Feed link placement in Euclidean graphs

Donkers, Huib

Award date:
2017

Feed Link Placement in Euclidean Graphs

Master's Thesis

Huib Donkers
h.t.donkers@student.tue.nl

Supervisors:
Joachim Gudmunsson
joachim.gudmundsson@sydney.edu.au
Mark de Berg
m.t.d.berg@tue.nl

July 21, 2017, Sydney

Abstract

We study the problem of connecting a new vertex p to a Euclidean graph G using k new edges,, called feed links which connect p to the boundary of the plane face f of G containing p. We aim to connect the vertex in such a way that the maximum dilation from p to a set of destination vertices in the graph is minimised, where we define dilation as the ratio between the distance in the graph and Euclidean distance. Previous work mainly focus on reducing the dilation to points on the boundary of f rather than points anywhere in G.

We give two different methods to solve this problem in polynomial time for a fixed k. We show that the problem is NP-hard when k is not fixed by a reduction from set cover. We give three different approximation algorithms that place k feed links such that the maximal dilation is approximated. The first algorithm gives a $(1+\varepsilon)$-approximation for arbitrarily small ε, however, its running time is still exponential in k. The second gives a $(t+1)$-approximation when f is a t-spanner and runs in $O(mn^2 \log n)$ time, where m denotes the number of vertices on the boundary of f and n denotes the total number of vertices in the graph. The third approximation algorithm also has a running time of $O(mn^2 \log n)$ and is obtained after a comparison between dilation and distance in the context of feed link placement. Its approximation ratio depends on the difference in Euclidean distances between p and the destination vertices.

Contents

Chapter 1

Introduction

Geometric networks often represent real-world network such as roads, railways or waterways. Many network analysis methods exist to help solve real-world problems involving these networks. Most of these methods require all important points in the network to be connected to the network. In some cases incomplete data may leave important points disconnected from the the network. This can be the result of omissions in the digitalisation process or lack of regular updates. In other cases one may wish to simplify a network before applying analysis methods. As an example consider the question of where to place a new hospital such that it can be reached within 45 minutes by at least a minimal number of citizens. Instead of calculating distances to a potential location of the new hospital for every individual citizen, one may group citizens together by postal-code. As a location representing this group of citizens, one can choose the centroid of the postal-code region for example. This centroid is not necessarily be connected to the road network.

In order to perform network analysis we need to connect all important points to the network. Different approaches have been proposed. Some simply move the point such that it lies on the network. Moving the point can have undesired consequences, so another strategy is to connect the point to the network by inserting a number of new edges called *feed links* in the network. Aronov et al. [1] introduces *dilation* as a way to verify and quantify how reasonable the resulting connections are and present a number of algorithms placing feed links such that the dilation from the disconnected point to the surrounding edges is minimised. Informally, dilation captures the relative detour between any two points in a network. It is the ratio between the distance in the graph and the crow flight distance.

In this thesis we will also use dilation to quantify how reasonable the connection is. However, where Aronov et al. [1] take the maximum dilation between the disconnected point and points on the surrounding edges, we consider the maximum dilation between the disconnected point and a given set of destination points anywhere in the network. Generally people want a good connection to locations like shopping centres, hospitals and their work. Therefore considering a set of destinations may yield more reasonable connections in certain situations.

More formally, we will consider the problem of connecting a point p to a Euclidean graph G using k feed links such that the dilation between p and a set of destination points D is minimised. In a graph G, dilation between point a

and b is defined as follows:

$$\frac{\text{dist}_G(a,b)}{|ab|}$$

where $|ab|$ denotes the Euclidean distance between a and b and $\text{dist}_G(a,b)$ denotes the length of the shortest path from a to b in the graph G. We may omit G when it is clear from the context which graph is referred to. Specifically we are interested in minimising the maximum dilation between p and any point in D.

Like in Aronov et al. [1], feed links can only connect to a point on the boundary of the face containing p, not to a point elsewhere in the graph. However, feed links may intersect the boundary in order to reach a part of the boundary not "visible" from p.

In this thesis we present two algorithms that give a feed link placement that minimises the maximum dilation to the destination points. The first runs in $O((2mn^2 + mn)^{k-1}mn \log n)$, the second find the optimal solution by solving the decision version of the problem $O(\log n)$ times. For this decision problem we present an algorithm based on Set Cover running in $O((mn)^k kn)$ time, yielding a total running time of $O((mn)^k kn)$ to find the optimal solution. We show that the feed link placement problem is NP-hard by a reduction from Set Cover, and present three approximation algorithms. The first gives an approximate solution with an arbitrarily small approximation ratio, but runs in exponential time. The other two run in polynomial time and give an approximate solution with a ratio depending on two different properties of the graph.

Chapter 2

Analysis

In this chapter we will introduce the exact problem description and some notation and definitions used throughout this thesis. We will also examine how the position of a feed link influences the dilation to individual destination points, which will help us reason about the maximum dilation when placing multiple feed links.

2.1 Notation and problem description

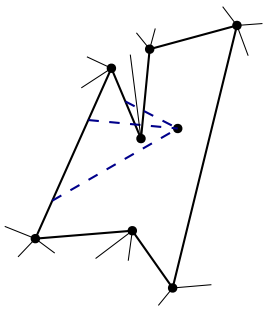
In a Euclidean graph $G = (V, E)$ we define the maximum dilation from $p \in V$ to a set of destination points $D \subseteq V$ as

$$\Delta_p(G, D) = \max_{v \in D} \frac{\text{dist}_G(p, v)}{|pv|}. \tag{2.1}$$

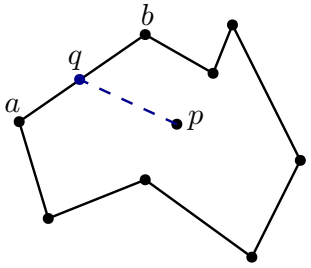
In this thesis we will consider Euclidean graphs where edges may intersect. In a Euclidean graph G we define a *plane face* as follows: G generates a planar subdevision of the plane $\mathcal{A}$. Any face f in $\mathcal{A}$ is a plane face in G if and only if all vertices on the boundary of f are in G.

For our purposes a *feed link* is a single edge connecting a given point p to a point q where p is located in a plane face f of G and q is located on the boundary of the face f. When f is concave, a feed link may intersect with the boundary of f to reach a point on the boundary not visible from p, see Figure 2.1a. Given a Euclidean graph $G = (E, V)$, a point p and a set of feed links F, we denote the graph obtained by inserting the feed links F in G by $G + F$. When all feed links simply connect to vertices of G, we can easily see that $G + F = (V \cup \{p\}, E \cup F)$. However when for some feed link $(p, q) \in F$ the point q is on the interior of an edge (a, b) of f, the graph $G + F$ also contains q as a vertex and the edge (a, b) is replaced by (a, q) and (q, b). See Figure 2.1b.

Definition 1. *Given a Euclidean graph $G = (E, V)$, a point p in a plane face f of G, a set of destination points $D \subseteq V$ and an integer k. The* Minimum Dilation Feed Link Placement *(*MinDFLP*) problem is to find a set F of k feed links connecting p to the boundary of f such that $\Delta_p(G + F, D)$ is minimised.*



(a) Three examples of allowed feed links in a plane face of a graph. The feed links are shown as a dashed line.



(b) When inserting a feed link (p, q) in a graph G we replace (a, b) by two edges (a, q) and (q, b).

Figure 2.1: Feed link mechanics.

Throughout this thesis we will use the following notations and definitions. Firstly n denotes the number of vertices in the graph and m denotes the number of vertices on the boundary of the plane face f containing p. Let $x_0, \ldots, x_{m-1}$ denote (in clockwise order) the vertices on the boundary of f. More generally let x_i denote the $(i-1 \mod m)$-th vertex (in clockwise order) on the boundary of f. Let $P(r)$ denote the point on the boundary, in clockwise direction, at distance r along the boundary of f from x_0 as in Savić and Stojaković [2]. See Figure 2.2. We also define the inverse $P^{-1}(v) = r$ where r is the smallest positive value for which $P(r) = v$.

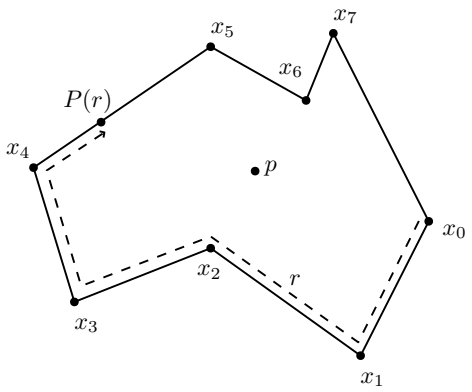


Figure 2.2: $P(r)$ is located on the boundary of f at distance r along the boundary from x_0

2.2 Dilation functions

In this section we investigate the influence of the position of a feed link on the dilation to an individual destination point. We assume we are given a MinDFLP problem instance (G, p, D, k).

For all $v \in D$ and $r \in \mathbb{R}$, let us define the dilation from p to a destination

point $v \in D$ via a feed link $(p, P(r))$ as follows:

$$\delta_v(r) = \frac{|pP(r)| + \text{dist}(P(r), v)}{|pv|}. \tag{2.2}$$

Note that we can now write

$$\Delta_p(G + F, D) = \max_{v \in D} \min_{(p,x) \in F} \delta_v(P^{-1}(x)). \tag{2.3}$$

As described in Savić and Stojaković [2], the function describing the length of the feed link $h(r) = |pP(r)|$ is a combination of hyperbolic functions, see Figure 2.3. Each hyperbolic function $h_i(r)$ corresponds to the length of a feed link connecting to the boundary edge between x_i and x_{i+1}. Let m_i denote the point on the line through x_i and x_{i+1} closest to p. We define $d_i = |pm_i|$ and $c_i = P^{-1}(x_i) + |x_i m_i|$, i.e. we have $P(c_i) = m_i$. We can now express the hyperbolic function $h_i(r)$ as follows:

$$\begin{aligned} h_i(r) &= \sqrt{|P(r)m_i|^2 + |pm_i|^2} \\ &= \sqrt{|P(r)P(c_i)|^2 + d_i^2} \\ &= \sqrt{(r - c_i)^2 + d_i^2}. \end{aligned} \tag{2.4}$$

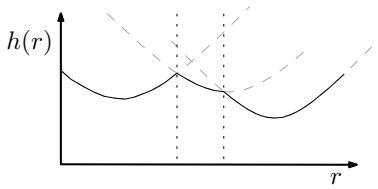


Figure 2.3: The function $h(r)$ is made up of a series of hyperbolic functions.

Similarly, for every $v \in D$ we can describe $\text{dist}(P(r), v)$ as a function of r. We know that $g_v(r) = \text{dist}(P(r), v)$ is a combination of linear functions. Between $P^{-1}(x_i)$ and $P^{-1}(x_{i+1})$ two of these linear functions form a peak. An example is shown in Figure 2.4. These peak shaped functions can be described as follows

$$\begin{aligned} g_{v,i}(r) &= \min\{|P(r)x_i| + \text{dist}(x_i, v), |P(r)x_{i+1}| + \text{dist}(x_{i+1}, v)\} \\ &= \min\{r - P^{-1}(x_i) + \text{dist}(x_i, v), -r + P^{-1}(x_{i+1}) + \text{dist}(x_{i+1}, v)\}. \end{aligned} \tag{2.5}$$

Let us now define for any $v \in D$ and $i \in \mathbb{Z}$, the dilation between p and v using a feed link connecting to $P(r)$ on the edge between x_i and x_{i+1}:

$$\delta_{v,i}(r) = \frac{h_i(r) + g_{v,i}(r)}{|pv|} \tag{2.6}$$

Lemma 1. *For any $i \in \mathbb{Z}$ and $v \in D$, $\delta_{v,i}(r)$ has no local minima other than its endpoints $\delta_v(P^{-1}(x_i))$ and $\delta_v(P^{-1}(x_{i-1}))$.*

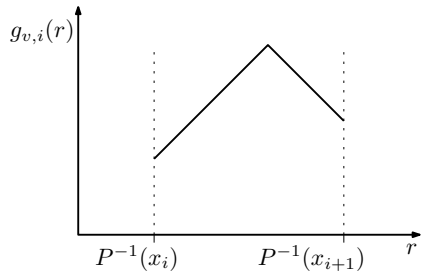


Figure 2.4: An example of what the plot of $g_{v,i}(r)$ can look like.

Proof. Note that $g_{v,i}(r)$ has no local minima other than its endpoints, it consists of one increasing part and one decreasing part. For the increasing part we know $\frac{\mathrm{d}g_{v,i}(r)}{\mathrm{d}r} = 1$ and for the decreasing part $\frac{\mathrm{d}g_{v,i}(r)}{\mathrm{d}r} = -1$. This can be derived from its formula, but it can also easily be seen from its definition $\mathrm{dist}(P(r), v)$: when $P(r)$ moves $\mathrm{d}r$ along the boundary of f, the distance to v increases or decreases by $\mathrm{d}r$.

Note also that for the hyperbolic function $h_i(r)$ we have $-1 \leq \frac{\mathrm{d}h_i(r)}{\mathrm{d}r} \leq 1$. Again this can be derived from the formula, but it is easier to see using the definition $h_i(r) = |pP(r)|$: when $P(r)$ moves $\mathrm{d}r$ along the boundary, $|pP(r)|$ increases or decreases by at most $\mathrm{d}r$.

We can conclude that $h_i(r) + g_{v,i}(r)$ is increasing if and only if $g_{v,i}(r)$ is increasing. And trivially the same applies to $\delta_{v,i}(r)$. Hence we can conclude that the local minima of $\delta_{v,i}(r)$ are in the same place as the local minima of $g_{v,i}(r)$. □

Corollary 1.1. *For any $v \in D$, the function $\delta_v(r)$ has local minima only at $r \in \{P^{-1}(x_0), \ldots, P^{-1}(x_{m-1})\}$.*

Lemma 2. *For any $i \in \mathbb{Z}$ and distinct $u, v \in D$, the functions $\delta_{u,i}(r)$ and $\delta_{v,i}(r)$ intersect at most twice.*

Proof. As previous observed, $\delta_{u,i}(r)$ and $\delta_{v,i}(r)$ both consist of an increasing part followed by a decreasing part. We will first show that the two increasing parts intersect at most once and that the two decreasing parts intersect at most once. It is easy to see that a decreasing part and an increasing part intersect at most once. From this we will conclude that $\delta_{u,i}(r)$ and $\delta_{v,i}(r)$ intersect at most three times. Finally we will prove that $\delta_{u,i}(r)$ and $\delta_{v,i}(r)$ cannot intersect three times.

We choose r_u such that $\delta_{u,i}(r)$ is increasing for $r < r_u$ and decreasing for $r > r_u$ and similarly, we pick r_v such that $\delta_{v,i}(r)$ is increasing for $r < r_v$ and decreasing for $r > r_v$, see Figure 2.5. For $r < r_u$ we know $\frac{\mathrm{d}g_{u,i}(r)}{\mathrm{d}r} = 1$ and for $r > r_u$ we know $\frac{\mathrm{d}g_{u,i}(r)}{\mathrm{d}r} = -1$. We derive

$$\begin{aligned} \frac{\mathrm{d}\delta_{u,i}(r)}{dr} &= \frac{1}{|pu|}\left(\frac{\mathrm{d}h_i(r) + g_{u,i}(r)}{\mathrm{d}r}\right) \\ &= \begin{cases} \frac{1}{|pu|}\left(\frac{\mathrm{d}h_i(r)}{\mathrm{d}r} + 1\right) & \text{for } r < r_u \\ \frac{1}{|pu|}\left(\frac{\mathrm{d}h_i(r)}{\mathrm{d}r} - 1\right) & \text{for } r > r_u \end{cases} \end{aligned} \tag{2.7}$$

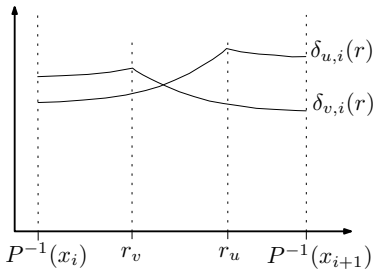


Figure 2.5: We choose r_u where $\delta_{u,i}(r)$ changes from increasing to decreasing, and we choose r_v where $\delta_{v,i}(r)$ changes from increasing to decreasing.

and similarly

$$\frac{\mathrm{d}\delta_{v,i}(r)}{dr} = \begin{cases} \frac{1}{|pv|}\left(\frac{\mathrm{d}h_i(r)}{\mathrm{d}r} + 1\right) & \text{for } r < r_v \\ \frac{1}{|pv|}\left(\frac{\mathrm{d}h_i(r)}{\mathrm{d}r} - 1\right) & \text{for } r > r_v \end{cases} \tag{2.8}$$

Without loss of generality assume $|pu| < |pv|$ meaning $\frac{1}{|pu|} > \frac{1}{|pv|}$. Hence for $r < \min\{r_u, r_v\}$ we know $\delta_{u,i}(r)$ is increasing faster than $\delta_{v,i}(r)$, so they can intersect at most once. Similarly for $r > \max\{r_u, r_v\}$ we know $\delta_{u,i}$ is decreasing faster than $\delta_{v,i}(r)$ so there can be at most one intersection for $r > \max\{r_u, r_v\}$.

As noted before, there can be at most one intersection for $\min\{r_u, r_v\} < r < \max\{r_u, r_v\}$ since one function is increasing and the other decreasing. So we know $\delta_{u,i}(r)$ and $\delta_{v,i}(r)$ intersect at most three times in total. We now prove by contradiction that they cannot intersect three times.

Assume $\delta_{u,i}(r)$ and $\delta_{v,i}(r)$ intersect three times. Under our previous assumption that $|pu| < |pv|$, we know that for the leftmost intersection $\delta_{u,i}(r)$ increases faster than $\delta_{v,i}(r)$. So between the leftmost and the middle intersection we know $\delta_{u,i}(r) > \delta_{v,i}(r)$, and therefore between the middle and rightmost intersection we know $\delta_{u,i}(r) < \delta_{v,i}(r)$. This means that for the rightmost intersection $\delta_{v,i}(r)$ decreases faster than $\delta_{u,i}(r)$. This is a contradiction with our earlier assumption that $|pu| < |pv|$. □

Corollary 2.1. *For all $u, v \in D$, $\delta_u(r)$ and $\delta_v(r)$ intersect at most $2m$ times.*

2.3 A general purpose initialisation algorithm

We will assume that the graph is given as a doubly connected edge list, but any data structure supporting efficient shortest path finding is sufficient. Let us also assume that the data structure enables us to find the boundary vertices $x_0, \ldots, x_{m-1}$ of f in linear time.

To be able to efficiently make use of the dilation functions $\delta_v(r)$ we precompute all parameters used to describe the functions. Of course $|pv|$ can be computed in constant time, so precomputing this does not change the running time. Similarly, for all $0 \leq i < m$ we can compute c_i, d_i and $P^{-1}(x_i)$ in constant time too. We are left with computing $\mathrm{dist}(x_i, v)$ for all i and $v \in D$. Running Dijkstra's algorithm [3], for all $0 \leq i < m$, gives us $\mathrm{dist}(x_i, v)$ for all $v \in D$ in $O(mn^2 \log n)$ time.

After this initialisation, we can find all intersections between $\delta_u(r)$ and $\delta_v(r)$ for all distinct $u, v \in D$ in $O(m)$ time by solving the equation $\delta_u(r) = \delta_v(r)$, that is, for all $0 \leq i < m$

$$\frac{h_i(r) + g_{u,i}(r)}{|pu|} = \frac{h_i(r) + g_{v,i}(r)}{|pv|}.$$

2.4 A visualisation

To gain more insight into the problem, we made a java application. In this this application we can create problem instances of MinDFLP. These problem instances can be saved as a simple text file. The main reason for this application is to get a better understanding of how the position of feed link influences dilation and shortest paths to destination points. One should see it as a visualisation of the problem and its properties rather than a solver, we did not implement any of the algorithm presented in this thesis other than the initialisation from Section 2.3: The application finds the boundary of the plane face containing p and runs Dijkstra's algorithm for every boundary vertex. Only after this initialisation can we show the shortest paths and dilation functions.

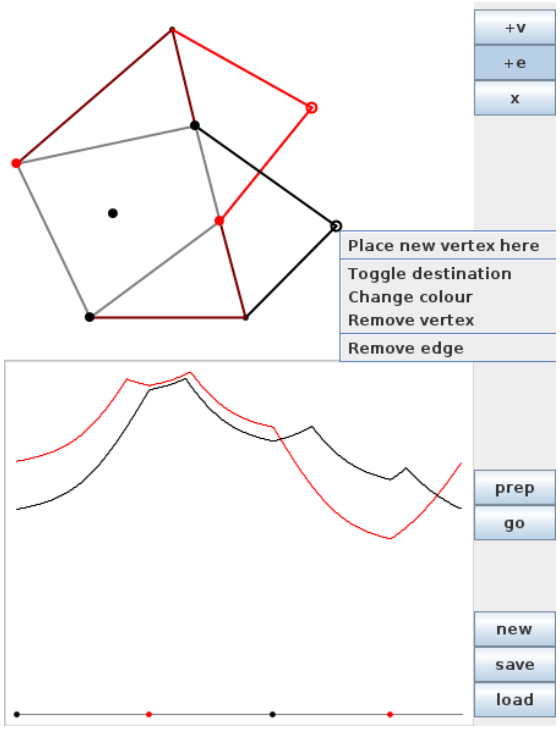


Figure 2.6: The java application. In the top half of the screen one can view and edge the graph, in the bottom half of the screen one can see the resulting dilation functions for each of the vertices marked as destination point.

The visualisation did not reveal any big surprises to us, but it gave a more realistic view on the shape of the dilation functions. Eventually the main role of the application was to quickly verify counter examples to some unproven claims.

Chapter 3

A first exact algorithm

In this chapter we present an algorithm solving MinDFLP exactly. We first give an algorithm placing one single feed link, before we give a general algorithm that works for any k. Both algorithms work by identifying a bounded number of points for a feed link to connect to.

3.1 Placing a single feed link

Let us first consider the case where $k = 1$: we place one single feed link. We can rewrite Equation 2.3 as

$$\Delta_p(G + \{(p, x)\}, D) = \max_{v \in D} \delta_v(P^{-1}(x)). \tag{3.1}$$

We can easily see that finding r such that $\max_{v \in D} \delta_v(r)$ is minimised, directly gives us $\{(p, P^{-1}(r))\}$ as an optimal solution.

Lemma 3. *$\max_{v \in D} \delta_v(r)$ has $O(mn)$ local minima, which can be found in $O(mn \log n)$ time when all parameters defining $\delta_v(r)$ for all $v \in D$ are known.*

Proof. The local minima of $\max_{v \in D} \delta_v(r)$ are only found where for some $v \in D$ the function $\delta_v(r)$ has a local minimum, or at an intersection of $\delta_u(r)$ and $\delta_v(r)$ for some distinct $u, v \in D$. By Lemma 1, for any $v \in D$, the function $\delta_v(r)$ can only have local minima at $r \in \{P^{-1}(x_0), \ldots, P^{-1}(x_{m-1})\}$. By Lemma 2 we have for every i, $0 \leq i < m$ there are at most two intersections between $\delta_u(r)$ and $\delta_v(r)$ between $P^{-1}(x_i)$ and $P^{-1}(x_{i+1})$ for any distinct $u, v \in D$. We are interested in the intersections that are part of the upper envelope of the functions $\delta_v(r)$ for all $v \in D$. By Theorem 1 from Davenport and Schinzel [4] there are $O(n)$ such intersections. Hence we have a total of $m + m \cdot O(n) = O(mn)$ local minima. After initialisation of $O(mn^2 \log n)$ time as described in Section 2.3, these intersections of the upper envelope can be found in $O(mn \log mn) = O(mn \log n)$ time using a simple divide and conquer algorithm as described in Computational Geometry: Algorithms and Applications [5]. □

By inspecting all local minima of $\max_{v \in D} \delta_v(r)$ we can determine the absolute minimum, yielding a value r that minimises $\max_{v \in D} \delta_v(r)$. As observed

earlier, this directly gives us an optimal solution. By Lemma 3 we know this can be done in $O(mn \log n)$ time. We obtain the following Theorem:

Theorem 1. *There exists an algorithm solving* MinDFLP *in* $O(mn \log n)$ *time for* $k = 1$ *after initialisation of* $O(mn^2 \log n)$ *time.*

3.2 Candidate set

To solve the problem for $k > 1$ we generalise the idea of the previous section. We identified the local minima of the upper envelope as a bounded set of feed links for which we knew at least one must be optimal. As a generalisation of this we define a *candidate set* in which each element represents a point (r, d) in the plot of the functions $\delta_v(r)$ for all $v \in D$. The value for r directly gives the feed link $(p, P(r))$.

Definition 2. *Let* (G, p, D, k) *be an instance of the* MinDFLP *problem. A set* $C \subseteq \mathbb{R}^2$ *is said to be a* candidate set *when there exists a subset* $S \subseteq C$ *such that*

- $F = \{(p, P(r)) | (r, d) \in S\}$ *is an optimal solution to the* MinDFLP *problem instance, and*
- *in the graph* $G + F$ *the following holds: For all* $(r, d) \in S$ *we have* $d = \max_{v \in D_r} \delta_v(r)$ *where* D_r *denotes the set of all* $v \in D$ *for which the shortest path from* p *to* v *includes* $P(r)$*.*

An element of a candidate set can be seen as a point in the plot of the dilation functions, and we will call them *candidate points*, see Figure 3.1. We will now show that the set of all intersections of the dilation functions combined with all intersections of the dilation functions with vertical lines for any r associated with a boundary vertex.

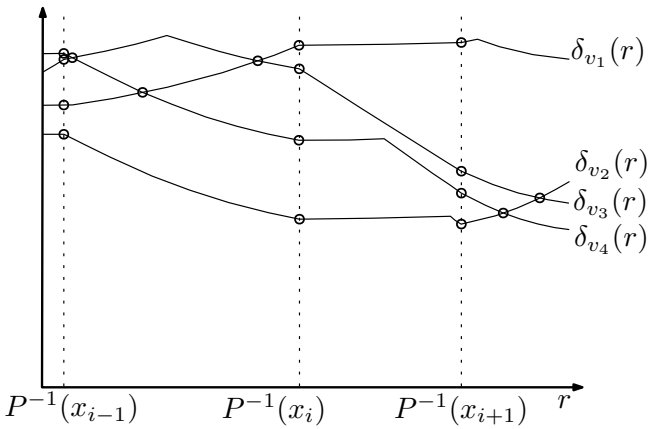


Figure 3.1: Part of the plot of the dilation functions of four destination points. Candidate points are shown as circles.

Lemma 4. *The set* $X = \{(r, d) | \exists_{u,v \in D} \delta_u(r) = \delta_v(r) = d \wedge u \neq v\} \cup \{(P^{-1}(x_i), d) | 0 \leq i < m \wedge \exists_{v \in D} \delta_v(P^{-1}(x_i)) = d\}$ *is a candidate set.*

Proof. Suppose F is an optimal solution to the MinDFLP problem with maximum dilation OPT. For each feed link e in F we define $D_e \subseteq D$ as the set of destination points for which the shortest path from p to $v \in D_e$ includes e. We will prove the lemma by constructing a set $S \subseteq X$ as in Definition 2.

Initialise $S \leftarrow X \cap \{(r,d) | (p, P(r)) \in F \wedge d = \max_{v \in D_e} \delta_v(r)\}$, i.e. S contains all candidate points that relate to a feed link in F.

Consider all feed links in F not related to a candidate point in X: let $F' = F \setminus \{(p, P^{-1}(r)) | (r,d) \in X\}$. For all feed links $e = (p,x) \in F'$ there exists exactly one $v \in D_e$ for which $\delta_v(P^{-1}(x)) = \max_{u \in D_e} \delta_u(P^{-1}(x))$. Note that if there were two then $(P^{-1}(x), \delta_v(P^{-1}(x)))$ would be a candidate point in S. Since $v \in D_e$ we know

$$\delta_v(P^{-1}(x)) \leq OPT. \tag{3.2}$$

Let $(r_a, d_a), (r_b, d_b) \in X$ be two candidate points with $\delta_v(r_a) = d_a$ and $\delta_v(r_b) = d_b$ such that r_a is maximal but smaller than $P^{-1}(x)$ and r_b is minimal but larger than $P^{-1}(x)$, see Figure 3.2. We will assume that $\delta_v(r_a) \leq \delta_v(r_b)$, the proof for the opposite is symmetric. We add (r_a, d_a) to S.

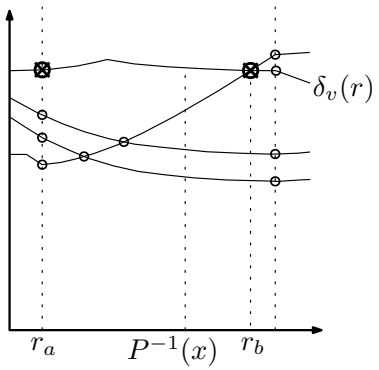


Figure 3.2: Candidate points (r_a, d_a) and (r_b, d_b) are indicated with a $\otimes$.

Note that by definition of r_a and r_b there is no $u \in D$ such that $\delta_u(r) = \delta_v(r)$ for any $r_a < r < r_b$. Since we know that all dilation functions are continuous we can conclude that for any $u \in D$ for which $\delta_u(P^{-1}(x)) < \delta_v(P^{-1}(x))$ the following holds

$$\delta_u(r_a) \leq \delta_v(r_a). \tag{3.3}$$

Note also that by Corollary 1.1 we know that $\delta_v(r)$ can only have local minima for $r \in \{P^{-1}(x_0), \ldots, P^{-1}(x_{m-1})\} \subseteq X$. It follows that $\delta_v(r)$ cannot have a local minimum for $r_a < r < r_b$ and we obtain

$$\delta_v(r_a) \leq \delta_v(P^{-1}(x)), \tag{3.4}$$

because if this does not hold we have that $\delta_v(P^{-1}(x))$ is smaller than both $\delta_v(r_a)$ and $\delta_v(r_b)$ meaning $\delta_v(r)$ must have a local minimum for $r_a < r < r_b$, contradicting our earlier observation.

Equations 3.2, 3.3 and 3.4 yield that for all $u \in D_e$ we have

$$\delta_u(r_a) \leq \delta_v(r_a) \leq OPT \tag{3.5}$$

and since $d_a = \delta_v(r_a)$, we know that $d_a = \max_{u \in D_e} \delta_u(r_a)$.

After considering every feed link in F' in this manner we have generated our final set S of size k. It follows from the discussion above that the set of feed links $\{(p, P(r))|(r, d) \in S\}$ is an optimal solution, and we also know that in graph $G + \{(p, P(r))|(r, d) \in S\}$ we have $d = \max_{v \in D_e} \delta_v(r)$ for all $e = (r, d) \in S$. □

Lemma 5. *The candidate set X from Lemma 4 contains at most $2mn^2 + mn$ points and can be found in $O(mn^2)$ time after the initialisation described in Section 2.3.*

Proof. We prove this in two parts, first we give a bound on the size of the set, then we prove the running time.

By Lemma 2 $\delta_{u,i}(r)$ and $\delta_{v,i}(r)$ intersect at most twice for any distinct $u, v \in D$ and $0 \leq i \leq m$. So $\delta_u(r)$ and $\delta_v(r)$ intersect at most $2m$ times. This gives us a total of $2m \cdot |D|^2$ intersections. We know that $|D| \leq n$ and trivially $|\{(P^{-1}(x_i), d)|0 \leq i < m \wedge \exists_{v \in D} \delta_v(P^{-1}(x_i)) = d\}| \leq mn$ so we can conclude X has size at most $2mn^2 + mn$.

We know that after the initialisation described in Section 2.3 we can find all intersections of $\delta_u(r)$ and $\delta_v(r)$ for all distinct $u, v \in D$ and in $O(m)$ time. Since there are $O(|D|^2) = O(n^2)$ pairs of $u, v \in D$ we need $O(mn^2)$ time to find all intersections. The other mn points can trivially be obtained in $O(mn)$ time by evaluating $\delta_v(r)$ for all $v \in D$ and $r \in \{P^{-1}(x_0), \ldots, P^{-1}(x_{m-1})\}$. Hence in total we spend $O(mn^2)$ time after initialisation. □

With a small candidate set X that can be found quickly, we can perform an exhaustive search to find an optimal solution: we try all $|X|^k$ subsets $F' \subseteq X$ of size k and calculate in $O(kn)$ time the value of the solution. This naive algorithm takes $O((2mn^2 + mn)^k kn)$. We can improve on this running time by applying the algorithm from Theorem 1 to pick the last feed link. To do this we need to find which destination points will need to use the final feed link. This can easily be done by using the second value of the candidate points (r, d): we know for all $v \in \delta_v(r) > d$ that the shortest path from p to v uses a different feed link. This way we find all $v \in D$ that need to use the last feed link.

Theorem 2. *Algorithm 1 solves* MinDFLP *in $O((2mn^2 + mn)^{k-1} mn \log n)$ time.*

Proof. Let us first argue correctness. We know from the definition of a candidate set that there exists a set $S' \subseteq X$ such that

- $F = \{(p, P(r))|(r, d) \in S'\}$ is an optimal solution, and
- in the graph $G + F$ the following holds: For all $(r, d) \in S$ we have $d = \max_{v \in D_r} \delta_v(r)$ where D_r denotes the set of all $v \in D$ for which the shortest path from p to v includes $P(r)$.

Let OPT denote the optimal value. It is easy to see that for all $(r, d) \in S'$ we have $d \leq OPT$. Algorithm 1 examines all subsets $S \subseteq X$ of size $k-1$, therefore we will at some point in our algorithm deal with the case where $S \subseteq S'$. We know there must exist an optimal solution that includes feed links $\{(p, P(r))|(r, d) \in S\}$ and has one additional feed link that is included in the shortest path from p to any $v \in Z$, where Z denotes the set of destination points $v \in D$ for which $\delta_v(r) > d$ for all $(r, d) \in S$. Observe that for all other $v \in D \setminus Z$ there exists

a $(r, d) \in S$ such that $\delta_v(r) \leq d \leq OPT$, so for the final feed link it suffices to optimise for only the destination points in Z. Hence choosing the final feed link reduces to a MinDFLP problem instance with $k = 1$. After solving this, we have an optimal solution, which will be stored in S^* and eventually returned.

We will now prove the running time. Lines 2, 3, 4 and 5 take $O(mn^2 \log n)$, $O(mn^2)$, $O(1)$ and $O(1)$ time respectively. The loop on line 6 is executed $|X|^{k-1}$ times. Line 7 iterates over all $v \in D$ and $(r, d) \in S$ taking $O(kn)$ time. Line 8 takes $O(mn \log n)$ time as per Theorem 1, and line 9 takes constant time. Calculating $\max_{(r,d)\in S} d$ and $\{(p, P(r))|(r, d) \in S\}$ takes $O(k)$ time, so lines $10 - 12$ take $O(k)$ time. Hence the body of the loop from line 6 takes $O(kn) + O(mn \log n) + O(1) + O(k) = O(mn^2 \log n)$ time in total. We can conclude the complete algorithm runs in $O((2mn^2 + mn)^{k-1} mn \log n)$ time. □

Algorithm 1 A first algorithm solving MinDFLP

Input: Graph G with a face described by vertices $\langle x_0, \ldots, x_{m-1}\rangle$, point p, destination points D, integer k.

Output: A set F of k feed links such that $\Delta_p(G + F, D)$ is minimised.

1: **function** ExhaustiveSearch(G,p,D,k, $\langle x_0, \ldots, x_{m-1}\rangle$)
2: Initialise ▷ As described in Section 2.3
3: $X \leftarrow$ find candidate set ▷ As described in Lemma 4 and 5
4: $F^* \leftarrow \emptyset$
5: $OPT^* \leftarrow \infty$
6: **for all** $S \subseteq X$ **with** $|S| = k - 1$ **do**
7: $Z \leftarrow \{v \in D|\forall_{(r,d)\in S} \delta_v(r) > d\}$
8: $(p, x) \leftarrow$ solve $(G, p, Z, 1)$ with the algorithm from Theorem 1
9: $S \leftarrow S \cup \{(P^{-1}(x), \Delta_p(G + \{(p, x)\}, Z))\}$
10: **if** $\max_{(r,d)\in S} d < OPT$ **then**
11: $F^* \leftarrow \{(p, P(r))|(r, d) \in S\}$
12: $OPT^* \leftarrow \max_{(r,d)\in S'} d$
13: **return** F^*

Chapter 4

Decision problem

4.1 Intervals

Let us consider the decision version of the MinDFLP problem, i.e. does a feed link placement exist such that the maximum dilation is at most some given threshold α. We will refer to the decision version as DilFLP. An instance of the decision problem is given as (G, p, D, k, α).

For every $v \in D$ we can identify which values of r give us $\delta_v(r) \leq \alpha$. Since $\delta_v(r)$ has at most m local minima (Lemma 1) there are at most m ranges of r for which $\delta_v(r) \leq \alpha$. We will call these ranges *intervals*, and we denote them as (r_a, r_b, v) where r_a, r_b are the extremes of the range, and v is the associated destination point. So for every interval (r_a, r_b, v) we have $\delta_v(r_a) = \delta_v(r_b) = \alpha$ and for all $r_a \leq r \leq r_b$ we have $\delta_v(r) \leq \alpha$. Let I denote the set of all intervals for all destination points. We say a feed link (p, x) *covers* a destination point $v \in D$ if and only if there exists an interval $(r_a, r_b, v) \in I$ with $r_a < P^{-1}(x) < r_b$, i.e. there is a path from p to any of these destination points with dilation at most α.

Lemma 6. *The set of I has size at most mn and can be computed in $O(mn^2 \log n)$ time.*

Proof. By Corollary 1.1 we know that $\delta_v(r)$ has m local minima, hence for any α there can be at most m intervals where $\delta_v(r) < \alpha$. Since we have $O(n)$ destination points, the total number of intervals is $O(mn)$.

In order to find all intervals, we need to determine, for every $v \in D$, which values for r yield that $\delta_v(r) \leq \alpha$. This can be done in linear time by solving the equation $\delta_v(r) = \alpha$ after the initialisation step described in Section 2.3. This directly gives us all intervals in $O(mn^2 \log n + mn) = O(mn^2 \log n)$ time. □

4.2 Set cover

Using the set I of all intervals, as described in Section 4.1, we can determine for each value of r the set of destination points that have dilation at most α when inserting the feed link $(p, P^{-1}(r))$. In fact, we can divide the boundary in non overlapping regions, each associated with a set of destinations points C that have dilation at most α when inserting a feed link from p to any point on the segment. We will denote these segments as (r_a, r_b, C), where $P(r_a)$ and $P(r_b)$

are the end points of the segment. We will refer to C as the *covering set* of the segment.

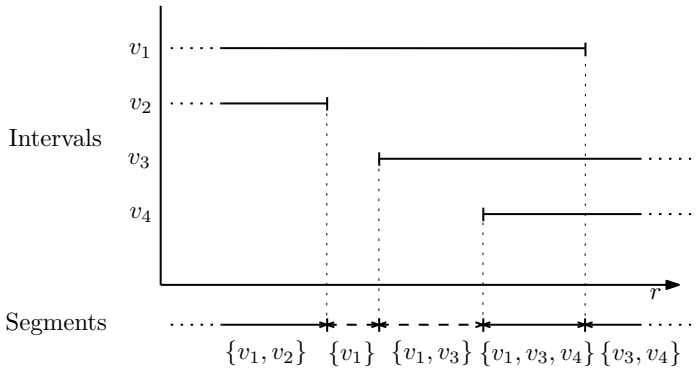


Figure 4.1: The relation between intervals of individual destination points and segments with their covering sets. In this figure we can also observe that any segment that ends at the startpoint of an interval has a covering set that is a strict subset of the covering set of the next segment.

If we find k such segments for which the union of their covering sets include all destination points, we can directly conclude that a solution to the DILFLP problem instance exists, namely k feed links connecting p with an arbitrary point on each of the k segments. So after finding all segments with their cover sets, the problem is reduced to a set cover problem. We can find all these segments in linear time after sorting using a simple scan over the start and endpoints of the intervals. Note that for two segments (r_a, r_b, C) and (r'_a, r'_b, C') such that $C' \subseteq C$, we can simply ignore the second segment, since connecting p to a point on the first segment will also cover all destination points in C'. Using this observation the can easily be adapted to report a smaller set of segments. This algorithm is described in Algorithm 2.

Algorithm 2 Algorithm to find all relevant segments with their covering set

```
1:  function FindSegments(G,p,D,k,α)
2:      I ← set of intervals                          ▷ As described in Section 4.1
3:      E ← {(r_a, v, startpoint)|(r_a, r_b, v) ∈ I} ∪ {(r_b, v, endpoint)|(r_a, r_b, v) ∈
    I}
4:      Sort E
5:      C ← {v ∈ D|δ_v(0) ≤ α}
6:      r ← 0
7:      for all (r', v, t) ∈ E in increasing order do
8:          if t = endpoint then
9:              Report segment (r, r', C)
10:             C ← C \ {v}
11:         else
12:             C ← C ∪ {v}
13:         r ← r'
```

Lemma 7. *Algorithm 2 runs in $O(mn^2 \log n)$ time and finds at most mn segments such that there exists a set of k segments such the union of their covering sets includes all destination points in D if and only if there exists a feed link placement F such that $\Delta_p(G + F, D) \leq \alpha$.*

Proof. We know from Lemma 6 that $|I| \leq mn$, and since there are only $|I|$ *endpoint*-events in E, we can easily see that the algorithm reports at most mn segments. We can also see that segments are not reported for *startpoint*-events. In this case we know that the covering set of the next segment will include all elements of the covering set of the current segment, meaning it can be ignored.

From Lemma 6 we also know that line 2 takes $O(mn^2 \log n)$ time. It is easy to see that line 3 takes $O(|I|) = O(mn)$ time. Sorting E takes $O(|E| \log |E|) = O(mn \log mn)$ time, and since $m < n$ we know $O(mn \log mn) = O(mn \log n)$. Initialisation at line 5 and 6 take $O(n)$ time. The body of the for loop (lines 8-13) takes constant time, and since the body is executed $|E|$ times we obtain a total running time of $O(mn^2 \log n) + O(mn \log n) + O(n) + |E| \cdot O(1) = O(mn^2 \log n)$. □

We can now solve the set cover problem: given mn subsets of D, find k subsets $C_1, \ldots, C_k$ such that $C_1 \cup \ldots \cup C_k = D$. This is a set cover problem and can be solved naively in $O((mn)^k kn)$ time, simply by examining every possible combination of k subsets and checking in $O(kn)$ time whether their union equals D. We obtain the following theorem:

Theorem 3. DILFLP *can be solved in $O((mn)^k kn)$ time.*

4.3 Finding an optimal solution using the decision problem

Given an algorithm for the decision problem, we can solve the optimisation problem by executing this algorithm $O(\log n)$ times. First, recall from Section 3.2 that we can find a candidate set X of size at most $2mn^2 + mn$. By definition there exists a subset $S \subseteq X$ such that $F = \{(p, P(r)) | (r, d) \in S\}$ is an optimal solution for MINDFLP. The value of this optimal solution can directly be deduced from S:

$$\Delta_p(G + F, D) = \max_{(r,d) \in S} d \tag{4.1}$$

It follows that there must be an element $(r, d) \in X$ such that d is the value of the optimal solution. We can use the algorithm for DILFLP to check the existence of a solution for all $\alpha \in \{d | (r, d) \in X\}$, meaning we run this algorithm $|X|$ times to find the smallest value for α for which a solution exists. It is easy to see that we can also apply binary search to find this value for α, meaning only $O(\log mn^2) = O(\log n)$ executions of the algorithm are required. We obtain the following theorem:

Theorem 4. *Given an algorithm solving* DILFLP *with running time $O(f(k, m, n))$, there exists an algorithm solving* MINDFLP *with a running time of $O(mn^2 \log n + f(k, m, n) \cdot \log n)$.*

Using Theorem 3 and Theorem 4 we find a new algorithm solving MINDFLP. This improves our earlier bound from Theorem 2:

Theorem 5. *There exists a problem solving* MinDFLP *running in* $O((mn)^k kn \log n)$ *time.*

When an algorithm for DilFLP does not give an exact solution, but a ρ-approximation, this binary search method will find a solution with a value between OPT and $\rho \cdot OPT$, since a ρ-approximation algorithm for DilFLP will give solutions for any $\alpha \geq \rho \cdot OPT$ and will always return 'no' for $\alpha < OPT$. This means that an approximation algorithm for DilFLP can be used for an approximation algorithm for MinDFLP.

Theorem 6. *Given a* ρ*-approximation algorithm for* DilFLP *with running time* $O(f(k, m, n))$*, there exists a* ρ*-approximation algorithm solving* MinDFLP *in* $O(mn^2 \log n + f(k, m, n) \log n)$ *time.*

Chapter 5

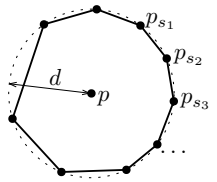
NP-hardness

The algorithms presented so far all run in exponential time suggesting the problem might be NP-hard. We will now show that this is indeed the case. We reduce an arbitrary instance of the Set Cover problem to an instance of the DilFLP problem.

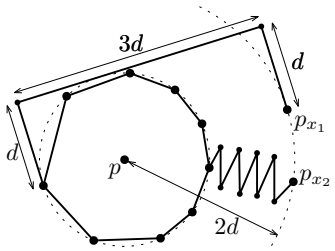
Theorem 7. DilFLP *is NP-hard*

Proof. Let $(S,\mathcal{U},k)$ be an instance of the Set Cover problem, i.e. S is a family of subsets of $\mathcal{U}$ and we need to find a subset $S^* \subseteq S$ of size k such that $\cup_{s\in S^*} s = \mathcal{U}$.

We will now construct a graph G, starting with a point p. For each $s \in S$ we add a vertex p_s at distance d from p and insert edges between them, forming a cycle around p, see Figure 5.1a. For each $x \in \mathcal{U}$ we place a vertex p_x at distance $2d$ from p. For each $s \in S$ and $x \in s$ we connect p_s with p_x using a path of length $5d$, this can always be done without intersecting the face containing p, see Figure 5.1b.



(a) For $s_1, s_2, \ldots \in S$, vertices $p_{s_1}, p_{s_2}, \ldots$ are placed at distance d from p forming a cycle around p.



(b) For all $x_1, x_2, \ldots \in \mathcal{U}$, vertices $p_{x_1}, p_{x_2}, \ldots$ are placed at distance $2d$ from p. A path of length $5d$ from some p_s, $s \in S$, to some p_x, $x \in \mathcal{U}$, can always be created without intersecting the plane face containing p.

Figure 5.1: Construction of the graph G.

Now if the DilFLP problem instance $(G, p, \{p_x | x \in \mathcal{U}\}, k, 3)$ is a 'yes'-instance, there exists a set of k feed links such that the dilation from p to

any p_x (for $x \in \mathcal{U}$) is at most 3. The only way to achieve this is by connecting p to a vertex p_s on the boundary of the face from which there is a path of length $5d$ to p_x. If there are k such vertices p_s, then this directly gives us k sets $s \in S$ such that their union equals $\mathcal{U}$.

If on the other hand $(G, p, \{p_x | x \in \mathcal{U}\}, k, 3)$ is a 'no'-instance, there does not exist a feed link placement such that the dilation from p to p_x is at most 3, for all $x \in \mathcal{U}$, it directly follows that there does not exist a set of k vertices p_s (with $s \in S$) on the boundary that have a path of no longer than $5d$. Therefore there does not exist a set of k sets $s \in S$ such that their union equals $\mathcal{U}$.

Construction of the graph can be done in polynomial time and converting the solution of the DilFLP problem into a solution of the Set Cover problem is also done in polynomial time. Therefore if DilFLP can be solved in polynomial time then so can Set Cover. We know that Set Cover is NP-hard [6], so we conclude that, unless P=NP, DilFLP cannot be solved in polynomial and is therefore also NP-hard. □

Chapter 6

Approximation algorithms

Because the problem is NP-hard and finding exact solutions costs can be slow, we will present a number of approximation algorithms that require less time to execute.

6.1 Discretisation gives a (1+ε)-approximation

Let us consider a problem instance (G, p, D, k) again, where p lies in the face f of G. To obtain a $(1+\varepsilon)$-approximation we discretise the problem as follows: insert Steiner points on each edge of f, separated by at most $\frac{1}{4}\varepsilon l$ where l is the length of the edge. For each edge we need at most $4/\varepsilon$ Steiner points, so in total we need $4m/\varepsilon$ Steiner points to cover the whole boundary of f. See Figure 6.1.

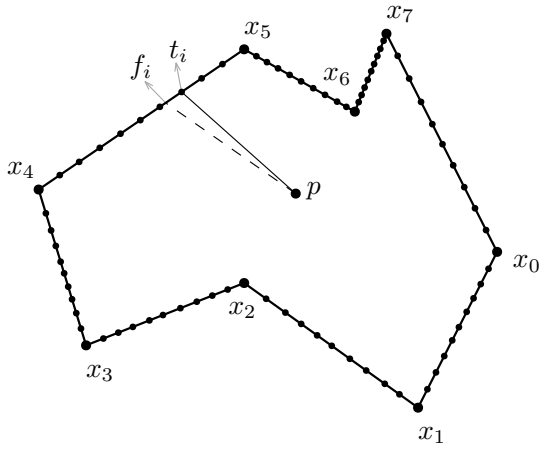


Figure 6.1: The boundary of the plane face containing p covered with Steiner points. In this example we chose $\varepsilon = \frac{4}{9}$. We can also see an example of how t_i is obtained from f_i.

Lemma 8. *For any feed link placement $F = \{(p, f_1), \ldots, (p, f_k)\}$, there exists a feed link placement $T = \{(p, t_1), \ldots, (p, t_k)\}$ such that $t_1, \ldots, t_k$ are Steiner points or boundary vertices, and $\Delta_p(G + T, D) \leq (1 + \varepsilon) \cdot \Delta_p(G + F, D)$.*

Proof. To prove this, we will construct a feed link placement $T = \{(p, t_0), \ldots, (p, t_l)\}$ such that the lemma holds. For every $(p, f_i) \in F$ we have one of three cases:

- If f_i is a Steiner point or boundary vertex, then we take $t_i = f_i$,
- If f_i is a point between a Steiner point and a boundary vertex, then let t_i be this boundary vertex,
- If f_i is a point between two Steiner points, then let t_i be the Steiner point that is closest to one of either vertices at the endpoints of the edge, see Figure 6.1.

It directly follows that each feed link connects p to a boundary vertex or Steiner point. To prove that $\Delta_p(G + T, D) \leq (1 + \varepsilon) \cdot \Delta_p(G + F, D)$, we prove for an arbitrary $v \in D$ that $\text{dist}_{G+T}(p, v) \leq (1 + \varepsilon) \cdot \text{dist}_{G+F}(p, v)$. Let (p, f_i) be the feed link used in the shortest path from p to v in graph $G + F$. Let (x_a, x_b) be the boundary edge containing t_i. Without loss of generality, assume $|x_a t_i| \leq |x_b t_i|$.

Let ω denote the shortest path from p to v in graph $G + F$. We know the first three vertices visited by ω are either $\langle p, f_i, x_a \rangle$ or $\langle p, f_i, x_b \rangle$. Consider the path ω' from p to v in graph $G + T$ where ω' differs from ω only in the second vertex: the second vertex in ω' is t_i instead of f_i. The way we constructed T, we know that t_i is in between x_a and f_i. So in the case that the first three vertices of ω are $\langle p, f_i, x_a \rangle$, we know ω' is shorter than ω.

Consider the case where the first three vertices of ω are $\langle p, f_i, x_b \rangle$. Recall that we have $|x_a t_i| \leq |x_b t_i|$ and $|x_a x_b| = l$. It follows that $|x_b t_i| \geq \frac{1}{2} l$ which implies the following:

$$\text{length}(\omega) \geq \frac{1}{2} l. \tag{6.1}$$

Observe also that $|p t_i| \leq |p f_i| + \frac{1}{4} \varepsilon l$ and $|t_i x_b| \leq |f_i x_b| + \frac{1}{4} \varepsilon l$. It directly follows that

$$\text{length}(\omega') \leq \text{length}(\omega) + \frac{1}{2} \varepsilon l. \tag{6.2}$$

From Equation 6.1 and Equation 6.2 we obtain

$$\text{length}(\omega') \leq (1 + \varepsilon) \, \text{length}(\omega)$$

and since ω is the shortest path from p to v in $G + F$, it follows that

$$\text{dist}_{G+T}(p, v) \leq \text{dist}_{G+F}(p, v). \tag{6.3}$$

□

An approximation algorithm is directly obtained from Lemma 8: try every feed link placement connecting to only Steiner points or boundary vertices. As observed before we need $4m/\varepsilon$ Steiner points to cover the whole boundary, meaning we need to try $(4m/\varepsilon)^k$ feed link placements. Calculating the maximum dilation for each placement takes $O(kn)$ time, after initialisation as described in Section 2.3. Hence this approximation algorithm runs in $O((4m/\varepsilon)^k kn + mn^2 \log n)$ time.

Theorem 8. *A $(1+\varepsilon)$ approximation algorithm exists for* MinDFLP *that runs in $O((4m/\varepsilon)^k kn + mn^2 \log n)$ time.*

6.2 A $(t+1)$-approximation when the face is a t-spanner

In this section we present an approximation algorithm for DilFLP. If the algorithm returns "no", there is no solution with k feedlinks and a maximum dilation α. If the algorithm return "yes", it has found a solution using at most k feedlinks such that the maximum dilation is at most $(t+1)\cdot\alpha$, where for all pairs of points a, b on the boundary of f we have $\text{dist}(a,b)/|ab| \leq t$, i.e. f is a t-spanner.

First, recall from Lemma 6 that for each destination point $v \in D$ we can construct a set I of intervals (r_a, r_b, v) such that when we have a feed link (p, r) with $r_a \leq r \leq r_b$, the dilation from p to v is at most α. When I contains only one interval for each destination point, the problem simplifies significantly, since we now simply have to "hit" *every* interval with a feedlink. In reality however there may be up to m intervals for each $v \in D$. Let us define a *gap* as the region between two intervals for the same destination point: (r_b, r_c, v) is a gap when there exist $r_a, r_b \in \mathbb{R}$ such that $(r_a, r_b, v) \in I$ and $(r_c, r_d, v) \in I$. The principle of our approximation algorithm is based on closing all but one of these gaps.

Let t be the smallest value for which f is a t-spanner, that is, the dilation between any two points on the boundary of f is at most t. We will show that for each destination point $v \in D$ there is at most one gap (r_a, r_b, v) such that $\delta_v(r) > (t+1)\cdot\alpha$ for some r with $r_a < r < r_b$. It follows that it is always possible to merge all the intervals into one single interval (r_a, r_b, v), such that for all $r_a \leq r \leq r_b$ we have $\delta_v(r) \leq (t+1)\cdot\alpha$.

Lemma 9. *When f is a t-spanner, there is at most one gap (r_a, r_b, v) such that $\delta_v(r) > (t+1)\cdot\alpha$ for some $r_a < r < r_b$.*

Proof. First note that there is at most one gap that is larger than half the length of the boundary of the face. Let (r_a, r_b, v) be an arbitrary gap smaller than half the boundary. Let $a = P(r_a)$ and $b = P(r_b)$. Since the boundary is a t-spanner, we know that the distance along the boundary from a to b is at most $t \cdot |ab|$.

Second, note that $\delta_v(r_a) = \delta_v(r_b)$, so we obtain

$$|pa| + \text{dist}(a, v) = |pb| + \text{dist}(b, v). \tag{6.4}$$

Next we pick a point c between a and b, such that

$$\text{dist}(c, a) + \text{dist}(a, v) = \text{dist}(c, b) + \text{dist}(b, v). \tag{6.5}$$

See Figure 6.2.

Note that $|pa| + |pb| \geq |ab|$ and $\text{dist}(a, v) + \text{dist}(b, v) \geq |ab|$. Using this, we establish the following:

$$\begin{aligned}
|pa| + \text{dist}(a, v) &= \frac{1}{2}(|pa| + \text{dist}(a, v) + |pa| + \text{dist}(a, v)) \\
&= \frac{1}{2}(|pa| + \text{dist}(a, v) + |pb| + \text{dist}(b, v)) \\
&\geq \frac{1}{2}(|ab| + |ab|) \\
&= |ab|
\end{aligned}$$

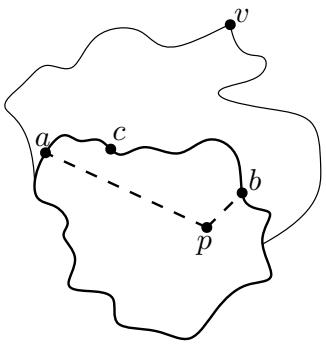


Figure 6.2: Connecting a feed link to a point between a and b results in a dilation between p and v larger than α. We prove that this dilation will not exceed $(t+1)\cdot\alpha$.

We can now derive an upper bound for the distance in the graph from p to v via a feedlink connected to c.

$$\begin{aligned}
|pc| + \mathrm{dist}(c,v) &\leq |pc| + \mathrm{dist}(c,a) + \mathrm{dist}(a,v) \\
&= \frac{1}{2}(|pc| + \mathrm{dist}(c,a) + \mathrm{dist}(a,v) + |pc| + \mathrm{dist}(c,a) + \mathrm{dist}(a,v)) \\
&= \frac{1}{2}(|pc| + \mathrm{dist}(c,a) + \mathrm{dist}(a,v) + |pc| + \mathrm{dist}(c,b) + \mathrm{dist}(b,v)) \\
&= \frac{1}{2}(2|pc| + \mathrm{dist}(c,a) + \mathrm{dist}(c,b) + \mathrm{dist}(a,v) + \mathrm{dist}(b,v)) \\
&\leq \frac{1}{2}(2|pc| + t\cdot|ab| + \mathrm{dist}(a,v) + \mathrm{dist}(b,v)) \\
&\leq \frac{1}{2}(|pa| + |ac| + |cb| + |bp| + t\cdot|ab| + \mathrm{dist}(a,v) + \mathrm{dist}(b,v)) \\
&= \frac{1}{2}(|ac| + |cb| + t\cdot|ab| + (|pa| + \mathrm{dist}(a,v)) + (|pb| + \mathrm{dist}(b,v)) \\
&\leq \frac{1}{2}(\mathrm{dist}(a,c) + \mathrm{dist}(c,b) + t\cdot|ab| + 2(|pa| + \mathrm{dist}(a,v))) \\
&\leq \frac{1}{2}(t\cdot|ab| + t\cdot|ab| + 2(|pa| + \mathrm{dist}(a,v))) \\
&= t\cdot|ab| + |pa| + \mathrm{dist}(a,v) \\
&\leq t\cdot(|pa| + \mathrm{dist}(a,v)) + |pa| + \mathrm{dist}(a,v) \\
&= (t+1)\cdot(|pa| + \mathrm{dist}(a,v)) \\
&= (t+1)\cdot\alpha|pv|
\end{aligned}$$

Finally we prove that for any point c' between a and b we have $|pc'| + \mathrm{dist}(c',v) \leq (t+1)\cdot\alpha$. Let's assume c' is between c and a. The proof for c' between c and b is symmetric. From the triangle equality it follows that $|pc'| - \mathrm{dist}(c,c') \leq |pc|$. Observe that $\mathrm{dist}(c',a) = \mathrm{dist}(c,a) - \mathrm{dist}(c,c')$, and it

follows that

$$\begin{aligned}|pc'| + \text{dist}(c', v) &\leq |pc'| + \text{dist}(c', a) + \text{dist}(a, v) \\ &= |pc'| + \text{dist}(c, a) - \text{dist}(c, c') + \text{dist}(a, v) \\ &\leq |pc| + \text{dist}(c, a) + \text{dist}(a, v) \\ &\leq (t+1)\alpha|pv|.\end{aligned}$$

We can conclude that for all gaps (r_a, r_b, v) smaller than half the boundary of the face, we have that $\delta_v(r) \leq (t+1) \cdot \alpha$ for all $r_a < r < r_b$, and therefore there can be at most one gap (r_a, r_b, v) for which $\delta_v(r) > (t+1) \cdot \alpha$ with $r_a < r < r_b$. □

For each gap (r_a, r_b, v), one can calculate the maximum $\delta_v(r)$ for $r_a < r < r_b$. We can then close all gaps except for the one with the largest maximal dilation to obtain one large interval (r_a, r_b, v) for each $v \in D$. Lemma 9 implies that $\delta_v(r) \leq (t+1)\alpha$ for all $r_a < r < r_b$.

Now the problem is reduced to a problem where we have to pick k locations on a cyclic set of intervals, such that every interval is hit. For the non-cyclic version of this problem, there is a simple greedy algorithm: scan the endpoints of the intervals in order, and for each end point we encounter we add it to the solution and remove all intervals being hit. In the cyclic version of this problem, we cannot define a first endpoint, however after any first choice, the remaining intervals are no longer cyclic. So to solve the cyclice version, we can check every first choice (there are n choices) after which we run the linear time greedy algorithm on the left over, non-cyclic intervals. Hence, the cyclice interval hitting problem can be solved in $O(n \log n + n^2) = O(n^2)$ time.

Recall from Lemma 6 that to find all intervals, we need $O(mn^2 \log n)$ time. To find all gaps we need to sort the intervals and then apply a trivial linear scan. So finding the gaps takes $O(n + mn \log mn) = O(mn \log n)$ time. Finding the gap with the worst dilation for each $v \in D$ simply means finding the r for which $\delta_v(r)$ is maximal. This can be done in $O(m)$ time for each v, so we need $O(mn)$ in total. We obtain the following theorem:

Theorem 9. *There exists a $(t+1)$-approximation algorithm for* DilFLP *that runs in $O(mn \log n)$ time after initialisation of $O(mn^2 \log n)$.*

This algorithm can be used to solve MinDFLP using the method explained in Section 4.3.

Theorem 10. *There exists a $(t+1)$-approximation algorithm for* MinDFLP *that runs in $O(mn^2 \log n)$ time when the face containing p is a t-spanner.*

Proof. By Theorem 9 and Theorem 6 we obtain running time $O(mn \log^2 n)$ after initialisation, which requires $O(mn^2 \log n)$ time. So the initialisation step dominates the running time. □

Chapter 7

Distance and dilation

In this chapter we are interested in the relation between dilation and distance in the context of placing feed links. Studying feed link placement for distance may yield new insights for feed link placement for dilation.

7.1 Equivalence of decision problems

To see how distance and dilation are related in the context of placing feed links, we consider two similar problems: DilFLP as described in Chapter 4, and a version of DilFLP where we focus on distance rather than dilation which we will call Distance Feed Link Placement (DisFLP). That is, given a planar graph G, a vertex p in a plane face f of G, a set of vertices $D \subseteq V(G)$ and an integer k the problems are defined as follows:

- In the DilFLP problem we have an instance (G, p, D, k, α) and want to know if there exists a feed link placement F such that

$$\max_{v \in D} \frac{\text{dist}_{G+F}(p, v)}{|pv|} \leq \alpha \tag{7.1}$$

- In the DisFLP problem we have an instance (G, p, D, k, β) and want to know if there exists a feed link placement F such that

$$\max_{v \in D} \text{dist}_{G+F}(p, v) \leq \beta \tag{7.2}$$

We show that these problems are equivalent with regard to their computational complexity, i.e., DilFLP is polynomial-time reducible to DisFLP and DisFLP is polynomial-time reducible to DilFLP.

Lemma 10. DilFLP *is linear-time reducible to* DisFLP.

Proof. Given an instance (G, p, D, k, α) of the DilFLP problem we construct an instance of the DisFLP problem. We initialise $D' \leftarrow \emptyset$ and $G' \leftarrow G$. Let q be the vertex in D furthest away from p. For all vertices $v \in D \setminus \{q\}$ we perform the following operation: insert a new vertex v' in G' at distance $\alpha \cdot |pq| - \alpha \cdot |pv|$ from v and insert an edge between v and v'. Add v' to the set D'. It can easily

be seen that obtaining G', D' and $\alpha \cdot |pq|$ can be done in linear time. We will now prove that $(G', p, D', k, \alpha \cdot |pq|)$ is a 'yes' instance of DisFLP if and only if (G, p, D, k, α) is a 'yes' instance of DilFLP.

Assume there exists a feed link placement F such that $\max_{v' \in D'} \text{dist}_{G'+F}(p, v') \leq \alpha \cdot |pq|$, then we know that in the graph $G' + F$ there is a path from p to any vertex $v' \in D'$ of length at most $\alpha \cdot |pq|$. The last edge in this path is from v to v' with length $\alpha \cdot |pq| - \alpha \cdot |pv|$, so by removing this edge from the path we obtain a path from p to v of length at most $\alpha \cdot |pq| - (\alpha \cdot |pq| - \alpha \cdot |pv|) = \alpha \cdot |pv|$. This path also exists in graph $G + F$ and it directly follows that the dilation from p to v in graph $G' + F$ is at most α. Hence if $(G', p, D', k, \alpha \cdot |pq|)$ is a 'yes' instance of DisFLP then (G, p, D, k, α) is a 'yes' instance of DilFLP.

Assume there exists a feed link placement F such that $\Delta_p(G + F, D) \leq \alpha$, then we know that in the graph $G + F$ there is a path from p to any vertex $v \in D$ of length at most $\alpha \cdot |pv|$. This path also exists in graph $G' + F$. If we append the edge (v, v') to this path we obtain a path from p to v' of length at most $\alpha \cdot |pv| + \alpha \cdot |pq| - \alpha \cdot |pv| = \alpha \cdot |pq|$, hence we can conclude that if (G, p, D, k, α) is a 'yes instance of DilFLP then $(G', p, D', k, \alpha \cdot |pq|)$ is a 'yes' instance of DisFLP. □

Lemma 11. DisFLP *is linear-time reducible to* DilFLP.

Proof. We take an approach similar to the proof in Lemma 10. Given an instance (G, p, D, k, β) of the DisFLP problem we construct an instance of the DilFLP problem. Initialise $D' \leftarrow \emptyset$ and $G' \leftarrow G$. For all vertices $v \in D$ we perform the following operation: insert a new vertex v' in G' at distance β from p and within distance β from v. If this is not possible then $|pv| > \beta$ implying that (G, p, D, k, β) is a 'no' instance and we can create any trivial 'no' instance of DilFLP, so we will assume that v' can be placed as described. We insert another new vertex v'' and two new edges (v, v'') and (v'', v') into G' such that $|vv''| + |v''v'| = \beta$. This is possible since $|vv'| \leq \beta$. Add v' to the set D'. It can easily be seen that obtaining G', D' can be done in linear time. We will now prove that $(G', p, D', k, 2)$ is a 'yes' instance of DilFLP if and only if (G, p, D, k, β) is a 'yes' instance of DisFLP.

Assume there exists a feed link placement F such that $\Delta_p(G' + F, D') \leq 2$, then we know that in the graph $G' + F$ there is a path from p to v' of length at most 2β (since $|pv'| = \beta$). We also know the last two edges of this path are (v, v'') and (v'', v') with a combined length of β. Therefore there must be a path in $G' + F$ from p to v of length β. This path also exists in graph $G + F$. Hence if $(G', p, D', k, 2)$ is a 'yes' instance of DilFLP then (G, p, D, k, β) is a 'yes' instance of DisFLP.

Assume there exists a feed link placement F such that $\max_{v \in D} \text{dist}_{G'+F}(p, v') \leq \beta$, then we know that in the graph $G + F$ there is a path from p to any vertex $v \in D$ of length at most β. This path also exists in graph $G' + F$. If we append the edges (v, v'') and (v'', v') to this path we obtain a path from p to v' of length at most 2β (since $|vv''| + |v''v'| = \beta$). Since $|pv'| = \beta$ we know that the dilation from p to v' in graph $G' + F$ is at most 2. We conclude that if (G, p, D, k, β) is a 'yes' instance of DisFLP then $(G', p, D', k, 2)$ is a 'yes' instance of DilFLP. □

Lemma 10 and Theorem 7 directly implies that DisFLP is also NP-hard.

Theorem 11. DisFLP*is NP-hard.*

We see that exact solution for DilFLP and DisFLP can easily be used to solve the other problem. However, this is not true for approxiate solutions. We present one way of using opproximate solution for one problem to find an approximate solution for the other. For this purpose, let $v_1, \ldots, v_n$ denote the vertices in D in order of Euclidean distance to p, e.g. v_1 is closest to p and v_n is farthest from p.

Theorem 12. *A ρ-approximation algorithm for* DisFLP *running in $O(f(k,m,n))$ time gives a $(1+(\rho-1)\frac{|pv_n|}{|pv_1|})$-approximation for* DilFLP *in $O(n+f(k,m,n))$ time.*

Proof. We use the same method as described in Lemma 10 to convert a DilFLPproblem instance into a DisFLP problem instance. If a ρ-approximation solution F for DisFLPis found we know that the paths from p to any $v' \in D'$ in graph $G'+F$ are of length at most $3 \cdot (\alpha \cdot |pq|)$, so for any $v \in D$ in graph $G+F$ we know

$$\begin{aligned}
\text{dist}_{G+F}(p,v) &\leq \rho \cdot (\alpha \cdot |pq|) - (\alpha \cdot |pq| - \alpha \cdot |pv|) \\
&= (\rho-1) \cdot (\alpha \cdot |pq|) + \alpha \cdot |pv| \\
&= |pv| \cdot (1 + (\rho-1) \cdot \frac{\alpha \cdot |pq|}{\alpha \cdot |pv|}) \cdot \alpha \\
&= |pv| \cdot (1 + (\rho-1) \cdot \frac{|pv_n|}{|pv|}) \cdot \alpha \\
&\leq |pv| \cdot (1 + (\rho-1) \cdot \frac{|pv_n|}{|pv+1|}) \cdot \alpha
\end{aligned}$$

and it directly follows that the dilation from p to any $v \in D$ in graph $G+F$ is at most $(1+(\rho-1) \cdot \frac{|pv_n|}{|pv_1|}) \cdot \alpha$.

The conversion for a DilFLP instance to a DisFLP instance takes $O(n)$ time so we obtain a total running time of $O(n+f(k,m,n))$. □

7.2 A 3-approximation for DisFLP

The findings in Section 7.1 suggest a better understanding of DisFLP could yield new results for solving DilFLP. In this section we will describe a simple 3-approximation algorithm for DisFLP. This is a greedy algorithm which in each step picks the best feed link for the destination point with the largest distance in the graph to p.

Theorem 13. *Algorithm 3 is a 3-approximation for* DisFLP *and runs in $O(kn)$ time after initialisation of $O(mn^2 \log n)$ time.*

Proof. To prove correctness we show that F^* as obtained by the algorithm is a 3-approximation with value $\max_{v \in D} d(v)$, meaning that if $\max_{v \in D} d(v) > 3\beta$ we can conclude no solution with value β exists. Consider an optimal feed link placement F with value OPT. We show that $\text{dist}_{G+F^*}(p,v) \leq 3 \cdot OPT$ for all $v \in D$. To do this we will need to new definitions:

- For all feed links in the approximation solution $f^* \in F^*$ let $v_{f^*} \in D$ denote the vertex found on line 6 of the algorithm directly prior to find f^* as a feed link minimising the distance (or dilation) to v_{f^*}.

Algorithm 3 A 3-approximation for DisFLP.

Input: Graph G with a face described by vertices $\langle x_0, \ldots, x_{m-1} \rangle$, point p, destination points D, integer k, real value β.

Output: A set F of k feed links such that $\max_{v \in D} \text{dist}_{G+F}(p, v) \leq 3\beta$ or a guarantee that there exists no feed link placement F of k feed links such that $\max_{v \in D} \text{dist}_{G+F}(p, v) \leq \beta$.

1: **function** ApproximateBestDistance(G,p,D,k,β, $\langle x_0, \ldots, x_{m-1} \rangle$)
2: Initialise ▷ As described in Section 2.3
3: $F^* \leftarrow \emptyset$
4: $d(v) \leftarrow \infty$ for all $v \in D$
5: **for all** $i = 1$ **to** k **do**
6: Find $v \in D$ with maximal $d(v)$
7: Find value for r that minimises $\delta_v(r)$
8: $F^* \leftarrow F^* \cup \{(p, P(r))\}$
9: **for all** $v \in D$ **do**
10: $d(v) \leftarrow \min\{d(v), |pv| \cdot \delta_v(r)\}$
11: **return** F^* if $\max_{v \in D} d(v) \leq 3\beta$, No otherwise

- For all feed links in the optimal solution $f \in F$ let $D_f \subseteq D$ denote the set of all vertices v for which the shortest path from p to v includes f.

Let $v \in D$ be an arbitrary vertex. There exists a path in graph $G + F$ from p to v of length at most OPT. Let $f = (p, x)$ be the first edge of this path. Note that $f \in F$. Consider the following two cases:

(1) There exists an $f^* \in F^*$ such that $v_{f^*} \in D_f$. We know that $\text{dist}_{G+F^*}(p, v_{f^*}) \leq OPT$ since F^* is chosen such that this distance is minimised. We also know that $\text{dist}_{G+F}(v_{f^*}, x) \leq OPT$ since $v_{f^*} \in D_f$. This also holds in the graph $G + F^*$. Similarly we know that $\text{dist}_{G+F^*}(x, v) \leq OPT$. We can conclude that the concatenation of the paths from p to v_{f^*}, from v_{f^*} to x and from x to v yields a path from p to v in graph $G + F^*$ of length at most $3 \cdot OPT$.

(2) There does not exist an $f^* \in F^*$ such that $v_{f^*} \in D_f$. This means that by the pigeon hole principle there must exist a $g \in F$ for which there exist two distinct $f^*, g^* \in F^*$ such that $v_{f^*} \in D_g$ and $v_{f^*} \in D_g$. Without loss of generality assume that feed link f^* was found by the algorithm before g^*. We know from case (1) that the presence of f^* implies that all $v' \in D_g$ there exists a path of length at most $3 \cdot OPT$ from p to v' starting with edge f^*. When the algorithm decides to add feed link g^*, it means that v_{g^*} was the destination point with the largest graph distance to p. We can conclude that directly prior to adding g^* all vertices $v'' \in D$ have graph distance at most $3 \cdot OPT$.

Hence F is a 3-approximation for DisFLP, and since $d(v)$ maintains the length of the shortest path in $G + F^*$ we can easily see that the value of solution F^* is $\max_{v \in D} d(v)$.

The running time of line 2 is described in Section 2.3 as $O(mn^2 \log n)$. Lines 3 and 4 take $O(1)$ and $O(n)$ time respectively. Line 6 takes $O(n)$ time, line 7 can be done in m time by Corollary 1.1. Line 8 takes $O(1)$ time. The loop of lines 9 and 10 takes $O(n)$ time since $\delta_v(r)$ can be evaluated in constant time

after initialisation from line 2. So the body of the outer loop takes $O(n)$ time in total. Since the loop executed k times we obtain a total running time of $O(mn^2 \log n + kn) = O(mn^2 \log n)$. □

Theorem 14. *There exists a* $(1 + 2\frac{v_n}{v_1})$*-approximation algorithm for* DILFLP *that runs in* $O(kn)$ *time after initialisation of* $O(mn^2 \log n)$ *time.*

Proof. This follows directly from Theorem 12 and Theorem 13. □

Theorem 15. *There exists a* $(1+2\frac{v_n}{v_1})$*-approximation algorithm for* MINDFLP *that runs in* $O(mn^2 \log n)$ *time.*

Proof. This follows directly from Theorem 4 and Theorem 14. □

7.3 Domination relation

Further research in how distances behave in the context of placing feed links reveals an interesting observation. Let us first define a relation between two vertices.

Definition 3. *Given a graph* G *with point* p *in face* f *of* G*, and* $u, v \in V(G)$*. Vertex* u c-dominates *vertex* v *if for all* x *on the boundary of* f *we have*

$$|px| + \text{dist}(x, u) \leq c \cdot (|px| + \text{dist}(x, v)). \tag{7.3}$$

In a c-approximation algorithm, when vertex u c-dominates vertex v, we can ignore v provided we minimise the distance p to u. We are interested to see what happens when two vertices do not dominate each other. Informally the following lemma describes that if a vertex v does not c-dominate a vertex u then there exists no feed link that is part of a short path from p to u and a short path from p to v.

Lemma 12. *Given a graph* G *with point* p *in face* f *of* G*, and* $u, v \in V(G)$*. For all values* c *and points* x_1, x_2 *on the boundary of* f *we have*

$$\begin{gathered} |px_1| + \text{dist}(x_1, v) > c \cdot (|px_1| + \text{dist}(x_1, u)) \\ \Rightarrow \\ \max\{|px_2| + \text{dist}(x_2, u), |px_2| + \text{dist}(x_2, v)\} > \frac{c-1}{2} \cdot (|px_1| + \text{dist}(x_1, u)) \end{gathered}$$

Proof. First note that

$$\begin{aligned} |px_1| + \text{dist}(x_1, u) + \text{dist}(u, v) &\geq |px_1| + \text{dist}(x_1, v) \\ &> c \cdot (|px_1| + \text{dist}(x_1, u)). \end{aligned}$$

It follows that

$$\text{dist}(u, v) \geq (c-1) \cdot (|px_1| + \text{dist}(x_1, u)).$$

Hence

$$\begin{aligned}\max\{|px_2| + \mathrm{dist}(x_2, u), |px_2| + \mathrm{dist}(x_2, v)\} &\geq \max\{\mathrm{dist}(x_2, u), \mathrm{dist}(x_2, v)\} \\ &\geq \frac{\mathrm{dist}(x_2, u) + \mathrm{dist}(x_2, v)}{2} \\ &\geq \frac{1}{2} \cdot \mathrm{dist}(u, v) \\ &> \frac{c-1}{2} \cdot (|px_1| + \mathrm{dist}(x_1, u)).\end{aligned}$$

□

Chapter 8

Conclusion

We studied the problem of connecting a disconnected point p to a network minimising the dilation between p and a set of destinations D in the network. This is a problem that needs to be solved when one wants to perform network analysis on incomplete or simplified data.

We presented an efficient algorithm to compute placement of a single feed link. We proved that the problem of placing multiple feed links is NP-hard and gave two different methods to solve this problem exactly in exponential time. The first was obtained after identifying a discrete number of feed link placements that is guarantees to include an optimal feed link placement. This algorithm runs in $O((2mn^2 + mn)^{k-1}mn \log n)$ time. A second exact algorithm was found after discovering an efficient way to use an algorithm solving the decision problem to solve the optimisation problem. This resulted in an improved running time of $O((mn)^k kn \log n)$.

Since the exact algorithms given in this thesis have impractically large running times, especially for larger values for k, we gave a number of approximation algorithms. A discretisation of the problem yields a $(1+\varepsilon)$-approximation algorithm that runs in $O((4m/\varepsilon)^k kn + mn^2 \log n)$ time. Subsequently we presented two and two polynomial time approximation algorithms. The first assumes f is a t-spanner, and gives a $(t+1)$-approximation in $O(mn^2 \log n)$ time. The second follows from a comparison between distance and dilation. We found a greedy algorithm giving a 3-approximation for the related problem of minimising distance to the destination points D. We found that this algorithm can be used to give a $(1 + 2\frac{|pv_n|}{|pv_1|})$-approximation in $O(mn^2 \log n)$ time, where v_1 denotes the vertex with the smallest Euclidean distance to p, and v_n the vertex with the largest Euclidean distance to p.

8.1 Further research and possible improvements

Some low hanging fruit is to show the existence of a smaller candidate set, resulting in a faster algorithm. We suspect the size of the the candidate set can be reduced by at least a factor 2 by removing all candidate points (r, d) from the set that relate to and intersection of two dilation functions δ_u and δ_v that are both increasing or both decreasing at r. A more dramatic improvement can be made if we can exclude a significant number of combinations of candidate

points.

For real world applications it is useful to study two related problems. Firstly, consider the case when one attempts to connect multiple disconnected points. A naive way to approach this problem is as a sequence of MinDFLP problems where we connect each point one by one. However, the order in which we add the points may yield significantly different results, since the inserted feed links may reveal shortest paths that were not present in the graph before. Additionally having multiple disconnected points in the same face requires a largely new approach to the problem.

Secondly, during our research some effort was put into the case where G is a planar graph. All algorithms presented in this thesis can solve this problem provided the feed links may intersect with existing edges in the graph. This is allowed in the model adopted by Aronov et al. [1]. When feed links are not allowed to intersect edges in the graph, the algorithms we presented are still of significance. Lee [7] presents a linear time algorithm that can be used to determine which regions of the boundary of the face f are visible from p. We believe all algorithms presented in this thesis can relatively easily be adapted to avoid the regions of the boundary of f not visible from p. Note that when G is planar, the number of edges in G can be bounded by $O(n)$ rather than $O(n^2)$, improving the running time of our initialisation algorithm to $O(mn \log n)$. An open problem that we were not able to make significant progress on is the question whether the planar case is NP-hard. It may be possible to exploit the planarity of the graph to obtain a polynomial time algorithm. It may be easiest to approach this planar version of the problem by considering the planar case of DisFLP. Although our proof for the reductions from DilFLP to DisFLP and vise versa assume intersecting edges are allowed, we believe a very similar approach will give the same result without intersecting edges. Therefore proving or disproving NP-hardness of DilFLP can be done by proving or disproving NP-hardness of DisFLP.

Bibliography

[1] Boris Aronov, Kevin Buchin, Maike Buchin, Bart M. P. Jansen, Tom de Jong, Marc J. van Kreveld, Maarten Löffler, Jun Luo, Rodrigo I. Silveira, and Bettina Speckmann. Connect the dot: Computing feed-links for network extension. *J. Spatial Information Science*, 3(1):3–31, 2011.

[2] Marko Savić and Miloš Stojaković. Linear time algorithm for optimal feed-link placement. *Computational Geometry*, 48(3):189 – 204, 2015.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[4] H. Davenport and A. Schinzel. A combinatorial problem connected with differential equations. *American Journal of Mathematics*, 87(3):684–694, 1965.

[5] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[6] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[7] D.T Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207 – 221, 1983.