MASTER

New results on feed-link placement

van Lieshout, T.

Award date:
2015

Link to publication
New results on feed-link placement

Master Thesis

Tim van Lieshout

Supervisors:
dr. J. (Joachim) Gudmundsson (University of Sydney)
prof.dr. M.T. (Mark) de Berg (TU/e)
dr. K.A. (Kevin) Buchin (TU/e)

final version

Sydney, June 2015
Abstract

Given a simple polygon $P$ in the plane and a source point $p$ in its interior, we aim to connect $p$ to the boundary $\partial P$ of the polygon using one or more edges called feed links, such that the detour from every point on $\partial P$ to $p$ is as small as possible. This detour is defined as the ratio between the shortest path in the resulting network, and the Euclidean distance. In doing so we look at a more general version of the problem of reducing the maximum detour presented in earlier papers, as well as a variation of this problem where we try to minimise the average detour.

We first prove that a feed link that solves one of these problems optimally might have arbitrary bad results for the other problem, and hence we need to look at these problems independently.

We then present an algorithm that places one feed-link optimally such that the average detour is minimised in linear time.

Next a variation of this algorithm is presented that reduces finding the optimal placement for $k$ feed links such that the average detour is minimised to solving $O(n^{2k})$ systems of $k$ equations in $k$ unknowns.

Finally we look at approximation algorithms for both minimising the average and minimising the maximum detour. This results in two algorithms. The first places $k$ feed links such that the average detour is a $(1 + \epsilon)$-approximation of the optimal solution in time $O(k^{2\epsilon} n)$. The second places $k$ feed links such that the maximum detour is a $(1+\epsilon)$-approximation of the optimal solution in time $O(k^{2\epsilon}(k + n))$. 
Preface

This master thesis describes the result of the research I have conducted at the University of Sydney for the past six months. Here I like to take the opportunity to thank all those that helped and supported me during this processes.

First of all I am immensely grateful to my daily supervisor Joachim Gudmundsson who graciously accepted to supervise me at the University of Sydney, and without whose feedback, discussions, and ideas this thesis would not be what it is today. Furthermore I would like to thank Mark de Berg and Kevin Buchin for always answering my questions and their great feedback on my ideas as well as on my thesis. I would also like to thank Mykola Pechenizkiy for joining the assessment committee.

My gratitude moreover reaches out to all the great people I met in Sydney who helped me find my feet in a foreign city. Special thanks go out to my new friend and former room mate Niraj who helped me with so many things it is to much too name them all here.

Finally I would like to thank my friends and family back home without whose support I would never have gotten to where I am today.

Tim van Lieshout, Sydney, May 2015
# Contents

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>vii</td>
</tr>
<tr>
<td>1  Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2  Preliminaries</td>
<td>3</td>
</tr>
<tr>
<td>3  Maximum versus average detour</td>
<td>5</td>
</tr>
<tr>
<td>4  Minimising the average detour with one feed link</td>
<td>9</td>
</tr>
<tr>
<td>4.1 Partitioning the polygon</td>
<td>9</td>
</tr>
<tr>
<td>4.2 The first interval</td>
<td>11</td>
</tr>
<tr>
<td>4.3 The remaining intervals</td>
<td>13</td>
</tr>
<tr>
<td>5  Generalising the algorithm</td>
<td>15</td>
</tr>
<tr>
<td>5.1 Dividing the polygon</td>
<td>15</td>
</tr>
<tr>
<td>5.2 Creating the formulas</td>
<td>18</td>
</tr>
<tr>
<td>5.3 Optimising the formula</td>
<td>19</td>
</tr>
<tr>
<td>6  Discretizing the problem</td>
<td>21</td>
</tr>
<tr>
<td>6.1 Finding the average detour</td>
<td>23</td>
</tr>
<tr>
<td>6.2 Finding the maximum detour</td>
<td>24</td>
</tr>
<tr>
<td>7  Conclusion</td>
<td>27</td>
</tr>
<tr>
<td>7.1 Further research</td>
<td>27</td>
</tr>
<tr>
<td>Bibliography</td>
<td>29</td>
</tr>
<tr>
<td>Appendix</td>
<td>31</td>
</tr>
<tr>
<td>A  Finding the derivative</td>
<td>31</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Geometric networks are often abstract representations of real-world networks such as roads, rivers, or railway networks. In such cases it can happen that locations of settlements are provided but the data describing roads is only partial. An example of this are accessibility studies that do not contain detailed network data, for instance because the data sets come from developing countries that are incomplete due to omissions in the digitization process.

When performing network analysis, however, it is necessary for all points of importance to be connected to the network. Therefore in this thesis we will focus on how we can connect each unconnected location, which we shall henceforth call a source points, to the rest of the network. There are numerous ways to do this. In the most simple approach these source points are simply snapped to the network. This might produce very unrealistic results, and also modifies the coordinates of the source points, which we assume are correct. The approach we will focus on in this thesis connects each source point directly to the network with one or more edges called feed links.

This approach was suggested by Dahlgren [4] and Ness and Brogaard [9] who simply connect each source point with an edge to the nearest known connected location. Ideally, however, the new connections should be reasonable in the sense that it is either conceivable that such connections exists in the real world or that adding them results in a similar network distance as could be assumed in the real (unknown) network. Aronov et al. [2], Savić and Stojaković [11] and Bose et al. [3] all try to establish methods that add feed links in a more reasonable way. Bose et al. [3] focus on adding a feed link such that the maximum network distance from the source point to any other point on the network is minimised. Aronov et al. [2] and Savić and Stojaković [11] focus on placing feed links that reduce the maximum detour from the source point to points on the boundary of the face it occurs in.

This detour is defined as follows. Let \( G := (V,E) \) be a plane graph in \( \mathbb{R}^2 \). Given any two points \( u,v \) on \( G \) the detour for these points is defined as \( \frac{\text{dist}(u,v)}{|uv|} \), where \( \text{dist}(u,v) \) is the length of the shortest path along \( G \), and \(|uv|\) denotes the Euclidean distance between these points. This is closely related to the well established concept of dilation used in things like spanners [1, 5, 7, 8]. The difference between the two concepts is that the detour is defined for any pair of points on a graph, while the dilation is only defined for every pair of vertices.

The rational for looking at the detour is as follows; In general people do not like detours, and there has been some work, eg. by Nordbeck [10] that shows that the detour in real world networks is typically small. Since this is the case we can reasonably assume that feed links that minimise the detour are more likely to represent connections that exist in the real network.

Aronov et al. [2] presents two distinct problems related to this maximum detour. The first problem they present is given a polygon \( P \) and a source point \( p \) inside the polygon place a feed link such that the maximum detour from \( p \) to any point on the polygon is minimised. The second problem they present is given a polygon \( P \), a source point \( p \), and a target detour \( c \) how to find the smallest number of feed links possible such that the detour from \( p \) to every point on the polygon is
CHAPTER 1. INTRODUCTION

at most \( c \). They then continue to present a number of results for each of these problems. For the first problem their main result is an algorithm that can find the optimal placement for a feed link that minimises the maximum detour in \( O(\lambda_7(n) \log(n)) \), where \( n \) is the number of vertices on the boundary of the face, and \( \lambda_7(n) \) is the maximum length of a Davenport-Schinzel sequence of order 7 on \( n \) symbols, a slightly superlinear function. In addition to this they present an algorithm that provides a 2-approximation for the problem in linear time and show how both of these algorithms can be adapted to take into account impenetrable obstacles within the polygon. For the second problem they present a linear time algorithm that places \( k + 1 \) feed links such that the detour from \( p \) to every point on the polygon is at most \( c \), where \( k \) is the minimum number of feed links needed to achieve this. In addition to this they also show that for arbitrary polygons this \( k \) might be as high as \( \frac{n^2}{2} \) even for extremely large values of \( c \), while for specific classes of polygon, such as convex polygons, a smaller number of feed links is always sufficient.

Savić and Stojaković [11] improve upon the results of Aronov et al. [2] on the problem of placing one feed link such that the maximum detour is minimised by presenting a linear algorithm that provides an optimal solution.

In this thesis we study two different problems that are closely related to the work of Aronov et al. [2] and Savić and Stojaković [11]. The first problem we study is a generalisation of the problem described in these papers where rather than placing one feed link to minimise the maximum detour we place \( k \) feed links to minimise the maximum detour. The second problem is a variation of this, where we look at how we can place \( k \) feed links that minimise the average detour.

The rest of this thesis is organised as follows. In Chapter 2 we introduce some notations and give a formal definition for the problems. Then in Chapter 3 we prove that minimising the maximum detour does not provide any constant-factor approximation for minimising the average detour, and minimising the average detour does not provide any constant-factor approximation for minimising the maximum detour. In Chapter 4 we provide an algorithm that places one feed link to minimise the average detour in linear time. Then in Chapter 5 we present a variation of this algorithm that reduces the problem of finding the optimal placement for \( k \) feed links to finding the solution to \( O(n^\epsilon) \) systems of \( k \) equations in \( k \) unknowns. Next in Chapter 6 we provide two algorithms that use a small number of Steiner points to place \( k \) feed links such that the maximum detour and the average detour respectively are within a \( 1 + \epsilon \) approximation of the optimum for \( \epsilon > 0 \). Finally in Chapter 7 we conclude by summarizing our results and discuss further work.
Chapter 2

Preliminaries

Given a simple polygon $P$ as a list of vertices $v_0, v_1, \ldots, v_{n-1}$ in clockwise order along its boundary $\partial P$, and given a source point $p$ that is within $P$. We want to connect $p$ to $\partial P$ with one or more edges, called feed links. See Figure 2.1 for an example.

![Figure 2.1: In this example the vertices $v_0, v_1, \ldots, v_8$ and the source point $p$ are the input. The edge $(p, q)$ is an example of a possible output.](image)

Given one such feed link $(p, q)$ with $q$ on $\partial P$, the detour $\delta$ for any point $r$ on $\partial P$ using this feed link is defined as follows:

$$\delta_q(r) = \frac{|pq| + \text{dist}(q, r)}{|pr|}$$

Here $|ab|$ denotes the Euclidean distance between points $a$ and $b$, and $\text{dist}(a, b)$ denotes the length of the shortest path along $\partial P$.

Now suppose we have a set of $k$ feed links. Then the detour of a point $r$ will be determined by the feed link that minimises the length of the path from $p$ to $r$. We will denote the detour of any point $r$ as $\delta_Q(r)$ where $Q := (q_0, q_1, \ldots, q_{k-1})$ is the set points where the feed links connect from $p$ to $\partial P$.

For any point on an edge of $P$ the Euclidean distance to $p$ changes as a function of the form $\sqrt{At^2 + Bt + C}$, where $A, B, C$ are constants and $t$ is the position of that point on the edge. Similarly, while all $k$ feed links are at a fixed position, as long as all points on an edge either have a shortest path clockwise around $\partial P$ to $p$ or they all have a shortest path counter clockwise
around $\partial P$ to $p$ the network distance to $p$ (defined below), changes linearly in the position of the point on the edge. The network distance between two points $a,b$ is defined as the length of the shortest path along the network from $a$ to $b$. As explained next by splitting the edges in $P$ in at most $n + 2k$ edges as in Aronov et al. [2] we can make sure that this is always the case.

For an example of this we look at Figure 2.1 shown above. For all edges shown, the Euclidean distance for any point on such an edge to $p$ can be expressed as a single function. For all edges except $(v_3, v_4)$ and $(v_8, v_9)$ the network distance for the points on the edge changes linearly. To make sure that the network distance for all edges changes linearly, we need to split the two aforementioned edges. As $q$ lies on the edge $(v_3, v_4)$ it needs to be split into $(v_3, q)$ and $(q, v_4)$ as the points on each side of $q$ have a different shortest path to $p$. The edge $(v_8, v_0)$ also needs to be split, as the shortest path to $p$ for some of these points goes over $v_8$ and for some it goes over $v_9$. This edge needs to be split into the edges $(v_8, m)$ and $(m, v_0)$, where $m$ is the point where the network distance to $p$ is the same over $v_8$ and $v_0$. We will call these edges that need to be split the connecting edge and the middle edge respectively.

Now using this, if we parametrize each edge by $t \in [0, 1]$, the network distance is a linear function $a + bt$ where $a, b \in \mathbb{R}_{>0}$ depend only on $P$, $p$ and the endpoints of the edge. The Euclidean distance has the form $\sqrt{At^2 + Bt + C}$ where $A, B,$ and $C$ are constants depending only on the coordinates of $p$ and the endpoints of the edge. As in Aronov et al. [2], given an edge and a feed link, that provides the shortest network distance to the source point, setting the derivative of the quotient to zero, we get as parameter values of a possible maximum $t = -bB + 2aC$ $2ab - aB$. When we insert $t$ back in the original quotient we get the maximum detour $\delta^m_q(e_i) = \delta^m_Q(e_i)$ for that edge. The maximum detour of the polygon $\delta^m_Q(P)$ is than simply

$$\max_{0 \leq i < n+2k} (\delta^m_q(e_i)).$$

Similarly, the average detour $\delta^av_q(e_i) = \delta^av_Q(e_i)$ for the edge $e_i$ can be found by taking the integral of the quotient

$$\delta^av_Q(e_i) := \int_0^1 \frac{at + b}{\sqrt{At^2 + Bt + C}} dt.$$  

Now let $|P|$ be the total length of all edges in $P$, we can then define the average detour of $P$ as

$$\delta^av_Q(P) = \sum_{0 \leq i < n+2k} \frac{\delta^av_Q(e_i) \cdot |e_i|}{|P|}.$$  

The goal in this thesis is to find algorithms that place feed links such that either $\delta^av_Q(P)$ or $\delta^av_Q(P)$ is minimised.
Chapter 3

Maximum versus average detour

A natural question to ask is whatever $\delta_{av}^Q(P)$ and $\delta_m^Q(P)$ are related. More precisely, let $(p, q^m)$ be the feed link that minimises the maximum detour, and let $(p, q^{av})$ be the feed link that minimises the average detour. Then we want to know whatever $\delta_{av}^Q(P)$ provides some approximation for $\delta_m^Q(P)$ and/or whatever $\delta_m^{av}(P)$ provides some approximation for $\delta_m^Q(P)$. In this chapter we will prove that this is not the case.

We do so by giving a counter example that is illustrated in Figure 3.1. In this example we will use $(0, 0)$ as coordinates for $p$ which gives the following coordinates for the vertices: $v_0$ is at $(1 - \gamma, 0)$, $v_1$ is at $(1 + \beta, -\epsilon)$, $v_2$ is at $(1, -\epsilon)$, $v_3$ is at $(-1, 0)$, $v_4$ is at $(1, \epsilon)$ and $v_5$ is at $(1 + \beta, \epsilon)$. Furthermore the circle in this example actually consists of an extremely large number of small straight line segments that zig zag such that the circumference of the circle is $2\sqrt{\beta}$. The variable $\epsilon$ is only used in this example to keep the polygon simple and should be regarded as infinitesimal.

To simplify the calculations and the instance however we will use the following two simplifications:

1. We will regard $\epsilon$ as 0 for the purpose of all calculations.
2. We will regard all edges between $v_2$ and $v_3$ together as one edge of length $\sqrt{\beta}$, and all edges between $v_3$ and $v_4$ together as one edge of length $\sqrt{\beta}$. Furthermore we will regard each point on these edges has having a distance of 1 to $p$.

It should be clear that neither of these simplifications will affect the results.

The idea in this example is that to minimise the maximum detour the feed link from $p$ will always connect to $v_0$, while to minimise the average detour the feed link is better of at $v_3$. Using this we can show that these properties are indeed unrelated by changing the values of $\beta$ and $\gamma$.

Lemma 1. If $\gamma \geq \frac{1}{2\sqrt{\beta}}$ the feed link that minimises the maximum detour must connect to $v_0$.

Proof. When the feed link connects to $v_0$, the maximum detour is clearly achieved at the point $v_3$ where the detour is $\sqrt{\beta} + 2\beta + 1$. Now if the feed link were to connect to some other point on the edge $(v_0, v_1)$ or $(v_0, v_5)$ this would only make the maximum detour worse, as the detour at $v_3$ would not change but the detour on all points on the far half of the circle would be worse. This leaves all points on $\partial P$ clockwise of $v_1$ and counter clockwise of $v_5$ as possible points where the feed link might connect. Observe, however, that in this case the detour at $v_0$ is at least $\frac{1 + 2\beta + \gamma}{1 - \gamma}$. Now if we can prove that $\sqrt{\beta} + 2\beta + 1 < \frac{1 + 2\beta + \gamma}{1 - \gamma}$ for $\gamma \geq \frac{1}{2\sqrt{\beta}}$ then we are done. Note that $\gamma \geq \frac{1}{2\sqrt{\beta}}$ implies that $\gamma(2\beta + \sqrt{\beta}) > \sqrt{\beta}$, so:

$$2\beta > 2\beta + \sqrt{\beta} - 2\beta\gamma - \sqrt{\beta}\gamma = (2\beta + \sqrt{\beta})(1 - \gamma)$$

which implies that:

$$\frac{2\beta}{1 - \gamma} > 2\beta + \sqrt{\beta}$$

New results on feed-link placement 5
For any

\[ \text{minimises the maximum and the feed link that minimises the average detour, which brings us to} \]

**Theorem 1.** For any \( \beta > 0 \) there is a polygon \( P \), and a point \( p \) such that \( \frac{\delta_{av}^P(p)}{\delta_{av}^P(P)} = \Omega\left(\frac{\sqrt{\beta}}{\log(\beta)}\right) \).

**Proof.** To prove this we will use a polygon that falls into the class described earlier. We take \( \gamma = 1 - \frac{1}{2\sqrt{\beta}} \). Note that for large enough \( \beta \) we have \( \gamma > \frac{1}{2\sqrt{\beta}} \). We will prove that if \( \beta \) increases the average detour achieved by connecting to \( v_3 \) is arbitrary much smaller than the average detour achieved by connecting to \( v_0 \). Note that since we simplified the polygon such that all edges between \( v_2 \) and \( v_3 \) are one edge, we will denote the length of this edge as \( |v_2v_3| \), the same holds for \( |v_3v_4| \).

We know that

\[
\delta_{av}^P(v_0) = \frac{2(\delta_{av}^P(v_0v_1)|v_0v_1| + \delta_{av}^P(v_1v_2)|v_1v_2| + \delta_{av}^P(v_2v_3)|v_2v_3|)}{2(|v_0v_1| + |v_1v_2| + |v_2v_3|)}
\]

\[
= \frac{\delta_{av}^P(v_0v_1)(\beta + 1 - \frac{1}{2\sqrt{\beta}}) + \delta_{av}^P(v_1v_2)\beta + \delta_{av}^P(v_2v_3)\sqrt{\beta}}{\sqrt{\beta} + 2\beta + 1 - \frac{1}{2\sqrt{\beta}}}
\]

\[
\delta_{av}^P(P) = \frac{2(\delta_{av}^P(v_0v_1)|v_0v_1| + \delta_{av}^P(v_1v_2)|v_1v_2| + \delta_{av}^P(v_2v_3)|v_2v_3|)}{2(|v_0v_1| + |v_1v_2| + |v_2v_3|)}
\]

\[
= \frac{\delta_{av}^P(v_0v_1)(\beta + 1 - \frac{1}{2\sqrt{\beta}}) + \delta_{av}^P(v_1v_2)\beta + \delta_{av}^P(v_2v_3)\sqrt{\beta}}{\sqrt{\beta} + 2\beta + 1 - \frac{1}{2\sqrt{\beta}}}
\]

Now with a slight abuse of notation, for \( v_0 \) we get that

\[ \delta_{av}^P(v_0v_1) = \delta_{av}^P(v_0v_5) = 1 \]  

and

\[ \delta_{av}^P(v_1v_2) = \delta_{av}^P(v_4v_5) = \int_0^1 \frac{\beta + 1 + \beta t}{\beta + 1 - \beta t} dt = \frac{2(\beta + 1)\log(\beta + 1)}{\beta} - 1 \]  

![Figure 3.1: Example where the goal of minimising the maximum detour may lead to a high average detour and vice versa.](image)
CHAPTER 3. MAXIMUM VERSUS AVERAGE DETOUR

and

\[ \delta_{v_0}^{av}(v_2v_3) = \delta_{v_0}^{av}(v_3v_4) = \int_0^1 (1 + 2\beta + \sqrt{3}t)dt = 1 + 2\beta + \frac{\sqrt{3}}{2}. \] (3.8)

Using 3.4 we obtain:

\[ \delta_{v_0}^{av}(P) = 1 - \frac{1}{2\sqrt{3}} + (2\beta + 2) \log(\beta + 1) + \sqrt{3} + 2\beta \sqrt{3} + \frac{\delta}{2} \] (3.9)

In this equation the enumerator is clearly dominated by the \(2\beta \sqrt{3}\) term as this is asymptotically the biggest term when \(\beta\) grows. As this is divided by \(2(\sqrt{3} + 2\beta + 1 - \frac{1}{2\sqrt{3}})\) this gives us that \(\delta_{v_0}^{av}(P) = \Theta(\sqrt{3})\).

Similarly for \(v_3\) we get:

\[ \delta_{v_3}^{av}(v_2v_3) = \delta_{v_3}^{av}(v_3v_4) = \int_0^1 (1 + \sqrt{3}t)dt = 1 + \frac{\sqrt{3}}{2} \] (3.10)

and

\[ \delta_{v_3}^{av}(v_1v_2) = \delta_{v_3}^{av}(v_4v_5) = \int_0^1 \frac{\sqrt{3} + 1 + \beta t}{1 + \beta t} dt = \frac{\log(\beta + 1)}{\sqrt{3}} + 1 \] (3.11)

and

\[ \delta_{v_3}^{av}(v_0v_1) = \delta_{v_3}^{av}(v_0v_5) = \int_0^1 \frac{1 + \sqrt{3} + \beta + (\beta + 1 - \frac{1}{2\sqrt{3}})t}{1 + \beta - (\beta + 1 - \frac{1}{2\sqrt{3}})t} dt = \log(\beta) + 2\log(\beta + 1) + \frac{\log(\beta + 1)(\log(\beta) + 2\log(\beta + 1) + \log(4))}{2\sqrt{3}(\beta + 1) - 1} - 1 + \log(4). \] (3.12)

Using 3.5 we obtain:

\[ \delta_{v_4}^{av}(P) = \left( \sqrt{3} + \frac{\beta}{2} + \frac{\beta \log(\beta + 1)}{\sqrt{3}} + \beta \log(\beta) + \log(\beta) - \frac{\log(\beta)}{2\sqrt{3}} + 2\beta \log(\beta + 1) + 2 \log(\beta + 1) + \frac{\log(\beta + 1)(\log(\beta) + 2\log(\beta + 1) + \log(4))}{2\sqrt{3}(\beta + 1) - 1} - 1 + \frac{1}{2\sqrt{3}} + \beta \log(4) + \log(4) - \frac{\log(4)}{2\sqrt{3}} + 2\beta \sqrt{3} + \frac{\delta}{2}\right) \] (3.13)

Observe however that asymptotically when \(\beta\) grows the biggest term in the enumerator is only \(2\beta \log(\beta + 1)\), which gives us that \(\delta_{v_0}^{av}(P) = \Theta(\log(\beta))\).

Therefore \(\delta_{v_0}^{av}(P) = \frac{\Theta(\sqrt{3})}{\Theta(\log(\beta))} = \Omega(\frac{\sqrt{3}}{\log(\beta)}).\) \(\square\)

Note that theorem 1 shows that when \(\beta \to \infty\) the ratio goes to \(\infty\) as well.

**Theorem 2.** For any \(\beta > 0\) there is a polygon \(P\), and a point \(p\) such that \(\frac{s_{\text{max}}^{av}(p)}{s_{\text{min}}^{av}(p)} = \Omega(\sqrt{3})\)
**Proof.** To see this we can use the same example as used in Theorem 1. In this case from Lemma 1 we know that the feed link that minimises the maximum detour connects to $v_0$, and has a detour of $\sqrt{\beta} + 2\beta + 1 = \Theta(\beta)$ as $\beta$ gets larger. From Theorem 1 we know however that the feed link that minimises the average detour does not connect to $v_0$. We also know that connecting to any other point on the edge $(v_0, v_1)$ or $(v_0, v_5)$ does not reduce the detour for any point. This means that the feed link that minimises the average detour connects to some point clockwise of $v_1$ and counter clockwise of $v_5$. Now since this is the case we know that the detour at $v_0$ for the feed link that minimises the average detour is at least

$$
\frac{1 + 2\beta + \gamma}{1 - \gamma} = 2 + 2\beta - \frac{1}{2\sqrt{\beta}} = 4\sqrt{\beta} + 4\beta\sqrt{\beta} - 1 = \Theta(\beta\sqrt{\beta})
$$

(3.14)

as $\beta$ gets larger. So therefore $\frac{\delta_{mqav}^m(P)}{\delta_{mqav}^m(P)} = \frac{\Theta(\beta\sqrt{\beta})}{\Theta(\beta)} = \Omega(\sqrt{\beta})$. 

---

New results on feed-link placement
Chapter 4

Minimising the average detour with one feed link

In this chapter we will present an algorithm that in $O(n)$ time places a feed link $f = (p, q)$ such that $\delta^w_q(P)$ is minimised. To do this we partition the boundary of the polygon into $O(n)$ non-overlapping intervals, after which for each interval we can find the optimal connection point in constant time.

4.1 Partitioning the polygon

For this problem we assume that as input we are given a polygon $P$ as a list of vertices $v_0, v_1, \ldots, v_{n-1}$ clockwise along its boundary $\partial P$ and a point $p$ inside $P$. Now let $\mu^+(a, b)$ be the distance from point $a$ to point $b$ in clockwise direction along $\partial P$. As in Savić and Stojaković [11] we define $P(r)$ with $r \in \mathbb{R}_{\geq 0}$ as the function returning the point on $\partial P$ such that:

$$\mu^+(v_0, P(r)) = r \mod |P|$$

(4.1)

See Figure 4.1.

![Figure 4.1: An example of the parametrization of $P$ with in green the point $P(r)$ with $\mu^+(v_0, P(r)) = r$](image)

We are interested in finding the point $P(r)$ such that the feed link $(p, P(r))$ gives the minimum average detour. In other words we are interested in finding the value of $r$ that minimises the
function

\[ F(r) := \delta_{P(r)}^\text{av}(P). \]  

(4.2)

Note that \( F(r) \) is obtained by adding the average detour of the edges \( e_i \) of \( P \), weighted by their length. More precisely

\[ F(r) = \sum_{i} \frac{|e_i|}{|P|} \delta_{P(r)}^\text{av}(e_i). \]  

(4.3)

The idea here is that if \( r \) varies such that \( P(r) \) moves along a single edge that \( F(r) \) is well behaved so we can minimise it. See Figure 4.2. To make this work however, we first need to partition the edge into smaller pieces.

To do this add a vertex \( v'_i \) to \( P \) for each vertex \( v_i \), such that \( v'_i = P(r_i + \frac{|P|}{2}) \) where \( v_i = P(r_i) \). Now we will define one interval \( I_j = (s_j, s_j + 1) \) for each consecutive pair of vertices \( v_j, v_{j+1} \) on \( \partial P \) such that \( v_j = P(s_j) \) and \( v_{j+1} = P(s_{j+1}) \). See Figure 4.3.

By defining the intervals as such we know that wherever the feed link connects anywhere within an interval the connecting edge and the middle edge will always remain the same. (Recall that the middle edge for a given point \( P(r) \) where the feed link connects, is the edge "opposite" to \( P(r) \), that is, the edge containing \( P(r + \frac{|P|}{2}) \).) We will use this property when creating the formula \( F(r) \) for each interval. Next we explain how we create these functions in more detail.
4.2 The first interval

To construct the functions mentioned above for each interval we partition $\partial P$ into four parts $Con, Mid, R$, and $L$, and create a function for each of these parts. To do this as efficiently as possible we update the formulas for each interval beyond the first rather than construct them from scratch. This section will describe how we construct the function for the first interval, while the next section will describe how we update the function for each of the remaining intervals.

Suppose the first interval is $(s_0, s_1)$. (Of course $s_0 = 0$ for the first interval, but it will be convenient for later use to use the more general form.) We then define the four parts mentioned above as follows. The first part $Con$ will only contain the (partial) edge $(P(s_0), P(s_1))$. The next part $L$ will contain all edges and partial edges clockwise from $P(s_1)$ and counter-clockwise from $P(s_0) + |\partial P|/2$. The third part $Mid$ will contain the (partial) edge $(P(s_0 + |\partial P|/2), P(s_1 + |\partial P|/2))$. The final part $R$ will contain all edges and partial edges clockwise from $P(s_1 + |\partial P|/2)$ and counter clockwise from $P(s_0)$. See Figure 4.4 for an example.

We want to describe $F(r)$ for $s_0 \leq r \leq s_1$. To this end it will be convenient to define $z := r - s$, and write the contribution of $Con, Mid$, $L$ and $R$ as function of $z$.

Now for any edge $e_i = (v_i, v_{i+1})$ where $v_{i+1}$ lies clockwise along $\partial P$ from $v_i$ we know that:

$$\delta_{P(z)}^{av}(e_i) = \int_0^1 \frac{a_i t + b_i}{\sqrt{A_i t^2 + B_i t + C_i}} dt$$

for some constants $a_i, b_i, A_i, B_i, C_i$. If $e_i$ is in $L$ we know that, $a_i = |e_i|$, and $b_i = \text{dist}(v_i, P(s_1)) + L(z)$ where $L(z) = s_1 - s_0 + |P(z)p| - z$. Using this we get that for any edge in $L$ the total contribution to the average detour is:
Chapter 4. Minimising the Average Detour with One Feed Link

Figure 4.4: An example polygon, where all (partial) edges in L are shown in blue, all (partial) edges in R are shown in orange, and the (partial) edges Con, Mid are shown dashed in green and red respectively.

\[ \frac{\delta_{av} P(z)}{|P|} = \frac{|c_i|}{|P|} \int_0^{\frac{1}{2}} \frac{|c_i|t + \text{dist}(v_i, P(s_1))}{\sqrt{A_i t^2 + B_i t + C_i}} \, dt + L(z)|c_i| \int_0^{\frac{1}{2}} \frac{1}{\sqrt{A_i t^2 + B_i t + C_i}} \, dt \]  

(4.5)

Now even without knowing the value of z we can calculate a large part of this formula. Let

\[ C_0(i) := |c_i| \int_0^{\frac{1}{2}} \frac{|c_i|t + \text{dist}(v_i, P(s_1))}{\sqrt{A_i t^2 + B_i t + C_i}} \, dt \]  

(4.6)

and let

\[ C_1(i) := |c_i| \int_0^{\frac{1}{2}} \frac{1}{\sqrt{A_i t^2 + B_i t + C_i}} \, dt, \]  

(4.7)

which are both constants. We then have that the contribution of the edge \( e_i \) to the average detour is:

\[ \frac{C_0(i) + C_1(i) \cdot L(z)}{|P|}. \]  

(4.8)

Using this we get that the total contribution of all edges in L is:

\[ \frac{C_0(L) + C_1(L) \cdot L(z)}{|P|}, \]  

(4.9)

where \( C_1(L) = \sum_i C_1(i) \) and \( C_2(L) = \sum_i C_2(i) \).

Similarly if \( e_i \) is in R we know that \( a_i = |c_i| \), and \( b_i = \text{dist}(v_{i+1}, P(s_0)) + R(z) \), where \( R(z) = z + |P(z)p| \). Following the same pattern as for L we get that:

New results on feed-link placement
CHAPTER 4. MINIMISING THE AVERAGE DETOUR WITH ONE FEED LINK

\[ C_2(i) := |e_i| \int_0^1 \frac{|e_i|t + \text{dist}(v_i+1, P(s_0))}{\sqrt{A_1t^2 + B_1t + C_1}} \, dt. \]  

(4.10)

Which gives us:

\[ \frac{C_2(R) + C_1(R) \cdot R(q)}{|P|}, \]  

(4.11)

where \( C_2(R) = \sum_i C_2(i) \) and \( C_1(R) = \sum_i C_1(i) \).

This leaves the formulas \( \text{Con}(z) \) and \( \text{Mid}(z) \) for \( \text{Con} \) and \( \text{Mid} \) respectively. Both need to be split into a right and a left part to calculate their detour. The left part for \( \text{Con} \) is the formula for the segment \( (P(s_0), P(z)) \) and the right part is the formula for the segment \( (P(z), P(s_1)) \). The left part for \( \text{Mid} \) is the formula for the segment \( (P(s_0 + \frac{|P|}{2}), P(s_0 + z + \frac{|P|}{2})) \) and the right part is the formula for the segment \( (P(s_0 + z + \frac{|P|}{2}), P(s_1 + \frac{|P|}{2})) \). For a given \( z \) these can be constructed in the regular way.

If we take this all together we get that for \( 0 \leq z \leq s_1 - s_0 \):

\[ \delta_{P(z)}^W(P) = \frac{C_0(L) + C_1(L) \cdot L(z) + C_2(R) + C_1(R) \cdot R(z) + \text{Con}(z) + \text{Mid}(z)}{|P|}. \]

(4.12)

Now to find the feed link with the minimum detour inside the interval we can take the derivative of this formula with respect to \( z \), refer to the Appendix A for the details of this derivative.

4.3 The remaining intervals

After constructing the formula for the first interval, the formulas for subsequent intervals can be constructed in constant time by using the formula from the previous interval. This can be done as follows. Recall that we had:

\[ \delta_{P(z)}^W(P) = \frac{C_0(L) + C_1(L) \cdot L(z) + C_2(R) + C_1(R) \cdot R(z) + \text{Con}(z) + \text{Mid}(z)}{|P|}. \]

(4.13)

We will first note that the formulas \( L(z), R(z), \text{Con}(z) \) and \( \text{Mid}(z) \) can all be constructed in a straightforward manner in constant time.

This leaves \( C_0(L), C_1(L), C_2(R) \) and \( C_1(R) \) that need to be updated. Since \( C_0(L) \) and \( C_2(R) \), and \( C_1(L) \) and \( C_1(R) \) are almost identical we will only go into detail on how we update \( C_0(L) \) and \( C_1(L) \).

Now let \( s_i, s_{i+1} \) be the numbers defining the interval \( I_i \) for which we already constructed the formula, and let \( s_{i+1}, s_{i+2} \) be the numbers defining the interval \( I_{i+1} \) we want to construct the formula for next. We then need to remove the (partial) edge \( (P(s_{i+1}), P(s_{i+2})) \) from \( L \), as this edge is in \( \text{Con} \) rather than in \( L \) for \( I_{i+1} \). Furthermore the edge \( (P(\frac{|P|}{2} + s_i), P(\frac{|P|}{2} + s_{i+1})) \) needs to be added to \( L \). Finally we need to modify \( C_0 \) as dist\((v_i, s_{i+1}) < \text{dist}(v_i, s_i) \) for all edges in \( L \). See Figure 4.5. We can do that as follows:

**Algorithm** \( \text{Update-C0-and-C1}(C_0, C_1, e_0, e_1) \)

**Input:** \( C_0 \) is \( C_0(L) \) from the previous interval, \( C_1 \) is \( C_1(L) \) from the previous interval, \( e_0 \) is the edge that needs to be removed from the formula, \( e_1 \) is the edge that needs to be added.

1. Let \( C := C_1 - |e_0| \frac{1}{\sqrt{A_1t^2 + B_1t + C_1}} \text{dist}(v_i+1, P(s_0)) \)
2. Let $C_0(L) := C_0 - C[e_0] - |e_0| \int_0^1 |e_0| \sqrt{A^2 + Bt + C} dt + |e_1| \int_0^1 |e_1| \sqrt{A^2 + Bt + C} dt$

3. Let $C_1(L) := C + |e_1| \int_0^1 \sqrt{A^2 + Bt + C} dt$

Figure 4.5: An example polygon, where all edges that are in $L$ for both the current and the previous interval are shown in blue, the edge that for the current interval is in $L$ but was not for the last interval in green, and the edge that for the current interval is not in $L$ but for the previous interval was in $L$ in red.

Now using this we can find the optimal feed link for the entire polygon by simply finding the feed link with the smallest average detour in each interval and then selecting the feed link with the smallest average detour from all of these. The final algorithm will look as follows

**Algorithm** FindBestFeedlink($P, p$)

**Input:** $P$ is the Polygon, $p$ is the point we want to attach to the polygon

1. Let $I$ be the list of intervals
2. Let $F$ be the formula for the first feed link.
3. Let $m$ be the minimum of $F$ within the interval
4. for each other interval in $I$
5. do Update $F$
6. Let $m'$ be the minimum of $F$ within the interval
7. Let $m$ be the minimum of $\min(m', m)$
8. return the feed link corresponding to $m$

Now in this algorithm the first two lines both take linear time, line 3 takes constant time, the for loop runs at most $2n$ times and everything inside the for loop is constant, and line 8 also takes constant time. This yields the following theorem:

**Theorem 3.** Given the boundary $\partial P$ of a simple polygon $P$ with $n$ vertices, and a point $p$ inside $P$, a feed-link that has an optimal average detour from $\partial P$ to $p$ can be computed in $O(n)$ time.
Chapter 5

Generalising the algorithm

Given a polygon $P$ as a list of vertices $v_0, v_1, \ldots, v_{n-1}$ clockwise along its boundary $\partial P$, a point $p$ inside $P$ and an integer $k$. We want to connect $p$ to $\partial P$ with $k$ feed links such that the average detour $\delta_{Q}^{av}(P)$ is minimised. To do this, we will describe an algorithm based on the one described in the previous chapter. Similar to before we want to divide the polygon such that we can create a well behaved function for each part, that we can minimise. This will reduce the problem to finding the solution to $O(n^2 k^k)$ systems of $k$ equations in $k$ unknowns. We will first look at how we divide the polygon. Next we will show how we can create the formula for each part. After which we will show how we can optimise these formulas.

5.1 Dividing the polygon

Let

$$F(r_0, r_1, \ldots, r_{k-1}) := \delta_{Q}^{av}(P)$$

where $Q := (P(r_0), P(r_1), \ldots, P(r_{k-1}))$ is the set of points where the feed links connect from $p$ to $\partial P$, we are interested in finding the values for $r_0, r_1, \ldots, r_{k-1}$ such that $F(r_0, r_1, \ldots, r_{k-1})$ is minimised.

The idea here is similar to before, if all $r_i$ vary such that every $P(r_i)$ moves along a single edge and every midpoint $m_i$ (defined below) moves along a single edge $F(r_0, r_1, \ldots, r_{k-1})$ is well behaved, and hence we can minimise it.

The midpoint $m$ between two adjacent feed links $(p, q_0)$, $(p, q_1)$ is defined as the point where the network distance from $m$ to $p$ over $(p, q_0)$ equals the network distance from $m$ to $p$ over $(p, q_1)$ so $m$ is the point where:

$$|pq_0| + \text{dist}(q_0, m) = |pq_1| + \text{dist}(q_1, m).$$

(5.2)

See Figure 5.1.

Unlike the algorithm for $k = 1$ however we can not define simple intervals, as the location of the midpoints is determined by the connection point of both adjacent feed links. See Figure 5.2. Instead we need to find all possible middle edges for each of the $O(n^k)$ combinations of $k$ connecting edges.

Given a combination of edges $con_0, con_1, \ldots, con_{k-1}$ on which we want the feed links to connect, let $v'_2, v'_{2i+1}$ be the vertices of the edge $con_i$, with $v'_{2i+1}$ laying clockwise along $\partial P$ from $v'_2$.

Now for every consecutive pair of these connecting edges we can find all possible middle edges as follows:

Let mid$^L_i$ be the edge containing the midpoint for the feed links connected to the vertices $v'_2$ and $v'_{2i+2 \mod (2k+2)}$, and let mid$^R_i$ be the edge containing the midpoint for the feed links connected to the vertices $v'_{2i+1}$ and $v'_{2i+3 \mod (2k+2)}$. See Figure 5.3. Using this we will define...
CHAPTER 5. GENERALISING THE ALGORITHM

Figure 5.1: Example of midpoints for 2 adjacent feed links, in this figure $|pq_0| + \text{dist}(q_0, m) = |pq_1| + \text{dist}(q_1, m)$

Figure 5.2: Left: a simple polygon with feed links $(p, P(r_0)), (p, P(r_1))$ connecting to $e_0$ and $e_4$ respectively. Right the plot of the areas the midpoint may lie in depending on $r_0$ and $r_1$.

the set of edges $E_i$ for $0 \leq i < k$ as all edges clockwise of mid$^L_i$ and counter clockwise of mid$^R_i$ including mid$^L_i$ and mid$^R_i$.

Now we will need to define one function for each element in the set:

$$I = E_0 \times E_1 \times \cdots \times E_{k-1}$$

(5.3)

As this represents all possible middle edges for this set of connecting edges.

Worst case this gives us $O\left(\frac{n^k}{k}\right)$ middle edges for each combination of connecting edges.

Note however that it is possible that not all of these combinations of middle edges have valid feed link placements. This happens in the following situation: Let $e_i$ be the part of con$_i$ where the feed link $(p, q_i)$ can connect such that one of the midpoints depending on that feed link lies on a specific middle edge $m$, and let $e'_i$ be the part of con$_i$ where the feed link $(p, q_i)$ can connect such that the other midpoint depending on that feed link lies on a specific middle edge $m'$. The feed link will than not have a valid feed link placement for that combination of middle edges if there is no overlap between $e'_i$ and $e_i$. See Figure 5.4 for an example. If this is the case we do not
need to create a formula for that element in $I$, and hence we can remove it.

Once we have removed these invalid combinations of middle edges we are left with all possible combination of middle edge for each combination of connecting edges, for which we can create a formula. As these formulas are only valid for one specific combination of middle and connecting edges, after we compute any solutions we will of course have to check if that solution in fact lies on those specific connecting and middle edges.
5.2 Creating the formulas

Let \( \text{con}_0, \text{con}_1, \ldots, \text{con}_{k-1} \) be a set of edges on the boundary of \( P \). And let \( I_0, I_1, \ldots, I_m \) be the combinations of middle edges corresponding to those connecting edges. To create a formula that calculates \( \delta_Q^\text{av}(P) \) for one element \( I_i \) in that list we will divide all edges into 3 groups. The first of these groups \( \text{CON} \) is the set of all connecting edges \( \text{con}_0, \text{con}_1, \ldots, \text{con}_{k-1} \). The second of these groups \( \text{MID} \) contains all middle edges \( \text{mid}_0, \text{mid}_1, \ldots, \text{mid}_{k-1} \). The third group \( \text{LR} \) simply contains all remaining edges. See Figure 5.5 for an example.

All edges in \( \text{LR} \) will be further subdivided into the groups: \( L_0, L_1, \ldots, L_{k-1}, R_0, R_1, \ldots, R_{k-1} \). In this subdivision each group of edges \( L_i \) containing all edges that lie clockwise of \( \text{con}_i \) and counter-clockwise of \( \text{mid}_i \), and each group of edges \( R_i \) containing all edges that lie counter-clockwise of \( \text{con}_i \) and clockwise of \( \text{mid}_{(i-1) \mod k} \).

Now before we can show the formula for such a combination of edges we need to define the variables \( z_0, z_1, \ldots, z_{k-1} \) with \( 0 \leq z_i \leq |\text{con}_i| \). These variables are used as follows: each feed link \((p, q_i)\) will connect to \( \text{con}_i \) such that \( |v_2 q_i| = z_i \).

Now let

\[
\delta_Q^\text{av} := \sum_{0 \leq i < k} \frac{G_i(z_i) + \text{Mid}_i(z_i, z_{(i+1) \mod k})}{|P|}
\]  

be the average detour of the polygon for a given combination of edges, with \( G_i(z_i) \) the total contribution of all edges that connect only to \( q_i \), and \( \text{Mid}_i(z_i, z_{(i+1) \mod k}) \) the contribution of the edge \( \text{mid}_i \).

Then we know that:

\[
G_i(z_i) = C_{i0}(L_i) + C_{i1}(L_i)L_i(z_i) + C_{i2}(R_i) + C_{i3}(R_i)R_i(z_i) + \text{Con}_i(z_i),
\]  

where all factors are almost identical to the ones described for one feed link in the previous chapter.

Figure 5.5: One possible combination of connecting and middle edges for \( k = 3 \), with the connecting edges in green and the middle edges in red. And the remaining edges in black.
CHAPTER 5. GENERALISING THE ALGORITHM

The main difference in the formula to find the average detour for \( k > 1 \) rather than \( k = 1 \), is the formula \( \text{Mid}_i(z, z_{(i+1) \mod k}) \), as this depends on the placement of 2 feed links. As was the case for \( k = 1 \) we will split this edge into 2 parts, with for one part all points having the shortest network distance to \( p \) over \( q_i \) and all points in the other part having the shortest network distance to \( p \) over \( q_{i+1} \). This can be done in a fairly straightforward way.

5.3 Optimising the formula

To find the optimal placement of feed links for each combination of edges, we need to find the minimum of the formula described in the previous section. To do this we need to take the partial derivatives of this function. This gives the following system of equations:

\[
\frac{\partial}{\partial z_0} (C_{01}(L_0)L_0(z_0) + C_{03}(R_0)R_0(z_0) + \text{Con}_0(z_0) + \text{Mid}_0(z_0, z_1) + \text{mid}_{(k-1)}(z_{k-1}, z_0)) = 0
\]

\[
\frac{\partial}{\partial z_1} (C_{11}(L_1)L_1(z_1) + C_{13}(R_1)R_1(z_1) + \text{Con}_1(z_1) + \text{Mid}_1(z_1, z_2) + \text{mid}_0(z_0, z_1)) = 0
\]

\[
\frac{\partial}{\partial z_2} (C_{21}(L_2)L_2(z_2) + C_{23}(R_2)R_2(z_2) + \text{Con}_2(z_2) + \text{Mid}_2(z_2, z_3) + \text{mid}_1(z_1, z_2)) = 0
\]

\[
\cdots
\]

\[
\frac{\partial}{\partial z_{(k-2)}} (C_{(k-2)1}(L_{(k-2)})L_{(k-2)}(z_{(k-2)}) + C_{(k-2)3}(R_{(k-2)})R_{(k-2)}(z_{(k-2)}) + \text{Con}_{(k-2)}(z_{(k-2)}) + \text{Mid}_{(k-2)}(z_{(k-2)}, z_{(k-1)}) + \text{mid}_{(k-3)}(z_{(k-3)}, z_{(k-2)})) = 0
\]

\[
\frac{\partial}{\partial z_{(k-1)}} (C_{(k-1)1}(L_{(k-1)})L_{(k-1)}(z_{(k-1)}) + C_{(k-1)3}(R_{(k-1)})R_{(k-1)}(z_{(k-1)}) + \text{Con}_{(k-1)}(z_{(k-1)}) + \text{Mid}_{(k-1)}(z_{(k-1)}, z_0) + \text{mid}_{(k-2)}(z_{(k-2)}, z_{(k-2)})) = 0
\]

Solving such a system of equations is outside of the scope of this project. In [6] they have a similar system of equations however and state that it can be solved by using elimination techniques which results in a polynomial with degree double exponential in \( k \).

Once we have solved this equation we need to check if any of our solutions are valid. For a solution to be valid for the combination of edges the following 2 conditions should apply:

1. \( 0 \leq z_i \leq |c_1| \) for \( 0 \leq i < k \)
2. the point \( r_i^* \) with \( |pf_i| + \text{dist}(f_i, r_i^*) = |pf_{(i+1) \mod k}| + \text{dist}(f_{(i+1) \mod k}, r_i^*) \) for \( 0 \leq i < k \) lies on the edge \( \text{mid}_i \).

In addition to finding all valid solutions we also need to check all corner cases. Since our solution space is a polygon in \( k \) dimensional space we need to check all \( k \) dimensional planes corresponding to the sides of this polygon. Because of this we get a total of \( O(3^k) \) recursions of this algorithm.

Once we found all valid solutions and checked all recursions the solution with the minimum detour found among those solutions is the optimal feed link placement for that combination of edges.

By doing this for all combinations of edges for we can find the optimum feed link placement to minimise the average detour. This implies the following theorem:

**Theorem 4.** Given the boundary \( \partial P \) of a simple polygon \( P \) with \( n \) vertices, a point \( p \) inside \( P \), and a parameter \( k > 1 \), finding a placement for \( k \) feed-link such that the average detour from \( \partial P \) to \( p \) is minimised can be reduced to the problem of finding the solution to \( O(n^{2k}) \) systems of \( k \) equations in \( k \) unknowns.
Chapter 6

Discretizing the problem

While the optimal placement for $k$ feed links that minimise the average detour can be found exactly with the algorithm that was described in the previous section, for most practical applications we expect the algorithm to be to slow, and to complex. It therefore makes sense to look at approximation algorithms for this problem. In this section we will look at an algorithm that places a small number of Steiner points on $\partial P$ and only consider feed links that connect to those points. By doing so we can get a $(1 + \epsilon)$-approximation of the best average detour, in $O(k(\frac{n}{\epsilon})^3)$ time. Using the same ideas we can get a $(1 + \epsilon)$-approximation of the best maximum detour, in time $O(n \epsilon (k + n))$. If the polygon is convex we can further reduce this to $O(k \cdot \max(n, \frac{1}{\epsilon})^3)$ for the average detour and $O(\max(n, \frac{1}{\epsilon}) \cdot (k + n))$ for the maximum detour.

These Steiner points are placed on each edge $e$ as follows. Place the first Steiner point $q_0$ on the point on $e$ closest to $p$. Place Steiner points $q_1, \ldots, q_\ell$ such that $\angle(q_i, p, q_0) = i\epsilon$ for $1 \leq i \leq \ell$. Next, place Steiner points $q'_1, \ldots, q'_{\ell'}$ such that $\angle(q_0, p, q'_i) = i\epsilon$ for $1 \leq i \leq \ell'$. See Figure 6.1.

![Figure 6.1: The placement of the Steiner points](image)

The set of Steiner points placed along the boundary of $\partial P$ is denoted $S$. As each edge is a straight-line segment one can bound the total number of Steiner points for any simple polygon by $O(\frac{n}{\epsilon})$. If the polygon is convex we can even use the tighter bound $O(\max(n, \frac{1}{\epsilon}))$ as the number of Steiner points is bounded by either $\frac{2\pi}{\epsilon}$ or $n$ whichever is greater.

We say that a feed-link set $F^*$ is discrete if every feed-link in $F^*$ has one endpoint in $S$.

**Lemma 2.** Given any feed-link set $F$ of size $k$ on a polygon $P$, there exists a discrete feed-link set $F^*$ of size $k$ such that for every point $q$ on $\partial P$ we have $\delta_{F^*}(q, p) \leq (1 + \epsilon)\delta_F(q, p)$.

**Proof.** Consider a feed-link set $F$ with the feed-links $\{s_1, \ldots, s_k\}$. Next we construct the set $F^*$. For each feed-link $s$ in $F$ add a feed-link $s'$ to $F^*$ as follows. Let $e$ be the edge of $P$ containing one endpoint of $s$ and let $q_0$ be the point on $e$ closest to $p$. Now, set the endpoint of $s'$ to be the
Steiner point in $S$ closest to the endpoint of $s$ along $e$ such that $p$ lies closer to the endpoint of $s'$ than the endpoint of $s$. See Figure 6.2. We will next prove the set $F^*$ fulfills the statement of the lemma.

![Figure 6.2: An example of the placement of the Steiner points, $F$ and $F^*$. Steiner points and $F^*$ are shown in red. $F$ is shown in green.](image1)

Note that there exists an integer $i > 0$ such that $\angle(q_0, p, s') = (i-1)\epsilon$ and $(i-1)\epsilon \leq \angle(q_0, p, s) < i\epsilon$. Now let $s''$ be the Steiner point in $S$ such that $\angle(q_0, p, s'') = i\epsilon$ then we know that if $|ps'| + |s's''| \leq (1 + \epsilon)|ps''|$ our lemma holds. Let $t$ be the point that lies on the line over $(p, s')$ such that $\angle(s', p, s'') = 90$. See Figure 6.3

![Figure 6.3: An example of $s', s''$ and $t$](image2)

Now we know that:
\[
|ps'| + |s's''| \leq |pt| + |t's''|
= \cos(\epsilon)|ps''| + \sin(\epsilon)|ps''| \\
\leq (1 + \sin(\epsilon))|ps''| \\
\leq (1 + \epsilon)|ps''|.
\] (6.1)

Let \(m\) denote the total number of Steiner points. As a direct result of this lemma we can then find both a feed link placement such that the maximum detour is a \((1 + \epsilon)\)-approximation of the optimal maximum detour and a feed link placement such that the average detour is a \((1 + \epsilon)\) approximation of the optimal average detour in \(O(m^k(k + n))\) time by simply calculating the respective detour for each possible discrete feed link placement. In Section 6.1 we will show how we can improve the running time of the algorithm to find this approximation for the average detour to \(O(k m^3)\), and in Section 6.2 will show how we can improve the running time of the algorithm to find the approximation of the maximum detour in \(O(m(k + n))\). Note that for both of these algorithms we assume that \(S\) is sorted clockwise along \(\partial P\).

### 6.1 Finding the average detour

To find a feed link placements which approximates the minimum average detour we will use preprocessing and dynamic programming.

The main observation here is that the average detour of the part of the polygon between 2 feed links only depends on the placement of those feed links. See Figure 6.4.

![Figure 6.4: As long as the two feed-links in red remain fixed, and no additional feed-links connect to the red part, the detour of that part will not change, no matter where feed-links connect in the green part](image)
CHAPTER 6. DISCRETIZING THE PROBLEM

Note that another way to express the minimum average detour is to say that it is the total detour divided by the length of the polygon.

Let \( q_i \) denote the \( i \)th element of \( S \). We can then define the function \( F(a, b) \) with \( a, b \in \mathbb{R} \) as the total detour of the part of \( \partial P \) between \( q_a \) and \( q_b \) in clockwise direction if there is a feed-link connected to \( q_a \) and a feed-link connected to \( q_b \). We will pre-compute this for all \( a, b \) such that \( 0 \leq a \leq m \) and \( 0 \leq b < m \), this can easily be done in \( O(m^2) \).

Now we can define the dynamic program \( G(a, \ell) \) as follows: \( G(a, \ell) \) returns the minimum total detour with \( \ell \) feed-links of the part of \( \partial P \) from \( q_a \) to \( q_0 \) clockwise, under the condition that one feed-link connects to \( q_0 \) and one feed-link connects to \( q_a \) and the other feed-links to some \( q_j \) with \( a < j < m \). This function looks as follows:

\[
G(a, \ell) = \begin{cases} 
\min_{a+i \leq m-\ell} (F(a, i) + G(i, \ell - 1)), & \text{if } \ell > 2 \\
\min_{a+i \leq m} (F(a, i) + F(i, 0)), & \text{otherwise}
\end{cases}
\] (6.2)

Now if we make a call to this program with \( G(0, k) \) we get that given that one feed-link is connected to \( q_0 \) the minimum total detour in \( O(km^2) \) time. From there it is trivial to get the corresponding feed-link placement. Now to get the optimal feed-link placement we can simply set every point in \( S \) once as \( q_0 \) and call \( G(0, k) \). The result is an algorithm with a running time of \( O(km^3) \), which presents us with the following theorems:

**Theorem 5.** Given the boundary \( \partial P \) of a simple polygon \( P \) with \( n \) vertices, a point \( p \) inside \( P \), and the parameters \( \epsilon > 0 \), \( 0 < k < 1 \), a placement for \( k \) feed-links can be found in \( O(k(\frac{n}{\epsilon})^3) \) time that provides a \((1 + \epsilon)\)-approximation of the minimum average detour from \( p \) to \( \partial P \).

**Theorem 6.** Given the boundary \( \partial P \) of a convex polygon \( P \) with \( n \) vertices, a point \( p \) inside \( P \), and the parameters \( \epsilon > 0 \), \( 0 < k < 1 \), a placement for \( k \) feed-links can be found in \( O(k \cdot \max(n, \frac{1}{\epsilon})^3) \) time that provides a \((1 + \epsilon)\)-approximation of the minimum average detour from \( p \) to \( \partial P \).

### 6.2 Finding the maximum detour

To reduce the running time of the algorithm that places the feed-links such that the maximum detour is minimised we will make use of the fact that the detour at any point can only be decreased by moving an adjacent feed-link closer to it.

Now to find a \((1 + \epsilon)\)-approximation of the maximum detour we first build two list \( L_q, R_q \) for each Steiner point \( q \). Let respectively \( L_q(i) \) and \( R_q(i) \) with \( i \in \mathbb{N} \) be the \( i \)th entries in those list. Furthermore let \( A_q(i) \) and \( B_q(i) \) respectively be the maximum detour of the \( i \)th edge clockwise and counter clockwise from that point. Now set \( L_q(0) = A_q(0) \) and \( L_q(i) = \max(L_q(i-1), A_q(i)) \), and similarly \( R_q(0) = B_q(0) \) and \( R_q(i) = \max(R_q(i-1), B_q(i)) \). These lists can easily be build in time \( O(\text{num}) \).

As before let \( q_i \) denote the \( i \)th element of \( S \). Place \( k \) feed-links \( s_0, s_1, \ldots, s_{k-1} \) such that \( s_i = (p, q_i) \). We can then find the point with maximum detour, move the first feed-link that lies counter clockwise of it to the next Steiner point in \( S' \), and if the resulting detour is smaller, store the feed-link placement. We can keep repeating this process until either the point where the maximum detour is achieved lies between two adjacent Steiner points or until \( s_{k-1} \) would connect to \( q_0 \). In more detail the algorithm looks as follows:

**Algorithm MinimiseMaxDetour()**

1. Create the lists containing the detour information.
2. Place the feed-links on the Steiner points.
3. For each pair of consecutive feed-links find the point that has the same distance to \( p \) over the polygon using either feed-link. And note how many edges these points lie from the Steiner point.
4. \( \text{opt} \leftarrow \) the initial position of the feed-links along with the initial maximum detour.

New results on feed-link placement
5. while the maximum detour does not lie between two feed links connect to adjacent Steiner points feed-links and $s_{k−1}$ is not connected to $q_0$
6. do Move the feed-link counter-clockwise from the max detour to the next Steiner point clockwise.
7. Re calculate the maximum detour.
8. if the max detour is smaller than the value in $opt$ store the feed-link position and the max detour in $opt$
9. return $opt$

Line 1 of this algorithm takes $O(nm)$ as we saw earlier. Line 2,3,4 and 9 all clearly take less than that. The for-loop on line 5 to 8 runs $O(km)$ times. Line 6 can clearly be done in constant time. Line 7 can also be done in constant time as when we only move 1 feed-link we have to update the position of two midpoints. This is done by keeping an index, and looking up the maximum detour in the lists we created in the pre-processing step. When we have done that we can then calculate the maximum detour of the edges containing these mid points. Once we have done that we can compare it with the maximum detours stored in the list created in step 1 to find the maximum detour. Line 8 can also be executed in constant time as we can simply keep track of the maximum detour in the lists we created in the pre-processing step. When we have done that we can compare it with the maximum detours stored in the list created in step 1 to find the maximum detour.

In the second state, $q_0, q_1, \ldots, q_{k−1}$ are all lie clockwise from the $q_0, q_1, \ldots, q_{k−1}$. If this is the case, then we move a feed-link we will get closer to the optimal solution. And at some point we will transition to the first state.

In the second state some $q_j = q_j'$, but for others $q_j'$ still lies clockwise of $q_j$. In this state if we move any feed-link for which $s_j \neq (p, q_j')$ we will get closer to the optimal solution. We will therefore argue that it is not possible to move a feed-link if $s_j = (p, q_j')$, unless we found an optimal solution.

Let $\delta_j$ be the maximum detour clockwise between $q_j$ and $q_{j+1} \mod k$. We than know that we can only move a feed-link $s_j$ if $\delta_j = \delta^p_0(P)$. Furthermore we know that if $q_j = q_j'$ then $\delta_j$ must be equal or less to the detour achieved clockwise between $s_j'$ and $s_{j+1} \mod k−1$ in the optimal solution as we know that $q_j = q_j'$ or $q_j'$ lies clockwise of $q_j$. Therefore if $q_j = q_j'$ and the next move the algorithm makes will be to move $s_j$ we know that $\delta^p_0(P)$ is at most the detour of the optimal solution, and we have therefore found an optimal solution (after which we will be in the fourth state until we eventually terminate).

After a certain number of moves in the second state if we do not find any other optimal solutions we will be in the third state, which is the state where $q_j = q_j'$ for $0 \leq j < k$, which is what we set out to prove (after which we will be in the fourth state until we eventually terminate).

This presents us with the following theorems:

**Theorem 7.** Given the boundary $\partial P$ of a simple polygon $P$ with $n$ vertices, a point $p$ inside $P$, and the parameters $\epsilon > 0$, $k > 1$, a placement for $k$ feed-links can be found in $O(\frac{n}{k}(n + k))$ time that provides a $(1+\epsilon)$-approximation of the minimised maximum detour from $p$ to $\partial P$.

**Theorem 8.** Given the boundary $\partial P$ of a convex polygon $P$ with $n$ vertices, a point $p$ inside $P$, and the parameters $\epsilon > 0$, $k > 1$, a placement for $k$ feed-links can be found in $O(max(n, \frac{1}{2} \cdot (n + k)))$ time that provides a $(1+\epsilon)$-approximation of the minimised maximum detour from $p$ to $\partial P$.
Figure 6.5: The possible states the feed-links of \( s \) can be in. Top left is the starting position, here non of the feed-links \( s_i \) are connected to the \( q'_i \). Top middle is the state where some of the feed-links \( s_i \) are connected \( q'_i \), but the others are still connected to Steiner points that lie counter clockwise of \( q'_i \). Top right is the state where all feed-links \( s_i \) are connected to the Steiner points \( q'_i \) (the optimal solution). Bottom left shows the is the state after the optimal solution has been found. Bottom right shows a state that can not happen if there is a unique solution, where some \( s_i \) connect to Steiner points clockwise from \( q'_i \) and some to Steiner points counter clockwise from the \( q'_i \).
Chapter 7

Conclusion

In this thesis we looked at how we can best use feed links to connect disconnected nodes to a network. We studied, how we can place these feed links such that the average detour is minimised, as well as how we can place these feed links such that the maximum detour is minimised. This gave us the following problem definitions:

1. Given a polygon $P$, a source point $p$, and a constant $k$, how can we place $k$ feed links such that the average detour from $p$ to every point on the boundary $\partial P$ of $P$ is minimised.

2. Given a polygon $P$, a source point $p$, and a constant $k$, how can we place $k$ feed links such that the maximum detour from $p$ to every point on the boundary $\partial P$ of $P$ is minimised.

We first showed was that finding a solution for one of these problems does not give us any approximation for the other problem. We showed this by providing an example where a feed link $(p, q^{av})$ that minimises the average detour connects to a different part of the polygon than the feed link $(p, q^m)$ that minimises the maximum detour. For this example we than showed that the average detour of $(p, q^{av})$ was arbitrary much worse than the one of $(p, q^m)$, while the maximum detour of $(p, q^{av})$ was arbitrary much worse than the one of $(p, q^m)$.

Next we provided an algorithm that solves the first problem for $k = 1$ in linear time. It does this by dividing the boundary $\partial P$ of the polygon into $O(n)$ intervals, for each of which we can than create a formula that calculates the average detour if the feed link connects within that interval.

We then looked at a variation of the algorithm to solve the first problem for $k > 1$. This variation of the algorithm also divided $\partial P$ into a number of intervals, for each of which we can then create a formula that calculates the average detour if the feed links connects within that interval. This algorithm is more of theoretical interest however as the number of intervals is $O(n^2 k^{-1})$ and optimising the formulas for each of these intervals also takes a long time.

We therefore also looked at approximations of this problem that would be more practical applicable. This resulted in an approach where we can place a small number of Steiner points to find a $(1 + \epsilon)$-approximation for both minimising the average, and the maximum detour. For general polygon the algorithm that minimises that average detour has a running time of $O(k \cdot (\frac{2n}{\epsilon})^3)$, while the algorithm that minimises the maximum detour has a running time of only $O((\frac{2}{\epsilon})(n + k))$. For convex polygons the running time of these algorithms is further reduced to $O(k \cdot \max(n, \frac{1}{\epsilon})^3)$, and $O(\max(n, \frac{1}{\epsilon}) \cdot (n + k))$ respectively.

7.1 Further research

The research that we have done here has been mainly focused on minimising the average detour, while previous work has been mainly focused on minimising the maximum detour. A possible opportunity for further research would be to see how these different approaches relate back to the real world, and which provide a better approximation of it.
Related to this one could look at a variation of this where instead of adding $k$ edges only a certain maximum edge weight could be added. Another avenue would be to look at if these algorithms can be further generalised to look at multiple source points or more general graphs. Furthermore at the moment there is no algorithm that solves the second problem exactly, which could also provide a good research opportunity. Finally it is of course always possible to see if one can find a better algorithm for the problems presented here.
Bibliography


New results on feed-link placement
Appendix A

Finding the derivative

In Chapter 4.2 we looked at how we can compose a formula for the average detour from a point $p$ inside a polygon $P$ to every point on its boundary $\partial P$ if the feed-link $(p, P(z))$ connects to a (partial) edge $(P(s_0), P(s_1))$ of $P$.

Remember that we had:

$$\delta^z_{P(z)}(P) = \frac{C_0(L) + C_1(L) \cdot L(z) + C_2(R) + C_1(R) \cdot R(z) + Con(z) + Mid(z)}{|P|},$$

(A.1)

where $z$ is the distance from $s_0$ to $P(z)$, $C_0(L), C_1(L), C_2(R), C_1(R)$ are constants that only depend on $P$ and $p$, $L(z) = s_1 - s_0 + |P(z)p| - z$, $R(z) = z + |P(z)p|$, and Con$(z)$ and Mid$(z)$ are the formulas for the connecting and middle edge respectively.

To find the optimal connection point within this interval we will take the derivative of the formula with respect to $z$. Since $C_0(L), C_1(L), C_2(R), C_1(R)$, and $|P|$ are all constant with respect to $z$, we only have to calculate the derivative of $L(z), R(z), Con(z), Mid(z)$.

Before we can calculate $\frac{d}{dz} R(z)$ and $\frac{d}{dz} L(z)$ however we need to rewrite the $|P(z)p|$ term in terms of $z$ which gives us $R(z) = z + \sqrt{A\frac{z^2}{(s_1-s_0)^2} + B\frac{z}{s_1-s_0} + C}$ and $L(z) = s_1 - s_0 + \sqrt{A\frac{z^2}{(s_1-s_0)^2} + B\frac{z}{s_1-s_0} + C} - z$. Here $A$, $B$, and $C$ are constants depending only on the coordinates of $p$ and the endpoints of the edge $(P(s_0), P(s_1))$. Using this we get that:

$$\frac{d}{dz} R(z) = \frac{2Az + Bs_1}{2(s_1 - s_0)\sqrt{A\frac{z^2}{(s_1-s_0)^2} + B\frac{z}{s_1-s_0} + C} + 1}$$

(A.2)

$$\frac{d}{dz} L(z) = \frac{2Az + Bs_1}{2(s_1 - s_0)\sqrt{A\frac{z^2}{(s_1-s_0)^2} + B\frac{z}{s_1-s_0} + C} - 1}$$

(A.3)

To calculate $\frac{d}{dz} Con(z)$ and $\frac{d}{dz} Mid(z)$ we first need to express the Euclidean distance for a point on an edge $(v_0, v_1)$ different namely as:

$$\sqrt{(p.x - v_0.x - (v_1.x - v_2.x)t)^2 + (p.y - v_0.y - (v_1.y - v_2.y)t)^2}$$

(A.4)

where $a.x, a.y$ respectively denote the $x$ and $y$ coordinates of a point $a$.

Now using this we can express the formula for $Con(z)$ and $Mid(z)$, for $Con(z)$ this is demon-
APPENDIX A. FINDING THE DERIVATIVE

strated below:

\[
Con(z) = \int_{0}^{1} z(t) + \sqrt{A\frac{z^2}{s^2} + B\frac{z}{s_1} + C}/
\]

\[
(p.x - P(s_0).x - (P(s_1).x - P(s_0).x)\frac{z}{s_1 - s_0} - (P(s_0).x - P(s_1).x)\frac{z}{s_1 - s_0})^2
\]

\[
+ (p.y - P(s_0).y - (P(s_1).y - P(s_0).y)\frac{z}{s_1 - s_0} - (P(s_1).y - P(s_0).y)\frac{z}{s_1 - s_0})^2 dt +
\]

\[
\int_{0}^{1} (s_1 - s_0 - z)(s_1 - s_0 - z)t + \sqrt{A\frac{z^2}{s_1^2} + B\frac{z}{s_1} + C}/
\]

\[
(p.x - P(s_0).x - (P(s_1).x - P(s_0).x)\frac{z}{s_1 - s_0} - (P(s_0).x - P(s_1).x)\frac{s_1 - z}{s_1 - s_0})^2
\]

\[
+ (p.y - P(s_0).y - (P(s_1).y - P(s_0).y)\frac{z}{s_1 - s_0} - (P(s_1).y - P(s_0).y)\frac{s_1 - z}{s_1 - s_0})^2 dt. \quad (A.5)
\]

For Mid(z) the formula is virtual identical except that instead of all P(s_0) terms it has P(s_0 + \frac{|P|}{2}) and instead of all P(s_1) terms it has P(s_1 + \frac{|P|}{2}).

Now by taking the derivative of this we can find z such that δP(z)(P) is minimised within the interval.