# Constructing Concise Convex Covers via Clique Covers

**Mikkel Abrahamsen**
University of Copenhagen, Denmark

**William Bille Meyling**
University of Copenhagen, Denmark

**André Nusser**
University of Copenhagen, Denmark

## Abstract

This work describes the winning implementation of the CG:SHOP 2023 Challenge. The topic of the Challenge was the convex cover problem: given a polygon $P$ (with holes), find a minimum-cardinality set of convex polygons whose union equals $P$. We use a three-step approach: (1) Create a suitable partition of $P$. (2) Compute a visibility graph of the pieces of the partition. (3) Solve a vertex clique cover problem on the visibility graph, from which we then derive the convex cover. This way we capture the geometric difficulty in the first step and the combinatorial difficulty in the third step.

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Mathematics of computing → Combinatorial algorithms

**Keywords and phrases** Convex cover, Polygons with holes, Algorithm engineering, Vertex clique cover

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2023.66

**Category** CG Challenge

**Supplementary Material**
*Software*: `https://github.com/willthbill/ExtensionCC`
archived at `swh:1:rev:ad78739911ab5733f600d4fbcb08acae6d69e115`
*Text (Thesis)*: `https://github.com/willthbill/ExtensionCC/blob/main/bachelorthesis.pdf`

**Funding** *Mikkel Abrahamsen*: Supported by Starting Grant 1054-00032B from the Independent Research Fund Denmark under the Sapere Aude research career programme. Part of BARC, supported by the VILLUM Foundation grant 16582.
*William Bille Meyling*: Supported by Starting Grant 1054-00032B from the Independent Research Fund Denmark under the Sapere Aude research career programme.
*André Nusser*: Part of BARC, supported by the VILLUM Foundation grant 16582.

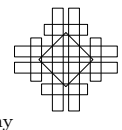
**Acknowledgements** We want to thank the CG:SHOP 2023 organizers and the other participants (especially Guilherme Dias da Fonseca) for creating such a fun challenge and for helpful comments on our write-up. We also want to thank Martin Aumüller and Rasmus Pagh for access and help with using their server. Finally, we want to thank Darren Strash for quick and last-minute support using the ReduVCC implementation.

## 1 Introduction

Covering a polygon with the minimum number of convex pieces is a fundamental problem in computational geometry and the problem chosen for the CG:SHOP 2023 Challenge. In this problem we are given a polygon $P$ (potentially with holes) and we have to find a smallest possible set of convex polygons whose union equals $P$. This problem is NP-hard [3] and was later shown to be even $\exists\mathbb{R}$-complete [1]. Note that in the Challenge, all coordinates of the solutions had to be rational, and then the decision problem is not even known to be decidable. Thus, it is not expected that there exists any fast algorithm that always finds







the optimal solution. Likewise, we are aware of no previously described algorithm which is fast in practice. In this 5th CG:SHOP Challenge, there were a total of 22 teams who signed up, out of which 18 submitted solutions. Our team – named *DIKU (AMW)* – obtained the highest total score, despite finding fewer smallest solutions than the runner-up [4], as we achieved significantly smaller solutions on many of the largest Challenge instances. See [5] for a survey about the Challenge.

Our algorithm consists of three steps. In the first step (see Section 2.1), the aim is to capture the geometry of the problem. We do this by partitioning the input polygon $P$ into triangles. Note that the corners of these triangles do not have to be corners of $P$ but can be Steiner points. In the second step (see Section 2.2), we move from the geometric structure to a combinatorial structure. We do this by computing a visibility graph $G$ of the partition, with each triangle corresponding to a vertex and an edge is inserted for two vertices only if the convex hull of the corresponding triangles lies within $P$ (i.e., the convex hull would be a valid piece of the convex cover). Finally, in the third step (see Section 2.3), we solve a combinatorial problem: we find a vertex clique cover (VCC) of $G$ with small cardinality. Recall that a *vertex clique cover* is a set of cliques in $G$, for which each vertex in $G$ appears in one of the cliques. When possible, we use the convex hull $H$ of the triangles of a clique $C$ as a piece of our convex cover. However, $H$ may intersect holes of $P$, which makes $H$ an invalid piece. This happens rarely for the Challenge instances, but in that case we split $C$ into smaller cliques. For us, the main insights and highlights of our approach are:

1. Assembling the pieces and the cover at the same time (instead of first deciding on the pieces and then assembling the cover) allows for great flexibility and adaptivity.
2. Reduction to a fundamental graph problem allows for usage of a powerful set of already existing tools.
3. As we can arbitrarily choose a partition in the first step of the algorithm, our approach is very adaptive with respect to input structure and instance size (simpler partitions can be chosen for larger instances).

## 2 Algorithm

In this section we describe our algorithmic approach to solve the convex cover problem.

### 2.1 Partition

First, we partition the polygon. While our approach in principle works with any kind of partition, for simplicity we only used partitions consisting of triangles. Recall that the goal of the partition is to obtain triangles from which we can later assemble good pieces for a convex cover and that the corners of these triangles are not restricted to lie on the corners of $P$. In fact, to obtain good solutions one often needs Steiner points.

The simplest partition that we use is a Delaunay triangulation. We prefer a Delaunay triangulation over an arbitrary triangulation because it leads to fat triangles, which we intuitively assume to create better pieces for the convex cover. The main issues of using a Delaunay triangulation of $P$ as partition are that its vertices are restricted to the corners of $P$ and that the pieces can be too coarse to merge into convex pieces. See Figure 6 for an example for which this leads to a suboptimal cover. Thus, the question is: which Steiner points should we introduce to obtain better solutions?

Consider a directed edge $e$ of $P$ and suppose that the interior of $P$ is to the left of $e$. We define the *extension* of $e$ to be the maximal directed segment $s$ such that $e \subseteq s \subseteq P$; see Figure 1 (middle). Note that a piece $Q$ of a convex cover can contain an interior point of

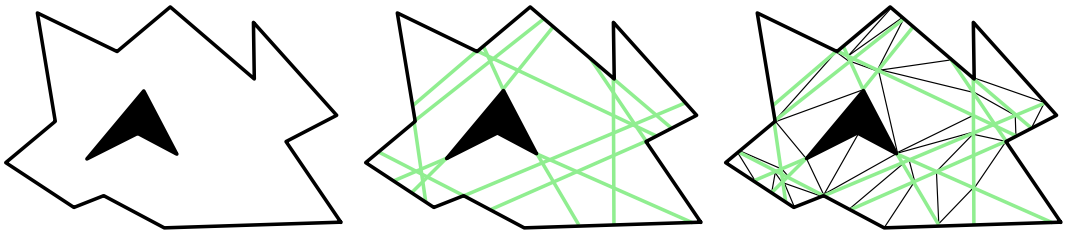


**Figure 1** Left: Polygon $P$ (left), extensions of $P$ (middle), and extension partition of $P$ (right).

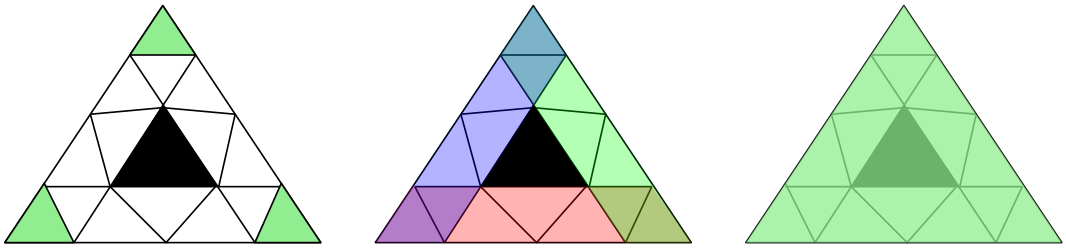


**Figure 2** For the set of green triangles (left), every pairwise convex hull is contained in $P$ (middle), but the convex hull of all the green triangles is not (right).

$e$ only if $Q$ does not contain a point to the right of $s$. Thus, intuitively it make sense to include pieces of the cover that are bounded by $s$. This intuition is captured by the *extension partition*, which is the constrained Delaunay triangulation of the extensions of all edges of $P$; see Figure 1 (right).

## 2.2 Visibility Graph

In order to create a convex cover, we first want to understand which triangles we can potentially combine to form pieces for the cover. Given a partition $\mathcal{P}$ of the polygon $P$ and two triangles $p, q \in \mathcal{P}$, we say that $q$ is *fully visible* from $p$ if every point in $p$ sees all of $q$ and *partially visible* if every point in $p$ sees some point in $q$. We define the visibility graph $G = (\mathcal{P}, E)$, which contains an edge $pq$ if the convex hull of $p \cup q$ is contained in $P$. We can compute $G$ naively by checking for each pair $p, q \in \mathcal{P}$ whether its convex hull is contained in $P$. However, the running time $\Omega(|\mathcal{P}|^2)$ renders this impractical. A simple observation comes in handy here: For any triangles $q \in \mathcal{P}$ fully visible from $p \in \mathcal{P}$, there exists a path from $p$ to $q$ in the dual graph[1] of $\mathcal{P}$ using only vertices that correspond to triangles that are partially visible from $p$. Thus, instead of checking all pairwise visibilities, we can simply perform a BFS on the dual graph, only using partially visible triangles and stop exploring on triangles that are not partially visible. While this significantly reduces the running time in practice, it can still be too expensive. For further speedup, we resort to building a subgraph of $G$ by only exploring fully visible triangles in the BFS. To speed up the visibility graph construction, we engineered fast visibility checks that we do not further describe here.

## 2.3 Compute Cover

We employ the following three steps to compute a convex cover using the visibility graph.

[1] The dual graph of a partition is defined as follows: the vertex set consists of the triangles of the partition and there is an edge between two vertices iff the two corresponding triangles touch.

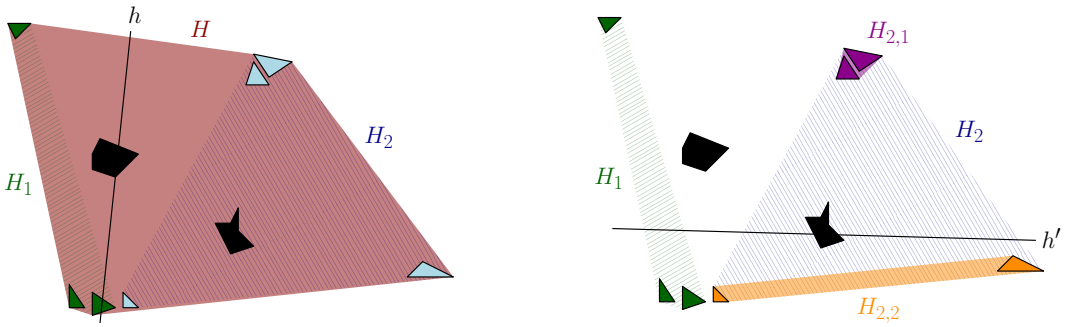



**Figure 3** Fixing an invalid clique: All visible triangles form the initial piece $H$ that we then split into pieces $H_1$ and $H_2$ using the half-plane $h$ (left). As $H_2$ still contains a hole, we again split it into pieces $H_{2,1}$ and $H_{2,2}$ using half-plane $h'$ (right). The result is the valid pieces $H_1$, $H_{2,1}$, and $H_{2,2}$.

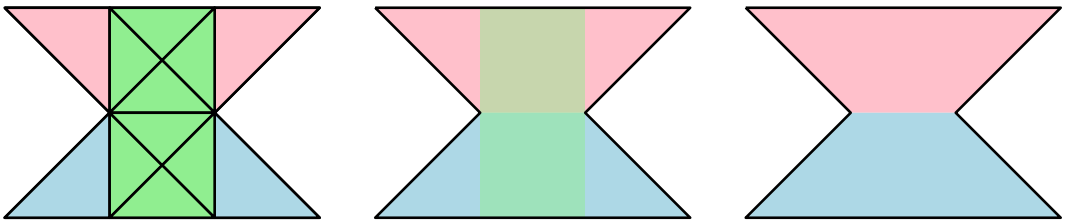


**Figure 4** Extension partition with suboptimal clique cover (left), the corresponding convex cover (middle), and a convex cover without the unnecessary green polygon (right).

**Compute Vertex Clique Cover:** We first compute a vertex clique cover (VCC) on the visibility graph. The problem of finding a minimum VCC is one of Karp's classical NP-hard problems, and there exists no $n^{1-\varepsilon}$-approximation algorithm for any $\varepsilon > 0$ unless P = NP. However, there exist implementations that compute small solutions on practical instances. Namely, Chalupa [2] presented a randomized clique-growing approach that was subsequently used as a subroutine by Strash and Thompson [8] in their state-of-the-art solver `ReduVCC` that uses sophisticated reduction rules with a branch-and-reduce approach.

**Fix Cover:** Recall that a clique $C$ corresponds to a set of triangles that are pairwise fully visible. We would like to use the convex hull $H$ of the triangles as a piece in our convex cover, but $H$ may not be contained in $P$; see Figure 2 for a simple example. While this rarely happens on the Challenge instances (see Section 3.3), we nonetheless have to post-process such pieces to obtain a feasible convex cover.

First, note that the only way a piece $H$ can be invalid is by containing a hole of $P$; in particular, it is not possible that $H$ intersects the unbounded region of the complement of $P$. We fix an invalid piece $H$ as follows: Pick any hole $h$ that invalidates $H$ and consider an arbitrary half-plane whose boundary intersects $h$. Now partition the triangles of $C$ according to whether they intersect the half-plane or not. This creates two new pieces, which both do not intersect the hole $h$ and which partition the remaining holes in $H$. We apply this procedure recursively to the new pieces (always reducing the number of holes intersected by these pieces by at least one) until all newly created pieces are valid; see Figure 3. We omit the proof of correctness of this procedure due to space constraints.

**Make Cover Minimal:** At this point, we may end up with a non-minimal cover, i.e., there may exist redundant pieces; see Figure 4. To make the solution minimal, we iterate over the pieces and remove them from the cover if their removal does not invalidate it.

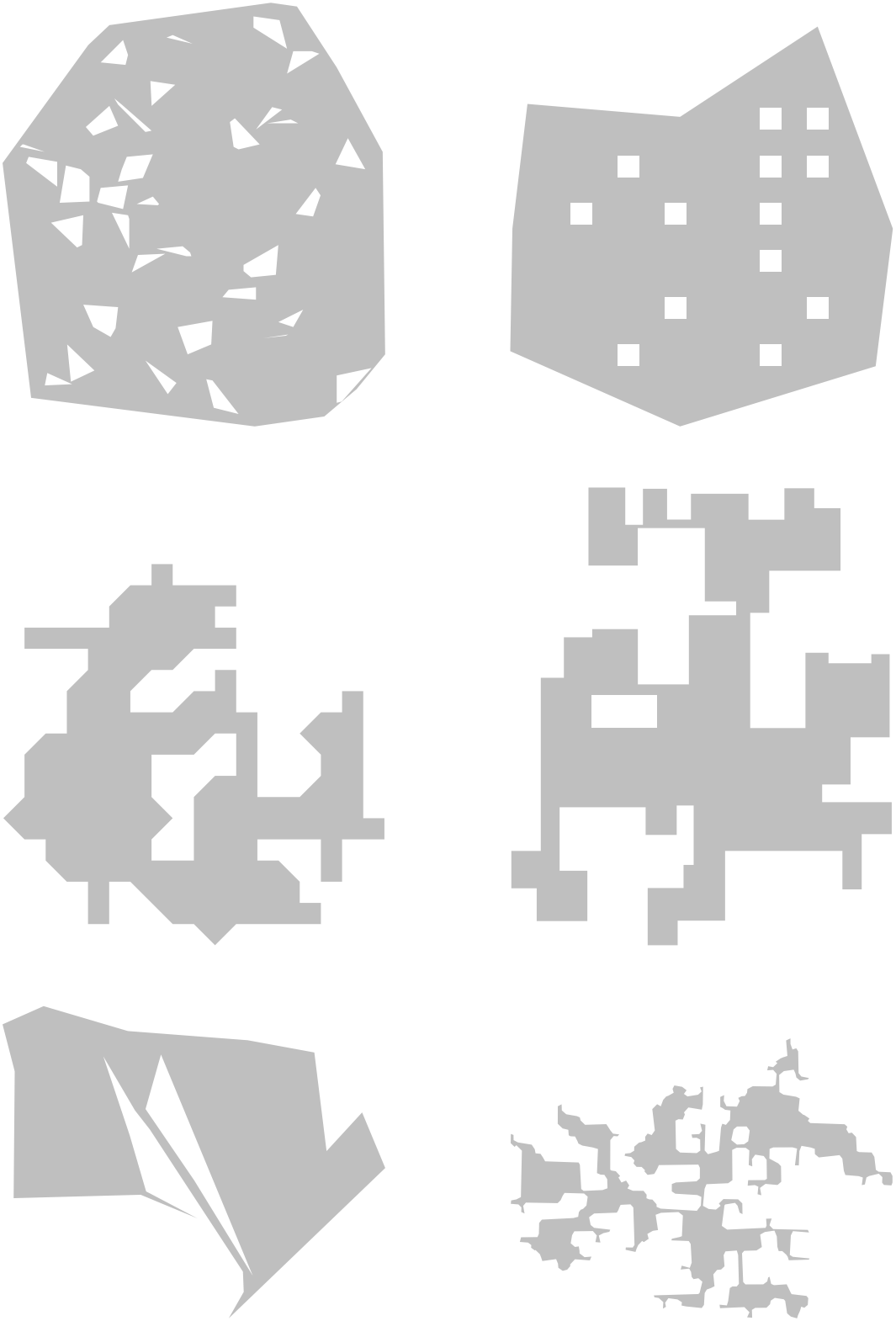


**Figure 5** The smallest instances of some of the Challenge instance types. From left to right and top to bottom, these are: `cheese`, `maze`, `octa`, `iso`, `fpg`, `srpg_mc`.

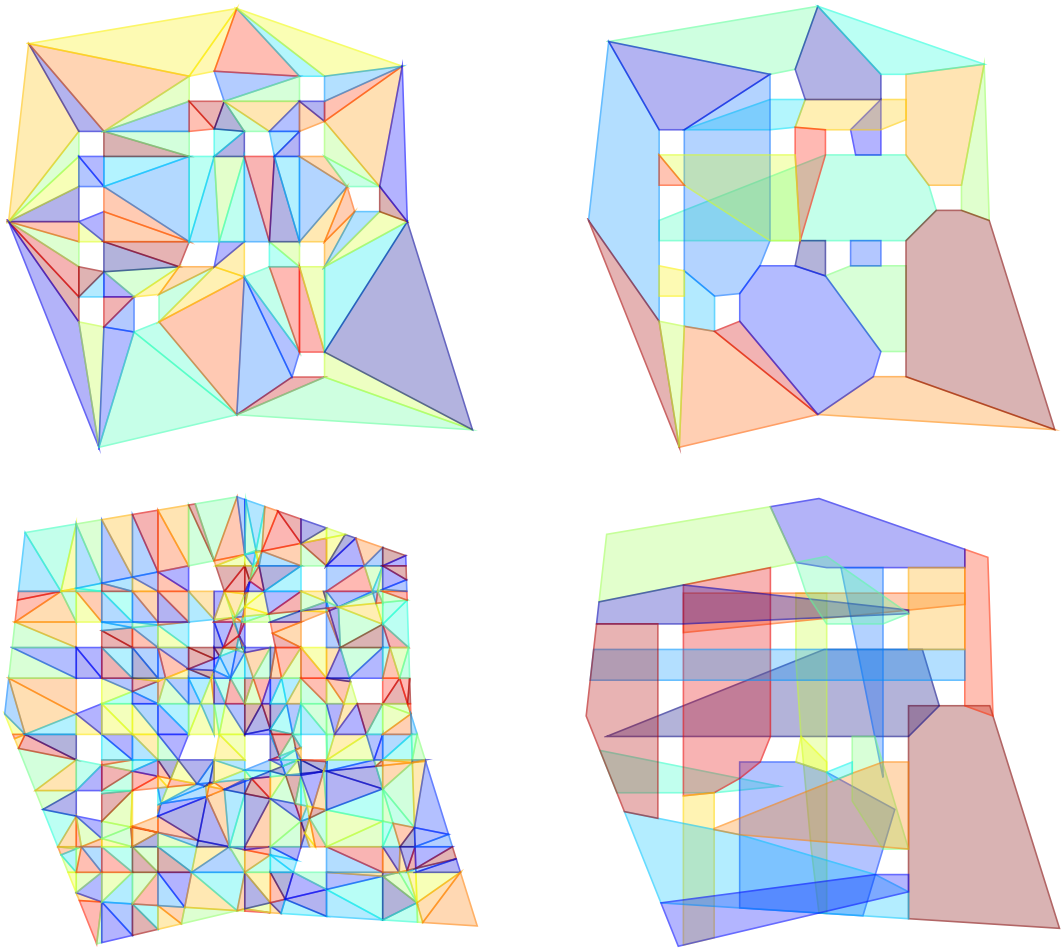


**Figure 6** The Delaunay triangulation (top left) and the resulting cover of size 27 (top right). The extension partition (bottom left) and the resulting cover of size 23 (bottom right).

# 3 Evaluation

## 3.1 Implementation and Data

The competition code is written in C++ and compiled using GCC 11.3 with `-O3` optimization turned on. We use CGAL [9] for all geometric primitives with a Kernel that uses a number type that saves numbers as fractions and performs exact computations. For the partitioning and to construct the visibility graph, we use the triangulation, visibility, and convex hull packages of CGAL [6, 7, 10]. To compute the vertex cover, we use ReduVCC [8]. We show different types of instances of the problem set in Figure 5.

## 3.2 Examples

An important part of our approach is the choice of the partition. In particular, while the Delaunay triangulation is fast to compute, the extension partition creates partitions with significantly more pieces and thus slows down our approach. To justify this kind of partition, it must lead to significantly better solutions. Figure 6 shows a cover of the same instance using the Delaunay partition and the extension partition – one can clearly see that the extension partition better adapts to the geometry of the input polygon. Unfortunately, extension

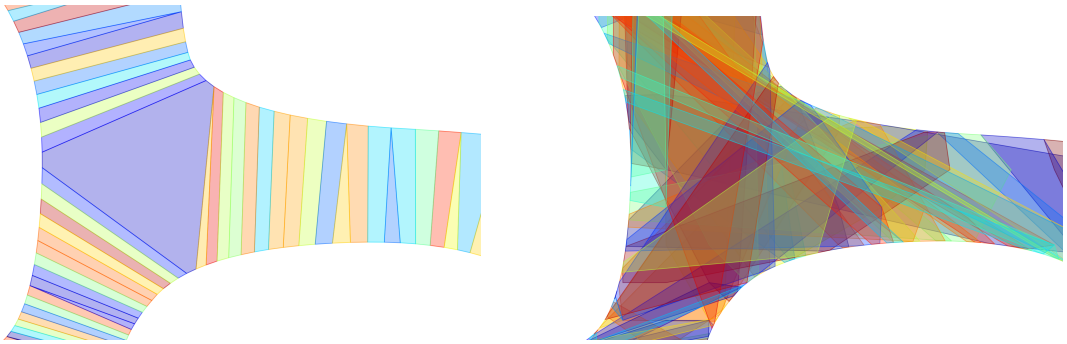


**Figure 7** Part of a polygon with multiple long concave chains. While the cover using a Delaunay triangulation almost exclusively creates pieces adjacent to only two concave chains (left), local triangulation allows for creation of pieces that contain parts of all three concave chains (right).

partitions can lead to a blow-up of the partition size that makes the approach practically infeasible for some instances. This blow-up happens when many extensions intersect. We circumvent this issue by computing restricted extension partitions in two different ways.

1. We want to preserve extensions locally. Thus, we only choose a subset of the extensions favoring short extensions. We either only insert extensions below a certain length, or we randomly sample extensions inversely proportional to their length.
2. Some Challenge instances have long concave chains on the boundary of $P$. Note that the midpoint of each edge of such a chain has to be part of a distinct piece in the convex cover. To allow for creation of pieces that combine segments from multiple concave chains, we locally triangulate long concave chains instead of creating extensions; see Figure 7.

## 3.3 Experiments

For this section, we selected a subset of the instances for more thorough experiments and subsequently only refer to these. See the sizes and types in the plots of Figure 8. Recall that we compute an intermediate, potentially infeasible solution via a vertex clique cover that is subsequently fixed. We argue above that we expect that only few cliques have to be fixed on practical instances. Indeed, on all except the `cheese` instances, the solution size increased by at most 6 pieces when fixing cliques, while most small instances did not have any invalid clique. However, the largest increase in solution size was for the largest `cheese` instances with an increase of 110 pieces. Our algorithm may create redundant cliques that are removed in a post-processing step, so it is interesting to consider how much this post-processing reduces the size of the solution. This decrease in pieces is very much dependent on the instance: While for `octa` the maximal decrease was 2 pieces, it was 83 pieces for `cheese` instances and all larger `cheese` instances saw significant improvements.

For the competition and our experiments we used a server with two Intel Xeon E5-2690 v4 CPUs with 14 cores (28 threads) each, and a total of 504GB RAM. All reported running times are single-threaded. The bottleneck of our approach is the visibility graph computation discussed in Section 2.2. To better understand the trade-off between running time, memory usage, and solution quality with respect to the choice of partition, we conduct experiments comparing Delaunay triangulation and extension partition; see Figure 8. The extension partition introduces a large overhead in running time and memory consumption compared to the Delaunay triangulation, but it reduces the solution size by a significant fraction. While for the extension partition the visibility graph computation clearly dominates the running time, for the Delaunay triangulation it only makes up 32.8% of the running time on average.

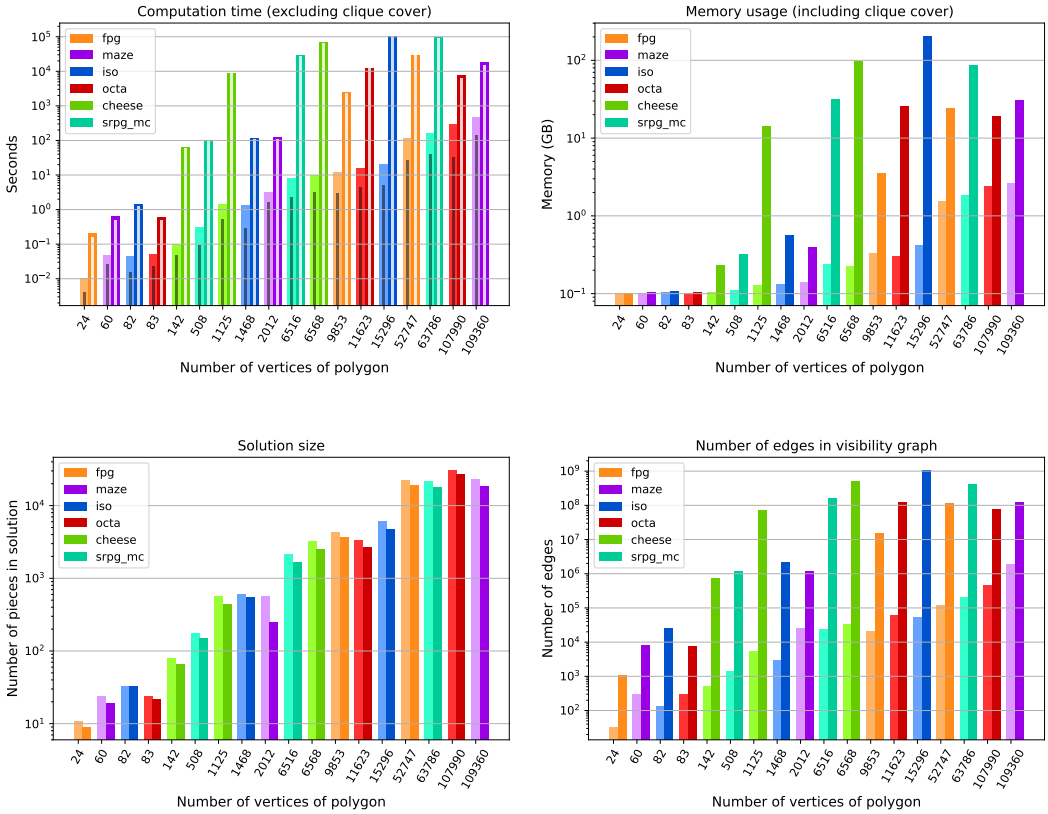



**Figure 8** Experiments with the Delaunay triangulation (left bars) and the extension partition (right bars) as underlying partitions for a selected set of instances. We measure the running time (top left; thin bars showing the running time of the visibility graph computation), memory usage (top right), solution size (bottom left), and number of edges in the visibility graph (bottom right).

## References


1 Mikkel Abrahamsen. Covering polygons is even harder. In *Symposium on Foundations of Computer Science (FOCS)*, pages 375–386, 2021. doi:10.1109/FOCS52979.2021.00045.

2 David Chalupa. Construction of near-optimal vertex clique covering for real-world networks. *Comput. Informatics*, 34(6):1397–1417, 2015. URL: http://www.cai.sk/ojs/index.php/cai/article/view/1276.

3 Joseph C. Culberson and Robert A. Reckhow. Covering polygons is hard. *J. Algorithms*, 17(1):2–44, 1994. doi:10.1006/jagm.1994.1025.

4 Guilherme D. da Fonseca. Shadoks approach to convex covering. In *Symposium on Computational Geometry (SoCG)*, volume 258, 2023. URL: https://pageperso.lis-lab.fr/guilherme.fonseca/CGSHOP23conf.pdf.

5 Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Stefan Schirra. Minimum coverage by convex polygons: The CG:SHOP Challenge 2023, 2023. arXiv:2303.07007.

6 Michael Hemmer, Kan Huang, Francisc Bungiu, and Ning Xu. 2D visibility computation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition, 2023. URL: https://doc.cgal.org/5.5.2/Manual/packages.html#PkgVisibility2.

7 Susan Hert and Stefan Schirra. 2D convex hulls and extreme points. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition, 2023. URL: https://doc.cgal.org/5.5.2/Manual/packages.html#PkgConvexHull2.

8 Darren Strash and Louise Thompson. Effective data reduction for the vertex clique cover problem. In *Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 41–53, 2022. doi:10.1137/1.9781611977042.4.

9 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition, 2023. URL: https://doc.cgal.org/5.5.2/Manual/packages.html.

10 Mariette Yvinec. 2D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition, 2023. URL: https://doc.cgal.org/5.5.2/Manual/packages.html#PkgTriangulation2.