Combinatorics of Beacon-based Routing in Three Dimensions[⋆,⋆⋆]

Jonas Cleve, Wolfgang Mulzer

Institut für Informatik, Freie Universität Berlin, Berlin, Germany

Abstract

A beacon $b \in \mathbb{R}^d$ is a point-shaped object in d-dimensional space that can exert a magnetic pull on any other point-shaped object $p \in \mathbb{R}^d$. This object p then moves greedily towards b. The motion stops when p gets stuck at an obstacle or when p reaches b. By placing beacons inside a d-dimensional polyhedron P, we can implement a scheme to route point-shaped objects between any two locations in P. We can also place beacons to guard P, which means that any point-shaped object in P can reach at least one activated beacon.

The notion of beacon-based routing and guarding was introduced in 2011 by Biro et al. [FWCG'11]. The two-dimensional setting is discussed in great detail in Biro's 2013 PhD thesis [SUNY-SB'13].

Here, we consider combinatorial aspects of beacon routing in three dimensions. We show that $\lfloor (m+1)/3 \rfloor$ beacons are always sufficient and sometimes necessary to route between any two points in a given polyhedron P, where m is the smallest size of a tetrahedral decomposition of P. This is one of the first results to show that beacon routing is also possible in higher dimensions.

Keywords: beacon routing, three dimensions, polytopes

1. Introduction

Visibility in the presence of obstacles is a classic notion in combinatorial and computational geometry [11]. Given a simple polygon P in the plane, two points p and q in P can *see each other* if and only if the line segment between p and q lies in P (considered as a closed region). The *visibility region* of a point $p \in P$ consists of all points $q \in P$ such that p and q can see each other. These basic

⋆Supported in part by DFG grant MU 3501/1 and ERC StG 757609.

⋆⋆A preliminary version appeared as J. Cleve and W. Mulzer. *Combinatorics of Beacon-based Routing in Three Dimensions*. Proc. 13th LATIN, pp. 346–360.

Email addresses: jonascleve@inf.fu-berlin.de (Jonas Cleve), mulzer@inf.fu-berlin.de (Wolfgang Mulzer)

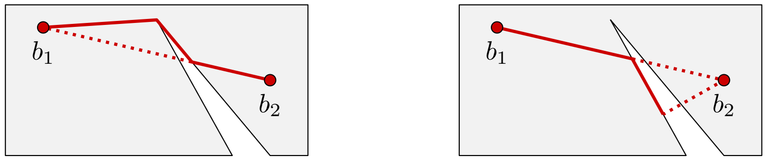


Figure 1: Attraction is not symmetric. In this two-dimensional example b_1 attracts b_2 (left) but b_2 does not attract b_1 (right).

definitions and their variants have spawned an active subarea of computational geometry, with whole textbooks devoted to it [11, 16].

In 2011, Biro et al. [6] introduced the concept of *beacon-based* visibility, where the objects take a more active role. A *beacon* $b \in \mathbb{R}^d$ is a point-shaped object in d-dimensional space. The beacon b can be *enabled* or *disabled*. Once b is enabled, it exerts a *magnetic pull* on any other point-shaped object p in $\mathbb{R}^d$. Then, the object p moves in the direction that most rapidly decreases the distance between b and p. In the simplest case, this motion proceeds along the line segment pb. If p encounters an obstacle that blocks the direct path along pb, then p slides along the boundary of the obstacle in the direction that most rapidly decreases the distance to b. If this is not possible, the motion ends, and we say that p gets *stuck*. If p does not get stuck, then it reaches b, and we say that p is *attracted* by b. See Fig. 1 for examples. The *attraction region* of b consists of all points that are attracted by b. This is an extension of classic visibility: the visibility region of b is a subset of the attraction region of b. However, unlike classic visibility, beacon attraction is not symmetric. Thus, it makes also sense to consider the *inverse* attraction region of a point p, i.e., the set of all beacon positions b such that b attracts p. Two examples of these regions can be found in Fig. 2.

The PhD thesis of Biro [5] constitutes the first in-depth study of beacon-based visibility. In particular, it considers beacon-based routing and guarding in (two-dimensional) polygonal domains. The idea of beacon-based routing is as follows: suppose we have a polygonal domain P that contains a set B of beacons, and suppose we want to route a point-shaped object p towards a target t. We assume that t can also act as a beacon, even if it is not contained in B. The routing proceeds by successive activation of beacons in $B \cup \{t\}$: a first beacon $b_1 \in B$ is enabled to attract p until it reaches b_1. Subsequently, b_1 is disabled,

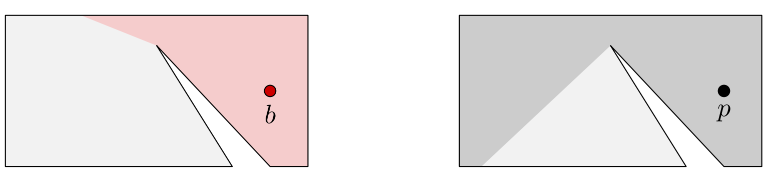


Figure 2: The *attraction region* of a beacon b (left) and the *inverse attraction region* of a point p (right).

and a second beacon $b_2 \in B$ is switched on, again attracting p until it reaches b_2. This is repeated until the last (implicit) beacon at t is enabled and finally attracts p to its location. The challenge is to devise a strategy for placing the beacons in P and for choosing a sequence of beacon activations such that it becomes possible to route between any two locations s and t in P. The size of B should be minimized. Note that we require that every activated beacon must attract p until it reaches the beacon's location. Only then are we allowed to enable the next beacon. Thus, if p gets stuck, the process ends and the routing is considered to be unsuccessful.

In beacon-based guarding (or coverage), the goal is to choose a minimum-size set B of beacons such that the union of the attraction regions for B covers the whole polygonal domain P. In this case, we say that B *covers* P. This is analogous to the classic art-gallery problem [16], using beacon-based visibility instead of straight-line visibility.

1.1. Related Work

Two dimensions. As mentioned above, a large part of the pioneering work on beacon-based routing and guarding was done by Biro and his co-authors [6–8]. An extensive collection of results can be found in Biro's PhD thesis [5].

Biro and his co-authors showed that $\lfloor n/2 \rfloor - 1$ beacons always suffice and sometimes are necessary for routing in a simple polygon with n vertices [7, Theorem 1]. We will discuss this result in more detail in Section 2. More generally, to route in a polygon with n vertices and h holes, $\lfloor n/2 \rfloor - h - 1$ beacons are sometimes necessary and $\lfloor n/2 \rfloor + h - 1$ beacons are always sufficient [7, Theorem 2]. For *orthogonal* polygons[1], they showed only a loose lower bound of $\lfloor n/4 \rfloor - 1$ beacons, leaving a larger gap for the routing problem [7, Theorem 3].

For beacon-based guarding of a simple polygon and of a polygon with h holes, they showed that $\lfloor 4n/13 \rfloor$ beacons are sometimes necessary, while $\lfloor (n+h)/3 \rfloor$ beacons are always sufficient [7, Theorem 5]. In particular, the upper bound for simple polygons is $\lfloor n/3 \rfloor$. For orthogonal polygons, they obtained an upper bound of $\lfloor n/4 \rfloor$ and a lower bound of $\lfloor (n+4)/8 \rfloor$ [7, Section 6].

Bae et al. [3] improved some of these bounds by showing that $\lfloor n/6 \rfloor$ beacons are sometimes needed and always sufficient for beacon-based guarding in orthogonal polygons. They also proved that if the polygon is monotone and orthogonal, the bound reduces to $\lfloor (n+4)/8 \rfloor$. The gap for routing in simple orthogonal polygons was finally closed by Shermer [18] who showed that $\lfloor (n-4)/3 \rfloor$ beacons are always sufficient and sometimes necessary.

Aldana-Galván et al. [1] extended the notion of coverage to both the interior and the exterior of a given polygon. They proved that $\lfloor n/4 \rfloor + 1$ vertex beacons always suffice to simultaneously cover the interior and exterior of an orthogonal polygon with n vertices (possibly with holes) [1, Theorem 1]. Table 1 gives an overview of the currently best results for routing and guarding in two dimensions.

[1] A planar polygon is *orthogonal* if all its edges are parallel to the x- or the y-axis.

		Bound		
Problem	Polygon type	Lower	Upper	Reference
Routing	Simple	$\lfloor n/2 \rfloor - 1$		[7, Thm 1]
	With holes	$\lfloor n/2 \rfloor - h - 1$ (*)	$\lfloor n/2 \rfloor + h - 1$	[7, Thm 2]
	Orthogonal	$\lfloor (n-4)/3 \rfloor$		[18]
Guarding	Simple	$\lfloor 4n/13 \rfloor$	$\lfloor n/3 \rfloor$ (*)	[7, Thm 5]
	With holes	$\lfloor 4n/13 \rfloor$	$\lfloor (n+h)/3 \rfloor$ (*)	[7, Thm 5]
	Orthogonal	$\lfloor n/6 \rfloor$		[3]

Table 1: The currently best results in two dimensions. The bounds marked (*) were conjectured to be tight by Biro [5, Conjectures 6.3.3, 7.3.7, and 7.3.9]

So far, we have only discussed results that give combinatorial bounds on the number of beacons needed to guard or to route in certain classes of polygons. Naturally, the notions of beacon-based routing and guarding also lead to interesting algorithmic questions. As is to be expected, several optimization problems associated with beacons are hard: Biro [5, Theorems 6.2.2, 6.2.3, and 6.2.4] showed that the All-Pair, All-Sink, and All-Source variants of the optimal beacon routing problem are NP-hard. In these problems, we are given a simple polygon P, and we need to find a minimum set B of beacons such that we can route between any pair of points in P; from a given location $s \in P$ to all other points in P; or from all points in P to a given location $t \in P$, respectively. Biro also showed that given a simple polygon P, it is NP-hard to find a minimum set of beacons that covers P [5, Theorem 7.2.1].

On the positive side, Biro et al. [8, Theorem 6] presented an algorithm to compute the attraction region of a given beacon in a polygon P with n vertices and h holes in $O(n + h \log^{1+\varepsilon} h)$ time and $O(n)$ space, for any fixed $\varepsilon > 0$. They also described how to find the *inverse* attraction region of a point in a polygon P with n vertices in $O(n^2)$ time [8, Theorem 8]. More generally, the inverse attraction region of a polygonal region R in P with m vertices can be computed in $O(n^2 m^2)$ time [8, Theorem 8]. As for routing, Biro et al. show how to find a *minimum-hop-beacon path* between two points s and t in a polygon with n vertices and h holes from a given set of m beacons in $O(m(n + h \log^{1+\varepsilon} h + m \log h))$ time [8, Theorem 11]. They also provide a $O(n^3)$-time 2-approximation algorithm for the case that the beacons can be placed arbitrarily inside the polygon. As the authors point out, this approximation algorithm can also be applied repeatedly to obtain a PTAS. More recently, Kostitsyna et al. [13] gave an optimal algorithm to compute the inverse beacon attraction region of a point in a simple polygon in $O(n \log n)$ time. Further algorithmic results can be found in Kouhestani's PhD thesis [14].

Three dimensions. This work is based on the Master's thesis of the first author [10] who presented the first combinatorial bounds for beacon-based routing in three dimensions. In his thesis, Cleve also showed that Biro's NP-hardness and APX-hardness results for optimum beacon routing extend to three dimensions, by a simple lifting argument [10, Section 4.3]. Finally, he constructed a three-dimensional polyhedron that cannot be guarded by placing a beacon at every vertex [10, Lemma 6.1]. Independently, and almost at the same time, Aldana-Galván et al. [2, Section 2] obtained a stronger result: there exists an *orthogonal* polyhedron that cannot be covered by beacons at every vertex. Furthermore, Aldana-Galván et al. [2, Theorem 1] showed that every *orthotree*[2] with n vertices can be covered by $\lfloor n/8 \rfloor$ beacons. They described a family of orthotrees where this number of beacons is needed. They also proved a tight bound of $\lfloor n/12 \rfloor$ becons for *well-separated* orthotrees.[3] Shortly afterwards, Aldana-Galván et al. [1] introduced the notion of *edge beacons*. Here, every point of an edge e may exert a magnet pull on a point-shaped object p, and p always moves towards the point on e closest to it. Aldana-Galván et al. prove that $\lfloor m/12 \rfloor$ edge beacons are always sufficient and sometimes $\lfloor m/21 \rfloor$ edge beacons are necessary to cover an orthogonal polyhedron with m edges [1, Theorems 3 and 4]. If both the interior and the exterior of an orthogonal polyhedron should be covered simultaneously, $\lfloor m/6 \rfloor$ is a tight bound for the number of edge beacons required [1, Theorem 5].

2. Preliminaries

We begin by reviewing the proof that $\lfloor n/2 \rfloor - 1$ beacons are needed for routing in a simple polygon with n vertices [7, Theorem 1]. This serves two purposes: on the one hand, the argument serves as a starting point for our three-dimensional bound; on the other hand, it provides an opportunity to correct a slight gap in the published proof by Biro et al. [7].[4]

2.1. Two-dimensional Upper Bound

The following theorem states the main result for beacon-based routing in two dimensions.

Theorem 1 (Biro et al. [7, Theorem 1]). *Let P be a simple polygon with n vertices. Then, $\lfloor n/2 \rfloor - 1$ beacons are sometimes necessary and always sufficient to route between any two points in P.*

The strategy of Biro et al. [7] is as follows: they triangulate P to obtain a partition into $n - 2$ triangles. Then, they place the beacons in P with an inductive strategy. In each step, one beacon b is placed, and at least two triangles are removed. They claim that there is always a way to position b on the boundary of the remaining polygon such that the whole interior of the removed triangles

[2] An orthotree is an orthogonal polyhedron made out of boxes that are glued face to face and whose dual graph is a tree.

[3] An orthotree is well-separated if its dual graph has the property that all neighbors of a vertex with degree strictly greater than 2 have degree at most 2.

[4] This issue and a possible fix have also been discovered by Tom Shermer, a fact personally communicated to us by Irina Kostitsyna [12], but as far as we know, no updated version of the proof has been published to date.

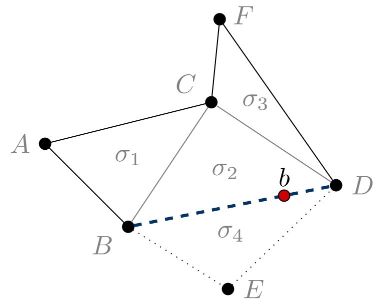


Figure 3: The situation analyzed by Biro et al. [7]. Here, b can be placed near D so that b can see every point inside $ABDFC$. The edges AB, AC, CF, and DF are boundary edges and BD is a diagonal.

can be seen from b. The inductive procedure ends as soon as no more triangles are left. Biro et al. conclude that $\lfloor n/2 \rfloor - 1$ beacons suffice for routing.

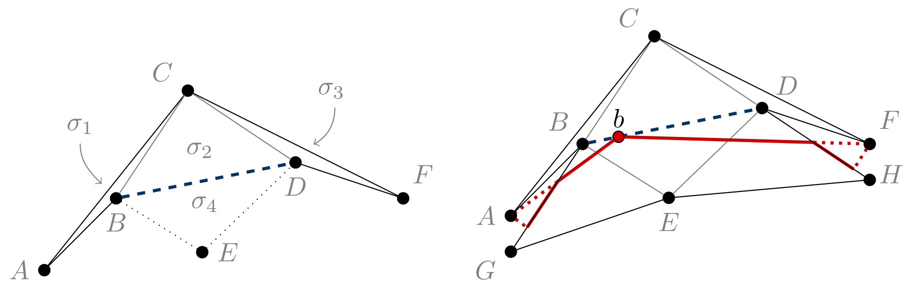
The technical heart of the argument lies in an analysis of different triangle configurations. The goal is to show that by placing a single beacon, at least two triangles can be removed. One configuration is as follows:[5] we have a central triangle $\sigma_2 = \triangle BCD$ with two adjacent triangles $\sigma_1 = \triangle ABC$ and $\sigma_3 = \triangle CDF$. Biro et al. [7] would like to argue that one can position a beacon b on the free edge BD of σ_2 such that the whole polygon $ABDFC$ is completely visible to b; see Fig. 3. More precisely, their reasoning goes like this:

> The location b along BD is chosen so the pentagon $ABDFC$ is visible to b. This is always possible, by placing b on the correct side of lines CF and AC. Then, any point in triangles $\triangle ABC$, $\triangle BCD$, $\triangle CDF$ can be routed to or from b as b is *visible* to each point in those triangles. — [7, p. 2]

However, the condition that b lies to the right of AC and to the left of FC is not sufficient for the whole pentagon $ABDFC$ to be visible from b. For this, b must also be to the left of AB and to the right of FD, i.e., in the visibility cone of both σ_1 and σ_3. Figure 4a shows a situation where this cannot be done: the line through B and D limits the visibility of any beacon b in the relative interior of the line segment BD. Moreover, if we place b at B or at D, then b still cannot see the full pentagon.

Nonetheless, visibility is not actually required; mutual attraction would be enough for the argument to go through. In fact, we can always place b so that it attracts all points inside the pentagon $ABDFC$. Unfortunately, the inverse does not hold. Consider Fig. 4b: unless b is placed at B, a point-shaped object at b that is attracted by A will get stuck on the line segment BG; and analogously

[5]We follow the notation of the original work [7].



(a) Here, b cannot be placed on BD to see the full pentagon.

(b) No matter where b lies on BD, it cannot be attracted by both A and F.

Figure 4: It is not always possible to place one beacon b on the line segment BD such that it attracts and is attracted by all points inside the pentagon $ABDFC$.

for D and F. Since b cannot be placed simultaneously at both B and D, the requirement that b is attracted by both A and F cannot be fulfilled.

Nevertheless, Theorem 1 still holds, as we will show in the following lemma. For completeness, we present the proof in full detail, and we indicate where we depart from the original argument of Biro et al. [7, Theorem 1].

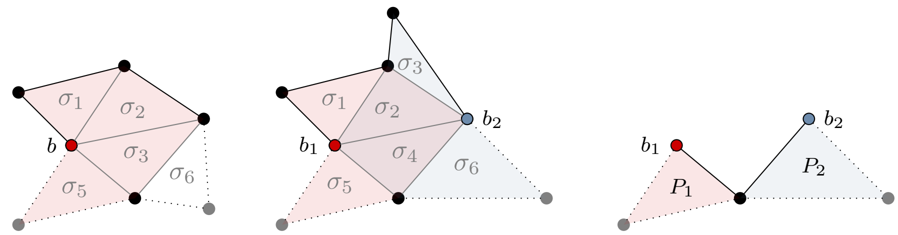
Lemma 2 (Two-dimensional upper bound)**.** *Let P be a simple polygon with $n \geq 2$ vertices. Then, $\lfloor n/2 \rfloor - 1$ beacons are always sufficient to route between any two points in P.*

Proof. The proof proceeds by induction on n. For the base case, we assume that $2 \leq n \leq 4$. If $n \in \{2, 3\}$, then P is either a line segment or a single triangle. In both cases, P is convex and no beacon is needed. For $n = 4$, we let d be a diagonal of P.[6] We place one beacon at an arbitrary point b on d. Then, every point $p \in P$ can see b, which means that p and b mutually attract. Thus, we can route from every $s \in P$ to every $t \in P$ via b.

Now suppose that $n > 4$ and assume that Lemma 2 holds for all simple polygons with at most $n - 1$ vertices. We triangulate P and consider the dual graph T of the triangulation: the triangles constitute the nodes, and two nodes are adjacent if and only if the corresponding triangles share an edge in the triangulation. As P is simple, T is a tree with $n - 2$ nodes and maximum degree 3. We take an arbitrary leaf of T, and we declare it the root. Let σ_1 be a triangle that corresponds to a deepest leaf in T. Let σ_2 be the parent triangle of σ_1. There are two cases:

Case 1: the triangle σ_1 is the only child of σ_2. Let σ_3 be the parent triangle of σ_2. Then, the triangles σ_1, σ_2, and σ_3 share a common vertex v, and we place a beacon b at v; see Fig. 5a. Next, we remove from P the parts of σ_1 and σ_2

[6] A diagonal is a line segment whose endpoints are vertices of P and whose relative interior lies in the interior of P.



(a) The beacon b covers at least three triangles: σ_1, σ_2, σ_3.

(b) The two beacons b_1 and b_2 cover σ_1, σ_2, σ_3, and σ_4 and both neighbors of σ_4.

(c) After removing σ_1 to σ_4 two (possibly empty) polygons P_1 and P_2 remain.

Figure 5: The two possible configurations in the inductive step are shown in (a) and (b). (c) shows the situation of (b) after removing the triangles.

that do not belong to another triangle of P. This gives a simple polygon P_1 with $n_1 = n - 2$ vertices. By the inductive hypothesis, there is a set B_1 of at most

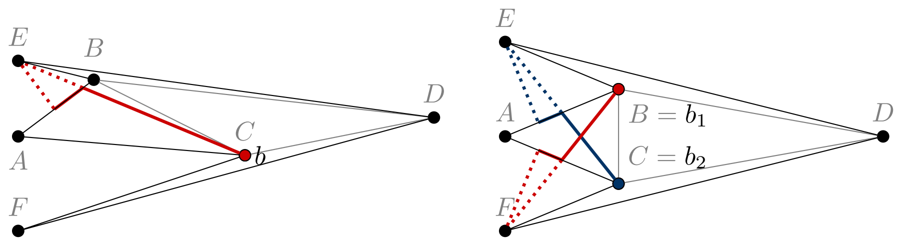
$$\left\lfloor \frac{n_1}{2} \right\rfloor - 1 = \left\lfloor \frac{n-2}{2} \right\rfloor - 1 = \left\lfloor \frac{n}{2} \right\rfloor - 2$$

beacons that allows us to route between any two points in P_1. We set $B = B_1 \cup \{b\}$. Then, we have $|B| \leq \lfloor n/2 \rfloor - 1$.

It remains to show that we can use B to route between any two points in P. By the inductive hypothesis and because b lies in σ_3 which remains in P_1, we can route between b and any point in P_1. Furthermore, due to convexity of triangles, every point $p \in \sigma_1 \cup \sigma_2$ can see b, and thus p can attract b and can be attracted by it. Hence, we can route between any pair of points in P using B.

Case 2: the triangle σ_2 has a second child σ_3. This is the erroneous case in Biro et al. [7, Theorem 1]. Let σ_4 be the parent triangle of σ_2. Since σ_1 is a deepest leaf in T, if follows that σ_3 is also a leaf; see Fig. 5b. Instead of placing a single beacon and removing three triangles, as suggested by Biro et al. [7, Theorem 1], we place two beacons b_1, b_2 and remove four triangles. The beacon b_1 is placed at the common vertex of σ_1, σ_2, and σ_4 (marked red), and b_2 is placed at the common vertex of σ_3, σ_2, and σ_4 (marked blue). If σ_4 has more neighbors, they are also covered by $\{b_1, b_2\}$, see Fig. 5b.

We remove from P the set $(\sigma_1 \cup \sigma_2 \cup \sigma_3) \setminus \{b_1, b_2\}$ and the interior of σ_4. This gives two polygons P_1 and P_2 with one common vertex, see Fig. 5c. Possibly, P_1 or P_2 (or both) degenerates to a line segment from b_1 or b_2 to the common vertex. Let $n_1 \geq 2$ be the number of vertices of P_1, and $n_2 \geq 2$ the number of vertices of P_2. We have $n_1 + n_2 = n - 2$, since we removed three vertices, and since P_1 and P_2 share one vertex to be counted twice. As $n_1 \leq n - 1$ and $n_2 \leq n - 1$, we can apply the inductive hypothesis to P_1 and P_2. This gives two sets $B_1 \subset P_1$ and $B_2 \subset P_2$ of beacons with $|B_1| \leq \lfloor n_1/2 \rfloor - 1$ and $|B_2| \leq \lfloor n_2/2 \rfloor - 1$. We set



(a) $\angle FCB > 3\pi/2$ and b is placed at C. However, despite $\angle EBC \leq 3\pi/2$, a beacon at E cannot attract an object at b.

(b) $\angle FCB \leq 3\pi/2$ and $\angle EBC \leq 3\pi/2$. The beacon is to be placed arbitrarily at B or C. However, for both positions it cannot be attracted by either F or E.

Figure 6: Two counterexamples for the alternative proof of Biro et al. [6].

$B = B_1 \cup B_2 \cup \{b_1, b_2\}$. Then,

$$\begin{aligned}|B| = |B_1| + |B_2| + 2 &\leq \left\lfloor \frac{n_1}{2} \right\rfloor - 1 + \left\lfloor \frac{n_2}{2} \right\rfloor - 1 + 2 \\ &= \left\lfloor \frac{n_1}{2} \right\rfloor + \left\lfloor \frac{n_2}{2} \right\rfloor \leq \left\lfloor \frac{n_1 + n_2}{2} \right\rfloor = \left\lfloor \frac{n-2}{2} \right\rfloor = \left\lfloor \frac{n}{2} \right\rfloor - 1.\end{aligned}$$

It remains to show that we can route between any two points in P. By the inductive hypothesis, and since b_1 lies on the boundary of P_1 and b_2 on the boundary of P_2, we can route between b_1 and any point in P_1, and between b_2 and any point in P_2. Moreover, since b_1 and b_2 both lie in σ_2, they can see and thus attract each other. Also, since every removed triangle σ_1, σ_2, σ_3, and σ_4 contains either b_1 or b_2, every point in $\bigcup_{i=1}^{4} \sigma_i$ can attract and be attracted by b_1 or b_2. It follows that for every point $p \in P$, we can route between p and b_1 or between p and b_2. Since we also can route between b_1 and b_2, it follows that we can route between any two points in P. $\square$

Remark. The *extended abstract* for the original paper by Biro et al. from 2011 [6], available on Irina Kostitsyna's ResearchGate profile, contains an alternative proof for Theorem 1. This version handles Case 2 slightly differently. However, we believe that it is susceptible to the same issues as the more recent version of the proof [7]. More precisely, in the alternative proof, the authors use the same notation as in Fig. 3. They say that if $\angle FCB > 3\pi/2$, the beacon b should be placed at C. From this, it follows that $\angle CBE \leq 3\pi/2$. The authors claim that then, "all points inside $\triangle BDE$ can reach b and vice versa". However, Fig. 6a shows a case where E cannot attract b. A similar counterexample applies for the symmetric case where $\angle EBC > 3\pi/2$ and b is placed at B. If both $\angle FCB \leq 3\pi/2$ and $\angle EBC \leq 3\pi/2$, then b is to be placed "arbitrarily at either B or C", but Fig. 6b shows a configuration where both positions cannot be attracted by all points inside the four triangles.

2.2. Tetrahedral Decompositions

To generalize the proof strategy from Theorem 1 to $\mathbb{R}^3$, we need a three-dimensional analogue of polygon triangulation: the decomposition of a bounded polyhedron into tetrahedra. This creates several difficulties that are not present in the two-dimensional case. In 1911, Lennes [15] showed that there are polyhedra that cannot be decomposed into tetrahedra without additional *Steiner points*. In fact, it is NP-complete to decide whether a tetrahedral decomposition without Steiner points exists [17]. The *size* of a tetrahedral decomposition is the number of tetrahedra contained in it. Unlike in two dimensions, the size of a tetrahedral decomposition may significantly exceed the number of vertices in the polyhedron. Chazelle [9] showed that for any n, there exists a polyhedron with $\Theta(n)$ vertices for which any decomposition into convex parts needs at least $\Omega(n^2)$ pieces. On the other hand, Bern and Eppstein [4, Theorem 13] described how to decompose any polyhedron into $O(n^2)$ tetrahedra using $O(n^2)$ Steiner points. Furthermore, a tetrahedral decomposition clearly must have size at least $n - 3$. A single polyhedron may have different tetrahedral decompositions of varying sizes. For example, the triangular bipyramid can be decomposed into two or three tetrahedra [17, p. 228]. Thus, our bounds will be in terms of the minimum size of a decomposition rather than the number of vertices. Steiner points are allowed.

To extend the ideas for two dimensions to $\mathbb{R}^3$, we must understand the dual graph of a tetrahedral decomposition. This graph is defined as follows:

Definition 3. Given a tetrahedral decomposition $\Sigma = \{\sigma_1, \dots, \sigma_m\}$ of a three-dimensional polyhedron, the *dual graph* $D(\Sigma)$ of Σ is the undirected graph with vertex set $\{\sigma_1, \dots, \sigma_m\}$ in which there is an edge between two distinct tetrahedra σ_i and σ_j if and only if σ_i and σ_j share a triangular facet.

Similarly to the two-dimensional case, the dual graph $D(\Sigma)$ of a tetrahedral decomposition has maximum degree 4. However, unlike in two dimensions, $D(\Sigma)$ is not necessarily a tree. The following lemma provides a tool for placing beacons in connected subgraphs of $D(\Sigma)$.

Lemma 4. *Let Σ be a tetrahedral decomposition of a three-dimensional polyhedron, and let $D(\Sigma)$ be the dual graph of Σ. Consider a set $S \subseteq \Sigma$ of tetrahedra such that the induced subgraph $D(S)$ of $D(\Sigma)$ is connected. Then,*

(i) if $|S| = 2$, the tetrahedra in S share a triangular facet;

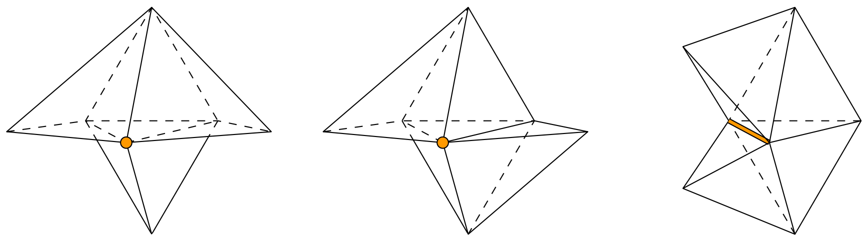
(ii) if $|S| = 3$, the tetrahedra in S share one edge; and

(iii) if $|S| = 4$, the tetrahedra in S share at least one vertex.

Proof. We consider the three cases separately.

Case (i): this follows directly from Definition 3.

Case (ii): since $D(S)$ is connected and since $|S| = 3$, there is a tetrahedron $\sigma \in S$ adjacent to the other two. By Definition 3, this means that σ shares a



(a) One tetrahedron in the center has all other tetrahedra as neighbors.

(b) Two tetrahedra with one and two tetrahedra with two neighbors.

(c) All four tetrahedra share one edge.

Figure 7: The three possible configuration for a polyhedron with a decomposition into four tetrahedra. The shared vertex or edge is marked.

facet with each of the other two tetrahedra. Since σ is a tetrahedron, any two facets in σ share an edge. The claim follows.

Case (iii): see Fig. 7. Let $S' \subset S$ be three tetrahedra in S so that $D(S')$ is connected. By (ii), the tetrahedra in S' share an edge e. By Definition 3, the remaining tetrahedron in $S \setminus S'$ shares a facet f with a tetrahedron $\sigma \in S'$. Since e contains two vertices of σ while f contains three vertices, e and f must share at least one vertex. The claim follows. $\square$

3. An Upper Bound for Beacon-based Routing

We now give an upper bound on the number of beacons needed to route within a polyhedron, extending the strategy of Biro et al. [7], as described in Section 2, to three dimensions. We want to show the following:

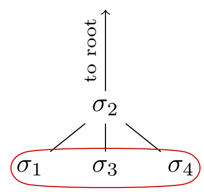
Theorem 5. *Let P be a three-dimensional polyhedron, and let Σ be a tetrahedral decomposition of P of size m. There is a set of at most $\lfloor (m+1)/3 \rfloor$ beacons that allows us to route between any pair of points in P.*

The rest of this section is dedicated to the inductive proof of Theorem 5. The following lemma constitutes the base case of the induction.

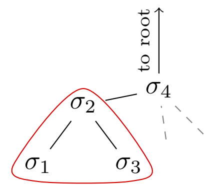
Lemma 6 (Base case)**.** *Let P be a three-dimensional polyhedron, and let Σ be a tetrahedral decomposition of P of size $m \leq 4$. There is a set of at most $\lfloor (m+1)/3 \rfloor$ beacons that allows us to route between any pair of points in P.*

Proof. If $m = 1$, then P is a convex tetrahedron, and no beacon is needed. If $m \in \{2, 3, 4\}$, we apply Lemma 4 to obtain a vertex v that is common to all tetrahedra in Σ. We place one beacon b at v. By convexity, every point in P can attract and be attracted by b, and the claim follows. $\square$

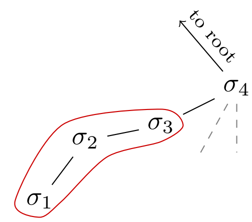
We proceed to the inductive step. For this, we consider a tetrahedral decomposition Σ of size $m > 4$. Our goal is to place k beacons, for some $k \geq 1$,



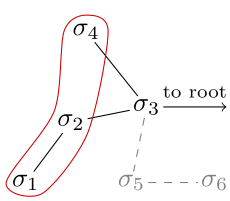
(a) Remove σ_1, σ_3, and σ_4 by placing a beacon where all four tetrahedra meet.



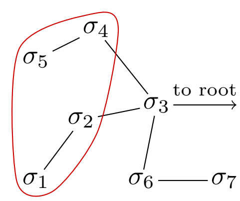
(b) Remove σ_1, σ_2, and σ_3 by placing a beacon where all four tetrahedra meet.



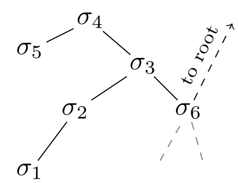
(c) Remove σ_1, σ_2, and σ_3 by placing a beacon where all four tetrahedra meet.



(d) Remove σ_1, σ_2, and σ_4 by placing a beacon where all four tetrahedra meet.



(e) Remove σ_1, σ_2, σ_4, and σ_5 by placing a beacon where σ_1 to σ_5 meet.



(f) The number and configuration of σ_6's children must be looked at.

Figure 8: The possible configurations in the first part of the inductive step.

such that the beacons lie in at least $3k + 1$ tetrahedra and therefore can attract and can be attracted by all points in those tetrahedra. Then, we remove at least $3k$ tetrahedra, leaving a polyhedron with a tetrahedral decomposition of size strictly less than m. We apply induction, and then show how to route between the smaller polyhedron and the removed tetrahedra.

To do this, we look at the dual graph $D(\Sigma)$ of Σ, as in Definition 3. Let T be a spanning tree of $D(\Sigma)$, rooted at an arbitrary leaf. We do not distinguish between nodes of T and the corresponding tetrahedra. Let σ_1 be a deepest leaf of T. If there are multiple such leaves, we choose σ_1 such that its parent σ_2 has the largest number of children, breaking ties arbitrarily. Fig. 8 shows the different cases how T can look like around σ_1 and σ_2. First, we focus on Figs. 8a to 8e. In all five cases, T must have at least one additional root node—either because $m \geq 5$ or because T is rooted at a leaf. The situation in Fig. 8f will be dealt with in Lemma 9.

Lemma 7 (Inductive step I)**.** *Let P be a three-dimensional polyhedron, and Σ a tetrahedral decomposition of P of size $m \geq 5$. Let T be a spanning tree of the dual graph $D(\Sigma)$, rooted at a leaf of T. Let σ_1 be a deepest leaf of T with the maximum number of siblings, and σ_2 its parent. Assume that one of the following conditions holds:*

(i) σ_2 has exactly three children σ_1, σ_3, and σ_4 (see Fig. 8a);

(ii) σ_2 has exactly two children σ_1 and σ_3, and a parent σ_4 (Fig. 8b);

(iii) σ_2 has exactly one child σ_1 and is the only child of its parent σ_3, whose parent is σ_4 (Fig. 8c);

(iv) σ_2 has exactly one child σ_1 and its parent σ_3 has two or three children at least one of which, say σ_4, is a leaf (Fig. 8d); or

(v) σ_2 has exactly one child σ_1 and its parent σ_3 has three children, each of which has a single leaf child (Fig. 8e).

Then, we can place a beacon b at a vertex of σ_1 such that b lies in at least four tetrahedra. After that, we can remove at least three of these tetrahedra so that T stays a tree and at least one remaining tetrahedron in T contains b.

Proof. We consider the cases individually.

Cases (i–iv): in each case, the induced subgraph on $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ is connected. Thus, Lemma 4(iii) implies that the four tetrahedra share a vertex v. We place b at v. After that, we remove either σ_1, σ_3, and σ_4 (case (i)); σ_1, σ_2, and σ_3 (cases (ii) and (iii)); or σ_1, σ_2, and σ_4 (case (iv)). In each case, we remove either only leaves or inner nodes with all their children. This means that the tree structure of T is preserved. Moreover, we only remove three of the four tetrahedra that contain b, so one of them remains in T.

Case (v): as shown in Fig. 8e, we have three connected sets, each containing σ_3, a child σ_i of σ_3, and σ_i's child: $\{\sigma_1, \sigma_2, \sigma_3\}$, $\{\sigma_5, \sigma_4, \sigma_3\}$, and $\{\sigma_7, \sigma_6, \sigma_3\}$. By Lemma 4(ii), each set has a common edge. These three edges all occur in σ_3, and since σ_3 is a tetrahedron, at least two of them share an endpoint v. Without loss of generality, let these be the common edges of $\{\sigma_1, \sigma_2, \sigma_3\}$ and of $\{\sigma_5, \sigma_4, \sigma_3\}$. We place b at v, and we remove σ_1, σ_2, σ_4, and σ_5. The beacon b is also contained in σ_3, which remains in T. □

The final configuration is shown in Fig. 8f. The following lemma provides an analysis of how the tetrahedra can intersect in this case.

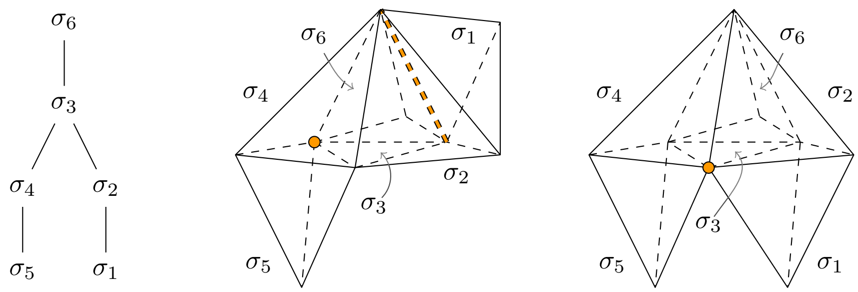
Lemma 8. *Let Σ be a tetrahedral decomposition of size 6, and suppose that $D(\Sigma)$ has a spanning tree as in Fig. 9a. Then at least one of the following holds:*

(i) σ_1, σ_2, σ_3, σ_4, and σ_5 have a common vertex; or

(ii) σ_3, σ_4, σ_5, and σ_6 share a common vertex v; σ_1, σ_2, σ_3, and σ_6 share a common edge e; and $v \cap e = \emptyset$. A symmetric situation is also possible.

Proof. Let $S_1 = \{\sigma_3, \sigma_4, \sigma_5, \sigma_6\}$ and $S_2 = \{\sigma_1, \sigma_2, \sigma_3, \sigma_6\}$. By Lemma 4, each set shares at least a vertex, but it may also share an edge. There are three cases:

Case 1: both S_1 and S_2 share an edge. These edges must belong to the triangular facet that connects σ_3 and σ_6. Thus, they share a common vertex, and (i) holds.



(a) The dual graph of the tetrahedral decomposition.

(b) The four tetrahedra on the left share a common vertex while the four tetrahedra on the right share a common edge.

(c) All tetrahedra but the rearmost tetrahedron σ_6 share one common vertex, marked in orange.

Figure 9: A tetrahedron σ_6 with a subtree of five tetrahedra. Figures (b) and (c) depict configurations that satisfy cases (ii) and (i) of Lemma 8, respectively.

Case 2: exactly one of S_1, S_2 shares an edge e, while the other shares only a vertex v. If $v \cap e = v$, then (i) applies, and if $v \cap e = \emptyset$, then (ii) holds—see Fig. 9b for an example.

Case 3: both S_1 and S_2 share only a vertex; see Fig. 9c. Let v be the vertex of σ_3 that is not in the facet shared by σ_3 and σ_6. In Fig. 9c, v is marked orange. Since σ_2 is adjacent to σ_3, it follows that σ_2 contains v and three of its four facets contain v. One of these facets is the shared facet with σ_3, and we claim that σ_1 is placed at one of the other two. Indeed, σ_1 cannot be located at the fourth facet of σ_2, since otherwise it would share an edge with σ_2, σ_3 and σ_6, which is ruled out by the current case. Thus, $v \in \sigma_1$, and a symmetric argument shows that $v \in \sigma_5$. It follows that (i) holds. $\square$

Now, we can proceed with the inductive step for the configuration from Fig. 8f. The problem is that to remove $\{\sigma_1, \dots, \sigma_5\}$, we need two beacons. However, this does not meet our goal of handling at least $3k$ tetrahedra by placing k beacons, for a $k \geq 1$. If we removed σ_6 and if σ_6 had additional children, the remaining dual graph might no longer be connected, and we could not continue with our induction. Thus, we must look at the (additional) subtrees of σ_6.

Since there are many possibilities, we wrote a short Python program to generate all the cases. Our program enumerates all rooted, ordered, ternary trees of height at most three. To each such tree, the program repeatedly applies Lemma 7 to prune subtrees. If this results in an empty tree, the case does not need to be considered. If not, we save the remaining tree for manual consideration, eliminating isomorphic copies of the same tree. The source code is in Appendix A. The program leaves us with nine different cases, shown in Fig. 10. In each case, the subtree from Fig. 8f is present. The following lemma explains how to place the beacons.

Lemma 9 (Inductive step II)**.** *Let P be a three-dimensional polyhedron, with a*

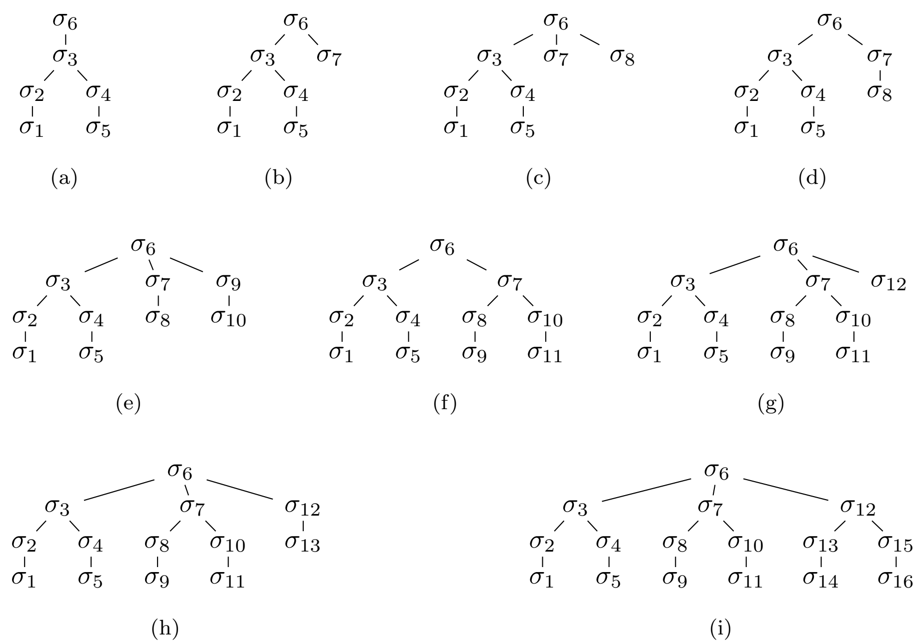


Figure 10: The "nontrivial" configurations of children of σ_6. The tree in (a) is a subtree of all configurations. In all cases, σ_6 has no other children than those shown here. Furthermore, since T is rooted at a leaf node, σ_6 needs to have an additional parent (except in case (a)).

tetrahedral decomposition Σ of size $m \geq 5$. Let T be a spanning tree of the dual graph $D(\Sigma)$, rooted at an arbitrary leaf. Let $T' \subseteq T$ be a subtree of T with height 3 for which Lemma 7 cannot be applied; see Fig. 10.

Then, there is a set B of $k \geq 2$ beacons that are vertices in at least $3k+1$ tetrahedra from T', such that the induced subgraph for B on Σ is connected. Furthermore, we can remove at least $3k$ tetrahedra, each containing a beacon from B, so that T remains connected and so that at least one remaining tetrahedron contains a beacon from B

Proof. We say that two beacons b_1 and b_2 *share an edge* or are *neighbors* if a a tetrahedron of Σ contains an edge between the vertices where b_1 and b_2 are placed. We go through the cases.

Fig. 10a: by Lemma 4(iii) the sets $\{\sigma_1, \sigma_2, \sigma_3, \sigma_6\}$ and $\{\sigma_6, \sigma_3, \sigma_4, \sigma_5\}$ each share one vertex, say v_1 and v_2, respectively. If $v_1 \neq v_2$, we set $B = \{v_1, v_2\}$. If $v_1 = v_2$, we set $B = \{v_1, w\}$, where w is any of the three other vertices of σ_6. If σ_6 has a parent tetrahedron, the shared facet contains three vertices of σ_6 and hence at least one beacon from B. Thus, by placing $k = 2$ beacons, we can remove the $6 = 3k$ tetrahedra $\{\sigma_1, \ldots, \sigma_6\}$.

Fig. 10b: we have the same situation as in Fig. 10a, except for the additional tetrahedron σ_7. We choose B as in Fig. 10a, and we observe that σ_6 contains two beacons. Thus, σ_7 contains at least one beacon from B. Hence, by placing $k = 2$ beacons, we can remove the $7 > 3k$ tetrahedra $\{\sigma_1, \ldots, \sigma_7\}$.

Fig. 10c: we apply the same argument as for Fig. 10b, observing that σ_8 must also contain a beacon from B. Thus, by placing $k = 2$ beacons, we can remove the $8 > 3k$ tetrahedra σ_1 to σ_8.

Fig. 10d: by Lemma 4(ii), the set $\{\sigma_6, \sigma_7, \sigma_8\}$ shares an edge e. We apply Lemma 8 to $\{\sigma_1, \ldots, \sigma_6\}$. This gives two cases. Case (i): $\{\sigma_1, \ldots, \sigma_5\}$ share a vertex v. As v is in σ_3, and as σ_3 shares a facet with σ_6, three neighboring vertices of v are in σ_6. The edge e contains at least one of those three neighbors. We call it w. We set $B = \{v, w\}$. Case (ii): we obtain a vertex v and an edge e' in σ_6, with $v \cap e' = \emptyset$. This covers three vertices of σ_6, so the edge e shares at least one vertex with v or with e'. To obtain B, we choose two vertices of σ_6 such that v, e', and e each contain at least one. In both cases, the beacons in B are neighbors. We place $k = 2$ beacons, and we remove the $7 > 3k$ tetrahedra $\{\sigma_1, \ldots, \sigma_5, \sigma_7, \sigma_8\}$.

Fig. 10e: by Lemma 4(ii), the sets $\{\sigma_6, \sigma_7, \sigma_8\}$ and $\{\sigma_6, \sigma_9, \sigma_{10}\}$ share edges e_1 and e_2, respectively. We apply Lemma 8 to $\{\sigma_1, \ldots, \sigma_6\}$. This again gives two cases. Case (i): $\{\sigma_1, \ldots, \sigma_5\}$ share a vertex v. We set $B = \{v, w_1, w_2\}$ such that w_1 and w_2 are vertices of σ_6, $|B| = 3$, and both edges e_1 and e_2 contain at least one beacon. As in Fig. 10d, v is a neighbor of w_1 or w_2. Furthermore, w_1 and w_2 are neighbors because they are vertices of σ_6. Case (ii): we obtain a vertex v and an edge e in σ_6, with $v \cap e = \emptyset$. We set $B = \{v, w_1, w_2\}$, where w_1 and w_2 are vertices of σ_6, such that $|B| = 3$ and such that all edges e, e_1, and e_2 contain at least one beacon. Since all beacons lie in σ_6, they are mutual neighbors. In both cases, we place $k = 3$ beacons such that every tetrahedron contains at least one. We remove the $9 = 3k$ tetrahedra $\{\sigma_1, \ldots, \sigma_{10}\} \setminus \{\sigma_6\}$.

Fig. 10f: we apply Lemma 8 to $\{\sigma_1, \ldots, \sigma_6\}$ and to $\{\sigma_6, \ldots, \sigma_{11}\}$. There are several cases. Case (i): each of $\{\sigma_1, \ldots, \sigma_5\}$ and $\{\sigma_7, \ldots, \sigma_{11}\}$ share a vertex, say v_1 and v_2, respectively. By the argument from Fig. 10d, three neighboring vertices of v_1 and three neighboring vertices of v_2 are vertices of σ_6. Thus, there is a vertex v of σ_6 that is a neighbor of v_1 and of v_2. We set $B = \{v, v_1, v_2\}$. Case (ii): without loss of generality, the set $\{\sigma_1, \ldots, \sigma_5\}$ shares a vertex v_1 and the set $\{\sigma_6, \ldots, \sigma_{11}\}$ has a vertex v_2 and an edge e in σ_6, with $v_2 \cap e = \emptyset$. Then, at least one of the three vertices of σ_6 that are neighbors of v_1 is covered by $v_2 \cup e$. We set $B = \{v_1, v_2, w\}$, where w is an endpoint of e. Case (iii): $\{\sigma_1, \ldots, \sigma_6\}$ have a vertex v_1 and an edge e_1 in σ_6 and $\{\sigma_6, \ldots, \sigma_{11}\}$ have a vertex v_2 and an edge e_2 in σ_6. We choose for B three vertices of σ_6 such that v_1, v_2, e_1, and e_2 each contain at least one beacon. In all cases, we place $k = 3$ beacons, so that B is connected and every tetrahedron in $\{\sigma_1, \ldots, \sigma_{11}\}$ contains at least one beacon. We remove $10 > 3k$ tetrahedra: all but σ_6.

Fig. 10g: this is similar to Fig. 10f. We only describe how to ensure that B contains a vertex of σ_{12}. In Case (i), v can be placed at two vertices. We choose the vertex that lies in σ_{12}. This is always possible, as σ_{12} contains three of the four vertices of σ_6. In Case (ii), we choose w as an endpoint of e that lies in σ_{12}. The same argument as before applies. In Case (iii), B must contain a vertex of σ_{12}, since three beacons are at vertices of σ_6. Thus, by placing $k = 3$ beacons, we remove $11 > 3k$ tetrahedra: all but σ_6.

Fig. 10h: initially, we choose a set of beacons B' as in Fig. 10f, at first ignoring σ_{12} and σ_{13}. By Lemma 4(ii), $\{\sigma_6, \sigma_{12}, \sigma_{13}\}$ shares an edge e'. If e' is covered by B', we set $B = B'$. If not, we set $B = B \cup \{w\}$, where w is an endpoint of e'. Thus, by placing $k \leq 4$ beacons, we may remove $12 \geq 3k$ tetrahedra: all but σ_6.

Fig. 10i: let $S_1 = \{\sigma_1, \ldots, \sigma_6\}$, $S_2 = \{\sigma_6, \ldots, \sigma_{11}\}$, $S_3 = \{\sigma_6, \sigma_{12}, \ldots, \sigma_{16}\}$. Also, let $S_1' = S_1 \setminus \{\sigma_6\}$, $S_2' = S_2 \setminus \{\sigma_6\}$, and $S_3' = S_3 \setminus \{\sigma_6\}$. We apply Lemma 8 to S_1, to S_2, and S_3. There are several cases. Case (i): S_1', S_2', and S_3' each share a vertex, say v_1, v_2, and v_3. By the argument of Fig. 10d, each of v_1, v_2, v_3 has three neighbors that are vertices of σ_6. Thus, they have one common neighbor vertex w in σ_6. We set $B = \{v_1, v_2, v_3, w\}$. Case (ii): without loss of generality, S_1' and S_2' each share a common vertex, say v_1 and v_2, and for S_3 we obtain a vertex v_3 and an edge e_3 in σ_6, with $e_3 \cap v_3 = \emptyset$. We set $B = \{v_1, v_2, v_3, w\}$, where w is an endpoint of e. Since v_3 and w are in σ_6, it follows that v_1 and v_2 have a neighboring beacon in σ_6. Case (iii): without loss of generality, S_1' has a common vertex v_1 and S_2 and S_3 each have a vertex v_2 and v_3 as well as an edge e_2 and e_3, all four in σ_6. We place a beacon at v_1 and three beacons at vertices of σ_6 such that v_2, v_3, and both edges e_1 and e_2 contain at least one beacon. Since three neighbors of v_1 are in σ_6, the beacon at v_1 has at least one beacon neighbor in σ_6. Case (iv): S_1, S_2, and S_3 each have a vertex and an edge in σ_6. We place three beacons so that all of them are covered. In all cases, we place $k \leq 4$ beacons to remove $15 > 3k$ tetrahedra: all but σ_6. □

We are now ready to prove Theorem 5:

Proof (of Theorem 5). We use induction on the size of the tetrahedral decomposition. The base case is in Lemma 6. Next, we assume that the inductive hypothesis (Theorem 5) holds for all polyhedra that have a tetrahedral decomposition of size less than m. Consider a spanning tree T of the dual graph $D(\Sigma)$ of the tetrahedral decomposition Σ, rooted at an arbitrary leaf. Let σ_1 be a deepest leaf. If σ_1 is not unique, choose one with the largest number of siblings, breaking ties arbitrarily. We can then apply either Lemma 7 or Lemma 9, to obtain the following:

(i) we have placed a set B of $k \geq 1$ beacons at vertices of Σ, and we have removed at least $3k$ tetrahedra;

(ii) every removed tetrahedron contains at least one beacon in B;

(iii) the induced subgraph on B on the vertices and edges of Σ is connected;

(iv) there is a beacon $b \in B$ in the remaining polyhedron P'.

By (i), the new polyhedron P' has a tetrahedral decomposition of size $m' \leq m - 3k < m$. Thus, by the inductive hypothesis, we need

$$k' = \left\lfloor \frac{m'+1}{3} \right\rfloor \leq \left\lfloor \frac{m-3k+1}{3} \right\rfloor = \left\lfloor \frac{m+1}{3} \right\rfloor - k$$

beacons to route between any pair of points in P'. Since $k' + k = \lfloor (m+1)/3 \rfloor$, we do not exceed the claimed amount of beacons. By the inductive hypothesis and (iv), it follows in particular that we can route from any point in P' to the beacon $b \in B$ and vice versa. From (ii), we know that for every point p in the removed tetrahedra, there is a beacon $b' \in B$ such that p attracts b' and b' attracts p. Finally, due to (iii), we can route between all beacons in B. In conclusion, we can route between any pair of points in P. This completes the inductive step. $\square$

Observation 10. Theorem 5 also implies that $\max\{1, \lfloor (m+1)/3 \rfloor\}$ beacons are sufficient to guard a polyhedron with a tetrahedral decomposition of size m. We need at least one beacon to cover the polyhedron, and placing them as in the previous proof is enough.

4. A Lower Bound for Beacon-based Routing

Our next goal is to obtain a lower bound for the number of beacons needed to route in three-dimensional polyhedra. We first give an alternative proof for the lower bound of $\lfloor n/2 \rfloor - 1$ beacons for routing in two dimensions. Our construction is similar to the one by Shermer [18] for orthogonal polygons. We present a family of spiral-shaped polygons for which we will then argue that $\lfloor n/2 \rfloor - 1$ beacons are needed for routing between a specific pair of points.

Definition 11. Given $c \in \mathbb{N}_{>0}$ the *c-corner spiral polygon* is a simple polygon with $n = 2c + 2$ vertices $s = r_0, r_1, \ldots, r_c, t = r_{c+1}, q_c, q_{c-1}, \ldots, q_1$, in clockwise order. The polar coordinates of the vertices are as follows:

- $r_k = \big(\lfloor k/3 \rfloor + 1; k \cdot 2\pi/3\big)$, for $k = 0, \ldots, c+1$; and
- $q_k = \big(\lfloor k/3 \rfloor + 1.5; k \cdot 2\pi/3\big)$, for $k = 1, \ldots, c$.

The trapezoids $\triangle r_k q_k q_{k+1} r_{k+1}$, for $k = 1, \ldots, c-1$ and the two triangles $\triangle s r_1 q_1$ and $\triangle t r_c q_c$ are called the *hallways.*

An example for $c = 5$ is shown in Fig. 11, with a placement of five beacons to route from s to t.

Lemma 12 (Two-dimensional lower bound)**.** *Let $c \in \mathbb{N}_{>0}$ and let P be a c-corner spiral polygon. Let $B \subset P$ be a set of beacons that lets us route from s to t. Then, we have $|B| \geq c$.*

Proof. We shoot three rays from the origin with angles $\pi/3\pi$, π, and $5\pi/3$; see Fig. 11. Each edge of P is intersected by exactly one ray. For $k = 1, \ldots, c+1$, the intersection of a ray with the edge $r_{k-1}r_k$ is called a_k and the intersection with the edge $q_{k-1}q_k$ is called b_k. We divide P into $c+2$ subpolygons $C_0, \ldots, C_{c+1}$ by drawing the line segments $a_k b_k$, for $k = 1, \ldots, c+1$. This gives two triangles C_0 and C_{c+1}, with s and t, respectively, and c subpolygons $C_1, \ldots, C_c$, called

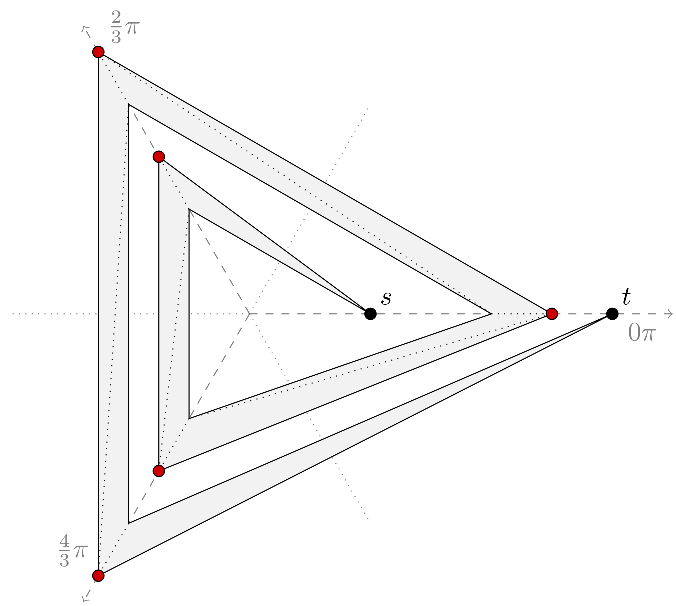
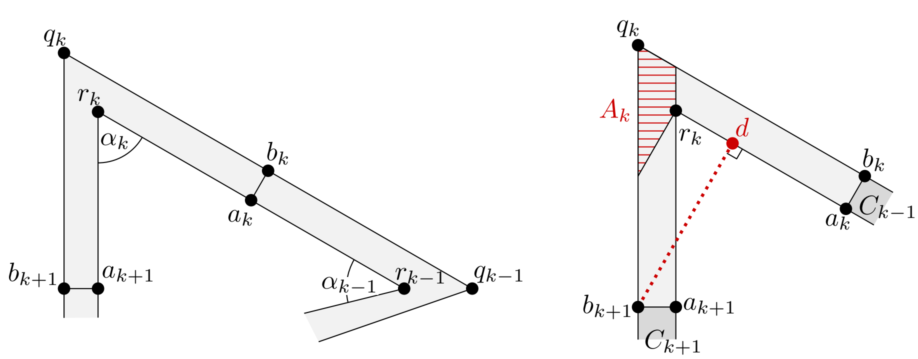


Figure 11: A 5-corner spiral polygon for which five beacons (marked in red) are necessary to route from s to t.



(a) Notation for the triangular spiral.

(b) The complete corner C_k.

Figure 12: A more detailed look at the parts of the spiral polygon.

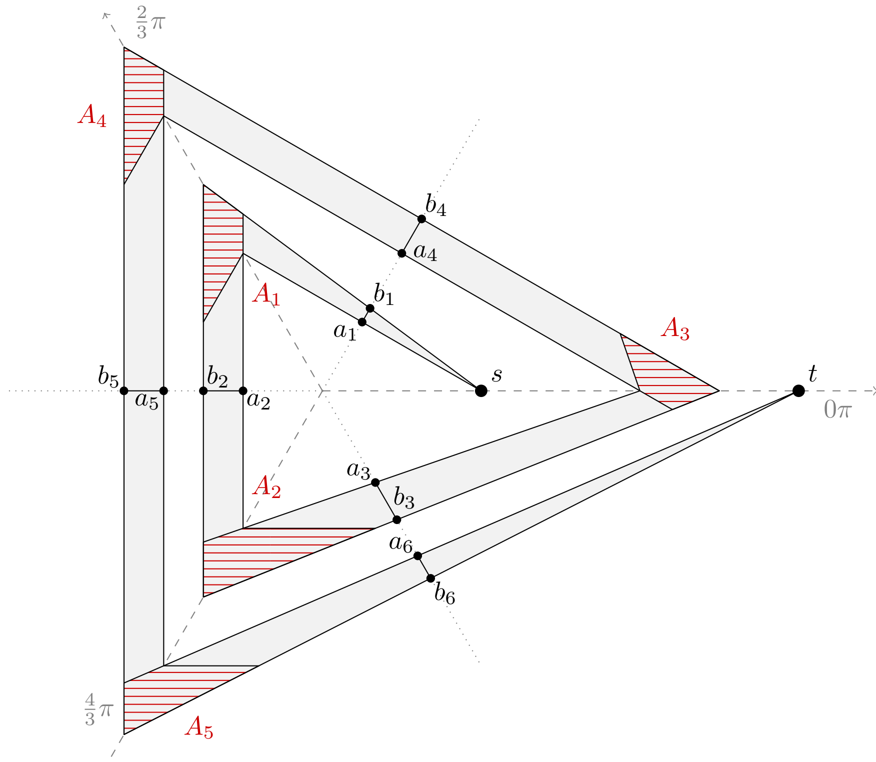


Figure 13: A 5-corner spiral polygon which shows the possible locations of the needed beacons to route through each corner when routing from s to t.

the *complete corners* of P; see Fig. 12a. We show that for $k = 1, \ldots, c$, there must be at least one beacon from B in $C_k \setminus (a_k b_k \cup a_{k+1} b_{k+1})$.

Suppose we route a point-shaped object p from s to t with the help of B. Fix a complete corner C_k, $1 \leq k \leq c$, as in Fig. 12b. Consider the last time the object p crosses $a_k b_k$. At this point, p is attracted by a beacon $b \in B$, and as we require that p moves all the way to b, the beacon b must lie in a complete corner C_ℓ, with $\ell \geq k$ (and b is not on the line segment $a_k b_k$). In fact, b can only be in C_k or in C_{k+1}, since otherwise it is clearly not possible that p reaches b along an attraction path. Thus, for p to reach b, it must be the case that either $a_k b_k$ is directly visible from b, or that the closest point to b on $r_k a_k$ is r_k. Otherwise, p would get stuck on $r_k a_k$, see Fig. 12b. The hatched region A_k in Fig. 12b shows the possible positions of b under these constraints. If this region is disjoint from $a_k b_k \cup a_{k+1} b_{k+1}$ the claim follows immediately.

In Fig. 13 we can see all A_k for $1 \leq k \leq c+1$ for $c = 5$. Clearly none of the A_k intersect $a_k b_k$. We show that none of the A_k intersect $a_{k+1} b_{k+1}$ for each of the three directions:

(i) $k = 1, 4, 7, \ldots$: The A_k are congruent since the angle α_k is always exactly

$\pi/3$. Hence, as can be observed in Fig. 13, for increasing k the distance from A_k to $a_{k+1}b_{k+1}$ increases. Since A_1 does not intersect a_2b_2 the same holds true for all $k = 1, 4, 7, \ldots$.

(ii) $k = 2, 5, 8, \ldots$: The boundary edge of A_k which could intersect $a_{k+1}b_{k+1}$ is always horizontal. As long as b_{k+1} lies above this boundary edge no intersection is possible. This is the case for A_2 (as visible in Fig. 13). Since the length of the hallways increases and the angle α_k decreases for increasing k it is always the case that b_{k+1} lies above the horizontal bounding edge of A_k. Hence, none of the A_k intersect $a_{k+1}b_{k+1}$ for $k = 2, 5, 8, \ldots$.

(iii) $k = 3, 6, 9, \ldots$: A_3 clearly does not intersect a_4b_4. However, as k grows, the angle α_k increases towards $\pi/3$ and the A_k grow towards a shape that is congruent with A_1. Since the hallways become larger and larger, even putting a rotated copy of A_1 at A_3 would not give an intersection with a_4b_4.

It follows that $|B| \geq c$. □

We now extend this proof to three dimensions. For this, we first define a c-corner spiral *polyhedron*.

Definition 13. Given $c \in \mathbb{N}_{>0}$ the *c-corner spiral polyhedron* is a polyhedron with $n = 3c+2$ vertices $s = r_0, r_1, \ldots, r_c, t = r_{c+1}, q_1, \ldots, q_c$, and $z_1, \ldots, z_c$. The coordinates of s, t, q_k, and r_k, for $k = 1, \ldots, c$, are the same as in Definition 11, with the z-coordinate set to 0. The z_k are positioned above the corresponding r_k, i.e., $z_k = r_k + \begin{pmatrix}0\\0\\1\end{pmatrix}$, for $k = 1, \ldots, c$. The edges and facets are given by the following tetrahedral decomposition:

- The start and end tetrahedra are $\triangle r_1q_1z_1s$ and $\triangle r_cq_cz_ct$.
- The *hallway* between two triangles $\triangle r_kq_kz_k$ and $\triangle r_{k+1}q_{k+1}z_{k+1}$ consists of the three tetrahedra $\triangle r_kq_kz_kr_{k+1}$, $\triangle r_{k+1}q_{k+1}z_{k+1}q_k$, and $\triangle q_kz_kr_{k+1}z_{k+1}$, for $k = 1, \ldots, c-1$.

For $c = 1$, the c-corner spiral polyhedron has two tetrahedra. For $c > 1$, we add $c-1$ hallways, each with three tetrahedra. This means that a c-corner spiral polyhedron has a tetrahedral decomposition of size $m = 3c-1$. Thus, by Definition 13, the number of tetrahedra in terms of the number of vertices is $m = 3 \cdot (n-2)/3 - 1 = n - 3$, the smallest number possible for a given n.

Lemma 14 (Lower bound). *Let $c \in \mathbb{N}_{>0}$ and let P be a c-corner spiral polyhedron. Let B be a set of beacons that lets us route from s to t. Then, $|B| \geq c$.*

Proof. We show that a projection B' of B onto the xy-plane maintains the attraction regions. It then follows from Lemma 12 that $|B| = |B'| \geq c$.

Note that the only reflex edges in P are the edges $e_k = r_kz_k$ for all $k = 1, \ldots, c$. Look at a beacon $b \in B$ and its projection $b' \in B'$. If a point p is visible from b it must be visible from b' as well: Since the hallways are convex objects and the only edges that could prevent visibility are the vertical reflex edges $r_k z_k$ a vertical translation of b to b' cannot inhibit visibility.

If a point p is attracted by b (but not visible from b) it must be attracted by b' as well. Each such attraction goes through exactly one reflex edge: at least one since p is not visible and at most one since two reflex edges in P together form angles larger than π. The movement of p is a movement (possibly of length zero) until it hits a face $f_k = r_k z_k r_{k+1} z_{k+1}$ at point q. It then slides along f_k until it hits one of the boundary edges w.l.o.g. $e_k = r_k z_k$ at point u. It then moves directly towards b.

Since f_k is orthogonal to the xy-plane if p is attracted by b' it will hit f_k at a point q' which can be obtained by moving q down along the z-axis. Hence the point then slides from q' along f_k towards e_k where it reaches at a point u' which (again due to e_k being orthogonal to the xy-plane) can be obtained by moving u down along the z-axis. It then moves directly towards b'.

Thus the set B can only attract what B' can. Since B' lies in the xy-plane and a cross section of P along the xy-plane gives exactly a c-corner spiral *polygon* P'. By Lemma 12 we obtain then that $|B| = |B'| \geq c$, as claimed. □

5. A Tight Bound for Beacon-based Routing

We combine the results from Section 3 and Section 4 into a tight bound:

Theorem 15. *Let P be a three-dimensional polyhedron, and m the smallest size of a tetrahedral decomposition of P. Then, it is always sufficient and sometimes necessary to place $\lfloor (m+1)/3 \rfloor$ beacons to route between any pair of points in P.*

Proof. The upper bound was shown in Theorem 5. For the lower bound, we consider the c-corner spiral polyhedron P_c with $c = \lfloor (m+1)/3 \rfloor$. By Definition 13, P_c has a smallest tetrahedral decomposition of size $m' = 3c - 1$. Furthermore, by Lemma 14, we need at least c beacons to route in P_c. This also shows that P_c does not have a tetrahedral decomposition with size strictly less than m', since otherwise Theorem 5 would yield a contradiction.

Due to the rounding we might have $m' = m - 1$ or $m - 2$. We then look at the $(c+1)$-corner spiral P_{c+1} that consists of three tetrahedra more than P_c. More specifically, the last hallway of P_{c+1} consists of the three tetrahedra $\sigma_1 = \triangle r_c q_c z_c r_{c+1}$, $\sigma_2 = \triangle q_c z_c r_{c+1} z_{c+1}$, and $\sigma_3 = \triangle q_c r_{c+1} q_{c+1} z_{c+1}$. The tetrahedron σ_1 is already present in P_c. Hence, for $m' = m - 1$, we add σ_2, and for $m' = m - 2$, we add σ_2 and σ_3 to P_c. Since for each additional tetrahedron we also need to add one additional vertex (z_{c+1} for σ_2 and q_{c+1} for σ_3), there is no decomposition of the resulting polyhedron into less than m tetrahedra.

Additionally, the resulting polyhedron also needs at least c beacons because the added tetrahedra cannot lower the number of beacons needed. □

6. Conclusion

We have shown that, given a tetrahedral decomposition of a polyhedron P of size m, we can place $\lfloor (m+1)/3 \rfloor$ beacons to route between any pair of points in P. We also constructed a family of polyhedra where this is also necessary.

A lot of questions that have been studied in two dimensions remain open for the three-dimensional case. For example, the complexity of finding an optimal beacon set to route between a given pair of points remains open. Additional open questions concern the efficient computation of attraction regions (computing the set of all points attracted by a single beacon) and of beacons kernels (all points at which a beacon can attract all points in the polyhedron).

Furthermore, Cleve [10] showed that not all polyhedra can be covered by vertex beacons and Aldana-Galván et al. [1,2] showed that this is even true for orthogonal polyhedra. Given a polyhedron P with a tetrahedral decomposition of size m, it remains open whether it is possible to guard P with fewer than $\max\{1, \lfloor (m+1)/3 \rfloor\}$ beacons as in Observation 10.

References

[1] I. Aldana-Galván, J. L. Álvarez Rebollar, J. C. Catana Salazar, N. Marín Nevárez, E. Solís Villarreal, J. Urrutia, and C. Velarde. Beacon coverage in orthogonal polyhedra. In *Proc. 29th Canad. Conf. Comput. Geom. (CCCG)*, pages 166–171, 2017.

[2] I. Aldana-Galván, J. L. Álvarez-Rebollar, J. C. Catana-Salazar, N. Marín-Nevárez, E. Solís-Villarreal, J. Urrutia, and C. Velarde. Covering orthotrees with guards and beacons. In *Proc. 17th Spanish Meeting Comput. Geom. (EGC)*, pages 29–32, 2017.

[3] S. W. Bae, C.-S. Shin, and A. Vigneron. Tight bounds for beacon-based coverage in simple rectilinear polygons. In *Proc. 12th Lat. Am. Symp. Theor. Inf. (LATIN)*, pages 110–122, 2016.

[4] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. *Computing in Euclidean geometry*, 4:47–123, 1995.

[5] M. Biro. *Beacon-Based Routing and Guarding.* PhD thesis, State University of New York at Stony Brook, 2013.

[6] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell. Beacon-based routing and coverage. In *Proc. 21st Fall Workshop Comput. Geom. (FWCG)*, 2011.

[7] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell. Combinatorics of beacon-based routing and coverage. In *Proc. 25th Canad. Conf. Comput. Geom. (CCCG)*, pages 129–134, 2013.

[8] M. Biro, J. Iwerks, I. Kostitsyna, and J. S. B. Mitchell. Beacon-based algorithms for geometric routing. In *Proc. 13th Int. Symp. Algorithms Data Struct. (WADS)*, pages 158–169, 2013.

[9] B. Chazelle. Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. *SIAM J. Comput.*, 13(3):488–507, 1984.

[10] J. Cleve. Combinatorics of beacon-based routing and guarding in three dimensions. Master's thesis, Freie Universität Berlin, 2017.

[11] S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

[12] I. Kostitsyna. Personal communication. 2019.

[13] I. Kostitsyna, B. Kouhestani, S. Langerman, and D. Rappaport. An optimal algorithm to compute the inverse beacon attraction region. In *Proc. 34th Int. Symp. Comput. Geom. (SoCG)*, pages 55:1–14, 2018.

[14] B. Kouhestani. *Efficient algorithms for beacon routing in polygons*. PhD thesis, Queen's University, Kingston, Ontario, 2013.

[15] N. J. Lennes. Theorems on the simple finite polygon and polyhedron. *Am. J. Math.*, 33(1/4):37, 1911.

[16] J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, 1987.

[17] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete Comput. Geom.*, 7(3):227–253, 1992.

[18] T. C. Shermer. A combinatorial bound for beacon-based routing in orthogonal polygons. In *Proc. 27th Canad. Conf. Comput. Geom. (CCCG)*, pages 213–219, 2015.

Appendix A. Program Code to Generate Trees

```python
#!/usr/bin/env python3
"""Generate all dual graph configurations we need to look at."""

from itertools import product

from graphviz import Digraph


############################################################################
# Tree structure.
############################################################################
class Node:
    """A tree structure which allows pruning of unneeded subtrees."""

    # ======================================================================
    # General tree structure.
    # ======================================================================

    def __init__(self):
        """A new node is simply a leaf."""
        self.nodes = []

    def add(self, n=1):
        """Append n additional children and return self."""
        for _ in range(n):
            self.nodes.append(Node())
```

```
        return self

    def append(self, node):
        """Append a node or an iterable of nodes and return self."""
        try:
            for n in node:
                self.nodes.append(n)
        except TypeError:
            self.nodes.append(node)
        return self

    def is_leaf(self):
        """Return whether this node is a leaf, i.e., has no children."""
        return not self.nodes

    # ======================================================================
    # Graphviz.
    # ======================================================================

    def to_dot(self, graph=None, prefix=''):
        """Return a Graphviz representation of the tree."""
        if graph is None:
            graph = Digraph()
        self._dot_recursion(graph, 1, prefix)
        return graph

    def _dot_recursion(self, graph, current, prefix=''):
        """Recursively create Graphviz tree."""
        graph.node(prefix + str(current), label=str(current))
        this_number = current
        current = current + 1
        for child in self.nodes:
            current, child_number = child._dot_recursion(graph, current,
                                                         prefix)
            graph.edge(prefix + str(this_number), prefix + str(child_number))
        return current, this_number

    # ======================================================================
    # Pruning of "easy" cases.
    # ======================================================================

    def prune(self):
        """Remove subtrees that are easily removed."""
        # First try to remove subtrees.
        if self._prune() is None:
            return None

        # Call prune() for all children and filter out children that were.
        # removed
        self.nodes = list(filter(lambda x: x is not None,
                                 map(Node.prune, self.nodes)))

        # Sort children after pruning to have a canonical structure.
        self.nodes.sort()

        # Try pruning easy subtrees again. Maybe pruning the children created
        # a prunable configuration again.
        return self._prune()

    def _prune(self):
        """Remove subtrees that are easily removed."""
        if len(self.nodes) == 3:
            if all(n.is_leaf() for n in self.nodes):
                # Case (i): Figure 5.4(a): This is s2
                #   Three children that are leaf nodes: Remove all of them.
                self.nodes = []
            elif all(len(n.nodes) == 1 and n.nodes[0].is_leaf()
                     for n in self.nodes):
```

```
                # Case (iii)(3): Figure 5.4(e): This is s3
                #   Three children with one child leaf each: Remove two
                #                                            children.
                self.nodes.pop()
                self.nodes.pop()
        if len(self.nodes) == 2:
            if all(n.is_leaf() for n in self.nodes):
                # Case (ii): Figure 5.4(b): This is s2
                #   Two children that are leaf nodes: Remove both including
                #                                     the parent node.
                return None
        if len(self.nodes) == 1:
            if len(self.nodes[0].nodes) == 1:
                if self.nodes[0].nodes[0].is_leaf():
                    # Case (iii)(1): Figure 5.4(c): This is s3
                    #   A chain of three nodes: Remove all of them.
                    return None
        if len(self.nodes) >= 2:
            leaves = [n for n in self.nodes if n.is_leaf()]
            leaves2 = [n for n in self.nodes if len(n.nodes) == 1 and
                       n.nodes[0].is_leaf()]
            if leaves and leaves2:
                # Case (iii)(2): Figure 5.4(d): This is s3
                #   One leaf child and one child with a single leaf child:
                #       Remove both children.
                self.nodes.remove(leaves[0])
                self.nodes.remove(leaves2[0])

        # Return self to indicate that the node itself is not to be removed.
        return self

    # ======================================================================
    # Make trees comparable.
    # ======================================================================

    def __eq__(self, other):
        """
        Compare equality of two nodes.

        Two nodes are equal if they have the same number of children and
        every child is equal to the respective child of the other node.
        """
        if other is None:
            return False
        if len(other.nodes) != len(self.nodes):
            return False
        for this, that in zip(self.nodes, other.nodes):
            if this != that:
                return False
        return True

    def __lt__(self, other):
        """
        Compare whether a node is smaller than another node.

        A node is smaller then another node if it has more direct children or
        if any of the children is smaller than the respective other child.
        """
        if len(self.nodes) != len(other.nodes):
            return len(self.nodes) > len(other.nodes)

        for this, that in zip(self.nodes, other.nodes):
            if this < that:
                return True
            if that < this:
                return False

        return True
```

```python

    # ======================================================================
    # String representation and hash value for uniqueness.
    # ======================================================================

    def __str__(self):
        """Generate a bracket term representing the tree."""
        return '(' + ''.join(str(n) for n in self.nodes) + ')'

    def __repr__(self):
        """Terminal representation."""
        return str(self)

    def __hash__(self):
        """Hash value for uniqueness."""
        return hash(str(self))


############################################################################
# Generate all trees with certain maximum depth.
############################################################################
def all_trees(depth):
    """
    Yield all trees with a given maximum depth.

    The trees are created recursively by appending combinations of trees of
    depth-1 to a node.
    """
    if depth == 1:
        # Create a node with 0, 1, 2, and 3 children.
        for i in range(4):
            yield Node().add(i)
    else:
        # Append 0, 1, 2, or 3 children.
        for number_of_children in range(4):
            # Create as many iterators of the next lower depth as there
            # should be children appended.
            next_level_iterators = []
            for _ in range(number_of_children):
                next_level_iterators.append(all_trees(depth - 1))

            # Combine all possible combinations of the iterators and add them
            # to a new node.
            for subtrees in product(*next_level_iterators):
                yield Node().append(subtrees)


def iterator_len(iterator):
    """
    Return the number of elements in an iterator.

    The iterator is consumed by calling this function.
    """
    length = 0
    for _ in iterator:
        length += 1
    return length


############################################################################
# Main program.
############################################################################
if __name__ == '__main__':
    # The maximum depth of the tree is 3
    depth = 3
    number_of_combinations = iterator_len(all_trees(depth))
    # Start with the first tree
    current = 1
```

```
    # A container for all distinct non-prunable trees
    trees = set()

    # Iterate through all different trees of maximum depth
    for tree in all_trees(depth):
        # Prune "easy" cases
        tree = tree.prune()
        # Add tree to set of trees if it was not pruned completely
        if tree is not None:
            trees.add(tree)

        # Debug output
        print('\r{percent:.2f}% ({current} / {all}) - trees: {trees}'
              .format(percent=100 * current / number_of_combinations,
                      current=current,
                      all=number_of_combinations,
                      trees=len(trees)),
              end='', flush=True)
        current += 1

    # Sum up the number of trees
    print()
    print(len(trees), 'trees')

    # Create a document with all non-prunable trees
    g = Digraph()
    prefix = 1
    for tree in trees:
        tree.to_dot(g, str(prefix) + '_')
        prefix += 1
    g.render('trees')
```