Translational Polygon Covering using Intersection Graphs

Karen Daniels[1] *Rajasekhar Inkulu*

Abstract

We investigate the following translational 2D polygon covering problem: for a collection of covering polygons and a target point (or polygon) set, decide if the covering polygons can be translated to cover the target set. If the answer is "yes," produce translations for a cover. We prove this problem is NP-complete and provide an incrementally constrained algorithm whose search space exploration is guided by intersection graph-based heuristics. Our algorithm is approximate for polygonal targets.

1. Background

Given 2D *covering polygons* $Q = \{Q_1, Q_2, \ldots, Q_m\}$ and 2D *target point (or polygon) sets* $P = \{P_1, P_2, \ldots, P_n\}$, the translational 2D covering problem seeks a vector[2] of translations $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$ such that: $P \subseteq \bigcup_{1 \le j \le m} \gamma_j(Q_j)$.

Figure 1 is an example for n=1 and m=3.

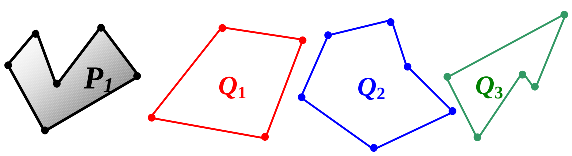

Figure 1(a): Sample P and Q

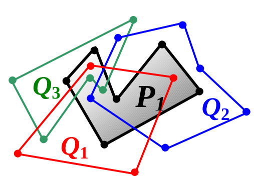

Figure 1(b): Translated Q Covers P

Translational 2D covering problems arise in applications such as telecommunications, spatial query optimization, publish/subscribe middleware, and military sensor coverage and targeting. Previous 2D covering work focuses on specialized forms of P and Q, often addresses the optimization problem of minimizing the number of covering polygons, and identifies many NP-complete covering problems. [7] surveys results for covering the plane with congruent convex shapes. [1] summarizes early hardness results for coverings that are also decompositions. [2] surveys some work on covering and closely related problems.

Section 2 proves that our problem is NP-complete. Section 3 describes our incrementally constrained algorithm. The algorithm is approximate for polygonal targets. Unlike previous 2D covering work, our algorithm is designed to handle point-set or polygonal P and does not require the covering polygons in Q to be identical. Its design also does not require P or Q to have special structure (e.g. rectangular, convex, star-shaped or orthogonal). However it does rely on convex decomposition of the polygons in Q. Section 4 adds intersection graph-based heuristics to the algorithm to guide search space exploration for polygonal P. Section 5 discusses our current implementation and test results, which are for convex covering polygons in Q and polygonal, connected P. The paper concludes with future work in Section 6.

2. NP-Completeness

Theorem: Translational 2D polygon covering is NP-complete.

Proof Sketch (see [2] for details):
The problem is in NP because, given a translation vector γ, we can construct $\cup_j (\gamma_j(Q_j))$ and test whether or not each P_i is inside $\cup_j (\gamma_j(Q_j))$ in time that is polynomial in the number of vertices of P and Q. A reduction from PARTITION [5] similar to that used in [3] establishes NP-hardness for polygonal P, even when n=1 and all polygons in P and Q are restricted to rectangles. Hardness for point-set P follows by viewing BOX-COVER[3] [4] as a special case of our problem.☐

3. Incrementally Constrained Algorithm

Here we provide an overview of our incremental covering algorithm; details appear in [2]. Since our problem is NP-complete, our goal is to decide

---

[1] The authors are in the Computer Science Department, University of Massachusetts, Lowell, MA.

[2] Note that *one* vector is sufficient; we do not seek *all* possible solutions.

[3] In a 2D instance of the NP-complete BOX-COVER problem [4] we are given a finite set of (integer) points $X = \{X_1, X_2, \ldots, X_n\}$, a finite set of boxes $Y = \{Y_1, Y_2, \ldots, Y_m\}$ with which to cover the points, and we seek γ such that $X \subseteq \cup \gamma_j(Y_j)$.

quickly if a cover exists for "easy" instances that are either combinatorially small, clearly infeasible, or feasible with many closely related solutions. We explore the covering search space by incrementally constraining translations of the covering polygons. We view a constraint as an assignment: a requirement that a certain polygon of Q cover a particular point in the target set P.

Definition3.1: Let Ψ be a subset of the points of P. An *assignment* f is a mapping $f : \Psi \rightarrow 2^Q$. The size of f is $\sum_{p \in \Psi} |f(p)|$. An assignment is *feasible* if $\exists \gamma$ such that $\forall p \in \Psi$ and $\forall Q_j \in f(p)$, $p \subseteq \gamma_j(Q_j)$. A feasible assignment is a *covering assignment* if $\exists \gamma'$ such that $P \subseteq \bigcup_{1 \le j \le m} \gamma'_j(Q_j)$.

The existence of a covering assignment is a necessary and sufficient condition for a covering. Consequently, if there exists $\Psi \subseteq P$ for which no feasible assignment exists, then the problem instance itself is infeasible. Since the number of candidate assignments is in $O((2^m-1)^N)$, where N is the total number of points of P involved in assignments, for feasible instances we try to find a small feasible assignment that is also a covering assignment. For infeasible instances we try to demonstrate infeasibility quickly either through a simple area comparison or by identifying some small Ψ for which no feasible assignment exists.

The high-level algorithm flow is as follows. Select a small subset Ψ of the points of P. Generate assignments for Ψ. Try to find a feasible assignment. Discard infeasible assignments. If no feasible assignment is found for Ψ, report that no cover exists. If a feasible assignment is found, test if its feasibility also implies that it is a covering assignment. If so, report that a cover has been found. Otherwise, either try to find another feasible assignment for Ψ or add a point of P to Ψ and execute the algorithm on the revised Ψ. For polygonal targets this process terminates based on an area tolerance and is therefore approximate.

In more detail, our starting Ψ consists of the vertices of the convex hull of P, denoted by *CH(P)*. Assignments for Ψ are generated in lexicographic order. To test an assignment's feasibility we observe that finding γ_j such that $p \subseteq \gamma_j(Q_j)$ is equivalent to finding γ_j such that $(p \cap \gamma_j(Q_j)) \neq \emptyset$. Since the Minkowski Sum[4] operator $\oplus$ can be used to find translations that guarantee intersection, the set of feasible γ_j for p is $p \oplus (-Q_j)$. If we define a "pseudo-inverse" $f^{-1}(Q_j)$ to be the set of all points of Ψ mapped by f to Q_j, then for each j the set of all feasible γ_j for Ψ is $f^{-1}(Q_j) \oplus (-Q_j)$. The assignment is feasible if and only if $(f^{-1}(Q_j) \oplus (-Q_j)) \neq \emptyset$ for every j.

We leverage convexity to test if feasibility of an assignment implies feasibility for an assignment superset. This rests on the following result for convex Q_j. For this reason the algorithm uses a convex decomposition of each Q_j.

Claim 3.1: If Q_j is convex and f is a feasible assignment for Ψ, then $CH(f^{-1}(Q_j)) \subseteq \gamma_j(Q_j)$.

Proof: Since f is feasible, γ_j is a translation of Q_j such that the points of $f^{-1}(Q_j)$ are covered by $\gamma_j(Q_j)$. Translate Q_j by γ_j. A convex set that covers a point set also covers the convex hull of that point set. The result now follows from the convexity of Q_j. □

When applied for all j, Claim 3.1 implicitly expands Ψ by identifying a set of convex regions of P that are covered under the current assignment. In some cases this implies that a small feasible assignment is also a covering assignment. When needed, we explicitly expand Ψ by adding to it a point in an uncovered region of P, selecting an assignment for it and testing its feasibility.

4. Intersection Graph-Based Heuristic for Polygonal *P*

For polygonal P, each choice of translations for Q has an associated intersection graph[5]. This motivates us to use intersection graphs to heuristically guide our initial search space exploration. We first consider assignments for Ψ = *vertices(CH(P))* derived from certain intersection graph topologies. If these do not yield a cover, we proceed as in Section 3. More formally, given polygonal P, Q, and translation vector γ, let R be the partition of P induced by the boundaries of the translated Q_js:

---

[4] The Minkowski Sum [6] of two point sets A and B is given by: $A \oplus B = \{a + b \mid a \in A, b \in B\}$.

[5] Intersection graphs have been used previously in the context of covering. See [4], for example.

$$R = \partial(P) \cup (P \cap (\bigcup_{1 \le j \le m} \partial(\gamma_j(Q_j))))$$

where $\partial(X)$ denotes the boundary of *X*.

Definition 4.1: An *intersection graph* on *P*, *Q*, and γ is an undirected graph containing one node for each region of *R* and an edge connecting each pair of nodes whose regions share an edge in *R*.

Definition 4.2: $S_k = \{Q_j \mid \gamma_j(Q_j) \cap R_k \neq \emptyset\}$ is the *intersection set* of a region R_k of *R*.

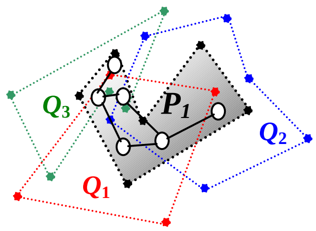

Figure 2(a): Intersection Graph for Figure 1(b)'s Cover

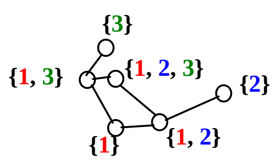

Figure 2(b): Intersection Sets for Figure 2(a)

Definition 4.3: An acyclic, connected intersection graph is a *linear intersection graph* if and only if the maximum degree of any node is 2.

Definition 4.4: Let *I* be a connected intersection graph. Let *D* be the set of single-degree nodes of *I*. *I* is a *star intersection graph* if *D* is non-null and is also the set of nodes of *I* having an intersection set of size one.

Using these definitions we heuristically consider a set of linear intersection graphs and, from these, construct small assignments for Ψ = *vertices(CH(P))*. If these do not yield a cover, we consider star intersection graphs. If these do not yield a cover, we proceed as in Section 3.

In the linear intersection graph heuristic, we consider a set of linear intersection graphs, each induced by a trapezoidal decomposition of *P* in which each vertex of each region is a vertex of *CH(P)*. The intersection set for each graph node consists of a single Q_j. This determines an assignment for Ψ. We then use techniques from the incremental algorithm of Section 3 to enforce the intersection constraint associated with each intersection set and to test if that assignment is feasible. Since each assignment is based on an intersection graph for a decomposition of *P*, assignment feasibility in this case also implies a covering.

In the star intersection graph heuristic, we consider a set of star intersection graphs. However, in the star case we do not base the graph on a decomposition of *P*. We only specify intersection sets for graph nodes in *D*. By definition, each such intersection set consists of a single Q_j. We associate each node of *D* with a vertex of *CH(P)*. This determines an assignment for Ψ. We then apply feasibility test techniques from the incremental algorithm of Section 3 as in the case of linear intersection graphs. However, since each assignment is not based on an intersection graph for a decomposition of *P*, assignment feasibility in the star case does not necessarily imply a cover. We must therefore test if a feasible assignment is also a covering assignment.

5. Implementation and Results

Our code is in C++ and uses the LEDA and CGAL libraries. Our current implementation is for polygonal, connected *P* and convex Q_js.

Table 1 shows the number of assignments explored by the algorithm for a representative set of test cases for a single η-vertex, connected polygon *P* and *m* convex polygons of *Q*. The number of assignments is given both with and without intersection graph heuristics. When using the heuristics, we first apply the linear heuristic, then (if needed) the star heuristic, and then (if needed) the approach of Section 3.

In cases where the heuristics find a cover using either a linear or a star intersection graph, the algorithm explores fewer assignments with the heuristics than without the heuristics. Note the algorithm can sometimes decide that no cover exists without examining all possible assignments.

#	m	η	P con-vex?	# Assign. with heuristics[6]	# Assign. without heuristics	Cover exists ?
1	2	6	yes	60+65	728	yes
2	2	5	yes	60+82+	729	no
3	2	7	no	30	1334	yes
4	2	5	no	16+7	52	yes
5	2	5	yes	11	174	yes
6	2	5	yes	40+60+	140	yes
7	2	6	yes	3	349	yes
8	3	6	yes	107	33714	yes
9	4	6	yes	24	-	yes
10	6	12	yes	15197	-	yes

Table 1: Test Results for η-vertex Polygonal, Connected *P* and *m* Convex Q_js

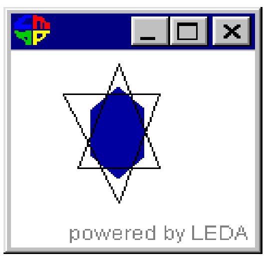

Fig 3a: Table Row 1

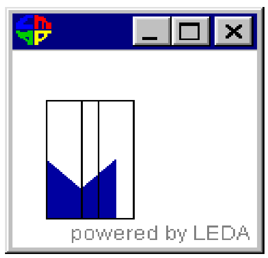

Figure 3b: Table Row 4

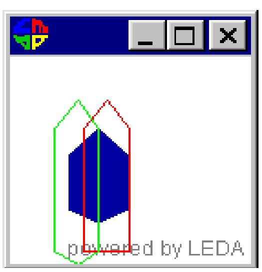

Figure 3c: Table Row 7

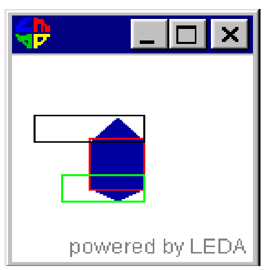

Figure 3d: Table Row 8

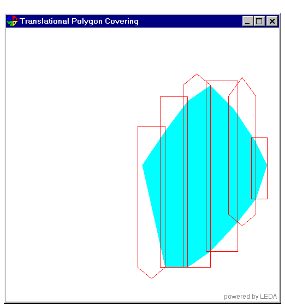

Figure 3e: Table Row 10

[6] In the X+Y+ notation, X is the number of assignments explored using linear intersection graphs. Y is the number of assignments explored using star intersection graphs. A second '+' indicates that no cover was found using the heuristics.

6. Conclusion and Future Work

This paper provides an incremental algorithm for the NP-complete translational 2D polygon covering problem. The algorithm incrementally generates assignments that constrain translations of the covering polygons. It can sometimes decide that no cover exists without examining all possible assignments. The algorithm is approximate for polygonal targets. In our test cases in which intersection graph heuristics find a cover, the algorithm explores fewer assignments with the heuristics than without them. Our next implementation will support convex decomposition of nonconvex covering polygons. We plan to develop heuristics for additional types of intersection graphs. We also plan to extend our algorithm to 3D.

References:

[1] Culberson, J. and Reckhow, R. "Covering Polygons is Hard." In *Proceedings of 29th IEEE Conference on Foundations of Computer Science*, 1988.

[2] Daniels, K.M., Inkulu R. "An Incremental Algorithm for Translational Polygon Covering" Technical Report 2001-001, Dept. of Computer Science, UMass Lowell, April, 2001 (see http://www.cs.uml.edu/~kdaniels).

[3] Daniels, K.M. and Milenkovic, V.J., *Multiple Translational Containment: Approximate and Exact Algorithms*. Proceedings of the 6th Annual ACM-SIAM SODA, January 22-24, 1995, p. 205-214.

[4] Fowler, R., Paterson, M., and Tanimoto, L., "Optimal Packing and Covering in the Plane are NP-Complete", *Information Processing Letters*, 12(3):133-137, June, 1981.

[5] Garey, M. and Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company, New York, 1979.

[6] Minkowski,H. "Volumen und Oberflache." *Mathematische Annalen*, 57:447-495, 1903.

[7] Toth, G. F. "Packing and Covering." In *Handbook of Discrete and Computational Geometry*, Goodman, J. and O'Rourke, J. (editors). CRC Press, New York, 1997.

Acknowledgement: The authors thank Diane Souvaine for initial discussions on this topic.