Approximating Robot Configuration Spaces with few Convex Sets using Clique Covers of Visibility Graphs

Peter Werner, Alexandre Amice, Tobia Marcucci, Daniela Rus, and Russ Tedrake

Abstract— **Many computations in robotics can be dramatically accelerated if the robot configuration space is described as a collection of simple sets. For example, recently developed motion planners rely on a convex decomposition of the free space to design collision-free trajectories using fast convex optimization. In this work, we present an efficient method for approximately covering complex configuration spaces with a small number of polytopes. The approach constructs a visibility graph using sampling and generates a clique cover of this graph to find clusters of samples that have mutual line of sight. These clusters are then inflated into large, full-dimensional, polytopes. We evaluate our method on a variety of robotic systems and show that it consistently covers larger portions of free configuration space, with fewer polytopes, and in a fraction of the time compared to previous methods.**

I. Introduction

Approximating complex sets as a union of simpler sets is a common pre-processing method for accelerating downstream computations. Familiar examples include clustering methods for high-dimensional data in machine learning [1, §2.5.2], approximating complex shapes via triangular meshes to facilitate graphics rendering [2], and describing geometries as unions of convex sets for efficient collision checking [3]. Similarly, recently developed methods for robot motion planning rely on (conservative) decompositions of the environment into convex sets to design smooth trajectories around obstacles using efficient convex optimization [4], [5], [6], [7], [8]. These motion planners have demonstrated great potential, and performance frequently superior to widely used sampling-based methods [5]. However, the decomposition of the environment that they require is a daunting task, often demanding a substantial degree of human supervision. This is largely due to the fact that the collision-free subset of a robot's configuration space ($\mathcal{C}^{\text{free}}$) is intractable to describe analytically [9, §3], even if the robot's task space is relatively simple (see Figure 1 for an example).

All authors are with CSAIL, MIT. The corresponding author is P.Werner, wernerpe@mit.edu. This work was supported in part by Amazon.com PO# #2D-06310236, Air Force Research Laboratory FA8750-19-2-1000, Office of Naval Research (ONR), Awards No. N00014-22-1-2121 and N00014-18-1-2830, and the Toyota Research Institute (TRI).

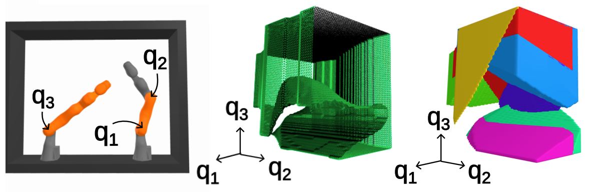


Fig. 1: The collision-free configuration space of a simple robot is decomposed into 7 polytopes, achieving around 92% coverage. *Left*: Robot with with 3 revolute joints q_1 to q_3. *Center*: Visualization of the full collision-free configuration space $\mathcal{C}^{\text{free}}$, given by the interior of the green mesh. *Right*: Approximate convex cover of $\mathcal{C}^{\text{free}}$ generated with the proposed method. See also: *Interactive visualization*, *video*.

In [10], the authors proposed an algorithm called IRIS to quickly compute large convex subsets of $\mathcal{C}^{\text{free}}$. This method takes as input a "seed" configuration and inflates a large polytopic region around this seed point using convex optimization. While originally limited to the case of convex obstacles, the IRIS algorithm has been recently extended to handle nonconvex obstacles and the complicated configuration spaces of multi-link robot manipulators [11], [12], [13]. The polytopes generated with these algorithms have been used with great success for motion planning in high dimensions [5], [6], [7]. Nevertheless, seeding collections of these regions so that they cover diverse areas of $\mathcal{C}^{\text{free}}$ remains a challenge: manual seeding is tedious, and naively growing regions around randomly chosen configurations leads to very inefficient decompositions.

In this paper, we propose an efficient method for the approximate decomposition of robot configuration spaces into few large convex sets, without any human supervision. A guiding illustration is shown in Figure 2.

Similar to some motion-planning algorithms [14], [15], our method constructs a visibility graph by sampling points in $\mathcal{C}^{\text{free}}$. The vertices of this graph are collision-free samples, and the edges connect pairs of points with mutual line of sight. The visibility graph contains rich information about the geometry of $\mathcal{C}^{\text{free}}$.

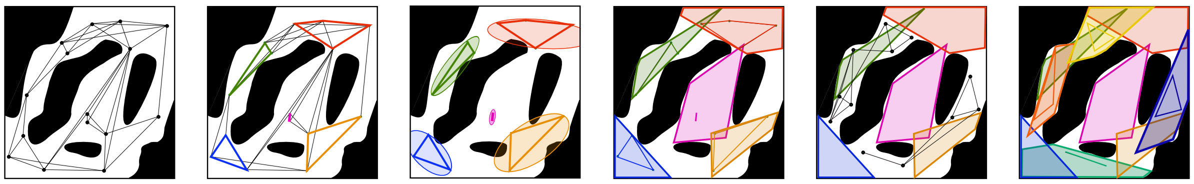


Fig. 2: Sketch of the proposed algorithm on a simple example. *First four figures*: Samples are drawn uniformly from $\mathcal{C}^{\text{free}}$ to build a visibility graph. The visibility graph is decomposed into five cliques. The principal directions and locations of the cliques are used to direct a region-inflation algorithm. *Remaining two figures*: This process is repeated until sufficient coverage is obtained by drawing new samples from the remaining free space, and repeating the previous steps.

In particular, our key observation is that, as the number of samples grows, fully connected subgraphs of this visibility graph (so-called "cliques") tend to represent better and better approximations of collision-free convex sets in the underlying configuration space. Our approach is to decompose the visibility graph into a small collection of large cliques. We then circumscribe the points in each clique with an ellipsoid by solving a convex-optimization problem. The center and the principal directions of these ellipsoids are subsequently used to initialize an inflation algorithm analogous to IRIS.

Through a large variety of experiments, we show that our algorithm outperforms the approaches that have been previously used in the literature to seed and inflate convex regions; both in terms of runtimes and number of regions used to cover the space. As an example, for a robot arm with seven degrees of freedom, and many task-space obstacles, our method requires approximately 46 regions and one hour of computations to cover 70% of $\mathcal{C}^{\text{free}}$. Whereas the approach outlined in [12] requires ten times more regions and is ten times slower.

II. Related Works

Finding the minimum convex cover of a set is a hard problem; even in the case of two-dimensional polygons, the problem is hard to solve exactly and approximately.[1] Nonetheless, finding low-cardinality convex covers of high-dimensional nonconvex spaces (both polygonal and non-polygonal) remains a problem of practical importance, for which a variety of approximate algorithms have been devised. In the following, we group these algorithms into two categories: ones that require explicit (e.g., analytic) descriptions of $\mathcal{C}^{\text{free}}$, and ones that only use implicit descriptions of $\mathcal{C}^{\text{free}}$ (and are hence suitable for most complex configuration spaces). In this literature review, we particularly focus on IRIS algorithms, due to their efficient scaling to high dimensions.

[1]More formally, the problem is $\exists\mathbb{R}$-complete [16], and therefore NP-hard, as well as APX-hard [17].

A. Algorithms Requiring Explicit Obstacle Descriptions

The recent work [18] constructs low-cardinality convex covers of two-dimensional polygons by first decomposing them into small convex pieces (for example triangles), and then by joining the pieces based on a small clique cover of an approximated set-to-set visibility graph. In three dimensions the approach in [19] can be used to decompose $\mathcal{C}^{\text{free}}$ into sets that are approximately convex, provided that a triangle mesh description is available. Finally, if the configuration-space obstacles are explicitly described as convex sets, the original IRIS algorithm from [10] can be used to inflate a large polytope in $\mathcal{C}^{\text{free}}$ around a specified seed point in arbitrary dimensions.

B. Algorithms Allowing Implicit Obstacle Descriptions

Most commonly, the explicit descriptions of the obstacles are only available in task space; while the collision-free configuration space $\mathcal{C}^{\text{free}}$ is defined implicitly through the robot's inverse kinematics, and intractable to describe analytically [9, §3]. Obtaining convex decompositions in this setting has been the focus of multiple works. The method in [20] uses visibility graphs and kernels to compute convex decompositions of three-dimensional spaces via sample-based collision checking. However, that method represents the convex sets through their vertices, and is inefficient in higher dimensions. In the family of IRIS algorithms, two methods can deal with implicit descriptions of $\mathcal{C}^{\text{free}}$: IRIS-NP [12] and C-IRIS [11], [13]. The former extends the original IRIS method [10] to arbitrary configuration spaces using nonlinear programming and inflates polytopes that are collision-free with high probability. The latter grows polytopes that are rigorously *certified* to be collision-free using a rational reparametrization of the configuration space and sums-of-squares programming.

III. Convex Covers, Visibility, and Cliques

In this section, we formally define our main problem: approximating the free configuration space $\mathcal{C}^{\text{free}}$ with a low-cardinality collection of convex sets. We also briefly review the main technical tools that we will use in the development of our algorithm, namely, visibility graphs and clique covers.

A. Problem Statement

Let $\mathcal{C}^{\text{free}} \subseteq \mathbb{R}^n$ be the collision-free subset of an n-dimensional configuration space, which we assume to have a well-defined finite volume. Let also α be a constant in the interval $(0, 1]$.

Definition 1 *An* α-approximate convex cover *of* $\mathcal{C}^{\text{free}}$ *is a collection of potentially overlapping convex sets* $\mathcal{R}_1, \dots, \mathcal{R}_N \subseteq \mathcal{C}^{\text{free}}$ *whose union covers at least an* α*-fraction of the volume of* $\mathcal{C}^{\text{free}}$*:*

$$\textbf{vol}\left(\bigcup_{i=1}^{N} \mathcal{R}_i\right) \geq \alpha\, \textbf{vol}\left(\mathcal{C}^{\text{free}}\right).$$

Our problem is to find an α-approximate convex cover of minimum cardinality N.

Problem: MIN α-APPROXCONVEXCOVER

$$\text{minimize } N \text{ subject to}$$
$$\textbf{vol}\left(\bigcup_{i=1}^{N} \mathcal{R}_i\right) \geq \alpha\, \textbf{vol}\left(\mathcal{C}^{\text{free}}\right),$$
$$\mathcal{R}_i \subseteq \mathcal{C}^{\text{free}}, \quad \forall i = 1, \dots, N.$$

In practice, we are interested in solving this problem for values of α that are sufficiently high to accomplish a task of interest, such as collision-free motion planning, but not so large that the cardinality N grows unreasonably. Indeed, when $\alpha = 1$, **MIN α-APPROXCONVEXCOVER** might not even have a finite solution.[2]

B. Visibility Graphs and Clique Covers

Our algorithm is based on the idea that clusters of points that see each other can approximate convex subsets of $\mathcal{C}^{\text{free}}$. Here, we formally define the notion of visibility as well as introduce some formal tools from graph theory to guide the development of our algorithm.

We begin by defining visibility in $\mathcal{C}^{\text{free}}$.

Definition 2 *Two points* $q, q' \in \mathcal{C}^{\text{free}}$ *are said to* see each other *if the entire line connecting them is collision-free:* $tq + (1-t)q' \in \mathcal{C}^{\text{free}}$ *for all* $t \in [0, 1]$. *Notice that this definition is symmetric in* q *and* q'.

[2]For $\alpha < 1$ a finite solution is guaranteed to exist. This can be seen by approximating the volume of $\mathcal{C}^{\text{free}}$ over finite hyperrectangle partitions with a lower Darboux integral.

We are now ready to define the visibility graph of a set of collision-free configurations.

Definition 3 *The* visibility graph *of a set of points* $q_1, \dots, q_K \in \mathcal{C}^{\text{free}}$ *is an undirected graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *with vertices* $\mathcal{V} = \{1, \dots, K\}$*, and with an edge* $\{i, j\} \in \mathcal{E}$ *for every pair of distinct points* q_i *and* q_j *that see each other.*

We show an example of a visibility graph in Figure 2. We note that clusters of points that can all see each other form a clique in the visibility graph.

Definition 4 *Let* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *be an undirected graph. A* clique $\mathcal{K}$ *is a subset of* $\mathcal{V}$ *where every pair of vertices is connected by an edge.*

Note that if a cluster of configurations can be placed in the same convex set, then these configurations must form a clique in the visibility graph. The second panel in Figure 2 highlights a collection of five cliques in the visibility graph that have this property. These five cliques form what is called a clique cover; which resembles a discrete analog of a convex cover.

Definition 5 *A collection* $\mathcal{T}$ *of cliques* $\mathcal{K}_1, \dots, \mathcal{K}_N$ *is a* clique cover *of a graph* $\mathcal{G}$ *if every vertex in the graph is contained in at least one clique.*

A natural discrete counterpart of the minimum convex cover is the **MINCLIQUECOVER** problem, where we look for the minimum number of cliques N required to cover a graph. Our observation is that, as the number of samples in the visibility graph increases, a minimum clique cover typically does an increasingly better job of approximating a minimum convex cover. Limitations of this analogy are discussed in §VI.

MINCLIQUECOVER is NP-complete [21]. There exist heuristics, such as [22], that attempt to solve **MIN-CLIQUECOVER** directly. Alternatively, one can greedily construct a clique cover by repeatedly eliminating the largest clique. The problem of finding the largest clique in a graph is called **MAXCLIQUE**. Even though this problem is also NP-complete [21], it is often substantially faster to solve in practice. We found the latter approach with exact solutions of **MAXCLIQUE** to perform particularly well on our problem instances.

IV. ALGORITHM

We now present our Visibility Clique Cover (VCC) algorithm, which consists of four main steps. First, we randomly sample a collection of points in $\mathcal{C}^{\text{free}}$ and construct their visibility graph. Second, we compute an approximate clique cover of the graph. Third, we summarize the geometric information of each clique using an ellipsoid. Fourth, we use these ellipsoids to initialize a polytope-inflation algorithm analogous to IRIS. This process is repeated until the generated set of polytopes $\mathcal{R}$ covers a given fraction α of $\mathcal{C}^{\text{free}}$. This procedure is summarized in Algorithm 1, and the remainder of this section details the individual steps.

Algorithm 1: VISIBILITYCLIQUEINFLATION

```
Input :
$\alpha$: coverage threshold
$K$: number of samples per iteration
$s_{\min}$: minimum clique size
Output:
$\mathcal{R}$: set of convex polytopes approximating $\mathcal{C}^{\text{free}}$
Algorithm:
$\mathcal{R} \leftarrow \emptyset$
while CHECKCOVERAGE($\mathcal{R}$) $\leq \alpha$ do
    $\mathcal{G} \leftarrow$ SAMPLEVISIBILITYGRAPH($K, \mathcal{R}$)
    $\mathcal{T} \leftarrow$ TRUNCATEDCLIQUECOVER($\mathcal{G}, s_{\min}$)
    $\mathcal{B} \leftarrow$ MINVOLUMEELLIPSOIDS($\mathcal{T}$)
    $\mathcal{R} \leftarrow \mathcal{R} \cup$ INFLATEPOLYTOPES($\mathcal{B}$)
end
return $\mathcal{R}$
```

A. Sampling the Visibility Graph

At the beginning of every iteration of VCC, the subroutine SAMPLEVISIBILITYGRAPH samples K configurations uniformly at random from the portion of $\mathcal{C}^{\text{free}}$ that is not already covered by the polytopes in $\mathcal{R}$. Then, it constructs the visibility graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, by checking for collisions along the line segments connecting each pair of sampled configurations. Currently, this is performed using sampling-based collision checkers. Exact visibility checking is possible using methods such as [23], [24, §5.3.4], or [13].

B. Truncated Clique Cover

In the subroutine TRUNCATEDCLIQUECOVER we approximately cover the visibility graph with a collection of cliques, each of which contain at least $s_{\min}$ vertices. We construct this approximate cover $\mathcal{T}$ greedily, by solving a sequence of **MAXCLIQUE** problems. Each instance of **MAXCLIQUE** is formulated as an integer linear program

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{K} b_i && \text{(1a)} \\ \text{subject to} \quad & b_i + b_j \leq 1, \qquad \forall \{i, j\} \in \bar{\mathcal{E}}, && \text{(1b)} \\ & b_i \in \{0, 1\}, \qquad \forall i = 1, \dots, K. && \text{(1c)} \end{aligned}$$

A binary decision variable b_i is added for each vertex. The role of this variable is to take unit value if and only if vertex i is included in the clique. The set $\bar{\mathcal{E}}$ contains all the pairs of vertices $\{i, j\}$ such that $i \neq j$ and $\{i, j\} \notin \mathcal{E}$. Therefore the first constraint ensures that two vertices are selected only if they share an edge.

After solving the integer program (1), the clique found is removed from the graph and added to the clique cover. Since small cliques are not informative, we stop this iterative process when the largest clique left in the graph is smaller than a given threshold $s_{\min}$. For this reason, our clique covers will be truncated, i.e., generally, not all vertices will be contained in one of the cliques.

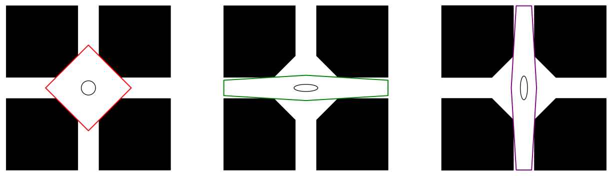


Fig. 3: The growth direction of an IRIS region can be guided by the initial distance metric. IRIS is initialized with three ellipsoids with same center but different principal axes, resulting in polytopes that cover different portions of $\mathcal{C}^{\text{free}}$.

C. Summarizing Cliques with Ellipsoids

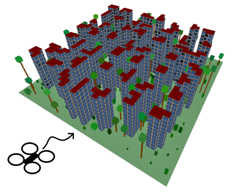
In the subroutine MINVOLUMEELLIPSOIDS, we solve a semidefinite program to enclose each clique with an ellipsoid of minimum volume [25, §8.4.1]. This collection $\mathcal{B}$ of ellipsoids allows us to summarize the geometry of each clique with a point and a set of principal directions, which are then used to initialize the region-inflation algorithm. For the upcoming computations, it is necessary that the center of each ellipsoid is not in collision; if this is not the case, we recenter the ellipsoid around the vertex in the clique that is closest to its center.

D. Inflating Polytopes

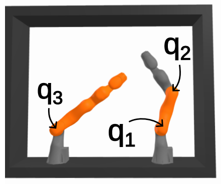
In the last step of VCC, the subroutine INFLATEPOLYTOPES inflates a large collision-free polytope around the center of each ellipsoid induced by a clique.

Let us initially assume that the obstacles are convex. Consider a single ellipsoid. Using convex optimization we compute the point in each obstacle that is closest to the center of the ellipsoid, according to the distance metric induced by the ellipsoid. These points anchor separating hyperplanes between the ellipsoid center and the obstacles, which form a polytope of obstacle-free space. These steps are repeated for each ellipsoid (i.e., for each clique) to obtain a collection of collision-free polytopes that we add to the set $\mathcal{R}$.

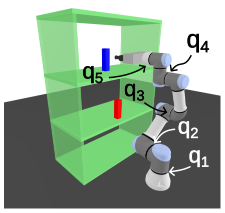
These computations correspond to a single iteration of the IRIS algorithm [10], and ensure that the largest uniformly-scaled, collision-free version of the ellipsoid is contained in the resulting polytope. Figure 3 illustrates



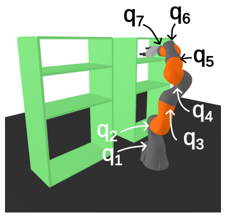
(a) `Village`



(b) `3DOF Flipper`



(c) `5DOF UR5`



(d) `7DOF IIWA`

Domain	`Village`		`3DOF Flipper`		`5DOF UR5`		`7DOF IIWA`	
Algorithm	IOS	VCC (ours)	IOS	VCC (ours)	IOS	VCC (ours)	IOS	VCC (ours)
# regions $\lvert\mathcal{R}\rvert$	198.0±13.6	**93.9**±7.5	10.4±1.9	**6.7**±0.5	90.5±18.8	**35.1**±1.6	482.6±83.8	**46.3**±4.5
runtime [s]	2.7e3±2.9e2	**1.7e3**±3.4e2	2.0e2±2.9e1	**4.9e1**±7.1	1.12e4±2.1e3	**5.5e2**±6.6e1	8.9e4±1.9e4	**5.7e3**± 1.7e3
coverage threshold α	0.8	0.8	0.9	0.9	0.75	0.75	0.7	0.7
# visibility vertices K	-	500	-	500	-	1000	-	1500
min. clique size s_{min}	-	10	-	10	-	10	-	20

TABLE I: Comparison of our Visibility Clique Cover (VCC) algorithm to the Iterative Obstacle Seeding (IOS) method from [12], across four different environments. All experiments are repeated ten times. The numbers in the first two rows indicate the mean and the empirical standard deviation over the trials. We observe that VCC achieves the given coverage targets with 1.6 to 10.4 times fewer regions, and between 1.4 to 20 times faster than IOS. The environment names are linked to interactive visualizations.

how the initial metric is fundamental in guiding the shape of the regions generated by IRIS. Traditionally, the IRIS algorithm was initialized with an uninformed ellipsoidal metric and needed to run for multiple (expensive) iterations in order to expand and cover a larger volume of space; in VCC we require only a single IRIS iteration.

When the obstacles are not convex, we run one iteration of the nonlinear programming variant IRIS-NP [12] instead. Alternatively, C-IRIS [11], [13] could be employed to obtain certifiably collision-free regions.

E. Convergence Check

The subroutine CHECKCOVERAGE estimates the fraction of $\mathcal{C}^{\text{free}}$ covered by the regions in $\mathcal{R}$, and terminates our algorithm if this value exceeds the threshold α. Computing this fraction exactly is impractical, and so we resort to randomized methods. The coverage is estimated by drawing a large number of M samples in $\mathcal{C}^{\text{free}}$, and computing the ratio of samples that land in at least one of the regions in $\mathcal{R}$. More sophisticated checks, such as one-sided Bernoulli hypothesis testing, are possible.

F. Completeness

Analogous to [26], [27], VCC is probabilistically complete under mild assumptions. This means as the number of iterations goes to infinity, the probability of completely covering $\mathcal{C}^{\text{free}}$ goes to one.

V. EXPERIMENTS

As there are no direct baseline methods, we compare our VCC algorithm against an extension of the method in [12, §III.D]. The natural extension of this approach is to iteratively grow polytopes around uniformly sampled points from the uncovered free space using IRIS, while treating previously computed regions as obstacles. This process is repeated until the desired coverage is met. We call this approach Iterative Obstacle Seeding (IOS).

In the following, all experiments are implemented in Drake [28], and all computations are performed on a single desktop computer with an Intel Core i9-10850K CPU and 32 Gb of RAM. We solve all **MAXCLIQUE** instances to global optimality using Gurobi [29].

We evaluate VCC and IOS on four environments: `Village`, `3DOF Flipper`, `5DOF UR5`, and `7DOF IIWA`. The dimension n of these environments ranges from 3 to 7. For each environment, we run the two algorithms ten times and report their performance in Table I. The `Village` environment from [8] contains only convex obstacles and is compatible with the original IRIS [10]. All other examples involve the configuration spaces of robotic manipulators and therefore IRIS-NP [12] is employed. The number of samples used in the convergence check in Algorithm 1 is $M = 5000$.

VCC meets the required coverage threshold with significantly fewer regions and in a substantially shorter amount of time. Notably, in the most challenging benchmark, `7DOF IIWA`, VCC requires 10 times fewer regions and meets the required coverage of 70% around 16 times faster. This substantial speedup can be attributed to two key factors. First, the region inflation in VCC is parallelized. Second, IRIS is the most computationally expensive step. While VCC only requires a single iteration of IRIS per region, IOS can require up to around five IRIS iterations per region, due to the initialization with a potentially uninformative spherical metric.

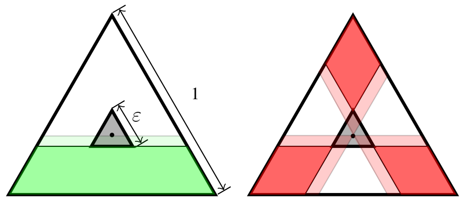


Fig. 5: Maximum cliques of infinitely dense visibility graphs can enclose holes, and do not necessarily correspond to collision-free convex sets. The largest collision-free convex region (green trapezoid) has a smaller area than the union of red parallelograms when $0 < \varepsilon \leq 1 - \sqrt{5/6}$. In this case, the convex hull of the maximum clique must enclose the hole.

VI. Limitations of Approximating Convex Sets with Cliques

Despite the strong performance of our algorithm, the hardness of solving **Min α-ApproxConvexCover** makes it possible to construct simple examples that highlight pitfalls of our heuristic approach. In this section, we discuss holes in $\mathcal{C}^{\text{free}}$ which leads to one such pitfall that is particularly insightful.

While every convex set in $\mathcal{C}^{\text{free}}$ naturally corresponds to a clique in the visibility graph, a clique in the visibility graph does not necessarily correspond to a convex set in $\mathcal{C}^{\text{free}}$. The convex hull of a clique can enclose holes. This problem persists even if we sample arbitrarily dense visibility graphs, and if we restrict the analysis to maximum cliques. A visual proof is shown in Figure 5, which illustrates a triangular configuration space with a triangular hole of size ε. As ε goes to zero, the largest convex subset of $\mathcal{C}^{\text{free}}$ is the green trapezoid, whose area approaches $5/9$ of the total area of $\mathcal{C}^{\text{free}}$. On the other hand, a larger subset of mutually visible points is given by the union of the three red parallelograms, whose area approaches $6/9$ of the total area. Therefore, if configurations are sampled uniformly at random, an optimum solution of **MaxClique** will almost surely enclose a hole as the number of samples goes to infinity. A similar construction can be used to show an analogous discrepancy between the minimum convex cover of $\mathcal{C}^{\text{free}}$ and **MinCliqueCover**.

In principle, this problem can be addressed by solving a modified version of **MaxClique** that better captures the notion of a convex set. In short, we require that every vertex of the visibility graph that is contained in the convex hull of the maximum clique must be a member of the clique. For an infinitely dense visibility graph, this ensures that the maximum clique cannot enclose holes.

We enforce the contrapositive of the latter condition through linear constraints that separate all non-clique members q_i, with $i \in \{1, \ldots, K\}$, from the points in the clique with a hyperplane $\mathcal{H}_i = \{q \mid c_i^T q + d_i = 0\}$, parameterized by the decision variables $(c_i, d_i) \in \mathbb{R}^{n+1}$.

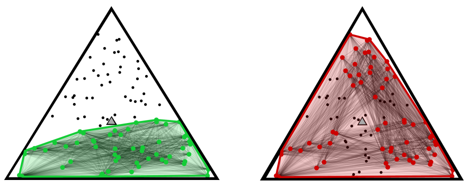


Fig. 6: A visibility graph with 100 random vertices in the triangular environment from Figure 5. Solving the maximum clique problem (1) with the additional constraints (2) yields a clique with 56 vertices (shown in green), that closely approximates the corresponding convex set in Figure 5. Solving only problem (1), yields a clique with 63 vertices (shown in red) that, however, encloses the central hole.

The resulting mixed-integer linear optimization extends (1) by adding the additional separation constraints

$$c_i^T q_i + d_i \geq 1 - b_i, \qquad \forall\, i = 1, \ldots, K, \qquad (2a)$$

$$c_i^T q_j + d_i \leq M(1 - b_j), \qquad \forall i, j = 1, \ldots, K, \qquad (2b)$$

where M is a large enough constant (e.g. the maximum distance between all pairs of vertices).

When the point q_i is not in the clique ($b_i = 0$) and the point q_j is in the clique ($b_j = 1$), these constraints read $c_i^T q_i + d_i \geq 1$ and $c_i^T q_j + d_i \leq 0$. Therefore, the hyperplane $\mathcal{H}_i$ separates q_i from q_j. On the other hand, these constraints are seen to be redundant for all other possible values of the binaries b_i and b_j. In Figure 6, we demonstrate how this extension prevents a maximum clique from enclosing holes in a finite-sample regime.

In practice, solving (1) with constraints (2) is too expensive for the problems in Table I. Nonetheless, we observed that in the first **MaxClique** problem (1) only an average of 0.1% of the vertices were excluded from the clique that could not be separated from it with a hyperplane. Subsequent cliques, and improvements to the formulation (2), demand a more nuanced discussion and are subject to future work.

VII. Conclusions

We have proposed an algorithm for approximately decomposing complex configuration spaces into small collections of polytopes. Our algorithm uses clique covers of visibility graphs as an effective heuristic for obtaining local information about $\mathcal{C}^{\text{free}}$, and for seeding a region-inflation algorithm. The parallels between convex sets in $\mathcal{C}^{\text{free}}$ and cliques in visibility graphs have also been discussed. Our experiments demonstrate that VCC reliably finds approximate convex covers of $\mathcal{C}^{\text{free}}$ with fewer regions and in less time than previous approaches.

Acknowledgements

The authors would like to thank Steven M. LaValle, Thomas Cohn, Annan Zhang, and Fabian Dickhardt for the many fruitful discussions.

References

[1] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[2] M. Pharr, C. Kolb, R. Gershbein, and P. Hanrahan, "Rendering complex scenes with memory-coherent ray tracing," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 101–108.

[3] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[4] T. Marcucci, J. Umenberger, P. A. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *arXiv preprint arXiv:2101.11565*, 2021.

[5] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *arXiv preprint arXiv:2205.04422*, 2022.

[6] T. Cohn, M. Petersen, M. Simchowitz, and R. Tedrake, "Non-euclidean motion planning with graphs of geodesically-convex sets," *arXiv preprint arXiv:2305.06341*, 2023.

[7] V. Kurtz and H. Lin, "Temporal logic motion planning with convex optimization via graphs of convex sets," *arXiv preprint arXiv:2301.07773*, 2023.

[8] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," *arXiv preprint arXiv:2305.01072*, 2023.

[9] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[10] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 109–124.

[11] A. Amice, H. Dai, P. Werner, A. Zhang, and R. Tedrake, "Finding and optimizing certified, collision-free regions in configuration space for robot manipulators," in *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 328–348.

[12] M. Petersen and R. Tedrake, "Growing convex collision-free regions in configuration space using nonlinear programming," *arXiv preprint arXiv:2303.14737*, 2023.

[13] H. Dai, A. Amice, P. Werner, A. Zhang, and R. Tedrake, "Certified polyhedral decompositions of collision-free configuration space," *arXiv preprint arXiv:2302.12219*, 2023.

[14] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[15] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[16] M. Abrahamsen, "Covering polygons is even harder," in *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 375–386.

[17] S. J. Eidenbenz and P. Widmayer, "An approximation algorithm for minimum convex cover with logarithmic performance guarantee," *SIAM Journal on Computing*, vol. 32, no. 3, 2003.

[18] M. Abrahamsen, W. Bille Meyling, and A. Nusser, "Constructing concise convex covers via clique covers (cg challenge)," in *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[19] K. Mamou and F. Ghorbel, "A simple and efficient approach for 3d mesh approximate convex decomposition," in *2009 16th IEEE international conference on image processing (ICIP)*. IEEE, 2009, pp. 3501–3504.

[20] A. Sarmientoy, R. Murrieta-Cidz, and S. Hutchinsony, "A sample-based convex cover for rapidly finding an object in a 3-d environment," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3486–3491.

[21] R. M. Karp, *Reducibility among combinatorial problems*. Springer, 2010.

[22] D. Strash and L. Thompson, "Effective data reduction for the vertex clique cover problem," in *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 2022, pp. 41–53.

[23] F. Schwarzer, M. Saha, and J.-C. Latombe, "Exact collision checking of robot paths," *Algorithmic foundations of robotics V*, pp. 25–41, 2004.

[24] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[25] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[26] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[27] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[28] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[29] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023.