Computing Optimal Shortcuts for Networks

Delia Garijo
Departamento de Matemática Aplicada I, Universidad de Sevilla, Spain
dgarijo@us.es

Alberto Márquez
Departamento de Matemática Aplicada I, Universidad de Sevilla, Spain
almar@us.es

Natalia Rodríguez
Departamento de Computación, Universidad de Buenos Aires, Argentina
nrodriguez@dc.uba.ar

Rodrigo I. Silveira
Departament de Matemàtiques, Universitat Politècnica de Catalunya, Spain
rodrigo.silveira@upc.edu

Abstract

We study augmenting a plane Euclidean network with a segment, called *shortcut*, to minimize the largest distance between any two points along the edges of the resulting network. Questions of this type have received considerable attention recently, mostly for discrete variants of the problem. We study a fully continuous setting, where all points on the network and the inserted segment must be taken into account. We present the first results on the computation of optimal shortcuts for general networks in this model, together with several results for networks that are paths, restricted to two types of shortcuts: shortcuts with a fixed orientation and simple shortcuts.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases graph augmentation, shortcut, diameter, geometric graph

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.15

Related Version A full version of the paper is available at [12], `https://arxiv.org/abs/1807.10093`.

Funding D. G. and R. S. were supported by project MTM2015-63791-R. R. S. was also supported by Gen. Cat. 2017SGR1640 and MINECO through the Ramón y Cajal program. A. M. was supported by project BFU2016-74975-P.

1 Introduction

A fundamental task in network analysis, especially in the context of geographic data (for instance, for networks that model roads, rivers, or train tracks), is analyzing how an existing network can be improved. This can arise in many different contexts: in relation to facility location analysis, for instance, to guarantee a certain maximum travel time from any point on the network to the nearest hospital, or in road network design problems, to decide where to add road segments to reduce network congestion [16].

Networks like the ones above are naturally modeled as a *geometric network*: an undirected graph whose vertices are points in $\mathbb{R}^2$ and whose edges are straight-line segments connecting pairs of points. Moreover, in many applications, it is reasonable to assign lengths to the edges equal to the Euclidean distance between their endpoints. These are called *Euclidean*

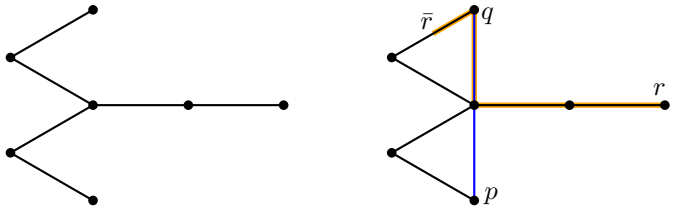

Figure 1 Left: Example of network with diameter 4 (each edge has unit length); in the highway model, no single segment insertion can improve the diameter. Right: in the planar model, segment pq is a shortcut, since its insertion reduces the diameter to $d(r, \bar{r}) = 3.5$. However, pq is not a shortcut in the highway model as its insertion increases the diameter to 5.

networks. When, in addition, there are no crossings between edges, the Euclidean network is said to be *plane*. Many problems in geographic analysis, for instance, those involving transportation networks, can be accurately modeled with a plane Euclidean network. In the following, we shall simply write *network*, it being understood as plane and Euclidean.

One of the most fundamental ways to improve a network is by adding edges. This increases the connectivity of the network and potentially can decrease travel times and congestion. The most studied criteria to measure network improvement, in the geometric setting, are related to distances. Particularly important is the maximum distance, or *diameter* of the network, which provides an upper bound on the distance between any two network points. Another important distance-related criterion in this context is the *dilation*, which captures the maximum detour between two points on the network.

In this work, we focus on the problem of adding edges to a network in order to improve its diameter. This can be seen as a variant of the Diameter-Optimal-k-Augmentation problem, which consists in inserting k additional segments into a graph, while minimizing the largest distance in the resulting network (see the survey [14] for more on augmentation problems over plane geometric graphs). More precisely, we study a continuous version of the problem for $k = 1$: we consider the addition of one segment, called *shortcut*, whose endpoints can be any two points (not necessarily vertices) on the network. A segment will be considered a shortcut only if its insertion improves the diameter of the resulting network. Note that the resulting network includes the points on the shortcut inserted.

Our goal is to find an *optimal* shortcut: one minimizing the diameter of the resulting network, over all possible shortcuts.

Two major variants of the problem arise, depending on how the shortcut is inserted into the network. In the first variant, which we call *highway model*, the crossings between the shortcut and the network edges do not form new network vertices: a path can only enter and leave the shortcut through its endpoints. In contrast, in the *planar model*, every crossing creates a new vertex, which can be used by paths in the network. Figure 1 illustrates some of the differences between the two models.

In this work, we focus on the planar model. This model is more general, and is applicable to a wider range of situations, like the addition of segments to road or pedestrian networks. From a theoretical point of view, the difference between the highway and planar model is important. The latter results in more complex problems, since the fact that a shortcut can be used only in part, implies that the structural information on how the distances in the network change after adding a segment is more difficult to maintain. Moreover, as we show in this work, many intuitive properties of shortcuts do not hold in the planar model anymore.

Related work. There has been a considerable amount of work devoted to the graph version of the Optimal-k-Augmentation problem. Due to space constraints, we only discuss the geometric version of the problem (i.e., where the graph is embedded in the plane), and where the continuous diameter is used (i.e., the distances are taken over all pairs of points in the network, as opposed to considering only pairs of vertices).

Most attention to the problem studied here has been on the highway model, for certain classes of graphs. For paths, De Carufel et al. [6] gave an algorithm to find an optimal shortcut in linear time, and also optimal pairs of shortcuts (i.e., $k = 2$) for convex cycles. Trees have been studied in a recent follow-up work [7], which presents an algorithm to find an optimal shortcut for a tree of size n in $O(n \log n)$ time. For circles, very recently Bae et al. [1] have analyzed how to add up to seven shortcuts in an optimal way.

For the planar model much less is known. Yang [15] designed three different approximation algorithms to compute an *optimal* shortcut for certain types of paths. Cáceres et al. [5] were the first to consider general networks, for which they show that one can find a shortcut in polynomial time if one exists (note that there are networks whose diameter cannot be improved by adding only one segment, e.g., a cycle), but they do not look for an optimal one.

Our results. We present the first study of optimal shortcuts in the planar model for general networks, and several improved results for paths. An important contribution of our work is to highlight many important differences between the highway and planar models, the latter resulting in considerably harder problems. In Section 2, we give a polynomial time algorithm to compute an optimal shortcut if one exists. Moreover, we present a discretization of the problem that immediately leads to an approximation algorithm for general networks, generalizing an existing result for paths [15]. Section 3 focuses on paths: we first show that the diameter of a path network after adding a shortcut can be computed in $\Theta(n)$ time. Then we improve the method of Section 2 for shortcuts of any fixed direction. Finally, we study simple shortcuts, a variant that has been studied before, which has applications in settings where the added edge cannot intersect the existing network.

Due to space limitations, most proofs are not included here; they can be found in [12].

1.1 Preliminaries

We will use $\mathcal{N} = (V(\mathcal{N}), E(\mathcal{N}))$ to denote a network with n vertices, and $\mathcal{N}_\ell$ for its *locus*, the set of all points of the Euclidean plane that are on $\mathcal{N}$. Thus, $\mathcal{N}_\ell$ is treated indistinctly as a network or as a closed point set. When $\mathcal{N}_\ell$ is a path, we use $\mathcal{P}_\ell$ instead of $\mathcal{N}_\ell$. Further, we write $a \in \mathcal{N}_\ell$ for a point a on $\mathcal{N}_\ell$, and $V(\mathcal{N}) \subset \mathcal{N}_\ell$.

A *path* P connecting two points a, b on $\mathcal{N}_\ell$ is a sequence $au_1 \dots u_k b$ such that $u_1u_2, \dots, u_{k-1}u_k \in E(\mathcal{N})$, a is a point on an edge ($\neq u_1u_2$) incident to u_1, and b is a point on an edge ($\neq u_{k-1}u_k$) incident to u_k. We use $|P|$ to denote the *length* of P, i.e., the sum of the lengths of all edges u_iu_{i+1} plus the lengths of the segments au_1 and bu_k. The length of a shortest path from a to b is the *distance* between a and b on N_ℓ. This distance is written as $d_{N_\ell}(a, b)$ or $d(a, b)$ when the network is clear, and whenever $ab \notin E(N_\ell)$, it is larger than $|ab|$, the Euclidean distance between the points.

The *eccentricity* of a point $a \in \mathcal{N}_\ell$ is $\mathrm{ecc}(a) = \max_{b \in \mathcal{N}_\ell} d(a, b)$, and the *diameter* of $\mathcal{N}_\ell$ is $\mathrm{diam}(\mathcal{N}_\ell) = \max_{a \in \mathcal{N}_\ell} \mathrm{ecc}(a)$. Two points $a, b \in \mathcal{N}_\ell$ are *diametral* whenever $d(a, b) = \mathrm{diam}(\mathcal{N}_\ell)$, and a shortest path connecting a and b is then called *diametral path*.

The diameter of $\mathcal{N}_\ell$, $\mathrm{diam}(\mathcal{N}_\ell)$, can be computed in polynomial time [5, 8]. Furthermore, the diametral pairs of $\mathcal{N}_\ell$ are either (i) two vertices, (ii) two points on distinct non-pendant

edges[1], or (iii) a pendant vertex and a point on a non-pendant edge [5, Lemma 6]. Thus, with some abuse of notation, in Section 2, we will say that a *diametral pair* $\alpha, \beta \in V(\mathcal{N}) \cup E(\mathcal{N})$ may be (i) vertex-vertex, (ii) edge-edge, or (iii) vertex-edge.

A *shortcut* for $\mathcal{N}_\ell$ is a segment s with endpoints on $\mathcal{N}_\ell$ such that $\text{diam}(\mathcal{N}_\ell \cup s) < \text{diam}(\mathcal{N}_\ell)$. We say that shortcut s is *simple* if its two endpoints are the only intersection points with $\mathcal{N}_\ell$, and s is *maximal* if it is the intersection of a line and $(\mathcal{N}_\ell \cup s)$, i.e., $s = (\mathcal{N}_\ell \cup s) \cap \ell$, for some line ℓ. A shortcut is *optimal* if it minimizes $\text{diam}(\mathcal{N}_\ell \cup s)$ among all shortcuts s for $\mathcal{N}_\ell$.

2 General networks

The main result in [5] states that one can always determine in polynomial time whether a network $\mathcal{N}_\ell$ has a shortcut (and compute one, in case of existence). In this section, we first prove the analogous result for optimal shortcuts. Our proof uses some ideas in [5] but captures the property of being optimal with a much shorter argument based on some functions defined in Lemma 1 below.

Let $\alpha, \beta \in V(\mathcal{N}) \cup E(\mathcal{N})$, and let $e = uv$ and $e' = u'v'$ be two edges of $\mathcal{N}$. When α is an edge, we use $\text{ecc}(u, \alpha)$ to indicate the maximum distance from u to the points on α (analogous for β and the remaining endpoints of e and e'); if α is a vertex, $\text{ecc}(u, \alpha) = d(u, \alpha)$. In general, $\text{ecc}(\alpha, \beta) = \max_{t \in \alpha, z \in \beta} d(t, z)$.

▶ **Lemma 1.** *Let $y = ax + b$ be a line intersecting edges $e = uv$ and $e' = u'v'$ on points p and q, respectively, and let $\alpha, \beta \in V(\mathcal{N}) \cup E(\mathcal{N})$. For each pair (w, z) with $w \in \{u, v\}$ and $z \in \{u', v'\}$, function $f^{w,z}_{\alpha,\beta}(a, b) = \text{ecc}(w, \alpha) + |wp| + |pq| + |qz| + \text{ecc}(z, \beta)$ is linear in b.*

The following theorem is the optimality version of Theorem 8 in [5].

▶ **Theorem 2.** *It is possible to determine in polynomial time whether a network $\mathcal{N}_\ell$ admits an optimal shortcut, and compute one in case of existence.*

It should be noted that the approach that leads to the preceding result, albeit polynomial, has a very high running time. A direct implementation involves $O(n^4)$ functions $f^{w,z}_{\alpha,\beta}(a, b)$ that must be computed, and this has to be done for $O(n^4)$ different cases. Moreover, each evaluation of $f^{w,z}_{\alpha,\beta}(a, b)$ takes $O(n^2)$ time. All in all, its running time would add up to $O(n^{10})$.

2.1 Discretizing the set of possible shortcuts: approximation

In light of the high running time of the previous approach, it becomes interesting to look for faster approximation algorithms. Moreover, given the continuous nature of the problem, it is natural to wonder to what extent the problem can be discretized. In other words, how good can shortcuts be if we restrict them to some discrete collection of segments? The most natural choice for such a collection is probably the segments defined by pairs of vertices u, v of $\mathcal{N}_\ell$, but this choice can lead to poor results, as the example in Figure 2(left) shows. In some cases, one can do better by considering the *maximal extensions* of the segments uv (i.e., the largest segment through uv with endpoints on $\mathcal{N}_\ell$), as Yang [15] did to obtain an additive approximation for paths. Unfortunately, as Figure 2(right) shows, maximal extensions do not work anymore as soon as $\mathcal{N}_\ell$ is a tree. However, in this section, we show that if one considers all extensions of segments defined by two vertices of $\mathcal{N}_\ell$, then it is possible to guarantee an approximation factor for general networks.

[1] An edge $uv \in E(\mathcal{N})$ is *pendant* if either u or v is a *pendant* vertex (i.e., has degree 1).

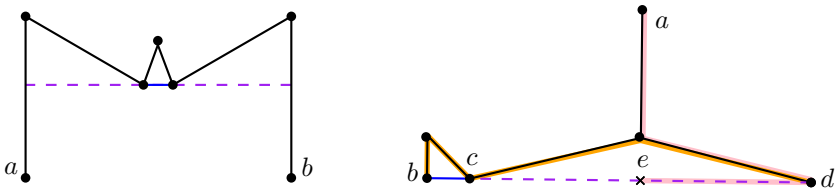

Figure 2 Left: the optimal shortcut is the dashed purple segment, which contains the blue segment. That blue segment (and any other segment between two vertices) gives a larger diameter as points a and b are diametral for both segments. Right: the original diameter is given by the orange path. The best shortcut connecting two vertices is bc. Contrary to intuition, extending bc to bd worsens the diameter, which becomes given by points a and e (pink path).

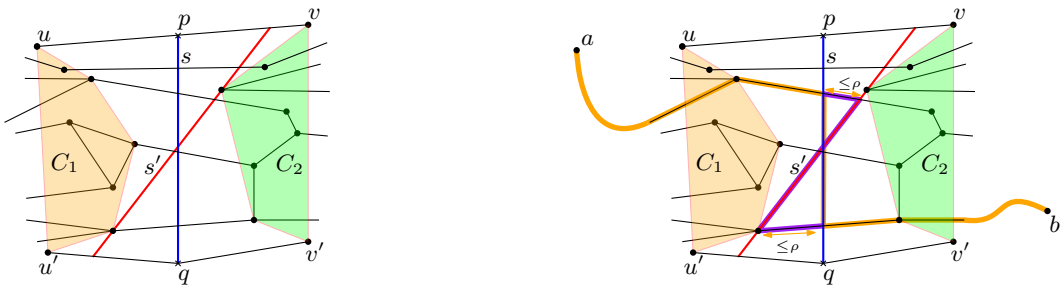

Figure 3 Left: approximating a shortcut s with a segment $s' \in \mathcal{S}_2$. Right: using s' instead s to go from a to b causes a detour of at most 4ρ (purple path).

Let $\mathcal{S}$ be the infinite set of segments with endpoints in $\mathcal{N}_\ell$, and let $\mathcal{S}_2 \subset \mathcal{S}$ be the subset of segments of $\mathcal{S}$ that contain two vertices of $\mathcal{N}_\ell$. The following proposition states that set $\mathcal{S}_2$ is an approximation of $\mathcal{S}$.

▶ **Proposition 3.** *Let ρ be largest edge length in $\mathcal{N}_\ell$. Then,* $\min_{s\in\mathcal{S}} \operatorname{diam}(\mathcal{N}_\ell \cup s) \leq \min_{s\in\mathcal{S}_2} \operatorname{diam}(\mathcal{N}_\ell \cup s) \leq \min_{s\in\mathcal{S}} \operatorname{diam}(\mathcal{N}_\ell \cup s) + 4\rho$.

Proof. The first inequality is straightforward. For the second, it suffices to prove that given $s = pq \in \mathcal{S} \setminus \mathcal{S}_2$ there exists $s' \in \mathcal{S}_2$ such that $\operatorname{diam}(\mathcal{N}_\ell \cup s') \leq \operatorname{diam}(\mathcal{N}_\ell \cup s) + 4\rho$.

Segment s may cross several faces of $\mathcal{N}_\ell$, refer to Figure 3. Consider the first and the last ones, say $\mathcal{F}_1$ and $\mathcal{F}_2$, together with the vertices of $\mathcal{N}_\ell$ that are adjacent to p and q in those faces: u, v in $\mathcal{F}_1$ and u', v' in $\mathcal{F}_2$. Let V_1 be the vertices of $\mathcal{N}_\ell$ in the quadrilateral $upqu'$ (including u and u'), and let C_1 be its convex hull. Analogously, we have V_2 and C_2 for the quadrilateral $vpqv'$. Note that both convex hulls may have one point in common. Extending one of the common internal tangents of C_1 and C_2 gives rise to a segment s' with endpoints on two of the edges of $\mathcal{F}_1$ and $\mathcal{F}_2$ containing points p and q. Observe that s' intersects all the edges of $\mathcal{N}_\ell$ that are crossed by s. Thus, this construction allows us to show that, for any two points $a, b \in \mathcal{N}_\ell$, the length of the shortest path between a and b that uses s' is at most 4ρ plus the corresponding length but using s. To do this, we first use the triangle inequality to compare the lengths of the used portions of segments s and s', which gives a difference of 2ρ, and then we add the two distances indicated in Figure 3(right). A similar argument is used for $a \in s'$ and $b \in \mathcal{N}_\ell$. ◀

The collection $\mathcal{S}_2$ is finite but quite large, it has size $O(n^4)$, which gives a time complexity of $O(n^6)$ to compute the optimal among the segments in $\mathcal{S}_2$ (there are $O(n^2)$ possible extensions per each pair of vertices, and for each of them one needs to compute the diameter from scratch in $O(n^2)$ time [13]).

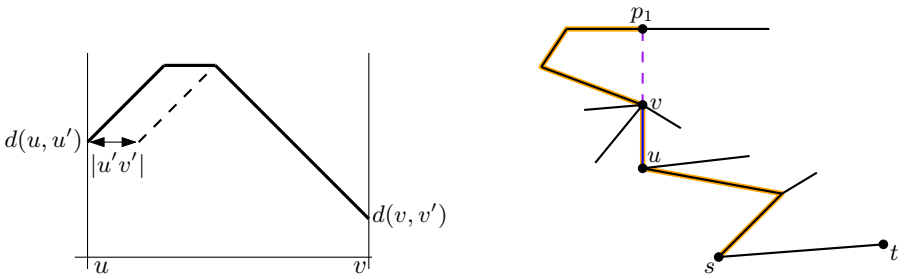

Figure 4 Left: function Φ_{uv}^{st}. Right: if the distance from p_1 to st decreases when adding $s_0 = uv$, then, after adding $s_1 = up_1$ it will become even smaller.

We would like to find a small subset of $\mathcal{S}_2$ that preserves the property in Proposition 3. Ideally, we would like to consider not all the extensions of a segment with endpoints in $V(\mathcal{N}_\ell)$ (that is exactly $\mathcal{S}_2$), but only the best extension for each segment. Unfortunately, this appears rather difficult: already for a tree with a single vertex of degree larger than two, it may happen that an extension of a segment gives a worse diameter than the segment itself, see Figure 2(right). However, we show next that we can speed-up the computation of the diameter for each extension in $\mathcal{S}_2$, saving a nearly-linear factor in the total running time.

Given a segment $s' = p'q'$, let r be the ray starting at p' and containing s', and let $\mathcal{P} = p_0, p_1, \ldots, p_k$ be the sorted list of intersection points of r with edges of $\mathcal{N}_\ell$ (note that $q' = p_j$ for some j). Segments $s_i = p'p_i$ are called extensions of s' to the *right*; the extensions to the *left* are defined similarly. Next we show how to speed-up the re-computation of the diameter of $\mathcal{N}_\ell \cup s_i$ as we insert $s_0, s_1, \ldots s_k$, in that order. To that end, we split the re-computation of distances into two parts: distances from points on s_i to points on $\mathcal{N}_\ell$, and distances (in $\mathcal{N}_\ell \cup s_i$) between two points on $\mathcal{N}_\ell$.

▶ **Lemma 4.** *Let u and v be vertices of $\mathcal{N}_\ell$. It is possible to compute the eccentricities of all the extensions to the right of segment uv in $O(n^2)$ time.*

Proof. As a preprocessing step, we store the distances from each vertex to all the other edges and the point at each edge attaining that maximum distance. This allows us to construct the functions $\Phi_{uv}^{st} : [0, 1] \to \mathbb{R}^+$ that encode the information of the maximum distance from each point on an edge uv to an edge st (see [13, Theorem 2] for their analytic expression). Their shape is as follows – see also Figure 4(left): let u' and v' be the farthest points to, respectively, u and v in edge st. Function Φ_{uv}^{st} increases uniformly from 0 and from 1 until the distance between both lines equals the distance between u' and v', at that moment it stabilizes horizontally. Thus, knowing the farthest points u' and v' to u and v in the segment st (and the distance between them), it is possible to build Φ_{uv}^{st} in constant time.

The main idea of the proof of this lemma is that it is possible to update each map Φ_{uv}^{st} for each extension of a segment again in constant time. Observe that Φ_{uv}^{st} encloses the information of the largest distance from any point of uv to the segment st.

In a first step, we insert segment $s_0 = uv$. As u and v are in $\mathcal{N}_\ell$, they belong to some edges g and g', and we use the information of Φ_g^{st} and $\Phi_{g'}^{st}$ to find the largest distance from u and v to st (in $\mathcal{N}_\ell$). With that information, we compute Φ_{uv}^{st} in constant time. Thus, the maximum eccentricity of the edge $s_0 = uv$ can be computed in linear time.

Observe that building the map Φ_{uv}^{st} it is possible to detect if $\text{ecc}(v, st)$ changes when adding s_0, so, we update the values of the distances from vp_0 to all the other edges, and the point on each edge giving that maximum distance (again, in linear time).

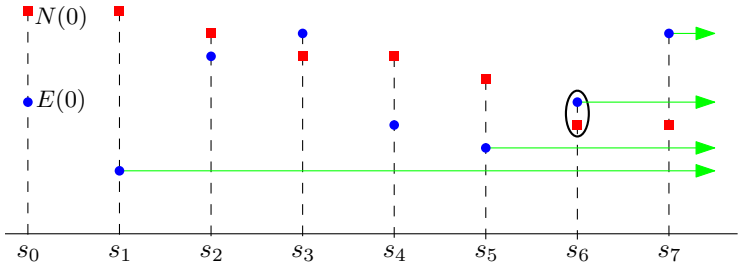

Figure 5 For each segment extension, we consider two values: $E(i)$ – maximum eccentricity on s_i – (blue points), and $N(i)$ – maximum eccentricity in $\mathcal{N}_\ell \cup s_i$ for points in $\mathcal{N}_\ell$– (red squares). For each s_i, the diameter of $\mathcal{N}_\ell \cup s_i$ is given by the maximum of these two values. Only the blue points with green arrows must be tested (they are the maximal blue points).

Of course, the addition of s_0 can change the eccentricity of p_1 with respect to some other edge (and the same with any of the other p_i's), but we will see that we do not need to update that information at this moment. Indeed, if the distance from p_1 to st changes when adding s_0, it can only decrease. Then, the addition of s_1 is going to make it even smaller – see Figure 4(right). Thus, in step i, we only need to update the information of the new vertex p_i, since by adding s_i, the value of p_{i+1} is going to be updated. ◀

▶ **Lemma 5.** *Let u and v be vertices of $\mathcal{N}_\ell$. It is possible to find the extension s of segment uv that minimizes* $\operatorname{diam}(\mathcal{N}_\ell \cup s)$ *in $O(n^3 \log n)$ time.*

Proof. The value of $\operatorname{diam}(\mathcal{N}_\ell \cup s)$ can be computed by calculating the eccentricity of segment s and comparing with the eccentricities in $\mathcal{N}_\ell \cup s$ of all the points in $\mathcal{N}_\ell$. Thus, for each extension s' of uv to the left, we compute the eccentricities $E(i)$ of all its extensions s_i to the right using Lemma 4, in $O(n^2)$ time. Let $N(i)$ be the maximum distance in $\mathcal{N}_\ell \cup s_i$ between pairs of points in $\mathcal{N}_\ell$. Our goal is to compute $\min_i \max\{E(i), N(i)\}$. Since $N(i)$ is a decreasing function as i grows, we do not need to compute $N(i)$ for all values of i, but only for those i for which $E(i)$ is maximal: there is no $j > i$ with $E(j) < E(i)$ (see Figure 5). Therefore, we can look for that minimum by binary search, computing $N(i)$ only for $O(\log n)$ values of i. Using [10], we can update the distances between vertices in quadratic time and then compute $N(i)$ also in quadratic time (the distance between pairs edge–edge and vertex-edge can be computed in constant time knowing the distance between vertices), giving a total time of $O(n^3 \log n)$. ◀

We thus obtain the main result in this section.

▶ **Theorem 6.** *Let ρ be the length of a longest edge of a network $\mathcal{N}_\ell$. Then, it is possible to find a segment s' such that* $\operatorname{diam}(\mathcal{N}_\ell \cup s') \leq \min_{s \in \mathcal{S}} \operatorname{diam}(\mathcal{N}_\ell \cup s) + 4\rho$ *in $O(n^5 \log n)$ time.*

This result immediately gives a simple approximation algorithm: subdivide each edge in $\mathcal{N}_\ell$ by adding dummy vertices such that the largest resulting edge length is ε. Then the previous theorem implies the following result, which is a generalization to general networks of the result for paths presented in [15, Theorem 8.1].

▶ **Corollary 7.** *Let ρ be the length of a longest edge of a network $\mathcal{N}_\ell$. Then, for any $0 < \varepsilon < \rho/2$ it is possible to find a segment s' such that* $\operatorname{diam}(\mathcal{N}_\ell \cup s') \leq \min_{s \in \mathcal{S}} \operatorname{diam}(\mathcal{N}_\ell \cup s) + 4\varepsilon$ *in $O((n\rho/\varepsilon)^5 \log(n\rho))$ time.*

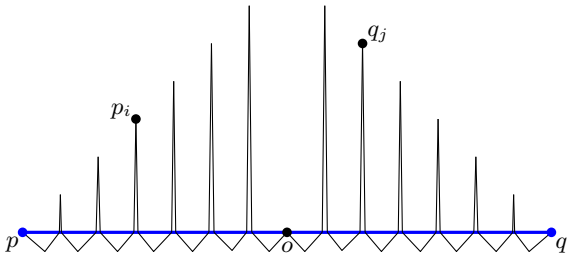

Figure 6 Schematic construction showing that the insertion of a shortcut pq can create $\Theta(n^2)$ diametral pairs. The distance between the top of one spike on the left of o and one on its right, like p_i and q_j, can be made to be $|pq|$, and equal to the diameter of $\mathcal{P}_\ell \cup pq$.

3 Path networks

In the remaining, we focus on networks that are paths. To illustrate the complexity of this seemingly simple setting, we begin by observing that the insertion of a shortcut to a path can create a quadratic number of diametral pairs; as illustrated in the construction in Figure 6. It consists of $\Theta(n)$ spikes placed symmetrically with respect to the midpoint of the shortcut, denoted with o. After inserting pq, each spike forms a face with a cycle of length roughly twice its height. The spikes are spaced by one unit each, while their heights are set such that the distance from o to the top of the spike is always the same, namely $|pq|/2$. In this way, for any two spike tops p_i and q_j on the left and right of o, respectively, the distance between p_i and p_j on $\mathcal{P}_\ell \cup pq$ is always equal to $|pq|$, which is also the diameter of $\mathcal{P}_\ell \cup pq$.

3.1 Diameter after inserting a shortcut

The diameter of $\mathcal{P}_\ell$ can be immediately computed in linear time, however, the addition of a shortcut s can create a linear number of new faces, thus in principle it is not clear whether $\mathrm{diam}(\mathcal{P}_\ell \cup \mathrm{s})$ can be computed in linear time, i.e., without computing the diameter between each pair of faces. The main result in this section is that this is still possible.

Path networks have the nice property that the maximal extension of an optimal shortcut is also optimal [15]. Thus, we can assume that $s = pq$ is maximal and horizontal. The insertion of s splits $\mathcal{P}_\ell$ into polygonal chains, which bound the different faces created. Our goal is to compute the pair of chains that have maximum distance in $\mathcal{P}_\ell \cup s$.

We number the polygonal chains from 0 to m in the order of their left endpoints from left to right along s (using right endpoints to disambiguate). Except for possibly the first and last, all chains have both endpoints on s. For the ith chain C_i, we denote its left and right endpoints by p_i^l and p_i^r, respectively. If the first vertex of $\mathcal{P}_\ell$ is not on s, we consider the path from its first vertex to the first intersection of $\mathcal{P}_\ell$ with s as a degenerate loop chain with equal left and right endpoints on s (analogous for the last vertex of $\mathcal{P}_\ell$). Refer to Figure 7.

Let $|C_i|$ be the length of C_i, let $L_i = |pp_i^l|$ and $R_i = |p_i^r q|$, and let s_i denote the segment $p_i^l p_i^r$. Note that $C_i \cup s_i$ forms a cycle. We use D_i for the distance on $\mathcal{P}_\ell \cup s$ from p_i^l to its furthest point $\bar{p}_i^l$ on $C_i \cup s_i$ (i.e., D_i is the semiperimeter of $C_i \cup s_i$).

We make some basic observations about the diameter between two chains, depending on their relative position. They reveal a key property of the problem: the linear ordering between chains induced by s defines uniquely how the diameter between two chains is achieved.

▶ **Observation 8** (Disjoint chains). *Let C_i, C_j be two chains of $\mathcal{P}_\ell \cup s$ with $s_i \cap s_j = \emptyset$ and s_i to the left of s_j. The diameter of $C_i \cup p_i^l p_j^r \cup C_j$ is $D_i + |p_i^r p_j^l| + D_j = D_i + R_i - R_j - |s_j| + D_j$.*

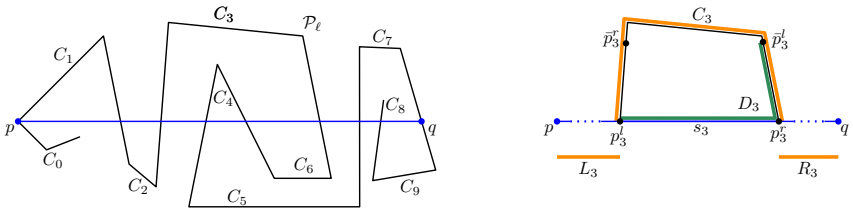

Figure 7 Left: chains created by s; C_0 and C_8 are degenerate chains. Right: detail for chain C_3, showing the cycle formed by $C_3 \cup s_3$. Thick lines are used here to denote distances.

▶ **Observation 9** (Nested chains). *Let C_i, C_j be two chains of $\mathcal{P}_\ell \cup s$ with $s_j \subset s_i$. The diameter of $C_i \cup s_i \cup C_j$ is $\frac{1}{2}(|C_i| + |p_i^l p_j^l| + |p_i^r p_j^r| + |C_j|) = \frac{1}{2}(|C_i| + L_j - L_i + R_j - R_i + |C_j|)$.*

▶ **Observation 10** (Overlapping chains). *Let C_i, C_j be two chains of $\mathcal{P}_\ell \cup s$ with $s_i \cap s_j \neq \emptyset$, $p_i^l \notin s_j$ and $p_j^r \notin s_i$. The diameter of $C_i \cup p_i^l p_j^r \cup C_j$ is $\frac{1}{2}(|C_i| + |p_i^l p_j^l| + |p_i^r p_j^r| + |C_j|) = \frac{1}{2}(|C_i| + L_j - L_i + R_i - R_j + |C_j|)$.*

Note that, while in the case of disjoint chains the diameter is achieved by a unique pair of points, that is not the case of nested and overlapping chains, for which an infinite number of diametral pairs of points may exist. Also, observe that expressions in Obs. 9 and Obs. 10 are the same except for adding up either $R_j - R_i$ or $R_i - R_j$. This difference only exists to differentiate between the two possible orders of the right endpoints of the two chains.

The algorithm for computing $\mathrm{diam}(\mathcal{P}_\ell \cup \mathrm{s})$ in linear time starts by going along $\mathcal{P}_\ell$ and computing all intersections with s in the order of $\mathcal{P}_\ell$. Then we apply a linear-time algorithm for Jordan sorting [11] to obtain the intersections in the order along s, say, from left to right. Within the same running time we can compute C_k and s_k. Next, we sweep the endpoints of the chains along s to compute, for each chain C_k, its furthest chain from the ones seen so far. To that end, certain information is computed and stored:

1. The furthest chain from C_k to the left, given by $\arg\max_{0 \le i < k} \alpha_i$, where $\alpha_i = D_i + R_i$. Similarly, we store the furthest chain to the right.
2. The furthest chain nested inside C_k. This is given by $\arg\max_{j \in N_k} \beta_j$, where $\beta_j = |C_j| + L_j + R_j$ and N_k is the set of indices of all chains nested inside C_k.
3. The furthest chain with one endpoint in C_k, and one outside: given by $\arg\max_{j \in O_k^r} \gamma_j$, where $\gamma_j = |C_j| + L_j - R_j$ and O_k^r is the set of indices of all overlapping chains with their left endpoint inside C_k and their right endpoint outside. Similarly, we store those with their left endpoints outside and the right one inside of s_k.

The computation of the information in (1) is straightforward when sweeping along s, say, from left to right. We just maintain the largest value of α_i seen so far as we sweep. The case of nested or overlapping chains, which is explained next, is more complicated because one needs the maximum restricted to those chains that are contained or overlap with C_k. For that reason, what we will do is to store β_j and γ_j values for *all* chain endpoints. Suppose that C_i starts to the left of C_j (the other case is analogous). We use a data structure for range minimum queries [2,3]. This allows to preprocess an array A in linear time in order to find the maximum value in any subarray $A[a, b]$ in $O(1)$ time. In our context, we need two such data structures. We use arrays A_n and A_o to store the maximum β and γ values defined above for the chains that are nested and overlapping, respectively. Each array has one position for each endpoint of a chain, thus $2m$ in total. The positions are as they appear sorted along s, from left to right. Refer to Figure 8. For a chain C_j, the position corresponding to its left endpoint has a value equal to β_j in the array A_n, and value γ_j in array A_o. The values corresponding to the right endpoints of the chains are not used, i.e., they have value $-\infty$, in both arrays. At each array position, we also store pointers to the corresponding chains.

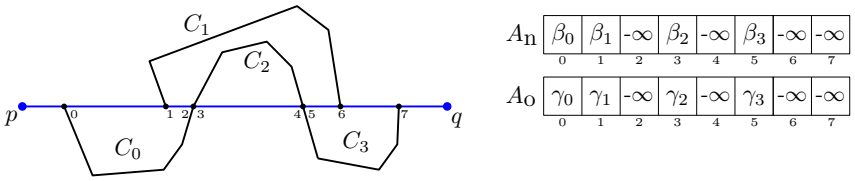

Figure 8 Example of arrays with distances to chains that are nested (A_n) and overlapping (A_o).

To find the nested or overlapping chain furthest from C_i we would like to perform one maximum range query in A_n and one in A_o, in both cases with a subarray corresponding to the interval between the endpoints of C_i. The goal is to use these queries to obtain the furthest chain of each type: nested and overlapping. However, there is an issue. In the way A_n and A_o are defined, the result of a range query cannot distinguish between nested or overlapping chains, it necessarily searches in both sets (i.e., $N_i \cup O_i^r$). Fortunately, the geometry of the problem guarantees that we can still use the result obtained, as we show next. The following lemma shows that if the furthest face is associated to a β_i value, then it must be nested, and similarly, if it is associated to a γ_i value, it must be overlapping.

▶ **Lemma 11.** *Let C_k be a chain with distance to C_i equal to $d^* = \max\{\max_{j\in(N_i\cup O_i^r)} |C_i| - L_i - R_i + \beta_j, \max_{j\in(N_i\cup O_i^r)} |C_i| - L_i + R_i + \gamma_j\}$. Then it holds: (i) if $d^* = |C_i| - L_i - R_i + \beta_k$, then $k \in N_i$; (ii) if $d^* = |C_i| - L_i + R_i + \gamma_k$, then $k \in O_i^r$.*

Therefore, when processing a chain C_k, we perform one maximum range query in A_n and one in A_o, and keep the maximum of those two values. Lemma 11 guarantees that the associated chain is the furthest one that is either nested or overlapping. Proceeding in an analogous way for the chains that are overlapping with one endpoint to the left of C_k, the furthest face from C_k of any of the three types (disjoint, nested, overlapping) can be found in $O(1)$ time, and the maximum distance between two chains can thus be found in linear time.

▶ **Theorem 12.** *For every path $\mathcal{P}_\ell$ with n vertices and a shortcut s, it is possible to compute the diameter of $(\mathcal{P}_\ell \cup s)$ in $\Theta(n)$ time.*

It is worth noting that the ideas used in this section do not extend to networks that are trees, since in that case the structural results in Observations 8–10 do not hold anymore.

3.2 Optimal horizontal shortcuts

In this section we compute an optimal horizontal shortcut for a path considerably faster than using the general method in Section 2. After a suitable rotation, this allows to find an optimal shortcut of any fixed orientation.

Assume as in Section 3.1 that shortcuts are horizontal and maximal, so they can be treated as horizontal lines. Now, consider the vertices in $\mathcal{P}_\ell$ sorted increasingly by y-coordinate, and let y_a, y_b, with $y_a < y_b$, be the y-coordinates of two consecutive vertices in that order. Observations 8–10 are stated in terms of chains, but they also apply to faces. Indeed, they imply that the distance between any two faces f_i and f_j is a linear function $d_{ij}(y)$ for $y_a \le y \le y_b$. Thus, each face f_i is associated with $k-1$ lines in 2D where k is the total number of faces (each line represents the corresponding function $d_{ij}(y)$ for $j \neq i$). Considering all faces, we obtain a set $\mathcal{L}$ of $\Theta(k^2)$ lines (note that $k = O(n)$). The optimal shortcut over all $y \in [y_a, y_b]$ is given by the minimum of the upper envelope of $\mathcal{L}$, which can be computed in $O(k^2 \log k)$ time [9]. If this is done with each of the $n-1$ horizontal strips formed by consecutive vertices of $\mathcal{N}_\ell$, the optimal horizontal shortcut is obtained in total $O(n^3 \log n)$

time. Now, this method can be improved if, instead of computing from scratch the upper envelope of $\mathcal{L}$ at each horizontal strip, we maintain the upper envelope between consecutive strips and only add or remove the lines that change when going from one strip to the next one. The changes between two consecutive strips are of three types: (i) one of the two line segments bounding a face within the strip changes; (ii) a face ends; (iii) a new face appears. In the worst case, $n-1$ lines are removed from $\mathcal{L}$ and another $n-1$ lines are added to $\mathcal{L}$. Maintaining the upper envelope of N lines is equivalent to maintaining the convex hull of N points in 2D, which can be done in amortized $O(\log N)$ time per insert/delete operation with a data structure of size $O(N)$ [4]. Since we have $N = O(n^2)$, we obtain the following:

▶ **Theorem 13.** *For every path $\mathcal{P}_\ell$ with n vertices, it is possible to find an optimal horizontal shortcut in $O(n^2 \log n)$ time, using $O(n^2)$ space.*

3.3 Optimal simple shortcuts

In this section we consider optimal *simple* shortcuts, i.e., we restrict the possible shortcuts to those whose interior does not intersect $\mathcal{N}_\ell$. We show that an optimal simple shortcut can be computed much faster if it exists. Note that one must distinguish between an *optimal simple shortcut* and a *simple optimal shortcut*. The first is a shortcut that is optimal in the set of simple shortcuts; this is different of being optimal in the set of all shortcuts and, in addition, to be simple. Interestingly, it is known that optimal simple shortcuts may not exist, even for paths [16] (e.g., when the only optimal shortcut goes through a vertex, see [12, Figure 12(a)])). It is not clear, however, what the conditions for a network $\mathcal{N}_\ell$ to have an optimal shortcut are, even restricted to simple shortcuts. The following proposition is a first approach to this question (note that its converse is not true, see [12, Figure 12(b)]).

▶ **Proposition 14.** *Let $\mathcal{N}$ be a network whose locus $\mathcal{N}_\ell$ admits a simple shortcut, and let $\overline{\mathcal{N}}$ be the network resulting from adding to $\mathcal{N}$ all edges of the convex hull of $V(\mathcal{N})$. If all faces of $\overline{\mathcal{N}}$ are convex, then $\mathcal{N}_\ell$ has an optimal simple shortcut.*

We now turn our attention to the computation of an optimal simple shortcut if one exists.

Let $s = pq$ be a simple shortcut for a path $\mathcal{P}_\ell$ with endpoints u, v. Suppose that point p is closer to u than q along $\mathcal{P}_\ell$; let $x = d(u, p)$ and $y = d(v, q)$. There is only one bounded face in $\mathcal{P}_\ell \cup s$ whose boundary is a cycle $C(p, q)$. Let $\overline{p}$ and $\overline{q}$ be the farthest points from, respectively, p and q on $C(p, q)$, and let $z = (d_{\mathcal{P}_\ell}(p, q) - |pq|)/2$. Note that $d(\overline{p}, \overline{q}) = |pq|$ and $z = d(p, \overline{q}) = d(\overline{p}, q)$. There are three candidates for diametral path in $\mathcal{P}_\ell \cup s$ (see [6]):

1. The path from u to v via s is diametral if and only if $z = \min\{x, y, z\}$,
2. the path from u to $\overline{p}$ via s is diametral if and only if $y = \min\{x, y, z\}$,
3. the path from v to $\overline{q}$ via s is diametral if and only if $x = \min\{x, y, z\}$.

Thus, $\operatorname{diam}(\mathcal{P}_\ell \cup s) \in \{x + y + |pq|, x + z + |pq|, y + z + |pq|\}$. For the highway model, it was proved in [6] that $\mathcal{P}_\ell$ has an optimal shortcut satisfying $x = y$, which allows to compute it in linear time. In the planar model the situation is more complicated but, in a similar fashion, we can prove the following lemma, which lead to Theorem 16.

▶ **Lemma 15.** *Let pq be an optimal simple shortcut for $\mathcal{P}_\ell$. The following statements hold.*

1. *If neither p nor q are vertices of $\mathcal{P}_\ell$ then $x = y = z$.*
2. *If p or q are vertices of $\mathcal{P}_\ell$ then the two smallest values among x, y, z are equal.*

▶ **Theorem 16.** *It is possible to decide whether a path $\mathcal{P}_\ell$ with n vertices has an optimal simple shortcut and compute one (in case of existence) in $O(n^2)$ time.*

4 Conclusions

In this work, we have presented the first results on the computation of optimal shortcuts for the planar model. We have shown that an optimal shortcut can be computed in polynomial time, and given a discretization of the problem that results in an approximation of the original continuous version. Even though the discretization obtained is too large to be of practical use, it is interesting from a theoretical point of view, and hopefully will be useful to obtain smaller discretizations in the future. We also presented new results for paths, including how to quickly compute the diameter after inserting a shortcut, the computation of an optimal shortcut of fixed orientation, and of an optimal simple shortcut. These are important first steps on a relevant and difficult problem, which leave many intriguing questions open. The existence of small discrete set of segments to approximate an optimal shortcut, or a fast algorithm to find an optimal shortcut for paths (any orientation), are some examples. Finally, the questions studied but for optimal sets of $k > 1$ shortcuts pose challenging open problems.

References

1 S. W. Bae, M. de Berg, O. Cheong, J. Gudmundsson, and C. Levcopoulos. Shortcuts for the Circle. In *Proceedings ISAAC 2017*, pages 9:1–9:13, 2017.

2 M. A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *J. Algorithms*, 57(2):75–94, 2005.

3 O. Berkman and U. Vishkin. Recursive Star-Tree Parallel Data Structure. *SIAM J. Comput.*, 22(2):221–242, 1993.

4 G. S. Brodal and R. Jacob. Dynamic Planar Convex Hull. In *Proceedings FOCS '02*, pages 617–626, Washington, DC, USA, 2002. IEEE Computer Society.

5 J. Cáceres, D. Garijo, A. González, A. Márquez, M. L. Puertas, and P. Ribeiro. Shortcut sets for the locus of plane Euclidean networks. *Appl. Math. Comp.*, 334:192–205, 2018.

6 J. L. De Carufel, C. Grimm, A. Maheshwari, and M. Smid. Minimizing the Continuous Diameter when Augmenting Paths and Cycles with Shortcuts. In *Proceedings SWAT 2016*, pages 27:1–27:14, 2016.

7 J. L. De Carufel, C. Grimm, S. Schirra, and M. Smid. Minimizing the Continuous Diameter when Augmenting a Tree with a Shortcut. In *Algorithms and Data Structures. WADS 2017. LNCS 10389*, pages 301–312, 2017.

8 C. E. Chen and R. S. Garfinkel. The generalized diameter of a graph. *Networks*, 12:335–340, 1982.

9 M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

10 Greg N. Frederickson. Fast Algorithms for Shortest Paths in Planar Graphs, with Applications. *SIAM J. Comput.*, 16(6):1004–1022, December 1987.

11 K. Y. Fung, T. M. Nicholl, R. E. Tarjan, and C. J. Van Wyk. Simplified linear-time jordan sorting and polygon clipping. *Information Processing Letters*, 35(2):85–92, 1990.

12 D. Garijo, A. Márquez, N. Rodríguez, and R. I. Silveira. Computing optimal shortcuts for networks. Preprint, 2018. https://arxiv.org/abs/1807.10093.

13 P. Hansen, M. Labbé, and B. Nicolas. The continuous center set of a network. *Discrete Applied Mathematics*, 30:181–195, 1991.

14 F. Hurtado and C. D. Tóth. Plane Geometric Graph Augmentation: A Generic Perspective. In *Pach J. (eds) Thirty Essays on Geometric Graph Theory*, pages 327–354. Springer, 2013.

15 B. Yang. Euclidean Chains and Their Shortcuts. *Theor. Comput. Sci.*, 497:55–67, July 2013.

16 H. Yang and M. G. H. Bell. Models and algorithms for road network design: a review and some new developments. *Transport Reviews*, 18(3):257–278, 1998.