Improved Bounds for Geodetic Hulls

No Author Given

No Institute Given

Abstract. In this paper we study geodetic convex hulls in graphs and prove new upper and lower bounds on the size of hull sets and related parameters. A set of vertices in a graph is *geodetic convex* if no shortest path between vertices in the set leaves the set. The *geodetic hull* of a vertex set is its smallest convex superset. A *hull set* is a set of vertices whose geodetic hull is the entire graph. Computing a hull set of minimum size poses an NP-hard problem. We present an exact kernelization algorithm which implies that the problem is FPT in the vertex cover number. Furthermore we provide new bounds on the size of a minimum hull set in relation to other graph parameters. For practical application, we design an improved heuristic as well as an algorithm to obtain strong instance-based lower bounds. We evaluate the efficiency and quality of the heuristic and the lower bound on a diverse set of graphs. Our experiments demonstrate that our methods outperform previous ones by a large margin.

Keywords: Geodetic hull · Geodetic iteration number · Minimum hull set.

1 Introduction

Convexity in Euclidean space has long been studied and applied to many problems to great benefit [21]. The underlying concept of convexity is the ternary relation of *betweenness*: A set $S \subseteq V$ is convex in a space V if for all $(a, b, c) \in$ betweenness (i.e., b lies between a and c) with a and c in S, b lies in S as well. For Euclidean space, betweenness is defined via straight-line segments. In graphs the most common notion of betweenness is the following. Vertex u lies *between* vertices v and w if it lies on a shortest path from v to w [20]. This definition gives rise to what is known as *geodetic convexity*, along with the *geodetic (convex) hull*. However fundamental properties of convexity in Euclidean space do not carry over to shortest paths in graphs. Thus the analysis of geodetic convexity deserves attention as a separate research topic. Indeed there are many applications of geodetic convexity, for example in data structure design [18], hole detection in sensor networks [17], and cluster recovery [6]. A central problem is to find a minimum set S of vertices whose geodetic hull is equal to the vertex set of the graph; such a set S is called a *hull set* of the graph [15]. Solving this problem is NP-hard [12,2,1,13,5], and to date no approximation algorithm is known. The goal of the paper is to offer new theoretical insights that facilitate practical computation of geodetic hulls.

Use \emph, not \textsl(?)

the ?

Make \paragraph to save space.

1.1 Related Work

Geodetic hulls were first defined ~~in~~ by Harary and Nieminen [20]. Not much later, Everett and Seidman [15] defined the hull number, characterized it for some graph classes, and gave bounds for it. ~~NP-hardness of~~ Computing the hull number ~~was proven in~~ turned out to be NP-hard [12]. Since then, the problem has proven intractable even for some restricted graph classes such as bipartite graphs [2] and chordal graphs [5], while it is solvable in polynomial time on others [12,2,3,13]. ~~In [24], a linear-time algorithm is presented that computes geodetic hulls~~ In outerplanar graphs, geodetic hulls can be computed in linear time [24].

Closely related to the hull set and the hull number is the geodetic set and the geodetic number. They were first defined ~~in~~ by H., L., and T. [19], who also showed that computing the geodetic number is NP-hard. Since then it was shown to be NP-hard as well on restricted inputs [14,10], and its parameterized tractability has been studied extensively as well [14,16]. Much research has been done on the geodeticity of certain vertex sets; in particular the graph contour was first defined and its geodeticity studied in [8]. One important aspect is the so-called *geodetic iteration number* (gin), which counts how often the interval operation has to be repeated on an input set until it is convex. It has been extensively researched under which circumstances the contour has gin 1 [9,7,4,23]. It had been conjectured for a long time that its gin is always at most 2, but [22] gave a counterexample with a contour that has gin 3. Inspired by this, [25] present further bounds on the gin of the contour, as well as further heuristics and bounds for hull sets.

define!

results?

define here!

1.2 Contribution

In this paper, we provide several new results on the hull number of general graphs and the gin of the contour, a particular hull set. First, we prove that computing a minimum hull set is FPT in the vertex cover number of the graph. This is the first parameterized complexity result for this problem. Next, we show a lower bound on the hull number that can be used to obtain instance-based lower bounds. We also provide novel bounds on the gin of the contour in relation to the graph diameter and the cardinality of the contour. Both of these results improve on the best previously known bounds shown in [25]. Moreover our bounds on the gin of the contour are tight. Furthermore we discuss heuristic approaches for computing small hull sets in practice and analyze their performance. Finally we compare our new bounds and algorithms with the existing ones in an extensive experimental study. As a main outcome, we observe that our heuristic is usually faster and produces ~~smaller~~ hull sets ~~than the computation of~~ the graph contour.
that are smaller than
than computing the graph contour

2 Basics

Throughout the paper we assume to be given a connected, undirected, and unweighted graph ~~G~~.

2.1 Convexity in Graphs and Euclidean Spaces

The central concept that we study in this paper is convexity, which is closely connected to the notion of intervals in graphs. The *(closed) interval* $I[S]$ of a vertex set S is the set of vertices that lie on a shortest v-w-path for some v, $w \in S$. If there is at most one shortest v-w-path for each pair of vertices v and w, the graph is also called *geodetic*. A vertex set S is *geodetic convex* if $I[S] = S$. The *geodetic iteration number* $\text{gin}(S)$ is the smallest k such that $I^k[S] = I^{k+1}[S]$, that is, the computation has *converged*. In that case the *geodetic hull* of S is $\text{hull}(S) = I^k[S]$. Let S and T be vertex sets; if $I[S] \supseteq T$, then S is a *geodetic set* of T; if $\text{hull}(S) \supseteq T$, then S is a *hull set* of T. The *hull number* $\text{hn}(T)$ is the size of the smallest hull set of T, and the *geodetic number* $\text{gn}(T)$ is that of the smallest geodetic set of T.

These concepts are derived from convexity in Euclidean space where a set of points is *convex* if no straight-line segment between points of the set leaves the set and the *convex hull* of a set S is defined as the smallest convex superset of S. A natural way to compute the convex hull is to repeatedly add all points on segments between points of the set to the set until the set does not change anymore. We call the number of iterations until convergence the *Euclidean iteration number* of the set. Obviously, this number is at most 2 for points in the plane. However, we show that in higher dimensional space, the iteration number can be larger but not arbitrarily so.

Lemma 1. *In d dimensions the Euclidean iteration number of a set of points is ~~less than or equal to~~ $\lfloor \log_2 d \rfloor + 1$. This bound is tight.*
at most

Proof (sketch). It suffices to consider d-simplices. Here we can represent each point in the convex hull as a $(d+1)$-tuple of barycentric coordinates. For the seed points, exactly one coordinate is 1, all others are 0. After the ith iteration, we have reached exactly the points with at most 2^i nonzero coordinates. Thus convergence occurs after iteration $\lceil \log_2(d+1) \rceil = \lfloor \log_2 d \rfloor + 1$. Refer to the appendix for details. □

For the analogue in graphs—the geodetic iteration number (gin)—the upper bound is $n-3$ for $n \geq 5$ [20]. This shows one of the many structural differences between convexity in Euclidean spaces and in graphs.

2.2 Properties and Computation of Geodetic Hulls

We now discuss some basic properties of geodetic hulls which we will later leverage in our algorithms.

Observation 1. *The hull operator* hull *is a closure operator, that is, for all vertex sets S and T with $S \subseteq T$, we have $S \subseteq \text{hull}(S)$ (extensivity),* $\text{hull}(S) \subseteq \text{hull}(T)$ *(monotonicity), and* $\text{hull}(\text{hull}(S)) = \text{hull}(S)$ *(idempotence).*

For computing the interval of a given node set S, we need to compute all shortest paths between its members. The standard method to compute shortest paths from v to the vertices in a set S is *breadth-first search* (BFS). Usually we record only one predecessor for each vertex; if we then collect the shortest v-w-path for all $w \in S$, we obtain a $\text{BFS}(v, S)$ *tree*. If we instead record a *set* of predecessors for each vertex, we can collect *all* shortest paths to vertices in S, which yields the $\text{BFS}(v, S)$ *DAG*.

Observation 2. *Any directed path in any BFS DAG is a shortest path in G.*

Now computing an interval $I[S]$ boils down to computing the $\text{BFS}(v, S)$ DAG for each vertex $v \in S$ and taking the union of the sets of vertices contained in these DAGs. If one uses a specialized vertex set data structure that implements operations in constant time, the whole procedure runs in time $\mathcal{O}(|S| \cdot m)$. , where $m = |E(G)|$. In order to compute the geodetic hull and the gin of S, we may iterate $I[\cdot]$ until convergence, counting the iterations, which takes time $\mathcal{O}(|\text{hull}(S)| \cdot \text{gin}(S) \cdot m)$. Note the dependency on the *output* size.

2.3 Minimum Hull Sets and the Graph Contour

A hull set of a graph G is a node subset S for which $\text{hull}(S) = V(G)$. Naturally one aims to find small hull sets.

Problem: MinimumHullSet
Input: An unweighted graph G and a positive integer k.
Question: Does a vertex set $S \subseteq V(G)$ exist such that $|S| \leq k$ and $\text{hull}(S) = V(G)$?

Ct(G) ?

The problem is NP-hard, even if the input is restricted to bipartite graphs [2].

The graph contour is always a hull set of the graph [8]. For its formal definition, let S be a set of vertices. For a vertex v, its *eccentricity* $\mathrm{ecc}_S(v)$ is the maximum distance from v to any other vertex in S. A path with strictly monotonically increasing eccentricities is called a *trail* of its start vertex; the *trail set* $\mathrm{tr}(v)$ of vertex v is then the set of vertices w for which a v-w-trail exists; A vertex is contained in the *contour* $\mathrm{Ct}(S)$ of S if it has maximum eccentricity among its neighbors in S; see Fig. 1 for an illustration. The contour can be computed in $\mathcal{O}(nm)$ by running BFS from each node to compute its eccentricity.

time

iff (Better: The contour of S consists of those vertices in S that have maximum ecc. among their neighbors.)

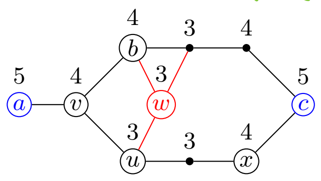


Fig. 1. A graph G with non-unique shortest paths between a and c. The numbers indicate the eccentricities. Its diameter is 5, its radius 3. Its contour is $\{a, b, c\}$. We have $\mathrm{gin}(\mathrm{Ct}(G)) = 2$, because $I[\mathrm{Ct}(G)]$ does not contain w. The path $\langle u, v, a\rangle$ is a trail.

Is $\mathrm{tr}(u) = \{u, v, a\}$ or $\{v, a\}$?

3 Finding Minimum Hull Sets

In this section we devise an exact kernelization algorithm for MinimumHullSet, which implies that the problem is FPT with respect to the vertex cover number vc. Our algorithm and its analysis are inspired by [16], which considers geodetic sets. In the following we focus on the necessary changes to make their approach work for MinimumHullSet. In particular, while one could adapt it straightforward, we employ a refinement which yields smaller kernels and whose reduction rules can be applied in fewer steps.

First, we need a few basic definitions. Let v and w be vertices. The *open neighborhood* $N(v)$ of v is the set of vertices that are adjacent to v. The *closed neighborhood* of v is $N[v] := N(v) \cup \{v\}$. If the subgraph induced by $N(v)$ is a complete graph, v is called *simplicial*. The vertices v and w are called *false twins* if $N(v) = N(w)$, and they are *true twins* if $N[v] = N[w]$. ~~It was proven in~~ [15] ~~that~~ If v is a simplicial vertex, then ~~each~~ every hull set of the graph contains v.

I'd make this a lemma so you can cite it below. It's used often.

Our algorithm works by first constructing a kernel whose size depends only on vc (Lemma 3). Then we ~~can~~ solve the kernel using brute force (Lemma 4).

Lemma 2. *If a graph G contains a set T of false twins that are not simplicial, then any minimum hull set of G contains at most two vertices of T.*

Proof. Let T ~~$= \{t_1, \ldots, t_h\}$~~ be a set of false twins in G which are neither simplicial nor true twins. Thus T forms an independent set, and there are two non-adjacent vertices x and y in the neighborhood of the vertices in T. Assume for a contradiction that $\overset{|T|}{h} \geq 3$ and that G has a minimum hull set H that contains at least three vertices of T~~; w.l.o.g. assume $\{t_1, \ldots, t_3\} \subseteq H$~~. We claim that $H' := (H \setminus$ T ~~$\{t_1, t_2, t_3\}$~~$) \cup \{x, y\}$ is still a hull set, contradicting the choice of H as a minimum hull set of G. To see this, notice that any vertex ~~from~~ in T lies on a shortest x-y-path. Thus $\mathrm{hull}(H') = \mathrm{hull}(I[H']) \supseteq \mathrm{hull}(H) = V(G)$, and H' is a hull set. □

Lemma 3. *MinimumHullSet parameterized by the vertex cover number* $\mathtt{vc}$ *admits a polynomial-time kernelization algorithm that returns an instance with* $\mathcal{O}(2^{\mathtt{vc}})$ *vertices.*

Proof (sketch). The kernelization algorithm exhaustively applies the following reduction rules in a sequential manner to the instance (G, k).

Reduction Rule 1. If G contains three simplicial vertices that are (false or true) twins, delete one of them from G and decrease k by one.
Reduction Rule 2. If G contains four vertices that are false twins, delete one of them from G.

With Lemma 2, correctness and runtime ? are analogous to [16]—with the exception of the fact that our Lemma 2 is an upper bounds of two, on the number of false non-simplicial twins. whereas the corresponding lemma from [16] is an upper bound of four. yields? As a consequence we can delete one of *four* false twins in reduction rule 2 instead of six. This means that we can apply reduction rule 2 by trying only $\binom{n}{4} = \Theta(n^4)$ candidates instead of $\binom{n}{6} = \Theta(n^6)$ and that the resulting kernel contains only $\mathtt{vc} + 3 \cdot 2^{\mathtt{vc}}$ vertices in the worst case instead of $\mathtt{vc} + 5 \cdot 2^{\mathtt{vc}}$. Further details can be found in the appendix. □

Lemma 4. *MinimumHullSet admits an algorithm running in time* $\mathcal{O}(n^{\mathtt{vc}+3})$.

Proof (sketch). Let X be a minimum vertex cover, and let S be the set of simplicial vertices. Then $X \cup S$ is a hull set. Since S ~~must be~~ is contained in ~~all~~ every hull set~~s~~, we need only try each candidate~~s~~ set $S \cup S'$ ~~for all~~ with $S' \subseteq V(G) \setminus S$ and $|S'| \leq |X|$. There are only $\mathcal{O}(n^{\mathtt{vc}})$ such ~~S'~~ candidate sets. For details refer to the appendix. □

We are now ready to state our main theorem.

Theorem 1. *MinimumHullSet admits an FPT algorithm running in time* $2^{\mathcal{O}(vc^2)} + n^{\mathcal{O}(1)}$.

Proof. Using Lemma 3 we compute a kernel with $n' = \mathcal{O}(2^{\mathtt{vc}})$ vertices in polynomial time. Then we find the answer for the kernel using Lemma 4 in time $\mathcal{O}(n'^{\mathtt{vc}+3}) = \mathcal{O}\big(2^{\mathtt{vc}^2} \cdot 8^{\mathtt{vc}}\big)$. □

4 Bounding the Hull Number

There is currently no polynomial-time approximation known for computing hull sets of minimum size. A first step towards such an algorithm is the establishing of lower bounds on the optimum solution size. One such bound namely ? was shown in [25]. The following lemma generalizes [25, Lemma 2] and enables us to state an improved lower bound, which is illustrated in Fig. 2.

Usually this has a different meaning. I'd call these vertices "boundary vertices".

Lemma 5. *Let $W \subseteq V(G)$, and let* $\mathrm{cut}(W)$ *be the set of cut vertices in W, that is, vertices that are adjacent to a vertex not in W. If $W \setminus \mathrm{hull}(\mathrm{cut}(W)) \neq \emptyset$, then every hull set of G must contain at least one vertex from $W \setminus \mathrm{hull}(\mathrm{cut}(W))$.*

Proof. Assume that H is a hull set of G, $H \cap$ ~~$(W \setminus \mathrm{hull}(\mathrm{cut}(W)))$~~ T $= \emptyset$, and $T := W \setminus \mathrm{hull}(\mathrm{cut}(W)) \neq \emptyset$. Define $\overline{T} := V(G) \setminus T$. Now take any pair of vertices $u, v \in \overline{T}$ and any shortest u-v-path P. If a vertex $w \in W$ lies on P, then the maximal subpath of P in W on which w lies must begin and end at vertices in $\{u, v\} \cup \mathrm{cut}(W) \subseteq \mathrm{hull}(\mathrm{cut}(W))$. By the definition of T, none of its vertices can lie on such a path. Hence $\overline{T}$ is convex. From $H \subseteq \overline{T}$ and the monotonicity of the hull operator (Observation 1), we can infer that $\mathrm{hull}(H) \subseteq \underbrace{\mathrm{hull}(\overline{T})}_{=\overline{T}} \subsetneq V(G)$. Therefore H is not a hull set. □

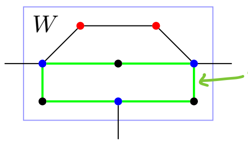


Fig. 2. A vertex set W (light blue box), its cut vertices $\mathrm{cut}(W)$ (dark blue dots), the shortest paths between them (green lines), and $T = W \setminus \mathrm{hull}(\mathrm{cut}(W))$ (red dots), of which at least one vertex must be contained in any hull set [25, Fig. 5].

earlier!

~~See Fig. 2 for a small example.~~ If we choose a suitable family $\{W_i\}_{i \in \{1,\dots,k\}}$ of vertex sets, we can use this to show that $\mathrm{hn}(G) \geq k$. Note that for distinct i and j, W_i and W_j may overlap in the hulls of their sets of cut vertices. The authors of [25] only show that every hull set must contain a vertex from each W_i; there W_i and W_j must not overlap for $i \neq j$ if we want to establish a lower bound on the hull number. While this may seem like a small difference, it enables us to include the entire hull of the cut vertices of W_1 in W_2 and so on, in the hope that a larger W_2 will have a smaller set of cut vertices.

consider to introduce $[k]$ as shorthand for this.

Now, in order to derive a lower bound from this, we must settle on a choice of W_i for some k and $i \in \{1, \dots, k\}$; then we can apply [25, Lemma 2] or our Lemma 5 to each W_i to obtain the corresponding set of cut vertices $\mathrm{cut}(W_i)$ and finally a lower bound of 0 or 1 for $|S \cap W_i|$ or $|S \cap (W_i \setminus \mathrm{hull}(\mathrm{cut}(W_i)))|$, respectively, for any hull set S. We use the same technique to find W_i as [25] but refine it. This technique works by finding several approximately geodetic convex vertex sets. Recall ~~Thinking back to~~ the analogy of convex point sets in Euclidean space, there we might solve th~~is~~e problem of subdividing a convex set into several convex sets by sampling some set of seed points and computing the corresponding Voronoi diagram. If we go back from Euclidean space to graphs, we arrive at the following. First, we find a hull set H by running 1-pruning from a vertex at the end of some long path. Then, we run BFS from all vertices in H simultaneously to find, for each vertex in the graph, the closest vertex in H. For $H = \{v_1, \dots, v_k\}$,

?

we then set W_i to the set of vertices that are closest to v_i. This is the approach from [25], where W_i and W_j are disjoint for distinct i and j.

On this we improve in two ways: On one hand we use our stronger Lemma 5, which lets us choose overlapping $\{W'_i\}_{i\in\{1,\ldots,k\}}$, as long as they share vertices only in the hulls of their cut vertices. To this end let $H_i := \text{hull}(\text{cut}(W'_i))$ and $B_i := W'_i \setminus H_i$ for $i \in \{1,\ldots,k\}$. We choose W'_i as the maximal connected subgraph containing W_i but containing no vertices from (1) $\bigcup_{j<i} B_j$ nor from (2) $\bigcup_{\ell>i} W_\ell$; this ensures sound $\{W'_i\}_{i\in\{1,\ldots,k\}}$, because $B_i \subseteq W'_i$ and B_j certainly will not overlap for $j < i$ by (1); moreover $W'_i \supseteq W_i$ is still possible because by (2), we never chose W'_j (again $j < i$) such that it contained a vertex from W_i. If we maintain the two sets $\bigcup_{j<i} B_j$ and $\bigcup_{\ell>i} W_\ell$ during the computation, we can find W'_i in linear time using a graph-search algorithm like BFS.

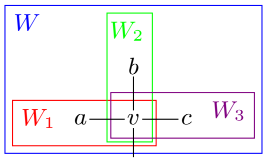


Fig. 3. A small example where we have $\text{cut}(W) = \text{hull}(\text{cut}(W)) = \{v\}$. $W \setminus \text{hull}(\text{cut}(W)) = \{a,b,c\}$ has three connected components, induced by $\{a\}$, $\{b\}$, and $\{c\}$, respectively. Hence we can choose $W_1 := \{v,a\}$, $W_2 := \{v,b\}$, and $W_3 := \{v,c\}$, which yields a lower bound of 1 for each of these three sets for a total of 3. ← Don't you mean W'_1, W'_2, W'_3 ?

Common notation for complete graphs.

On the other hand if we find that $W'_i \setminus \text{hull}(\text{cut}(W'_i))$ has multiple connected components $K_1, \ldots, K_c$, then we can conceptually choose $W'_{i1} := K_1 \uplus \text{hull}(\text{cut}(W'_i)), \ldots, W'_{ic} := K_c \uplus \text{hull}(\text{cut}(W'_i))$ and apply Lemma 5 to each of them, which yields a total of c vertices that must be contained in any hull set. Of course we need not compute c geodetic hulls for this; it suffices to compute $\text{hull}(\text{cut}(W'_i))$ and count the number of connected components in $W'_i \setminus \text{hull}(\text{cut}(W'_i))$, again using some graph-search algorithm running in linear time. See Fig. 3 for a small example, where [25] finds that at least one of the four vertices must be contained in any hull set, while we find that all three of the three leaves must be contained in any hull set.

Remark 1. Generalizing the example from Fig. 3 to a star graph with n vertices, the gap between the bound from [25] and ours is in $\Theta(n)$.

In Section 7.1 we $\overset{\text{will?}}{\text{shall}}$ evaluate the gap between the two bounds experimentally on different types of graphs.

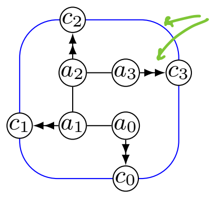
5 Bounding the Gin of the Contour

Next, we prove new bounds on the gin of the contour, which is well known to always be a hull set [8]. An important tool for the remainder of this section is the following lemma.

Lemma 6. *For any vertex $a_k \notin I^k[\mathrm{Ct}(G)]$ for $k \in \mathbb{N}$, there exist vertices $a_0, \ldots, a_{k-1}$ and $c_0, \ldots, c_k$ such that for all $i \in \{0, \ldots, k\}$ we have*

1. $a_i \notin I^i[\mathrm{Ct}(G)]$,
2. $c_i \in \mathrm{tr}(a_i) \cap \mathrm{Ct}(G)$,
3. $\mathrm{dist}(c_i, a_{i-1}) = \mathrm{ecc}(c_i)$ *if* $i > 0$, *and*
4. $\mathrm{ecc}(a_i) < \mathrm{ecc}(c_i) \leq \mathrm{ecc}(a_{i-1})$*; the second inequality holds only if* $i > 0$.

Moreover these vertices are pairwise distinct. See Fig. 4 for an illustration.



look like (special) edges of the graph

Fig. 4. A schematic illustration of the vertices and their relationships for $k = 3$ [25, Fig. 3]. The blue cycle depicts the contour. Straight lines indicate shortest paths, and lines with arrow heads represent trails.

Proof. If $k = 0$, then Property 3 is trivially true, and Property 1 holds by our assumption on a_k. Thus a_k has a neighbor with eccentricity $\mathrm{ecc}(a_k) + 1$; this is true for all vertices not in the contour. Therefore there exists a trail of a_k to a vertex in the contour; let this vertex be c_k, which fulfills properties 2 and 4.

Now assume $k > 0$. We show the properties for $i = k$; then they follow for $i < k$ by induction. We obtain c_k exactly as above, which yields properties 1 and 2 and $\mathrm{ecc}(a_k) < \mathrm{ecc}(c_k)$. Let a_{k-1} be a vertex with $\mathrm{dist}(c_k, a_{k-1}) = \mathrm{ecc}(c_k)$, implying Property 3. The first inequality of Property 4 can be shown exactly as above; the second holds because $\mathrm{ecc}(c_k) = \mathrm{dist}(c_k, a_{k-1}) \leq \mathrm{ecc}(a_{k-1})$. From $c_k \in \mathrm{tr}(a_k)$, we can infer $\mathrm{ecc}(a_k) = \mathrm{ecc}(c_k) - \mathrm{dist}(c_k, a_k)$. Together with $\mathrm{dist}(a_k, a_{k-1}) \leq \mathrm{ecc}(a_k)$, this yields $\mathrm{dist}(c_k, a_k) + \mathrm{dist}(a_k, a_{k-1}) \leq \mathrm{ecc}(c_k)$. This means that a shortest c_k-a_{k-1}-path contains a_k. If we had $a_{k-1} \in I^{k-1}[\mathrm{Ct}(G)]$, then consequently, we would also have $a_k \in I^k[\mathrm{Ct}(G)]$, a contradiction. Therefore we can apply the inductive hypothesis to a_{k-1}.

Property 4 implies that $a_0, \ldots, a_k$ are pairwise distinct, and the same holds for $c_0, \ldots, c_k$. Additionally $a_i \neq c_j$ for any i and j because $a_i \notin \mathrm{Ct}(G)$ but $c_j \in \mathrm{Ct}(G)$. □

The lemma generalizes [22, Lemma 2], where it is proven only for $k = 2$, and [25, Theorem 2], where it is shown for arbitrary $k \geq 2$ but only for $i \in \{k-2, k-1, k\}$. In the example in Fig. 1, we can apply this lemma with $k = 1$ to $w \notin I[\mathrm{Ct}(G)]$ to find $a_1 = w$, $c_1 = b$, $a_0 = x$, $c_0 = c$.

The following two lemmata show stronger bounds for the gin of the graph contour than the ones presented in [25] by leveraging new structural insights.

Lemma 7. *The gin of the contour* $\mathrm{Ct}(G)$ *is upper bounded by* $|\,\mathrm{Ct}(G)| - 1$.

Proof. Let $k := \mathrm{gin}(\mathrm{Ct}(G)) - 1$, and let a_k be a vertex that is not contained in $I^k[\mathrm{Ct}(G)]$. Lemma 6 implies that there are vertices $c_0, \dots, c_k$ in the contour with $\mathrm{ecc}(c_k) < \cdots < \mathrm{ecc}(c_0)$. Now take a vertex v' that realizes the eccentricity of c_0, i.e., $\mathrm{dist}(c_0, v') = \mathrm{ecc}(c_0)$. If v' is not in the contour, then we can follow a trail of v' to a vertex v that is contained in the contour and has greater eccentricity than c_0. If v' is already contained in the contour, let $v := v'$ instead. In either case, v is distinct from $c_0, \dots, c_k$. Thus we have $k + 2$ distinct vertices in the contour, so $|\,\mathrm{Ct}(G)| \geq k + 2 = \mathrm{gin}(\mathrm{Ct}(G)) + 1$. □

Note that this bound is tight for the example in Fig. 1, where $|\,\mathrm{Ct}(G)| - 1 = 3 - 1 = 2 = \mathrm{gin}(\mathrm{Ct}(G))$. [25] show merely that $\mathrm{gin}(\mathrm{Ct}(G)) \leq |\,\mathrm{Ct}(G)|$.

Lemma 8. *For any unweighted graph G with radius at least 2, the gin of the contour* $\mathrm{Ct}(G)$ *is strictly less than* $\mathrm{diam}(G)/2$.

Proof. If $\mathrm{rad}(G) > \mathrm{diam}(G)/2$, this follows immediately from [25] where it was shown that for any G, the gin of the contour $\mathrm{Ct}(G)$ is upper bounded by $\mathrm{diam}(G) - \mathrm{rad}(G)$. Otherwise we have $\mathrm{rad}(G) = \mathrm{diam}(G)/2$, since $2\,\mathrm{rad}(G) < \mathrm{diam}(G)$ is impossible. The rest of the proof is a generalization of the proof of [22, Theorem 3]. Suppose for contradiction that there is a vertex u that is not contained in $I^{\mathrm{rad}(G)-1}[\mathrm{Ct}(G)]$. It was proven in [25] that in any unweighted graph G and for any vertex $v \in V(G)$ with $\mathrm{ecc}(v) \geq \mathrm{diam}(G) - k$, we have $v \in I^k[\mathrm{Ct}(G)]$. With $k = \mathrm{rad}(G) - 1$, our u must have $\mathrm{ecc}(u) < \mathrm{diam}(G) - \mathrm{rad}(G) + 1$; this implies $\mathrm{ecc}(u) = \mathrm{rad}(G)$. Now take a vertex v with $\mathrm{ecc}(v) = \mathrm{diam}(G)$. There is a third vertex w with $\mathrm{dist}(v, w) = \mathrm{ecc}(v)$ and thus also $\mathrm{ecc}(w) = \mathrm{diam}(G)$. Due to their eccentricities, v and w lie in the contour. Since $\mathrm{dist}(v, u) + \mathrm{dist}(u, w) \leq 2\,\mathrm{ecc}(u) = 2\,\mathrm{rad}(G) = \mathrm{dist}(v, w)$, our u lies on a shortest v-w-path and is thus contained in $I[\mathrm{Ct}(G)]$, a contradiction. □

Note that this bound is tight for the example in Fig. 1, where $\mathrm{gin}(\mathrm{Ct}(G)) = 2 = \lfloor 5/2 \rfloor = \lfloor \mathrm{diam}(G)/2 \rfloor$, since the gin is integer. [25] show only “less than or equal” but not “strictly less than.”

6 Finding Small Hull Sets

While MinimumHullSet is—as we have shown above—tractable for input graphs with a small vertex cover number, this is a rather strong assumption for real-world data. As no approximation algorithms are known, oftentimes one simply computes the contour $\mathrm{Ct}(G)$ of the graph G as a heuristic. However, it takes time $\mathcal{O}(nm)$ to compute which is not practical for large inputs. A faster heuristic has been proposed in [25], which is based on the following observation.

Observation 3 ([25]). *Let G be a graph and $v \in V(G)$ a vertex. Let $L(v, S)$ denote the set of leaves in the* $\mathrm{BFS}(v, S)$ *tree. Then the set* $H := \{v\} \cup L(v, V(G))$ *is a hull set of G with* $\mathrm{gin}(H) = 1$.

The algorithm described in [25] is the straightforward application of Observation 3 and runs in $\mathcal{O}(m)$ time. However, upon closer inspection it is revealed that the following stronger lemma in fact implies Observation 3.

Lemma 9. *Let G be a graph, $S \subseteq V(G)$ a set of vertices, $v \in S$ a vertex, and $L'(v, S)$ the set of sinks in the* $\mathrm{BFS}(v, S)$ *DAG. Then $I[v, L'(v, S)] = S$.*

Proof. Let w be any vertex contained in S. If $w \in \{v\} \cup L'(v, S)$, it is clearly contained in the interval as well. Otherwise it is neither a source nor a sink in the $\mathrm{BFS}(v, S)$ DAG. Because it is contained in S, it must however be contained in this DAG. Now we follow a directed path in the DAG from w to some sink $\ell \in L'(v, S)$. With Observation 2, we have $w \in I[v, \ell]$ and hence $w \in I[v, L'(v, S)]$. □

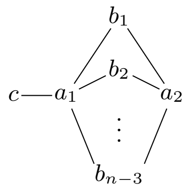
Even if we set $S = V(G)$, this is a stronger result than Observation 3, for in non-geodetic graphs we may have $L'(v, S) \subsetneq L(v, S)$. But we can also apply Lemma 9 with a different choice for v to the resulting hull set in order to obtain a smaller hull set with gin two. To define this algorithm, which we shall call *k-pruning*, in general, we start with the trivial hull set $H_0 := V(G)$ and apply the lemma to it k times for k different choices of v, resulting in a hull sets $H_1, \ldots, H_k$ with $\mathrm{gin}(H_i) \leq i$ for $i = 1, \ldots, k$. If at some point only vertices remain that have already been chosen as v, we may stop the computation early because further iterations certainly will not find any more vertices to prune. In this case we shall say the algorithm has *converged*. One can understand the case of n-pruning as exhaustive pruning until convergence. Clearly k-pruning runs in $\mathcal{O}(km)$ time.

Although k-pruning is generally much faster than the contour and experiments show that it also produces better results ([25] and Section 7.2), nevertheless there are input graphs for which even n-pruning will likely not find satisfactory hull sets.

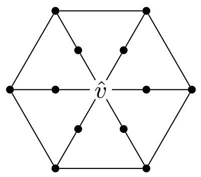
Lemma 10. *There is an infinite class $\{G_n\}_{n\geq 4}$ of graphs with n vertices such that G_n has a contour of constant size $|\mathrm{Ct}(G_n)| = 2$ but n-pruning with uniformly random choice of v will find a hull set of size $n - 2$ with high probability.*

Proof. We construct G_n as follows. Take the complete bipartite graph $K_{2,n-3}$ on the vertex sets $\{a_1, a_2\}$, $\{b_1, \ldots, b_{n-3}\}$, and add a vertex c adjacent to a_1; the resulting graph G_n is shown in Fig. 5a. We have $\mathrm{ecc}(a_2) = \mathrm{ecc}(c) = 3$, while all other eccentricities are 2; thus the contour of G_n is $\{a_2, c\}$. The first iteration of n-pruning will pick a vertex $v \in \{b_1, \ldots, b_{n-3}\}$ with high probability $(n - 3)/n$. In this case it will compute the hull set $H_1 = V(G_n) \setminus \{a_1, a_2\}$. Now we have $\mathrm{dist}(v, w) = 2$ for each pair of distinct vertices v and w contained in H_1. This means that n-pruning cannot prune any more vertices and $H_n = H_1$ with cardinality $n - 2$. □

Moreover both k-pruning (for any constant k) and the contour may produce arbitrarily bad results with respect to the optimal solution, and n-pruning may converge only after a linear number of iterations.



(a) Graph G_n with contour size 2, for which n-pruning will yield a bad result with high probability.



(b) The wheel graph W_6^2 with circumference 6 and radius 2.

Fig. 5. Two instances for which k-pruning performs badly.

Definition 1 (Wheel Graph). *For $c \geq 3$ and $r \geq 1$ let W_c^r denote the wheel graph with circumference c and radius r, which we construct as follows. We take a cycle C with c vertices (the* rim*), add a vertex $\hat{v}$ (the* axle*), and connect $\hat{v}$ to all vertices of C with paths of length r (the* spokes*). Figure 5b shows an example.*

Lemma 11. *For each positive integer k, there is an infinite class $\{G_{kr}\}_{r\geq 1}$ of graphs with $\Theta(kr^2)$ vertices and hull number $\Theta(k)$ but contour size $\Theta(kr)$ and k-pruning finds a hull set of size $\Omega(r)$.*

Proof. Choose $G_{kr} := W^r_{4(k+1)r}$, the wheel graph with circumference $4(k+1)r$ and radius r. Let $R := \{v_0, \dots, v_{4(k+1)r-1}\}$ be the set of vertices of degree 3 (the rim) in cyclic order, and let P_i be the set of inner vertices of the (unique) shortest $\hat{v}$-v_i-path (the ith spoke). For ease of notation, we shall treat indices modulo the circumference $4(k+1)r$. One can easily verify that $\operatorname{ecc}(v) = r + \operatorname{dist}(\hat{v}, v)$ for all vertices v. Because only vertices on the rim are not adjacent to a vertex that is further from $\hat{v}$, it follows that the contour of G_{kr} is the rim, which contains $4(k+1)r = \Theta(kr)$ vertices.

In order to bound the hull number of G_{kr} from above, consider the vertex set $H := \{\hat{v}, v_0, v_{2r}, \dots, v_{(2k+1)2r}\}$. For any v_i in H, it is clear that $I[v_i, v_{i+2r}]$ contains all vertices on the rim between v_i and v_{i+2r}, and hence $I[H]$ contains R. Since each vertex w on a spoke lies on the shortest path between the axle and a vertex on the rim, which are both contained in $I[H]$, w is certainly contained in $I^2[H]$. Because $I^2[H]$ contains the axle, the rim, and the spokes, it is equal to the vertex set $V(G)$, and we conclude that H is a hull set of size $2k+3$, that is, $\operatorname{hn}(G_{kr}) \leq 2k+3$.

To bound the hull number from below, we apply Lemma 5 to

$$W := \{\hat{v}, v_0, v_1, \dots, v_{4r-1}\} \cup P_0 \cup P_1 \cup \dots \cup P_{4r-1}$$

with $\operatorname{cut}(W) = \{\hat{v}, v_0, v_{4r-1}\}$ to find that $\operatorname{hull}(\operatorname{cut}(W)) = \operatorname{cut}(W) \cup P_0 \cup P_{4r-1}$, and thus any hull set of G_{kr} must contain some vertex from $W \setminus (\{\hat{v}, v_0, v_{4r-1}\} \cup P_0 \cup P_{4r-1}) = \{v_1, v_2, \dots, v_{4r-2}\}$. Analogously, we can convince ourselves that any hull set of G_{kr} must contain some vertex from $\{v_{4r+1}, \dots, v_{8r-2}\} \cup P_{4r+1} \cup \dots \cup P_{8r-2}$ and so on, for a total of $k+1$ sets that only overlap in cut vertices. It follows that $\operatorname{hn}(G_{kr}) \geq k+1$.

Now we turn to the result of k-pruning on G_{kr}. When we prune from a vertex v, observe that a vertex $u \neq v$ will be pruned only if it has a neighbor w that is further from v, that is, $\operatorname{dist}(v,u)+1 = \operatorname{dist}(v,w)$. This can only happen if $\operatorname{dist}(v,u) < \operatorname{ecc}(v)$. We claim that for any vertex $v \in V(G)$, there are at most $4r$ rim vertices distinct from v that fulfill this criterion. This means that the hull set H_k we get after pruning k times will contain at least $4(k+1)r - k \cdot 4r = 4r$ rim vertices.

To see that the claim is true, w.l.o.g., take a vertex $v \in \{\hat{v}, v_0\} \cup P_0$, and let $d := \operatorname{dist}(\hat{v}, v)$. Now take any rim vertex v_i, and let $a = \min\{i, 4(k+1)r - i\}$ be the length of the shortest v_0-v_i-path that uses only rim vertices. We show that there are at most $4r$ possible choices for v_i such that $\operatorname{dist}(v, v_i) < \operatorname{ecc}(v)$. It is clear that the shortest v-v_i-path either goes from v along P_0 to v_0 and from there along the rim to v_i, or it goes from v along P_0 to $\hat{v}$ and from there along P_i to v_i. By the first path, we have $\operatorname{dist}(v, v_i) \leq (r-d)+a$; by the second, we have $\operatorname{dist}(v, v_i) \leq d+r = \operatorname{ecc}(u)$. Only if $a \leq 2d$, can the first path be a shortest path. There are at most $4d+1$ choices for v_i for which this can be true. If $d < r$, then there are exactly $4d+1$ choices for v_i for which this is true, and $4d+1 \leq 4r$. If $d = r$, then v lies on the rim, and there are $4r+1$ choices for v_i for which this is true, but one of them is v, which cannot be pruned. This proves the claim and thus also the lemma. □

Lemma 12. *There is an infinite class $\{G_n\}_{n \text{ odd}}$ of graphs with n vertices for which n-pruning may prune a constant number of vertices in each iteration and the total number of iterations until convergence is in $\Omega(n)$.*

Proof. For odd n choose $G_n := W^1_{n-1}$, the wheel graph with circumference $n-1$ and radius 1. Let the rim vertices of G_n be $v_1, \ldots, v_{n-1}$ in cyclic order. We start with $H_0 = V(G_n)$. Assume that we prune from vertices $v_2, v_4, \ldots, v_{n-1}$ in that order to obtain $H_1, \ldots, H_{(n-1)/2}$. One can verify that $H_1 = H_0 \setminus \{\hat{v}, v_1, v_3\}$. Then, for each $i \geq 2$, we have $H_i = H_{i-1} \setminus \{v_{2i+1}\}$. The resulting hull set $\{v_2, v_4, \ldots, v_{n-1}\}$ is minimal, so the pruning has converged. Thus we get one pruning step in which we prune two vertices and $(n-3)/2$ pruning steps in each of which we prune one vertex. □

Now that we have seen the shortcomings of k-pruning—it may produce worse results than the contour, it may produce unsatisfactory results for small k, and it may converge slowly for large k—we give an example where k-pruning outperforms the contour in every metric.

Lemma 13. *There is an infinite class $\{G_n\}_{n \text{ even}}$ of graphs ~~with n vertices~~ such that G_n has* (i) *a hull set ~~H~~ of ~~constant~~ size 2 that is found by 1-pruning*, ~~*but*~~ and (ii) *a contour of size n* $= |V(G)|$.

Proof. For even n, choose $G_n := C_n$, the simple cycle with n vertices. Let $V(G_n) = \{v_1, \ldots, v_n\}$ in the order in which they appear on the cycle. Then $H := \{v_{n/2}, v_n\}$ is a hull set of G_n. If we run 1-pruning from v_n, it finds H; if we run 1-pruning from a different vertex, it finds the same hull set up to renaming the vertices. Since all vertices have eccentricity $n/2$, the contour is $\operatorname{Ct}(G_n) = V(G_n)$. □

7 Experimental Evaluation

We now evaluate our newly proposed lower bounds on the hull number based on Lemma 5 (Section 7.1) and the heuristic hull sets from Section 6 (Section 7.2). To this end we have selected a diverse set of eighty-one real-world and generated graphs from the second DIMACS challenge[1] on GraphColoring and the PACE 2021 challenge[2] on ClusterEditing. They can be categorized into six collaboration networks from arXiv.org and DBLP (collab), nine communication networks between humans or machines (comm), five product relation graphs from the online store Amazon (prod), thirty random graphs (rand), fourteen conflict graphs from register allocation for program compilation (reg), three sensor networks and complements thereof (sens), eleven social networks (soc), and finally a Latin square graph and two scheduling problems (other).

Maybe use a different font style (\sf?)

All algorithms were implemented in C++20 using no libraries apart from the C++ STL. They were compiled with GCC 14.1.1 and executed on a single core of an AMD Ryzen 7 3700X CPU clocked at 3.6 GHz with 128 G/B of memory.

7.1 Evaluating Lower Bounds on the Hull Number

We computed both bounds on our test instances with a time budget of one day per instance. ~~Even~~ | if not the entire graph was explored after one day, we stopped the computation then and retrieved the lower bound computed thus far. If it was 0 or 1, we set it to the trivial lower bound of 2; this also enables us to determine ratios of the bounds, which are plotted in Fig. 6. We found that ~~that~~ our new bound generally gives better results than the old one, by one or even two orders of magnitude in many cases. Only in few cases is the old bound better; all these are very large instances for which the new bound or both bounds could not explore the entire graph. We suspect that, given more time, our new bound would also find a larger value.

7.2 Evaluating Heuristic Hull Sets

In order to compare k-pruning and the contour experimentally, we run k-pruning for $k \in \{1, 2, 3, 4, n\}$ at least sixteen times[3] and record the minimum, average, and maximum size of the resulting hull set for each value of k. Figure 7 shows the results. Additionally we also compute the contour and its gin. For $k \leq 4$, the computation takes minutes at most, even for large instances. For each of the instances we set a time limit of one week, which the contour computation and n-pruning ran into for the largest instances. For all other instances, we found that the gin of the contour is at most 1.

First we investigate how much we gain from increasing k. We see a sharp decrease in size between small successive values of k, but already between three

[1] http://archive.dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/
[2] https://github.com/PACE-challenge/Cluster-Editing-PACE-2021-instances
[3] Exactly sixteen times for $k \geq 2$, and once for every possible start vertex for $k = 1$. For $k > 1$ trying all possible combinations of k start vertices quickly becomes infeasible.

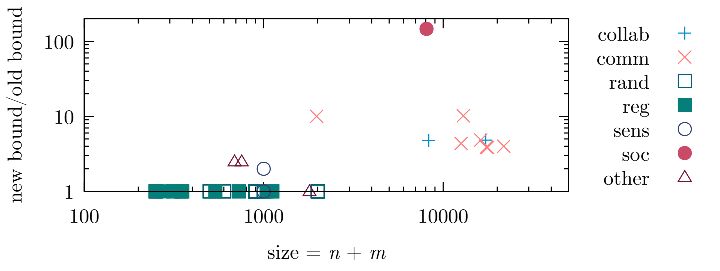


Fig. 6. The ratio of the new and old lower bound for each instance. A value greater than 1 means the new bound is stronger than the old one. Only the instances where the computation terminated within a day are shown.

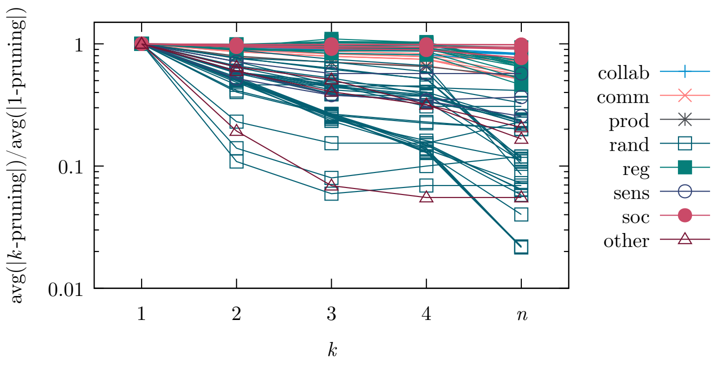


Fig. 7. Average quality of k-pruning in comparison with 1-pruning for $k \in \{1, 2, 3, 4, n\}$.

and four, the difference is not as great. The improvement we get from pruning is the greatest in (the) dense random graphs, while we get only a small improvement in sparse graphs such as the social networks; here there are few opportunities for pruning, and there are even many degree-1 vertices, which must be contained in any valid hull set. n-pruning does not always yield a significantly better result than 4-pruning. We conclude that in real-world applications picking a constant k, running k-pruning a constant number of times, and taking the best result seems to be the most sensible method.

Next we test how this approach compares to the contour. We find that for those instances where the contour includes a large fraction of the vertices in the graph, 4-pruning is consistently at least as good as the contour. But for the instances where the contour only consists of a very small number of vertices already, 4-pruning is not able to reliably find a hull set with a size in the same order of magnitude. These problematic instances are the register allocation problems, whose contour has size at most 10, and which have inspired Lemma 10.

If we want to give both algorithms—the contour and k-pruning—the same time budget and see which one performs better, we can run 1-pruning for each possible choice of the start vertex and take the optimum result. Then both algorithms need exactly n BFS runs. Our experiments show that in almost all cases does 1-pruning find an equivalent or better solution than the contour. It finds much better results especially on the instances with highly uniform degrees. This is not surprising insofar as the bad example from Lemma 10 has highly nonuniform degrees.

Finally we compare the sizes of our heuristic hull sets to the corresponding lower bounds we have computed for Section 7.1. To this end we take the ratio between the size of the smallest hull set any of our heuristics could find and the larger of our two lower bounds. See Fig. 8. We can see that whether the lower bound and the heuristic get close to each other or are far apart clearly depends on the graph type and size.

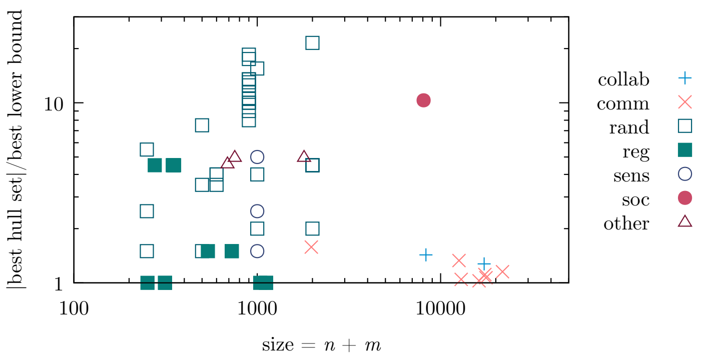


Fig. 8. How close do our heuristics approach our lower bounds? Only the instances where the computation terminated within a day are shown.

8 Discussion and Outlook

We have investigated several new approaches to hull sets, both in theory and experimentally, improving on those from [25], but also showing their limits. In addition we have done some theoretical work on the gin of the contour, which may aid in finding or ruling out a graph G with $\text{gin}(\text{Ct}(G)) > 3$. With our new bounds, we know that if $\text{gin}(\text{Ct}(G)) = 4$, then G must have diameter at least 9 and its contour must contain at least five vertices. The following questions are left to future research. Does a graph G with $\text{gin}(\text{Ct}(G)) > 3$ exist? More generally, is the gin of the contour bounded or unbounded? Can we establish a lower bound on MinimumHullSet parameterized by vc, analogously to [16], and show that our FPT-algorithm has asymptotically optimal running time? Can we find other parameters that allow for good parameterized running times? Can we prove or disprove the existence of an approximation for MinimumHullSet?

References

1. Albenque, M., Knauer, K.: Convexity in partial cubes: The hull number. Discrete Mathematics **339**(2), 866–876 (2016)
2. Araujo, J., Campos, V., Giroire, F., Nisse, N., Sampaio, L., Soares, R.: On the hull number of some graph classes. Theoretical Computer Science **475**, 1–12 (2013)
3. Araujo, J., Morel, G., Sampaio, L., Soares, R., Weber, V.: Hull number: P_5-free graphs and reduction rules. Discrete Applied Mathematics **210**, 171–175 (2016)
4. Artigas, D., Dantas, S., Dourado, M.C., Szwarcfiter, J.L., Yamaguchi, S.: On the contour of graphs. Discrete Applied Mathematics **161**(10), 1356–1362 (2013)
5. Bessy, S., Dourado, M.C., Penso, L.D., Rautenbach, D.: The geodetic hull number is hard for chordal graphs. SIAM Journal on Discrete Mathematics **32**(1), 543–547 (2018)
6. Bressan, M., Cesa-Bianchi, N., Lattanzi, S., Paudice, A.: Exact recovery of clusters in finite metric spaces using oracle queries. In: Belkin, M., Kpotufe, S. (eds.) Proceedings of 34th Conference on Learning Theory. Proceedings of Machine Learning Research, vol. 134, pp. 775–803. PMLR (2021)
7. Cáceres, J., Hernando, C., Mora, M., Pelayo, I.M., Puertas, M.L., Seara, C.: Geodeticity of the contour of chordal graphs. Discrete Applied Mathematics **156**(7), 1132–1142 (2008)
8. Cáceres, J., Márquez, A., Oellermann, O.R., Puertas, M.L.: Rebuilding convex sets in graphs. Discrete Mathematics **297**(1), 26–37 (2005)
9. Cáceres, J., Puertas, M.L., Hernando, C., Mora, M., Pelayo, I.M., Seara, C.: Searching for geodetic boundary vertex sets. Electronic Notes on Discrete Mathematics **19**, 25–31 (2005)
10. Chakraborty, D., Gahlawat, H., Roy, B.: Algorithms and complexity for geodetic sets on partial grids. Theoretical Computer Science **979**, 114217 (2023)
11. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: Further observations and further improvements. Journal of Algorithms **41**(2), 280–301 (2001)
12. Dourado, M.C., Gimbel, J.G., Kratochvíl, J., Protti, F., Szwarcfiter, J.L.: On the computation of the hull number of a graph. Discrete Mathematics **309**(18), 5668–5674 (2009)
13. Dourado, M.C., Penso, L.D., Rautenbach, D.: On the geodetic hull number of P_k-free graphs. Theoretical Computer Science **640**, 52–60 (2016)
14. Dourado, M.C., Protti, F., Rautenbach, D., Szwarcfiter, J.L.: Some remarks on the geodetic number of a graph. Discrete Mathematics **310**(4), 832–837 (2010)
15. Everett, M.G., Seidman, S.B.: The hull number of a graph. Discrete Mathematics **57**(3), 217–223 (1985)
16. Foucaud, F., Galby, E., Khazaliya, L., Li, S., Mc Inerney, F., Sharma, R., Tale, P.: Metric dimension and geodetic set parameterized by vertex cover (2024) $\leftarrow$ Ref. incomplete
17. Funke, S., Klein, C.: Hole detection or: "how much geometry hides in connectivity?". In: Proceedings of the 22nd Annual Symposium on Computational Geometry. pp. 377–385. SCG '06, Association for Computing Machinery, New York (2006)
18. Gallo, G.: An $\mathcal{O}(n \log n)$ algorithm for the convex bipartite matching problem. Operations Research Letters **3**(1), 31–34 (1984)
19. Harary, F., Loukakis, E., Tsouros, C.: The geodetic number of a graph. Mathematical and Computer Modelling **17**(11), 89–95 (1993)
20. Harary, F., Nieminen, J.: Convexity in graphs. Journal of Differential Geometry **16**(2), 185–190 (1981)
21. Lay, S.R.: Convex Sets and Their Applications. Wiley, New York (1982)

22. Mezzini, M.: On the geodetic iteration number of the contour of a graph. Discrete Applied Mathematics **206**, 211–214 (2016)
23. Mezzini, M., Moscarini, M.: On the geodeticity of the contour of a graph. Discrete Applied Mathematics **181**, 209–220 (2015)
24. Seiffarth, F., Horváth, T., Wrobel, S.: A fast heuristic for computing geodesic closures in large networks. In: Pascal, P., Ienco, D. (eds.) Discovery Science. pp. 476–490. Springer Nature Switzerland, Cham (2022)
25. Storandt, S.: Bounds and algorithms for geodetic hulls. In: Balachandran, N., Inkulu, R. (eds.) Algorithms and Discrete Applied Mathematics. pp. 181–194. Springer International Publishing, Cham (2022)

LNCS?
volume?

↑
CALDAM?

A Proofs

A.2 Basics

Proof (Lemma 1). First we show that the Euclidean iteration number of the set S of vertices of a d-simplex is $\lfloor \log_2 d \rfloor + 1$. On one hand consider the point at barycentric coordinates $\left(\frac{1}{d+1}, \ldots, \frac{1}{d+1}\right)$, which has $d+1$ nonzero coordinates. After i iterations only 2^i coordinates can be nonzero because in each iteration we take convex combinations of only pairs of points from the previous iteration and we begin with points in S with only one nonzero coordinate. It follows that this particular point is not yet reached after $\lceil \log_2(d+1) \rceil - 1 = \lfloor \log_2 d \rfloor$ iterations. On the other hand consider an arbitrary point p with at most 2^i nonzero coordinates. If $i = 0$, then $p \in S$ is reached after 0 iterations. Otherwise we can represent p as a convex combination of two points each of which has at most 2^{i-1} nonzero coordinates. By induction p is reached after at most i iterations. Our claim about the Euclidean iteration number of S follows, whence we know the bound is tight if it is correct. Now consider any set S of points and the convex polytope that is its convex hull. We can decompose this polytope into a union of simplices such that each point p in the convex hull lies in a simplex C whose vertices lie in S. By the above deliberations we find that p is reached after at most $\lfloor \log_2 d \rfloor + 1$ iterations if we start with the vertices of C. But since S is a superset, the upper bound is still correct. □

A.6 Finding Minimum Hull Sets

Proof (Lemma 3). The kernelization algorithm exhaustively applies the following reduction rules in a sequential manner to the instance (G, k).

Reduction Rule 1. If G contains three simplicial vertices that are (false or true) twins, delete one of them from G and decrease k by one.

Reduction Rule 2. If G contains four vertices that are false twins, delete one of them from G.

To see that Rule 1 is correct, assume that G contains three simplicial vertices u, v, and w that are twins. We show that G has a hull set of size k iff the reduced graph G' obtained by deleting u from G has a hull set of size $k-1$. For the forward direction let H be a hull set of G with $|H| = k$. Because they are simplicial vertices, H contains each of u, v, and w. Now let $H' := H \setminus \{u\}$. This set of size $k-1$ is a hull set of G' because any vertex of G' that lies on a shortest path in G between u and some vertex z also lies on a shortest path between v and z. Conversely if G' has a hull set H'' of size $k-1$, then clearly $H'' \cup \{u\}$ is a hull set of G with size k.

For Rule 2 assume that G contains four false twins that are not simplicial (since we have already applied Rule 1 exhaustively) as the set $T = \{t_1, \ldots, t_4\}$, and let G' be the reduced graph obtained by deleting t_1 from G. We show that G has a hull set of size k iff G' has a hull set of size k. For the forward direction let H be a minimum hull set of G with size k. By Lemma 2 H contains at most two vertices of T; w.l.o.g. t_1 and t_2 do not belong to H. Since the removal of t_1 from G does not change any distances, H is still a hull set of G'. Conversely let H' be a minimum hull set of G' with size k. Again, by Lemma 2, we may assume that one vertex among t_2, …, t_4 is not in H', w.l.o.g. $t_2 \notin H'$. Note that all vertices of G' are contained in $\text{hull}(H)$. In particular t_2 lies on a shortest path between two vertices x, $y \in \text{hull}(H)$. But then t_1 also lies on a shortest x-y-path. Hence H' is also a hull set of G.

Finally consider a reduced instance (G, k) to which neither of the reduction rules can be applied. If $k < 0$, we return a trivial no-instance. Otherwise let X be a minimum vertex cover of G with $|X| = \texttt{vc}$, and let $I \coloneqq V(G) \setminus X$ be an independent set of G. Notice that any set of false twins in I contains at most three vertices and corresponds to a unique subset of X. Hence G has at most $|X| + 3 \cdot 2^{|X|} = \texttt{vc} + 3 \cdot 2^{\texttt{vc}} = \mathcal{O}(2^{\texttt{vc}})$ vertices. □

Proof (Lemma 4). The algorithm starts by computing a minimum vertex cover X of G in time $\mathcal{O}(2^{\texttt{vc}} + n^2)$ using an FPT algorithm, for example the one in [11].

In polynomial time we find the set S of simplicial vertices of G. By [15, Theorem 2] any hull set of G contains S. Now note that $X \cup S$ is a hull set of G because any non-simplicial vertex $v \notin X$ has two non-adjacent neighbors x, $y \in X$ (else v would be simplicial or X not a vertex cover), and thus v lies on a shortest x-y-path of length two.

Hence, to enumerate all possible minimum hull sets, it suffices to enumerate each subset $S' \subseteq V(G) \setminus S$ with $|S'| \leq |X|$ and check whether $S \cup S'$ is a hull set. If one such set is indeed a hull set and has size at most k, we return yes; otherwise we return no.

Since $V(G) \setminus S$ has $\mathcal{O}(n^k)$ subsets of size k, we enumerate $\mathcal{O}(n^{\texttt{vc}})$ subsets in total. For each we compute the hull in time $\mathcal{O}(nm) = \mathcal{O}(n^3)$. □