therefore, as before, it is correct.

Dynamic structures for axis-aligned segment queries exist that take $O(\log^2 n + k)$ time for queries, insertion, and deletion[4]. Since we start with an empty structure, preprocessing time is irrelevant. When we find an intersection, we replace $O(k)$ boxes with a single box. Since there are $O(m) = O(n)$ boxes, and each box gets inserted and removed at most once, it follows that our algorithm takes $O(n \log^2 n)$ time to perform this process in total. The plane sweep takes only $O(n \log n)$ time. Therefore, our algorithm takes $O(n \log^2 n)$ time in total. This proves the following theorem.

Theorem 8 *The box-cover of a set of pairwise non-crossing trees with a total of n vertices can be computed in $O(n \log^2 n)$ time.*

5 Conclusions and Open Problems

We are able to compute the solutions to hull-cover and box-cover in $O(n \log^2 n)$ time. However this is not obviously optimal. It remains to be seen whether there are better algorithms for these problems.

While the hull-cover is a potentially powerful preprocessing step for computing the actual coverage, the relationship between the two is fairly weak. In the best case the hull-cover is the convex hull of the input, and the two are the same. However in the worst case the hull-cover is exactly the input, but the coverage is something of size $\Omega(n^4)$.

Given a set $\mathcal{O}$ of orientations, an $\mathcal{O}$-convex set S is a set of points such that every line with an orientation in $\mathcal{O}$ has either an empty or connected intersection with S. The $\mathcal{O}$-hull of a set T of points is then the intersection of all $\mathcal{O}$-convex sets that contain T. When $\mathcal{O} = \{[0, 180)\}$, the $\mathcal{O}$-hull is the convex hull. When $\mathcal{O} = \emptyset$, the $\mathcal{O}$-hull is the identity function. The $\mathcal{O}$-hull satisfies our properties for being well-defined [6]. However, an algorithm for the general $\mathcal{O}$-hull is not immediately obvious. Further, it is unclear as to whether there are other non-trivial well-defined covering functions beyond the $\mathcal{O}$-hull and the axis-aligned bounding box. The geodesic hull does satisfy our properties, but without a bounding domain the geodesic hull is just the convex hull. We know from the start of the paper that the minimum enclosing circle does not produce well-defined results, and a similar argument applies to the minimum enclosing square.

Remark that our proof that general ϕ-covers are well-defined does not rely on the fact that we are working in two dimensions. This allows us to easily extend the problem into higher dimensions, where the convex hull and bounding box still work. However,

while our technique for bounding boxes generalizes to d-dimensions nicely, our technique for the convex hull does not. Therefore, a technique for computing the hull-cover that generalizes well would be desirable.

Acknowledgement

Special thanks to Pat Morin for consultation on certain proofs.

References

[1] A. Beingessner and M. Smid. Computing the coverage of an opaque forest. pages 95–99. Canadian Conference on Computation Geometry, 2012.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2001.

[3] M. Ishaque, B. Speckmann, and C. Tóth. Shooting permanent rays among disjoint polygons in the plane. *SIAM Journal on Computing*, 41(4):1005–1027, 2012.

[4] M. Overmars. *The Design of Dynamic Data Structures*. Lecture Notes in Computer Science. Springer, 1983.

[5] F. Preparata and M. Shamos. *Computational geometry: an introduction*. Texts and monographs in computer science. Springer-Verlag, 1988.

[6] G. J. Rawlins and D. Wood. Restricted-oriented convex sets. *Information sciences*, 54(3):263–281, 1991.