Algorithm Engineering in Geometric Network Planning and Data Mining

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Martin P. Seybold

aus Ulm

Hauptberichter: Prof. Dr.-Ing. Stefan Funke

Mitberichter: Prof. Dr. Stefan Schirra

Tag der mündlichen Prüfung: 16.04.2018

Institut für Formale Methoden der Informatik

2018

Contents

List of Figures

Abstract

The geometric nature of computational problems provides a rich source of solution strategies as well as complicating obstacles. This thesis considers three problems in the context of geometric network planning, data mining and spherical geometry.

Geometric Network Planning In the d-dimensional GENERALIZED MINIMUM MANHATTAN NETWORK problem (d-GMMN) one is interested in finding a minimum cost rectilinear network N connecting a given set of n pairs of points in $\mathbb{R}^d$ such that each pair is connected in N via a shortest Manhattan path. The decision version of this optimization problem is known to be **NP**-hard. The best known upper bound is an $\mathcal{O}(\log^{d+1} n)$ approximation for $d > 2$ and an $\mathcal{O}(\log n)$ approximation for 2-GMMN.

In this work we provide some more insight in, whether the problem admits constant factor approximations in polynomial time. We develop two new algorithms, a 'scale-diversity aware' algorithm with an $\mathcal{O}(\mathcal{D})$ approximation guarantee for 2-GMMN. Here $\mathcal{D}$ is a measure for the different 'scales' that appear in the input, $\mathcal{D} \in \mathcal{O}(\log n)$ but potentially much smaller, depending on the problem instance. The other algorithm is based on a primal-dual scheme solving a more general, combinatorial problem – which we call PATH COVER. On 2-GMMN it performs well in practice with good a posteriori, instance-based approximation guarantees. Furthermore, it can be extended to deal with obstacle avoiding requirements. We show that the Path Cover problem is at least as hard to approximate as the HITTING SET problem. Moreover, we show that solutions of the primal-dual algorithm are $4\omega^2$ approximations, where $\omega \leq n$ denotes the maximum overlap of a problem instance. This implies that a potential proof of $\mathcal{O}(1)$-inapproximability for 2-GMMN *requires* gadgets of many different scales and non-constant overlap in the construction.

Geometric Map Matching for Heterogeneous Data For a given sequence of location measurements, the goal of the geometric map matching is to compute a sequence of movements along edges of a spatially embedded graph which provides a 'good explanation' for the measurements.

The problem gets challenging as real world data, like traces or graphs from the OpenStreetMap project, does not exhibit homogeneous data quality. Graph details and errors vary in areas and each trace has changing noise and precision. Hence, formalizing what a 'good explanation' is becomes quite difficult.

We propose a novel map matching approach, which locally adapts to the data quality by constructing what we call *dominance decompositions*. While our approach is computationally more expensive than previous approaches, our exper-

iments show that it allows for high quality map matching, even in presence of highly variable data quality without parameter tuning.

Rational Points on the Unit Spheres Each non-zero point in $\mathbb{R}^d$ identifies a closest point x on the unit sphere $\mathbb{S}^{d-1}$. We are interested in computing an ε-approximation $y \in \mathbb{Q}^d$ for x, that is *exactly* on $\mathbb{S}^{d-1}$ and has low bit-size. We revise lower bounds on rational approximations and provide explicit spherical instances.

We prove that floating-point numbers can only provide trivial solutions to the sphere equation in $\mathbb{R}^2$ and $\mathbb{R}^3$. However, we show how to construct a rational point with denominators of at most $10(d-1)/\varepsilon^2$ for any given $\varepsilon \in \left(0, \frac{1}{8}\right]$, improving on a previous result. The method further benefits from algorithms for simultaneous Diophantine approximation.

Our open-source implementation and experiments demonstrate the practicality of our approach in the context of massive data sets, geo-referenced by latitude and longitude values.

Pre Releases

Parts of this thesis were published in the following papers.

- Stefan Funke and Martin P. Seybold. The Generalized Minimum Manhattan Network Problem (GMMN) – Scale-Diversity Aware Approximation and a Primal-Dual Algorithm. In *Proceedings of the 26th Canadian Conference on Computational Geometry* (CCCG'14), Halifax, Nova Scotia, 2014.

- Daniel Bahrdt, Michael Becher, Stefan Funke, Filip Krumpe, André Nusser, Martin P. Seybold, and Sabine Storandt. Growing Balls in $\mathbb{R}^d$. In *Proceedings of the 19th Workshop on Algorithm Engineering and Experiments* (ALENEX'17), Barcelona, Spain, 2017.

- Martin P. Seybold. Robust Map Matching for Heterogeneous Data via Dominance Decompositions, In *Proceedings of the 2017 SIAM International Conference on Data Mining* (SDM'17), Houston, USA, 2017.

- Daniel Bahrdt and Martin P. Seybold. Rational Points on the Unit Sphere: Approximation Complexity and Practical Constructions. In *Proceedings of the 42nd International Symposium on Symbolic and Algebraic Computation* (ISSAC'17), Kaiserslautern, Germany, 2017.

Zusammenfassung

Die geometrische Gestalt von Berechnungsproblemen liefert vielfältige Lösungsstrategieen aber auch Hindernisse. Diese Arbeit betrachtet drei Probleme im Gebiet der geometrischen Netzwerk Planung, des geometrischen Data Minings und der sphärischen Geometrie.

Geometrische Netzwerk Planung Im d-dimensionalen Generalized Minimum Manhattan Network Problem (d-GMMN) möchte man ein günstigstes geradliniges Netzwerk finden, welches jedes der gegebenen n Punktepaare aus $\mathbb{R}^d$ mit einem kürzesten Manhattan Pfad verbindet. Es ist bekannt, dass die Entscheidungsvariante dieses Optimierungsproblems **NP**-hart ist. Die beste bekannte obere Schranke ist eine $\mathcal{O}(\log^{d+1} n)$ Approximation für $d > 2$ und eine $\mathcal{O}(\log n)$ Approximation für 2-GMMN.

Durch diese Arbeit geben wir etwas mehr Einblick, ob das Problem eine Approximation mit konstantem Faktor in polynomieller Zeit zulässt. Wir entwickeln zwei neue Algorithmen. Ersterer nutzt die ‘Skalendiversität’ und hat eine $\mathcal{O}(\mathcal{D})$ Approximationsgüte für 2-GMMN. Hierbei ist $\mathcal{D}$ ein Maß für die in Eingaben auftretende ‘Skalen’. $\mathcal{D} \in \mathcal{O}(\log n)$, aber potentiell deutlichen kleiner für manche Problem Instanzen. Der andere Algorithmus basiert auf einem Primal-Dual Schema zur Lösung eines allgemeineren, kombinatorischen Problems, welches wir Path Cover nennen. Die praktisch erzielten a posteriori Approximationsgüten auf Instanzen von 2-GMMN verhalten sich gut. Dieser Algorithmus kann für Netzwerk Planungsprobleme mit Hindernis-Anforderungen angepasst werden. Wir zeigen, dass das Path Cover Problem mindestens so schwierig zu approximieren ist wie das Hitting Set Problem. Darüber hinaus zeigen wir, dass Lösungen des Primal-Dual Algorithmus $4\omega^2$ Approximationen sind, wobei $\omega \leq n$ die maximale Überlappung einer Probleminstanz bezeichnet. Daher *müssen* potentielle Beweise, die konstante Approximationen für 2-GMMN ausschließen möchten, Instanzen mit vielen unterschiedlichen Skalen und nicht konstanter Überlappung konstruieren.

Geometrisches Map Matching für heterogene Daten Für eine gegebene Sequenz von Positionsmessungen ist das Ziel des geometrischen Map Matchings eine Sequenz von Bewegungen entlang Kanten eines räumlich eingebetteten Graphen zu finden, welche eine ‘gute Erklärung’ für die Messungen ist.

Das Problem wird anspruchsvoll da reale Messungen, wie beispielsweise Traces oder Graphen des OpenStreetMap Projekts, keine homogene Datenqualität aufweisen. Graphdetails und -fehler variieren in Gebieten und jeder Trace hat wechselndes Rauschen und Messgenauigkeiten. Zu formalisieren, was eine ‘gute Erklärung’ ist, wird dadurch schwer.

Wir stellen einen neuen Map Matching Ansatz vor, welcher sich lokal der Datenqualität anpasst indem er sogenannte Dominance Decompositions berechnet. Obwohl unser Ansatz teurer im Rechenaufwand ist, zeigen unsere Experimente, dass qualitativ hochwertige Map Matching Ergebnisse auf hoch variabler Datenqualität erzielbar sind ohne vorher Parameter kalibrieren zu müssen.

Rationale Punkte auf Einheitssphären Jeder, von Null verschiedene, Punkt in $\mathbb{R}^d$ identifiziert einen nächsten Punkt x auf der Einheitssphäre $\mathbb{S}^{d-1}$. Wir suchen eine ε-Approximation $y \in \mathbb{Q}^d$ für x zu berechnen, welche *exakt* auf $\mathbb{S}^{d-1}$ ist und niedrige Bit-Größe hat. Wir wiederholen untere Schranken an rationale Approximationen und liefern explizite, sphärische Instanzen.

Wir beweisen, dass Floating-Point Zahlen nur triviale Lösungen zur Sphären-Gleichung in $\mathbb{R}^2$ und $\mathbb{R}^3$ liefern können. Jedoch zeigen wir die Konstruktion eines rationalen Punktes mit Nennern die maximal $10(d-1)/\varepsilon^2$ sind für gegebene $\varepsilon \in \left(0, \frac{1}{8}\right]$, was ein bekanntes Resultat verbessert. Darüber hinaus profitiert die Methode von Algorithmen für simultane Diophantische Approximationen.

Unsere quell-offene Implementierung und die Experimente demonstrieren die Praktikabilität unseres Ansatzes für sehr große, durch geometrische Längen- und Breitengrade referenzierte, Datensätze.

Introduction

Algorithm Engineering denotes the complete process of actually finding solutions to problems raised by applications. This includes the interactions between formalizing a computational objective that meets application needs, analyzing theoretical aspects within the formalized frame and solving the objective with available machinery. While either one of these scientific directions has merit on its own, solely focusing on single aspects of this interaction can very well lead to huge gaps between algorithm theory and applicability or computation goals and application needs. For instance, an algorithm with a polynomial bound on the number of operations that has huge constants or an enormous degree might well be insufficient for even the smallest problem instances in an application domain. On the other hand, an **NP** hardness proof for a computational objective might well construct instances that do not occur in an application domain. A similar gap exists between application needs and formalized computation goals. A relaxed goal might allow a concise formalization and efficient, implementable algorithms, but lack to meet application needs. Conversely, heuristics that work well on machinery and instances of an application domain might well have an unclear computational objective – putting results for yet unobserved instances in the application domain in questionable light.

Many applications deal with objects of some geometry and the geometric approach to problems provides a rich source of solutions strategies as well as complicating obstacles. Each of the three chapters of this thesis eventually provides algorithms that allow for implementation and execution on contemporary computing hardware. The contents of Chapter 1 mainly relate to algorithm theoretic aspects, Chapter 2 to the aspect of objective formalization and Chapter 3 to aspects of computing machinery.

Model of Computation

In order to provide some concise statement on how complex the computational task is, one conveniently switches from Turing Machines to the RANDOM ACCESS MACHINE (RAM). The RAM can store arbitrary large integers in each of its infinitely many cells, perform exact arithmetic operations, comparisons and cell access, by the value of another, in unit time. By means of successive squaring, such machines can well double the bit-size of a number in each step. Note that despite such a model does not provide additional power in terms or problem decidability, such machines can decide **NP**-complete problems in polynomial time. E.g. the satisfiability problem in conjunctive normal form (CNF) allows arithmetic coding of the variables appearing in a clause as integers. Transferring the CNF in a disjunctive normal form via the distributive law for $\vee, \wedge$ can be achieved with the arithmetic operations $+, *$. The decision is due checking if there is a clause not simultaneously containing a variable and its negation by means of checking the respective bits of their arithmetic coding [Sch79].

Therefore, such unrealistic computational power is usually limited in the RAM by a cost measure of arithmetic operations that depends on the operands bit-size. A more pragmatical solution for algorithm analysis is to provide, along with bounds in uniform cost measure, an upper bound on the magnitude of the biggest integer that is computed during the algorithm – this bounds simulation time on non-uniform cost machines. Analyzing problems and algorithms in Euclidean geometry often requires to deal with $\sqrt[2]{\,}$, hence one allows the machine to exactly handle numbers of $\mathbb{R}$ in each cell – the Real RAM model. Unrealistic computational power appears in a similar fashion if exact rounding to the closest integer is allowed. Therefore, authors consider machines that can only apply rational functions to cells [BSS89] or limit the comparison operations to a fixed precision [BH98].

Chapter 1

The Generalized Minimum Manhattan Network Problem

A shortest Manhattan path for two points $s, t \in \mathbb{R}^d$ is a sequence of connected, axis parallel line segments of length $\|s - t\|_1$ and therefore contained in the axis aligned bounding box, called $\operatorname{box}(s, t)$. We study the d-dimensional Generalized Minimum Manhattan Network problem (d-GMMN):

Input: A set $R = \{(s_1, t_1), \dots, (s_n, t_n) \ : \ s_i, t_i \in \mathbb{R}^d\}$ of terminals pairs.

Goal: Determine a finite set N of axis parallel line segments of minimum total length that contains a shortest Manhattan path for each pair.

With $c(N) = \sum_{(p,q) \in N} \|p - q\|_1$, we denote the total length of the line segments in N. We further call $c(N)$ the *cost* of a solution, which is always non-negative.

An algorithm that outputs a solution within a number of operations, that is polynomial in the input size, is called approximation algorithm. Further, it is called *α-approximation* if, for all problem instances, the solution's cost is within a factor of α of the cost of an optimal solution. Hence, $\alpha \geq 1$ for minimization problems with non-negative cost functions.

The problem is closely related to the rectilinear Steiner network problem, where the goal is to connect designated pairs in a minimum cost network but not necessarily on shortest paths. In the context of circuit design ($d = 2$ or $d = 3$) – one of the main application areas of many Steiner-type problems – restricting to shortest paths for interconnection corresponds to keeping the latency low. See Figure 1.1 for an example.

Related Work

d-GMMN is a generalization of the d-dimensional Minimum Manhattan Network problem (d-MMN; all pairs over T are present in R, for a set of terminal points $T \subseteq \mathbb{R}^d$) and the d-dimensional Rectilinear Steiner Arborescence problem (d-RSA, each pair in R contains the origin as one of its elements). The

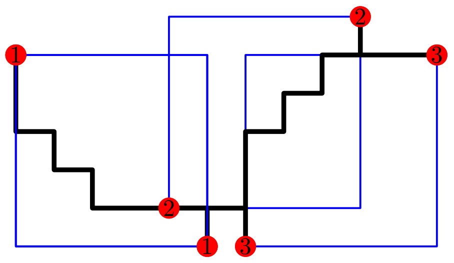


Figure 1.1: A 2-GMMN instance with 3 terminal pairs (red), axis-parallel bounding boxes of terminal pairs (blue), and line segments of a solution (black).

decision problems of 2-RSA and 2-MMN are strongly **NP**-complete [SS05, CGS11]. Unless $\mathbf{P} = \mathbf{NP}$, there is no algorithm scheme that achieves approximations within $(1+\varepsilon)$ of the optimal cost in a time that is polynomial in the number of terminals and in $1/\varepsilon$. Such algorithm schemes are called Fully Polynomial Time Approximation Scheme (FPTAS).

For 2-RSA several $\mathcal{O}(1)$-approximation algorithms are known. The algorithm given in [RSHS92] is conceptually simple and achieves a 2-approximation. There is also a polynomial time approximation scheme (PTAS) based on Arora's shifting-technique [LR00].

The situation for MMN problems is slightly different. We know that 3-MMN does not admit a PTAS, unless $\mathbf{P} = \mathbf{NP}$ [Eng10, MSU09]. [GLN01] gives an $\mathcal{O}(1)$-approximation. Several subsequent papers improve on the running time and the constant factor of the $\mathcal{O}(1)$-approximation. [CNV08] gives the first 2-approximation for 2-MMN. Their approach is based on a multiphase flow ILP formulation of polynomial size and iteratively rounding an optimal fractional solution. The authors introduce 2-GMMN in their discussion section and point out, that their approach might *not* translate easily.

Das et al. [DFK+17] provide a $\mathcal{O}(\log^{d+1} n)$ approximation for d-GMNN with n terminal pairs. They could even prove an $\mathcal{O}(\log n)$ upper bound for $d = 2$. Their approach follows the divide-&-conquer paradigm by subsequently solving sets of terminal pairs which can be connected via a common point in space; these base cases are solved with a known d-RSA approximation algorithm. They also provide an instance showing the analysis for their algorithm is essentially tight (not excluding other, better algorithms).

Contribution

We study theoretical aspects of the d-GMMN problem that lead to a somehow practical algorithm. On the theoretical side, we state simple decomposition and scaling properties that clarify that d-GMMN problem instances, whose associated geometric intersection graph has degree of at most Δ, allow simple $(\Delta + 1)$-approximations.

The results in Section 1.1.3 are along the lines of [DFK+17]. However, the approximation ratio of our 'scale-diversity aware' algorithm depends rather on the *scale diversity* $\mathcal{D}$ of the input, than on its arrangement. More concretely, we show that our algorithm computes an $\mathcal{O}(\mathcal{D})$ approximation to the 2-GMMN problem. Since $\mathcal{D} \in \mathcal{O}(\log n)$, this result always matches the result in [DFK+17], but is better if the scale diversity of R is small – e.g. $\mathcal{D} \in o(\log n)$.

Section 1.2 establishes the so-called Hanan property via a constructive plane sweeping argument. This reduces d-GMMN to a combinatorial problem on a graph of polynomial size, for fixed dimension d. We call this problem Path Cover. Regarding approximability in polynomial time, one may well assume that every polynomial time approximation algorithm for the geometric problem outputs a set of Hanan grid edges – one simply uses the plane sweeping as post-processing.

Apart from brute force algorithms, combinatorial problems allow for integer linear programming (ILP) formulations. Cut based formulations seem ill-suited to formalize the problem's shortest path requirements. We turn to a more flexible formulation that bases on, what we call, separation sets. Such formulations not only model the connectivity requirements of d-GMMN in the Hanan grid graph but the potentially more general Path Cover problem as well. We derive a primal-dual scheme for the Path Cover problem and show that solutions of this algorithm have costs within a factor of $4\omega^2$ of the optimum, where $\omega \in \{1, \ldots, n\}$ is the maximum number of pairs that can share an edge of the graph in solutions. In terms of the geometric intersection graph of the d-GMMN instance, ω denotes the clique number. Moreover, we extend the in-approximability results of the well known Hitting Set problem with a reduction to Path Cover.

On the practical side, the primal-dual approach produces lower bounds during its execution which – at least in our experiments with an implementation for $d = 2$ – turn out to be very close to the costs of the computed networks.

Chapter Outline and Pre-Releases

This chapter starts with observations on basic decomposition properties in Section 1.1 that lead to the 'scale-diversity aware' algorithm. Apart from the $(\Delta+1)$-approximation, these results can also be found in our contribution to the proceedings of the 26th Canadian Conference on Computational Geometry [FS14].

Section 1.2 establishes the Hanan property and thereby the reduction to the Path Cover problem. This statement, with focus on the existence of optimal solutions to the geometric d-GMMN problem, is contained in [FS14], as well. Our new results of Section 1.3 clarify that the (potentially) more general Path Cover problem is at least as hard to approximate as the Hitting Set problem. Section 1.4 reviews cut based combinatoric network design problems.

Section 1.5 describes the separation sets, which are the basis or our ILP formulation to the Path Cover problem. Section 1.6 describes the primal-dual method which is naturally linked to such separation set formulations. As presented here, the method avoids redundant constraints in the ILP formulation and now provides better lower bounds for a particular family of instances than our version in [FS14]. Section 1.7 provides a new, upper bound on the approximation ratio of the primal-dual algorithm for the Path Cover problem.

Section 1.8 discusses issues for a practical implementation and Section 1.9 provides experimental results for the primal-dual algorithm. Section 1.10 concludes this chapter with a brief summary on the approximability results for d-GMMN and potential extensions and improvements of the presented methods.

1.1 Scale-Diversity Aware Approximation

Let us first make some general observations about decomposition properties in the d-GMNN problem.

Lemma 1.1.1 ([FS14])**.** *The cost of an optimal solution to any subset R' of a GMMN instance R is a lower bound to the cost of an optimal solution for R.*

Proof. Consider an optimal solution N for R having a strictly lower cost than an optimal solution N' for R'. However, N contains a shortest path for each pair in R'. A contradiction to the optimality of N'. □

This simple lemma gives rise to the following decomposition property of GMMN instances.

Lemma 1.1.2 ([FS14])**.** *Let $R = R_1 \cup \ldots \cup R_k$ (not necessarily disjoint). If each N_i is an α_i-approximation for R_i, then $N = \bigcup_i N_i$ is a solution for R with $c(N) \leq \mathrm{OPT}(R) \cdot \sum_i \alpha_i$.*

Proof. N clearly connects each pair in R. By Lemma 1.1.1 we have $\mathrm{OPT}(R_i) \leq \mathrm{OPT}(R)$ and $c(N) \leq \sum_i c(N_i) \leq \sum_i \alpha_i \cdot \mathrm{OPT}(R_i) \leq \sum_i \alpha_i \cdot \mathrm{OPT}(R)$. □

This Lemma already provides some insight regarding approximability of d-GMMN. We consider the geometric intersection graph of the associated boxes of the terminal pairs in problem instance R. That is the simple, undirected graph $G = \big(\{v_1, \ldots, v_n\}, E\big)$ with $E = \{\{v_i, v_j\} : \mathrm{box}(s_i, t_i) \cap \mathrm{box}(s_j, t_j) \neq \emptyset\}$.

Theorem 1.1.3. *Let Δ denote the maximum degree of vertices in the geometric intersection graph of a d-GMMN instance R. Then R allows a $(\Delta+1)$-approximation within $\mathcal{O}(n^2)$ time.*

Proof. Building a graph representation for the instance R takes no more than $\mathcal{O}(n^2)$ operations. For a graph, with maximum degree Δ, it is simple to color the vertices with at most $\Delta+1$ colors, such that adjacent vertices are in different color classes [Die10]: We color the vertices in a fixed sequence $v_1, \dots, v_n$ by greedily assigning the smallest, free color to vertex v_i.

Now, the set of vertices with an equal color is an independent set. Moreover, a trivial assignment of line segments to these terminal pairs is an optimal solution for this subset of terminal pairs. Given above Lemmas 1.1.2 and 1.1.1, the union of $\Delta + 1$ trivial solutions provides a solution as stated. □

1.1.1 Shape Properties

Let us now turn to more shape-dependent properties of GMMN. We first show that if all terminal pairs $(s,t) \in R$ exhibit the same shape, that is, if in one dimension every terminal pair has 'about' the same extent, then we can decompose R into constantly many, not necessarily disjoint instances. Each of these instances has a very special structure, which allows for a constant approximation. Recall that we associate with each pair $(s,t) \in R$ the minimum area, axis-parallel box, having s and t as corners. Moreover, we denote the distance of points $(a_1,a_2),(b_1,b_2) \in \mathbb{R}^2$ in the dimension $j \in \{1,2\}$ with $d_j\big((a_1,a_2),(b_1,b_2)\big) = |a_j - b_j|$.

Lemma 1.1.4 ([FS14], Shapes in 2-GMMN)**.** *Let $\gamma \geq 0$ be constant and R a 2-GMMN instance. If for one dimension $j \in \{1,2\}$ each pair $(s,t) \in R$ has $\gamma < d_j(s,t) \leq 2\gamma$, then R can be decomposed into a constant number of (not necessarily disjoint) instances $R = R_0 \cup R_1 \cdots \cup R_5$. Boxes in each R_i have either a common axis parallel intersection line or no intersection at all.*

Proof. Let j be the dimension fulfilling the shape property. Consider lines that are axis-orthogonal to dimension j with distance γ in dimension j. These lines are axis-parallel to the other dimension. Each box of a pair $(s,t) \in R$ intersects at least one and at most three lines. If boxes r and r' contain lines i and i' respectively, then $|i - i'| \geq 6$ implies that $r \cap r' = \emptyset$. This gives rise to the decomposition where $R_i \subseteq R$ consists of all pairs that contain a line k with $k \equiv i \mod 6$ in their box. □

See Figure 1.2 for an illustration of this statement and proof.

1.1.2 Scale Properties

We call a d-GMMN instance R' *scaled*, if it is derived from an instance R by dividing each coordinate of a terminal by a fixed $\sigma > 0$. We have an one-to-one correspondence between the original and the scaled solutions, because a line

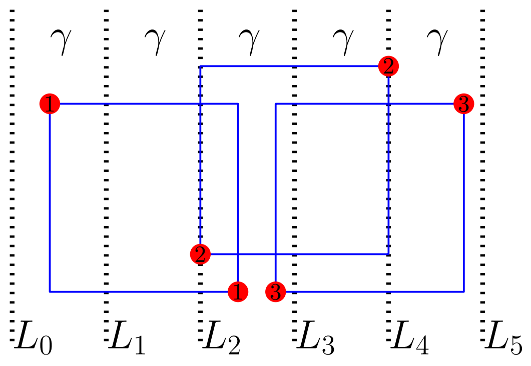


Figure 1.2: Illustration of Lemma 1.1.4 on shapes in a 2-GMMN problem with 3 pairs (blue rectangles) of terminals (red points).

segment between two points can be scaled up or down in the same way. Moreover, the cost of a network is σ times the cost of the corresponding network. If we have an instance of d-GMMN where some of the boxes associated with terminal pairs are very 'small' compared to the other boxes, we can essentially connect them naively without losing more than a constant factor in the total connection cost.

Lemma 1.1.5 ([FS14]). *Let R be a d-GMMN instance with $n = \max_{(s,t)\in R} ||s-t||_1$. Further, let $R_\varepsilon := \{(s,t) \in R : ||s-t||_1 \leq d\}$. If N' is an α-approximation for the instance $R \setminus R_\varepsilon$, then R can be approximated within $\mathcal{O}(\alpha)$.*

Proof. Since there exists a (s,t) with $||s-t||_1 = n$, we know that the cost of the optimal solution to R must be at least n. Connecting all terminal pairs in R_ε has cost at most $(n-1)d$. The lemma follows. □

1.1.3 Scale-Diversity Aware Approximation for 2-GMMN

The algorithm in this section is based on the combination of decomposition, shape and scale properties. For a set $U \subseteq (1,n]$ of numbers, we denote with

$$g(U) := |\{i \in \mathbb{N}_0 \mid \exists u \in U : 2^i < u \leq 2^{i+1}\}|$$

the *scale diversity* of U. Intuitively, $g(U)$ describes how many different magnitudes of numbers appear in the set U. Clearly $g(U) \in \mathcal{O}(\log n)$. In a 2-GMMN instance, each pair $(s,t) \in R$ naturally gives rise to 2 distance values $d_j(s,t)$ – their distance in the j-th coordinate. The preceding section argues that scaling, such that the biggest l_1 distance of a pair is exactly n, does not affect the form of solutions. After scaling, pairs with l_1 distance of no more than a constant can be neglected when aiming for a constant approximation. With $U_j := \{\ d_j(s,t) \in (1,n] : (s,t) \in R\ \}$, we denote the spread in dimension j of the pairs, after scaling. The quantity

$$\mathcal{D} := \max\{g(U_1), g(U_2)\}$$

denotes the *scale diversity* of R and essentially captures how many really different magnitudes of spreads with respect to the maximum extent appear in R.

We borrow the following lemma to solve the partitions in each of the six covers of one shape class. Note that this algorithm uses a PTAS (based on Arora's shifting technique) for 2-RSA as sub-procedure.

Lemma 1.1.6 (Lemma 7 in [DFK+17])**.** *Let R be a 2-GMMN instance. If all boxes of R have a common, axis-parallel intersection line, then R can be approximated within $\mathcal{O}(1)$.*

```
1. Scale the instance $R$ such that $|R| = \max_{(s,t)\in R}||s-t||_1$.
2. Partition $R$ into
   $R_\varepsilon = \{(s,t) \in R \ : \ ||s-t||_1 \leq 2\}$,
   $R_1 = \{(s,t) \in R \setminus R_\varepsilon : d_1(s,t) \geq d_2(s,t)\}$,
   and $R_2 = R \setminus (R_1 \cup R_\varepsilon)$.
3. Partition $R_1$ into $g(U_1)$ shape classes and solve.
4. Partition $R_2$ into $g(U_2)$ shape classes and solve.
5. Solve $R_\varepsilon$ trivially.
```

Algorithm 1: Scale-Diversity Aware Approximation

Essentially, after scaling we partition the problem instance into three instances R_ε, R_1 and R_2, where R_ε contains terminal pairs with 'very small' boxes, R_1 all terminal pairs whose box is wider than tall, and R_2 the remaining ones. Then for each R_i we consider the scale classes in dimension i and solve each of them using Lemmas 1.1.4 and 1.1.6. The R_ε are solved trivially.

Theorem 1.1.7 ([FS14])**.** *If R is a 2-GMMN instance with scale diversity $\mathcal{D} = \max\{g(U_1), g(U_2)\}$, then Algorithm 1 computes an $\mathcal{O}(\mathcal{D})$ approximate solution.*

Proof. The scaling property preserves optimality of solutions. The pairs in R_ε can essentially be ignored according to Lemma 1.1.5. R_1 can be decomposed into $g(U_1)$ shape classes of boxes. Using Lemma 1.1.4, each shape class in R_1 can again be decomposed into a constant number of instances each of which allows a partition in boxes sharing a common intersection line or not having an intersection at all. The disjoint parts of these instances with the common intersection line are solved with Lemma 1.1.6 within a constant factor of their optimum – their disjoint union remains within a constant factor of their optimum. Using Lemma 1.1.2 the approximation ratio of the possibly non-disjoint union follows. The same argument holds for R_2. □

Clearly $\mathcal{D} \in \mathcal{O}(\log n)$ but might be smaller in some applications. We give an example that occupies many shape classes. Let s_x denote the point $(x, 0)$ and t_y the point $(0, y)$. The arrangement $\{s_{n/2}, s_{n/4}, \ldots, s_{1/2}\} \times \{t_{n/2}, t_{n/4}, \ldots, t_{1/2}\}$ has an optimal solution of cost n. Consider the 2-GMMN instance R that contains $n/\log_2^2(n)$ disjoint copies of this arrangement ($|R| = n$). We have an optimal solution of cost $n^2/\log_2^2(n)$ and all shape classes in dimension 1 and 2 are occupied with $g(U_1), g(U_2) \in \Theta(\log n)$.

1.2 Reduction to a Combinatorial Problem

Let us first make the following observation which reduces the potentially very large number of line segments to consider for a solution network.

1.2.1 Restriction to the Hanan Grid

We consider the undirected, simple graph $\mathcal{H}(R)$ induced by the instance R. Let P_i be the projection of R onto the i-th coordinate. The vertices are the Cartesian product $\prod_{i=1}^{d} P_i$ and have an edge if and only if they are identical in all, but neighbored in one, coordinate. We call the difference in this coordinate the cost c_e of the edge e. Any d-GMMN instance with n pairs has a Hanan Grid of size at most $(2n)^d$ vertices and $\mathcal{O}(d(2n)^d)$ edges. This is polynomial for fixed d. See Figure 1.3 for an example. Given two vertices we call a simple path connecting them a *monotonous path* (m-path) if the sequence of the coordinates of the vertices along the path is monotonous in each dimension.

The Hanan grid is known to be a valuable tool for many geometric problems in the rectilinear setting. We adapt a simple proof that follows the argument for the 2D Rectilinear Steiner Tree Problem over weighted regions [Zac01]. A similar, but far more lengthly, argument for the d-dimensional Rectilinear Steiner Minimal Tree Problem is given in [Sny92].

Theorem 1.2.1 ([FS14]). *Let R be a d-GMMN instance and $\mathcal{H}(R) = (V, E)$ the associated Hanan grid. For any solution N to R, there is a solution $\overline{N} \subseteq E$ to R with*

$$c(\overline{N}) \leq c(N) .$$

The time to compute the edge set $\overline{N}$ is polynomial in the cardinality of N and E.

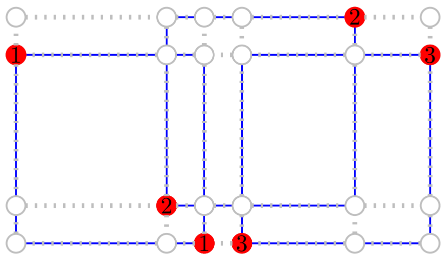


Figure 1.3: Example Hanan Grid $\mathcal{H}(R)$ for the 2-GMMN instance $R = \Big\{\{(0,1.5),(1.5,0)\},\{(1.2,.3),(2.7,1.8)\},\{(1.8,0),(3.3,1.5)\}\Big\}$. Terminals are indicated in red and terminal pairs with blue rectangles.

Given a solution N of line segments for R. We call a point in the cut of at least two segments a *node*. This proof constructs a set of line segments in which nodes coincide with vertices of $\mathcal{H}(R)$. We apply an argument that orthogonally sweeps a hyperplane over one dimension after the other (c.f. Figure 1.4):

Proof. Let N be a solution to R. We describe the sweep over the x_1 dimension. After the sweep, the x_1 coordinate of each node in N will be identical to one in P_1.

Consider the non-vertex nodes of N with maximum x_1 coordinates and their hyperplane h containing them. Now 'above' or 'below' h denotes that a point has a higher or respectively lower x_1 coordinate than h. Inductively, every node above h already has x_1-coordinates as desired. Let ε^+ denote the distance in x_1 to the next above coordinate in P_1 and ε^- the distance in x_1 to the smaller of either the next lower set of such nodes of N or to the next lower point in P_1. Let also S^+ and S^- denote the set of line segments parallel to x_1 and incident to a node in h above and below respectively. The change in total cost for jointly moving the line segments of N, that are contained in h, along the x_1 direction by $\delta \in [-\varepsilon^-, \varepsilon^+]$ is

$$\delta\Big(|S^-| - |S^+|\Big) .$$

If $|S^+| \geq |S^-|$ we simply move the segments in h upwards by ε^+ otherwise we move them downwards by ε^-. The total costs do not increase in either case. Since h does not contain terminals, jointly moving all nodes in h does not violate monotonicity of any m-path between terminals in N. After this sweep, the x_1 coordinates of a node in N is identical to one in P_1 and the total number of non-vertex nodes did not increase. After d sweeps, no non-vertex nodes are left. $\square$

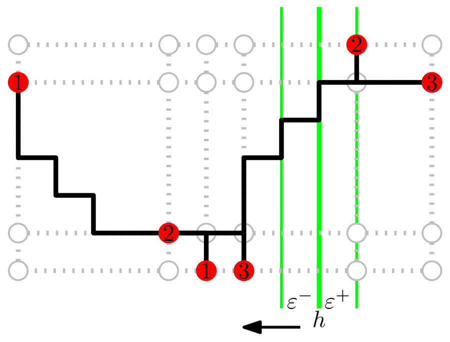


Figure 1.4: Illustration of the proof of Theorem 1.2.1.

The sweeping hyperplane method of above's proof is constructive. In terms of polynomial time approximability of d-GMMN, one may well assume that *any* d-GMMN approximation algorithm returns a set of edges of the respective Hanan grid as solution – Post-processing with aboves method requires only a polynomial time overhead and does not increase solution costs.

Corollary 1.2.2 ([FS14])**.** *For any d-GMMN instance R, there is an optimal solution $\overline{N}$ using only edges of $\mathcal{H}(R)$ as line-segments.*

This property of the Hanan grid enables us to compute an optimal solution to d-GMMN with a brute-force approach. Every feasible network consists of a covering of n m-paths, connecting a pair of terminals each. The number of m-paths for two vertices is generally exponential in n. For small instances however, one can enumerate all feasible networks living on the Hanan grid to find optimal solutions to the d-GMMN problem. Moreover, if the problem instance has a simple combinatorial structure, one can use a dynamic programming approach on the Hanan grid to obtain an optimal solution in polynomial time: one of these cases is that the intersection graph of the boxes, that are associated with the terminal pairs of a d-GMMN instance, has constant degree and constant tree-width. Such an overlay tree structure can guide a dynamic programming approach to solve level-wise in a leaf-to-root fashion by storing subtree costs for each of the constant interface configurations [Sch15].

1.2.2 The Path Cover Problem

Given an undirected graph $G = (V, E)$ with non-negative edge costs $c : E \to \mathbb{Q}$ and a set of n vertex pairs $\big\{\{s_i, t_i\} \subseteq V : 1 \leq i \leq n\big\}$. The goal of the Path Cover problem is to choose a path $p_i \subseteq E$, from the set of shortest paths between s_i

and t_i, for each $i \in \{1, \dots, n\}$ such that $\sum_{e \in N} c(e)$ is minimal, where $N = \bigcup_{i=1}^{n} p_i$ denotes the set of edges contained in these paths.

Clearly, the problem is trivial on instances with unique shortest paths between each vertex pair. Due to Corollary 1.2.2, d-GMMN is a special case of the path cover problem on the Hanan grid graph.

1.3 Reducing Hitting Set to Path Cover

We consider the unweighted Hitting Set problem over the universe $U = \{e_1, \dots e_m\}$. Given a family of n subsets $\mathcal{S} = \{S_i \subseteq U : 1 \leq i \leq n\}$, the goal is to find a minimum cardinality set $H \subseteq U$ such that $H \cap S_i \neq \emptyset$ for all i. This problem is equivalent to cardinality Set Cover by interchanging the role of sets and elements – The set cover problem seeks to cover all n elements with a minimum number sets. The greedy algorithm for set cover, that is repeatedly choosing a set that covers a maximum number of currently uncovered elements, is well known to provide approximations for the set cover problem no worse than a factor H_n [Vaz03]. Where $H_n = \sum_{i=1}^{n} 1/i$ denotes the n-th harmonic number and $H_n \leq 1 + \int_1^n \frac{1}{x} dx = 1 + \ln n$. [Fei98] shows that, for any constant $\delta > 0$, the existence of an $(1-\delta)\ln n$ approximation algorithm for cardinality set cover implies $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\mathcal{O}(\log\log n)})$. Even more, it is **NP**-hard to approximate within a factor $(1-\delta)\ln n$ for every $\delta > 0$ [DS14]. In this sense, the greedy algorithm provides the best achievable approximation factor for the set cover and hitting set problem.

Let $U_1 = \{u_{1,1}, \dots, u_{1,m}\}$ and $U_2 = \{u_{2,1}, \dots, u_{2,m}\}$ denote two disjoint copies of U. We define an undirected graph $G_{\mathcal{S}} = (V, E)$ on

$$V = \{s_1, t_1, \dots, s_n, t_n\} \cup U_1 \cup U_2 \ .$$

With $C_1 = \{\{s_i, u_{1,j}\} : e_j \in S_i\}$ and $C_2 = \{\{t_i, u_{2,j}\} : e_j \in S_i\}$, the edge set is

$$E = \big\{\{u_{1,i}, u_{2,i}\} : 1 \leq i \leq m\big\} \cup C_1 \cup C_2 \ .$$

See Figure 1.5 for an example. Let $\varepsilon = 1/(2\sum_{i=1}^{n} |S_i|)$. We set the edge costs to

$$c(e) = \begin{cases} \varepsilon & \text{for } e \in C_1 \cup C_2 \\ 1 & \text{otherwise} \end{cases}$$

and the shortest path requirement pairs to

$$C_1 \cup C_2 \cup \big\{\{s_i, t_i\} : 1 \leq i \leq n\big\}.$$

In this Path Cover problem instance, one is required to connect $n + 2\sum_{i=1}^{n} |S_i| \in \mathcal{O}(nm)$ node pairs with shortest paths in $G_{\mathcal{S}}$ (c.f. Section 1.2.2). Since all edges

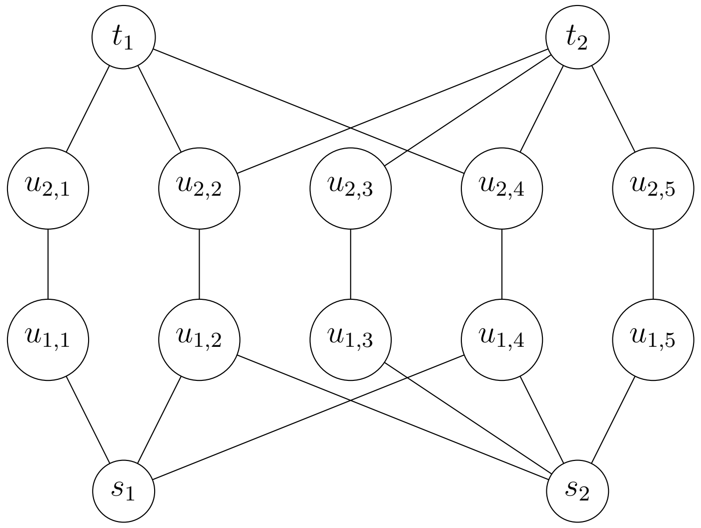


Figure 1.5: Graph of the hitting set instance $\mathcal{S} = \{S_1 = \{1, 2, 4\}, S_2 = \{2, 3, 4, 5\}\}$ on $U = \{1, 2, 3, 4, 5\}$.

have positive cost, there is exactly one shortest path for each pair in C_1 and in C_2. Hence each solution to the path covering problem contains all ε-cost edges $C_1 \cup C_2$ and has total cost of at least 1. The shortest path distance of s_i and t_i in $G_{\mathcal{S}}$ is $2\varepsilon + 1$ and there are $|S_i|$ different shortest paths for each $i \in \{1, \dots, n\}$. Moreover, we have a one-to-one correspondence between subsets of U and the subsets of the m edges of cost 1.

Considering a path covering solution $N \subseteq E$, we have a shortest path between each s_i, t_i pair. The cost 1 edges on these paths identify elements in U that hit each set S_i. On the other hand, every hitting set $H \subseteq U$ identifies $|H|$ of the cost 1 edges that augment the ε-cost edges to shortest paths for each s_i, t_i pair. The cost of such a Path Cover solution $N \subseteq E$ is

$$c(N) = \sum_{e \in N} c(e) = |H| + \varepsilon 2 \sum_{i=1}^{n} |S_i| = |H| + 1 \ .$$

For non-trivial instances of the cardinality hitting set problem, solutions H contain at least one element. Now, any polynomial time, factor α approximation algorithm for Path Cover also provides solutions to the cardinality hitting set problem of cost

$$|H| + 1 \leq \alpha c(N_{\text{OPT}}) = \alpha(|H_{\text{OPT}}| + 1) \leq 2\alpha |H_{\text{OPT}}| \quad .$$

From above's discussion on the approximability of the hitting set problem, it is unlikely to expect the existence of polynomial time algorithms with $\alpha < \ln \sqrt{n}$

for the general Path Cover problem. Note that this reduction easily translates to Path Cover problems in directed graphs. Moreover, strengthening the reduction to avoid the '+1' in the cost or having only n required vertex pairs is possible by expense of a slightly more complicated argument.

Unfortunately, this construction does not trivially extend to d-GMMN. An embedding of such graphs on grid graphs in $\mathbb{R}^d$, that preserves shortest path distances (to some extend), would establish a stronger in-approximability result for d-GMMN (c.f. Theorem 1.2.1). For general, planar graphs with non-negative weights it is already **NP**-hard to decide if there exists a geodesic embedding on a grid graph in $\mathbb{R}^2$ [KKRW10].

1.4 Review of Network Design Problems

A canonical framework to study combinatorial optimization problems is the theory of integer linear programming (ILP). The textbook [WS11] provides a comprehensive introduction. Network design problems seek to choose a minimum cost subset of graph edges under certain connectivity constraints. This broad category contains problems like the shortest s-t path problem or the **NP**-complete Steiner tree problem. Problems without special restrictions on the form of a connection path are well studied in form of *cut-sets*, which are subsets of the graph vertices that are connected under some subset of edges. Cut-sets provide a simple and flexible, combinatorial tool to formalize if a solution network meets or violates the problem-specific connectivity constraints. This section reviews some key properties, applied as requirements to cut-sets, that immediately establish constant factor approximations in polynomial time [Jai01].

Let (V, E) be an undirected graph with non-negative edge weights $c : E \to \mathbb{Q}^+$. The boundary-set $\delta(S)$ for a cut-set $S \subseteq V$ is given by the mapping

$$\delta(S) = \{e \in E \ : \ |e \cap S| = 1\} \ .$$

We introduce binary variables $x_e \in \{0, 1\}$ and constants $c_e = c(e)$ for each $e \in E$. The following cut-set formulation is common basis of many network design problems [WS11].

$$\begin{aligned} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \sum_{e \in \delta(S)} x_e \geq f(S) \qquad \forall\, S \in \mathcal{S} \\ & x_e \in \{0, 1\} \qquad \forall e \in E \end{aligned} \tag{1.1}$$

where $\mathcal{S}$ denotes a set of cut-sets and $f : 2^V \to \mathbb{N}$ is the *requirement function* describing how many edges of a specific boundary-set $\delta(S)$ are required in a feasible solution network.

Shortest s-t Path Problem The problem of finding a minimum cost s-t path in an undirected graph (V, E) with non-negative edge costs $c : E \to \mathbb{Q}^+$ can be considered as a network design problem [GW95] by requiring every cut-set, that separates s from t, to contain at least one edge. That is

$$\mathcal{S} = \{S \subseteq V \;:\; S \neq \emptyset, S \neq V\}$$
$$f(S) = \begin{cases} 1 & |S \cap \{s,t\}| = 1 \\ 0 & \text{otherwise} \end{cases} .$$

Generalized Steiner Network Problem In this problem one is given n vertex pairs $s_i, t_i \in V$ in an undirected graph (V, E) with non-negative edge costs $c : E \to \mathbb{Q}^+$. The goal is to find a minimum cost network $N \subseteq E$ that contains a path between every s_i-t_i vertex pair [WS11].

$$\mathcal{S} = \{S \subseteq V \;:\; S \neq \emptyset, S \neq V\}$$
$$f(S) = \begin{cases} 1 & \exists i \; |S \cap \{s_i,t_i\}| = 1 \\ 0 & \text{otherwise} \end{cases} .$$

1.4.1 Approximability from Requirement Functions

Further classification of combinatoric optimization problems with cut-set formulations is possible by distinguishing classes of requirement functions.

Definition 1.1. A requirement function $f : 2^V \to \mathbb{N}$ is called *proper*, if $f(V) = 0$ and the following two conditions hold.

1. For all $S \subseteq V$, we have $f(S) = f(V \setminus S)$.
2. For all $A, B \subseteq V$ with $A \cap B = \emptyset$, we have $f(A \cup B) \leq \max\{f(A), f(B)\}$.

There is a long line of work studying problems with proper requirement functions, which leads to a primal-dual algorithm (c.f. Section 1.6) to achieve constant factor approximations [GW95].

Definition 1.2. A requirement function $f : 2^V \to \mathbb{Z}$ is called *weakly-supermodular*, if $f(V) = 0$ and for every $A, B \subseteq V$ *one of* the following conditions holds.

1. $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$.

2. $f(A) + f(B) \leq f(A \setminus B) + f(B \setminus A)$.

This is a generalization over *supermodular* requirement functions, since only one of the conditions is required. In fact, every proper requirement function is weakly-supermodular [GGP+94]. The authors show as well, that the an extended primal-dual method provides factor $2H_{f_{\max}}$ approximations for these problems, where $H_n = 1 + 1/2 + \ldots + 1/n$ and $f_{\max} = \max_{S \subseteq V} f(S)$. Jain's iterative rounding method [Jai01] improves upon this and allows to find 2-approximations for such network design problems in polynomial time. The author provides a compressed formulation of the linear program and subsequently uses Tardos' polynomial time LP solver to derive 2-approximations for such network design problems in $\mathcal{O}(|V|^{10}|E|^7)$ time.

However, it is unclear how to capture the required monotonicity property of solution paths with boundaries of cut-sets. E.g. the cut-set $\{s\}$ that separates a s-t pair might well have a selected edge $e = \{s, v\}$ in it's boundary that is not monotonous for s-t. Moreover, following edges that cross the boundary of the cut-set $\{s, v\}$ might not provide a monotonous s-t path.

1.5 Separation Set Formulation for Path Cover

We turn to a more flexible modeling concept than boundary-sets of cut-sets. Given a graph (V, E) and two vertices $s, t \in V$, we call a set of edges $S \subseteq E$ a s-t *separation set*, if $(V, E \setminus S)$ contains no admissible s-t path. The Shortest s-t Path (c.f. Section 1.4) network design problem solely requires a simple path that connects the graph nodes s and t. Given any cut-set $A \subseteq V$ with $|A \cap \{s, t\}| = 1$, the boundary-set $\delta(A)$ is an example of a s-t separation set, since s and t are in different connected components of $(V, E \setminus \delta(A))$. However, smaller sets might well suffice for more restrictive requirements to s-t connectivity.

Definition 1.3. Let (V, E) be a graph with non-negative edge costs $c : E \to \mathbb{Q}$ and $s, t \in V$. A set $S \subseteq E$ is called a s-t separation set if the graph $(V, E \setminus S)$ contains no path, that is a shortest s-t path in (V, E).

There are no more than $2^{|E|}$ separation sets in a graph and adding edges to a separation set preserves the separation property. Hence the union of separation sets of different vertex pairs is still a separation set. A separation set is called minimal (under inclusion) if removing any edge of S allows a shortest s-t path in $(V, E \setminus S)$.

Finding *some* separation sets is simple for many combinatorial network design problems. If some graph traversal algorithm can test a subnetwork $N \subseteq E$ for admissible s-t paths, one can often extend the traversal to an **oracle algorithm**.

That is a polynomial time algorithm that either certifies N to be a feasible solution or provides separation sets $\{S_1, \ldots, S_k\}$ with $S_j \cap N = \emptyset$ for $j \in \{1, \ldots, k\}$.

Given an s-t vertex pair, we assign two labels to the vertices of (V, E). One label for the vertex' shortest-path distance from s and one for the distance to t. Considering the labels on adjacent vertices of an edge allows to judge if the edge is on a shortest s-t path. For the Hanan grid of a d-GMMN instance, such preprocessing is unnecessary as the spatial embedding of the vertices already provides such labels. In a query, the oracle is presented with a traversable edge set $N \subseteq E$ as input. A depth-first search from s either finds a shortest s-t path contained in (V, N) or reaches a set of edges $S \subseteq E \setminus N$ that extend shortest paths to (under N) unreachable vertices. We denote this edge set with S_s and a traversal from vertex t provides the set S_t. Hence, the described Path Cover oracle returns no more than $k = 2n$ separation sets and a single call to the depth-first search takes no more than $\mathcal{O}(|E|)$ time, given the discussed preprocessing labels. Section 1.8.1 describes an output sensitive method for sequenced calls to the separation set oracle when edges are added to N.

Figure 1.6 illustrates this on a 2-GMMN instance of three terminal pairs. The black edges denote the input subnetwork $N \subseteq E$. The 6 depth-first search calls return four separation sets, since terminal pair 3 has a monotonous path in N. The blue edges indicate the two separation sets of the s_1 and t_1 call (left part) and the separation sets of s_2, t_2 (right part).

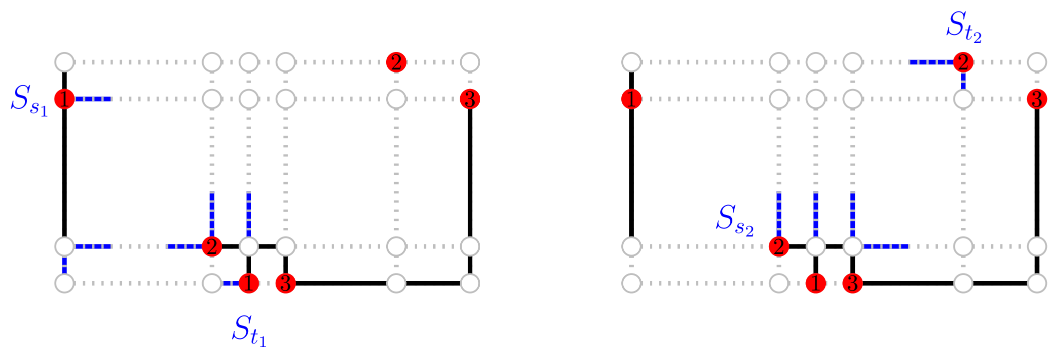


Figure 1.6: Example of separation sets (blue edges) from the oracle on a 2-GMMN instance with three terminal pairs. Black edges denote the input subnetwork $N \subseteq E$.

1.5.1 Linear Programming on Separation Set Constraints

If the connectivity requirements of a network design problems can be captured as separation sets, one immediately has a natural formulation as integer linear program:

$$\begin{aligned} \text{minimize} \ & \sum_{e\in E} c_e x_e & & \\ \text{subject to} \ & \sum_{e\in S} x_e \geq 1 & & \forall S \in \mathcal{S} \\ & x_e \in \{0,1\} & & \forall e \in E \end{aligned} \tag{1.2}$$

This primal ILP contains a binary variable $x_e \in \{0,1\}$ and a constant c_e for the associated cost of every graph edge $e \in E$. The set $\mathcal{S}$ contains all separation sets, hence we have no more than $2^{|E|}$ row constraints.

For the Path Cover problem with the required vertex pairs $s_i, t_i \in V$, we have

$$\mathcal{S} = \{S \subseteq E : S \text{ is a separation set for some } s_i\text{-}t_i \text{ pair}\} \ .$$

Solutions $N \subseteq E$ of the Path Cover problem instance and feasible assignments $\mathbf{x} \in \{0,1\}^{|E|}$ of the ILP have a one-to-one correspondence. Given a solution set $N \subseteq E$ for the Path Cover problem instance. Let $s, t \in V$ be a vertex pair that is separated by a row constraint, say S. Since N contains a shortest s-t path, at least one edge in N needs to cross S. Conversely, given a feasible assignment $\mathbf{x}$, calling the oracle algorithm on the corresponding edge set N cannot return any separation set. Hence, N contains a shortest path for each s_i-t_i pair.

The relaxation to non-integral variables $\tilde{x}_e \geq 0$ provides the primal LP. An optimal solution $\tilde{\mathbf{x}} \in \mathbb{R}^{|E|}$ to the LP provides a lower bound to the objective value of optimal integral solutions, since any optimal integral solution $\mathbf{x}$ is also feasible for the LP.

In the primal LP ($\min \mathbf{c} \cdot \tilde{\mathbf{x}},\ \mathbf{A} \cdot \tilde{\mathbf{x}} \geq \mathbf{1}$) the objective-coefficients are non-negative and the constraint-coefficients are either 0 or 1. Consider an inequality of the single row-constraint $S \in \mathcal{S}$ of $\mathbf{A}$ multiplied by some small $y \geq 0$. That is

$$y(\mathbf{A}_S \cdot \mathbf{x}) \geq y \cdot 1 \ . \tag{1.3}$$

For $y \leq \min\{c_e : e \in E\}$, the multiplied constraint coefficients $y\mathbf{A}_S$ are component-wise smaller than the objective coefficients $\mathbf{c}$. Therefore, the right hand side of inequality (1.3) is a lower bound to the objective value $\mathbf{c} \cdot \tilde{\mathbf{x}}$, for arbitrary non-negative assignments $\tilde{\mathbf{x}}$. Any conical combination (1.5) of row-constraints in which the coefficients are still component wise smaller than the objective-coefficients (1.4) provides this property. This canonical method leads to the dual LP, in which the objective is to find a maximal lower bound using such conical combinations of row-constraints:

$$\begin{aligned} \text{maximize} \quad & \sum_{S \in \mathcal{S}} y_S \\ \text{subject to} \quad & \sum_{S \in \mathcal{S} \,:\, e \in S} y_S \leq c_e \qquad \forall e \in E \qquad (1.4) \\ & y_S \geq 0 \qquad \forall S \in \mathcal{S} \qquad (1.5) \end{aligned}$$

This described property is well known as *weak duality* [Vaz03, Chapter 12.1]. We summarize these facts in the following Lemma.

Lemma 1.5.1. *For a feasible assignment* $\mathbf{y} \in \mathbb{R}^{|\mathcal{S}|}$ *for the dual LP and feasible assignments* $\tilde{\mathbf{x}}_{OPT} \in \mathbb{R}^{|E|}$, $\mathbf{x}_{OPT} \in \{0,1\}^{|E|}$ *with optimal objective value for the primal LP and ILP, respectively, we have*

$$\mathbf{y} \cdot \mathbf{1} \leq \mathbf{c} \cdot \tilde{\mathbf{x}}_{OPT} \leq \mathbf{c} \cdot \mathbf{x}_{OPT} \; .$$

The maximum ratio $(\mathbf{c} \cdot \mathbf{x}_{\mathrm{OPT}})/(\mathbf{c} \cdot \tilde{\mathbf{x}}_{\mathrm{OPT}})$, observed over all instances of a problem, is called the *integrality gap* of a particular formulation.

1.6 The Primal-Dual Method

The method is coined by extensive research [Wil02] on the Generalized Steiner Network Problem (see Section 1.4 for the definition). The idea of a primal-dual algorithm scheme is to start with a pair of assignments. An infeasible integral assignment to the primal ILP ($\mathbf{x} = \mathbf{0}$) and a feasible, however far from optimal, assignment ($\mathbf{y} = \mathbf{0}$) to the dual LP. By alternately improving the dual assignment and making the primal assignment more feasible, one obtains a lower bound that *might* have a close relation to the eventually feasible, primal assignment. If a polynomial time oracle reveals some violated constraints based on an infeasible primal assignment, storing the dual assignment sparsely or solely the dual objective value is sufficient.

We now describe the adaption to problems with separation set oracles, as discussed in Section 1.5. Given an infeasible, primal assignment $\mathbf{x} \in \{0,1\}^{|E|}$, we run a separation set oracle on the respective set of edges. The resulting violated separation sets $\mathcal{V} = \{S_1, S_2, \ldots S_k\} \subseteq \mathcal{S}$ contain edges, that *extend* admissible path pre- or suffixes that are contained in $\mathbf{x}$. Continuous, uniform increase of the dual variables $y_{S_1}, \ldots, y_{S_k}$ by some $\varepsilon > 0$ eventually packs a dual's inequality constraint tightly, with equality. Meaning for the inequality of some row constraint we have

$$c_e = \sum_{S \in \mathcal{S} \,:\, e \in S} y_S \; .$$

We add the edges with tight dual constraints to $\mathbf{x}$ (c.f. Line 9 in Algorithm 2) and repeat until primal feasibility is reached. This creates an ordered sequence for the edges of $\mathbf{x}$ and a lower bound $Y = \sum_{S \in \mathcal{S}} y_S$ value. After the sequence is feasible, unnecessary edges are pruned in reverse sequence order (c.f. Lines 15-19 in Algorithm 2).

In each step, one of the polynomially many edges is added to $\mathbf{x}$ and proceeding oracle calls do not contain edges of $\mathbf{x}$ in the separation sets. A run of this algorithm represents a feasible dual assignment. The initial dual assignment ($\mathbf{y} = \mathbf{0}$) is feasible and, by keeping track of the inequality slack, no dual constraint becomes violated by the variable increases. Hence the objective value of this dual assignment, stored in the variable Y, is a lower bound on the cost of a optimal integral solution (c.f. Lemma 1.5.1). The pruning phase preserves primal feasibility of $\mathbf{x}$.

```
1  $\mathbf{x} := \emptyset$ ;                                  /* Edge list */
2  $Y := 0$ ;                                   /* Dual objective value */
3  $\mathbf{c}' := \mathbf{c}$ ;                            /* Slack in dual constraints */
4  while $0 < \left|\mathcal{V} := \text{oracle}(\mathbf{x})\right|$ do
       /* Let $\nu : E \to \mathbb{N}$ denote the edge's frequency          */
5      $\nu(e) = |\{S_i \in \mathcal{V} \ : \ e \in S_i\}|$
6      $\varepsilon := \min\{ \mathbf{c}'[e]/\nu(e) \ : \ \nu(e) > 0 \}$
7      foreach $e \in E$ with $\nu(e) > 0$ do
8          Decrement $\mathbf{c}'[e]$ by $\varepsilon\nu(e)$
9          if $\mathbf{c}'[e] = 0$ then
10             $\mathbf{x}.\text{append}(e)$
11         end
12     end
13     Increment $Y$ by $\varepsilon|\mathcal{V}|$
14 end
15 foreach $e \in \mathbf{x}$ in LiFo order do
16     if $\mathbf{x} - \{e\}$ primal feasible then
17         $\mathbf{x}.\text{remove}(e)$
18     end
19 end
20 return $Y$ and $\mathbf{x}$
```

Algorithm 2: Primal-Dual Scheme on Separation Set Oracles. See Figure 1.7 for an example with the 2-GMMN separation set oracle.

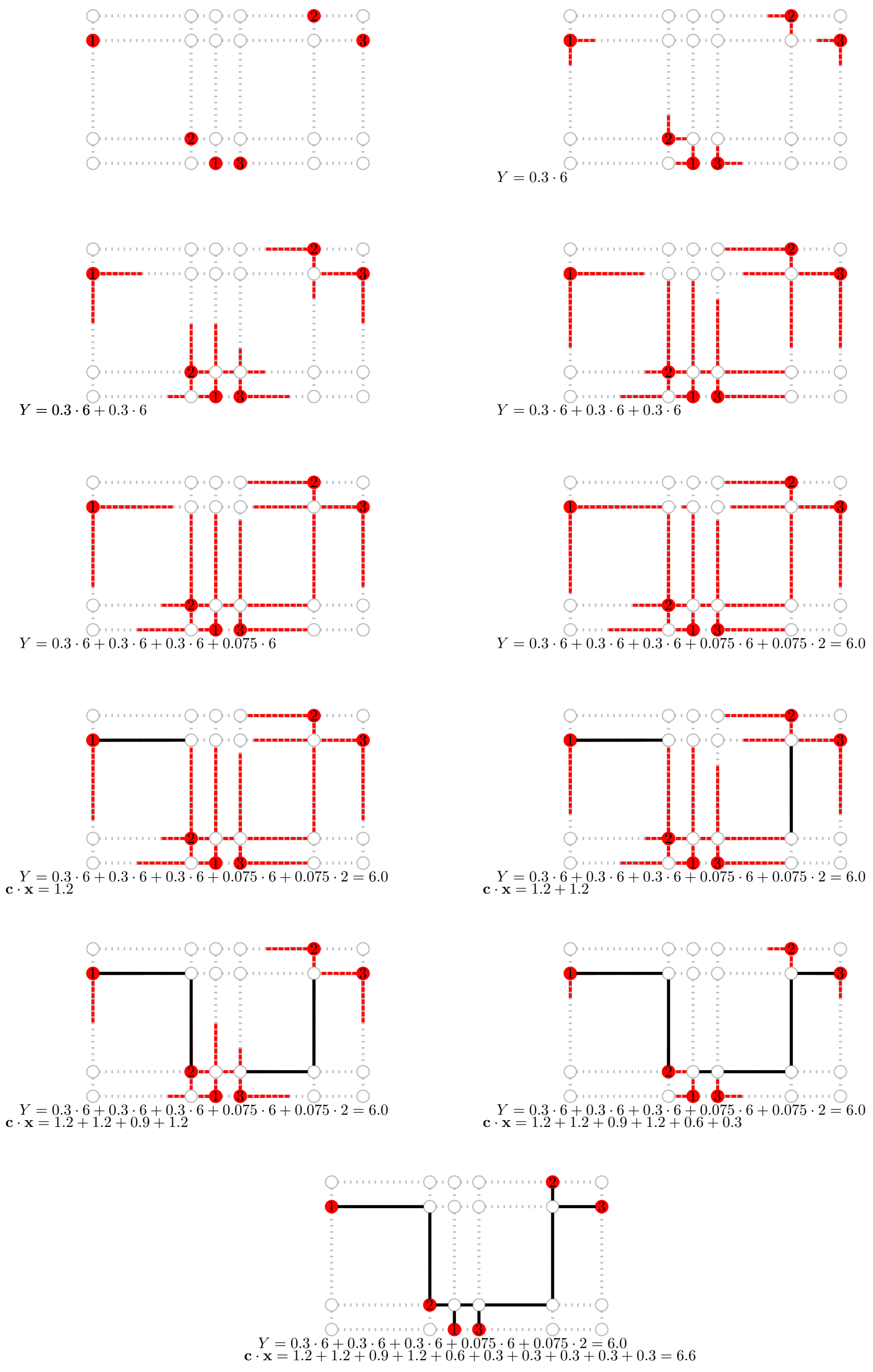

1
2
3
Y = 0.3 · 6
Y = 0.3 · 6 + 0.3 · 6
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6 + 0.075 · 6
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6 + 0.075 · 6 + 0.075 · 2 = 6.0
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6 + 0.075 · 6 + 0.075 · 2 = 6.0
c · x = 1.2
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6 + 0.075 · 6 + 0.075 · 2 = 6.0
c · x = 1.2 + 1.2
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6 + 0.075 · 6 + 0.075 · 2 = 6.0
c · x = 1.2 + 1.2 + 0.9 + 1.2
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6 + 0.075 · 6 + 0.075 · 2 = 6.0
c · x = 1.2 + 1.2 + 0.9 + 1.2 + 0.6 + 0.3
Y = 0.3 · 6 + 0.3 · 6 + 0.3 · 6 + 0.075 · 6 + 0.075 · 2 = 6.0
c · x = 1.2 + 1.2 + 0.9 + 1.2 + 0.6 + 0.3 + 0.3 + 0.3 + 0.3 + 0.3 = 6.6

Figure 1.7: Protocol of a run of Algorithm 2 on the 2-GMMN instance $R = \Big\{\{(0,1.5),(1.5,0)\},\{(1.2,.3),(2.7,1.8)\},\{(1.8,0),(3.3,1.5)\}\Big\}$. Steps are listed from left to right and top to bottom. The first 6 steps illustrate the dual's slack (partially red edges) during the while loop. Remaining steps illustrate the LiFo pruning decisions (black edges).

1.7 The Primal-Dual Analysis

The Primal-Dual method has several interesting features [Wil02]. Beside a solution for the primal ILP, the method also provides a feasible assignment for the dual LP. This establishes a lower bound on the optimal objective value of the LP relaxation of a particular problem instance. Thereby, a bound on the approximation quality of a solution is known after execution.

Let E_i denote the edge set stored in $\mathbf{x}$ in step i of the while loop and N the final edge set after the reverse pruning. With $N_i = N \setminus E_i$, we denote the edges added after step i and kept in the reverse pruning. We have

$$\begin{array}{rcccccccc} \emptyset = E_1 & \subseteq & E_2 & \subseteq & \dots & \subseteq & E_l & \subseteq & E_{l+1} \\ N = N_1 & \supseteq & N_2 & \supseteq & \dots & \supseteq & N_l & \supseteq & N_{l+1} = \emptyset \end{array}$$

and $E_i \cup N_i$ is a solution for each $i \in \{1, \dots, l+1\}$. Moreover, let $\mathcal{V}_i$ denote the set of separation sets and ε_i the increment in step i (c.f. lines 4 and 6 of Algorithm 2). The method, as used in Algorithm 2, provides two interesting properties

$$\sum_{S \in \mathcal{S}} y_S = \sum_{i=1}^{l} |\mathcal{V}_i| \varepsilon_i \tag{1.6}$$

$$c_e = \sum_{S \in \mathcal{S} \,:\, e \in S} y_S \qquad \forall e \in E_l \tag{1.7}$$

Equation 1.6 describes the contributions to the lower bound value over the steps i of the while loop in Algorithm 2 and Equation 1.7 provides means to charge the costs of a particular edge of a solution against a set of dual variables. This allows to rewrite the cost of a calculated solution $N \subseteq E_l$ to the primal ILP as

$$\begin{aligned} \sum_{e \in N} c_e &= \sum_{e \in N} \sum_{S \in \mathcal{S} \,:\, e \in S} y_S \\ &= \sum_{S \in \mathcal{S}} y_S \, |N \cap S| \\ &= \sum_{S \in \mathcal{S}} |N \cap S| \sum_{i \,:\, S \in \mathcal{V}_i} \varepsilon_i \\ &= \sum_{i} \varepsilon_i \left(\sum_{S \in \mathcal{V}_i} |N \cap S| \right) \end{aligned} \tag{1.8}$$

Given Lemma 1.5.1 and Equation 1.6, it is sufficient for an α approximation to bound the last sum in Equation 1.8 with $\alpha |\mathcal{V}_i|$ for each step $i \in \{1, \dots, l\}$ of the algorithm.

The Shortest s-t Path Problem (c.f. Section 1.4) provides a simple example with $\alpha = 1$. A separation set oracle for this problem simply determines the connected component $A \subseteq V$ of s in (V, E_i) and returns the boundary-set $\delta(A)$ as the minimal separation set, if t is in a different component. Hence, there is just one violated separation set $|\mathcal{V}_i| = 1$ in every step i and we denote the sequence of these separation sets with $S_1, \ldots, S_l$. The reverse pruning strategy provides $|N \cap S_i| = 1$, by an inductive argument. Say we keep edge e, which was added in step i, in the final set N. Then no other edge $e' \in S_j$ with $j \leq i$ is kept, if $e \in S_j$.

1.7.1 Path Cover Problem

The Path Cover problem (c.f. Section 1.2.2) is no simpler to approximate than the Hitting Set problem (c.f. Section 1.3). This section provides a bound on the worst-case approximation ratio of Algorithm 2 for the Path Cover Problem.

Let $G = (V, E)$ denote the graph, $c : E \to \mathbb{Q}$ the non-negative edge costs, and $\{s_i, t_i\}$ the n vertex pairs that require a shortest path in a solution network. With Π_i we denote the set of shortest s_i-t_i paths

$$\Pi_i = \{p \subseteq E \ : \ p \text{ is a shortest path for } s_i, t_i \text{ in } G\}$$

and $E(\Pi_i) = \bigcup_{p \in \Pi_i} p$ denotes the edges of shortest s_i-t_i paths. With the mapping $\sigma : E \to \{0, \ldots, n\}$, that is

$$\sigma(e) = \left| \left\{ i \in \{1, \ldots, n\} : e \in E(\Pi_i) \right\} \right| \quad ,$$

we describe how many pairs can share a particular edge. We define the *parameter* ω of an instance with $\omega = \max_{e \in E} \sigma(e)$. For problem instances that originate from a d-GMMN problem R, the parameter $\omega(R)$ simply captures the maximum overlap of the bounding boxes associated to terminal pairs. That is

$$\omega(R) = \max_{p \in \mathbb{R}^d} \left| \left\{ (s,t) \in R \ : \ p \in \text{box}(s,t) \right\} \right| \quad .$$

Let $r_i \in \{0, \ldots, n\}$ denote the number of s-t pairs without shortest path in the edge set E_i of step i. The separation set oracle, as introduced in Section 1.5, returns the set $\mathcal{V}_i$ of violated separation sets on input of E_i. That is $\mathcal{V}_i = \{S_1, \ldots, S_k\}$ where each S_j originates from the guided depth-first search starting from the terminal vertices. Hence $k \in \{1, \ldots, 2r_i\}$. Since a separation set $S \subseteq E$ for s_j-t_j is always a subset of $E(\Pi_j)$, we have

$$r_i/\omega \leq |\mathcal{V}_i| \leq 2r_i \tag{1.9}$$

for each step $i \in \{1, \ldots, l\}$. We now turn to a property of the reverse pruning.

Lemma 1.7.1. *For each step $i \in \{1, \ldots, l\}$, we have*

$$|N \cap E(\mathcal{V}_i)| \leq 2r_i \ .$$

Proof. Let $e_1, \ldots, e_l$ denote the sequence in which the edges were added. Note that $E_i = \{e_1, \ldots, e_{i-1}\}$ and $N_i = N \setminus E_i$. If edge e_j survives the reverse pruning, it has a list of s-t pairs as labels. Each such pair-label is witness that the respective pair has no shortest path in $(V, E_j \cup N_{j+1})$. Moreover, e_j is in the separation set of this pair in $\mathcal{V}_j$.

Suppose $|N_i \cap E(\mathcal{V}_i)| > 2r_i$. Then there are at least three edges $e_{j_1}, e_{j_2}, e_{j_3}$ with $j_1 < j_2 < j_3$ in this set intersection that have the same s-t pair-label. Note that $E_{j_2} \cup N_{j_2}$ contains a shortest s-t path p. Such a path consists of three parts $p = p_s p_m p_t$, where the prefix p_s and the suffix p_t are contained in E_{j_2} and the middle part $p_m = \{e_{j_2}, \ldots, e_{j_3}\}$ is contained in N_{j_2}. The s-t pair has at most two separation sets $S_s, S_t \in \mathcal{V}_i$ for any step i. W.l.o.g. $e_{j_2} \in S_s$ and $e_{j_3} \in S_t$. Hence e_{j_1} is in the separation set that contains e_{j_2} or e_{j_3}. However, a shortest s-t path contains at most one edge of a minimal s-t separation set. Hence, the $E_{j_1} \cup N_{j_2}$ contains a shortest s-t path via p_m and e_{j_1} is not labeled with this pair by the reverse pruning – a contradiction. □

Theorem 1.7.2. *Algorithm 2 provides solutions to the Path Cover problem within a factor $4\omega^2$ of the optimal cost, for instances with maximum overlap of ω.*

Proof. We use aboves standard Primal-Dual analysis and show that

$$\sum_{S \in \mathcal{V}_i} |N \cap S| \leq 4\omega^2 |\mathcal{V}_i|$$

holds for each step $i \in \{1, \ldots, l\}$. Let $E(\mathcal{V}_i) = \bigcup_{S \in \mathcal{V}_i} S$ denote the edges in some separation set of step i. Rewriting the sum provides

$$\begin{aligned}\sum_{S \in \mathcal{V}_i} |N \cap S| &= \sum_{e \in N \cap E(\mathcal{V}_i)} \Big|\{S \in \mathcal{V}_i : e \in S\}\Big| \\ &\leq \sum_{e \in N \cap E(\mathcal{V}_i)} 2\omega = \Big|N \cap E(\mathcal{V}_i)\Big| 2\omega \\ &\leq 4|\mathcal{V}_i|\omega^2\end{aligned}$$

The first inequality is due $\sigma(e) \leq \omega$ and that each pair has at most two sets in $\mathcal{V}_i$. The last inequality is due to Lemma 1.7.1 and the left side of inequality (1.9). □

See Figure 1.8 for an illustration of the proof.

Corollary 1.7.3. *For fixed d, instances of the d-GMMN problem with maximum overlap of ω allow $4\omega^2$ approximations in polynomial time.*

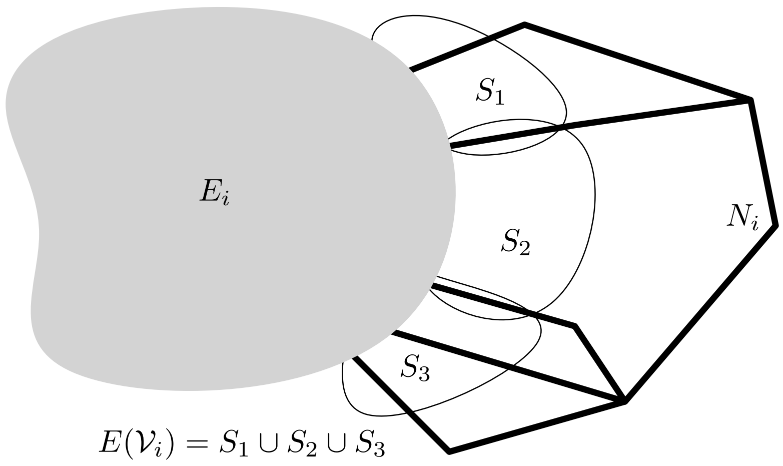


Figure 1.8: Illustration of the proof of Theorem 1.7.2

This corollary is not trivial regarding the decomposition results in Section 1.1. In terms of the geometric intersection graph, ω equals the size of a maximum clique. Another complexity parameter in the context of the d-GMMN problem is the maximum vertex degree Δ in the geometric intersection graph (c.f. Section 1.2.1). We have $\omega \leq \Delta + 1$. However, Δ might well be much bigger leading to poor approximations of the greedy coloring-decomposition of Theorem 1.1.3. There is extensive research on chromatic numbers of geometric intersection graphs [PKK+13]. [AG60] shows the existence of a $\omega(4\omega - 3)$ bound on the chromatic number of the geometric intersection graph of axis-aligned rectangles in $\mathbb{R}^2$. However, [Bur65] provides a construction of a family of axis-aligned boxes in $\mathbb{R}^3$ that have arbitrary large chromatic numbers.

1.8 Implementation for d-GMNN

Using floating point arithmetics of contemporary computing hardware is problematic for algorithms that rely on value increments or decrements. Operations with a relatively small operand might well lead to an unchanged value. The Primal-Dual scheme in Algorithm 2 only uses increments of the lower bound value Y by $\varepsilon|\mathcal{V}|$ and decrements of the slack value of a dual constraint by $\varepsilon\nu(e)$, where $\nu(e) \in \{1, \ldots, 2n\}$. Since all operands are positive, consequently incrementing less and decrementing more keeps the dual assignment $\mathbf{y}$ feasible. Moreover, the value Y is a lower bound to the objective value of this assignment. Using certain rounding modes allows to avoid the use of arbitrary precision arithmetic libraries in this case. A more pragmatical way is to over- and underestimate ε and the arithmetic result by the machine precision. Even tough these lower bound values hold for the problem instance, the edge sequence, which is basis of the reverse pruning, is affected by these rounding errors.

A naive implementation for the oracle calls of Algorithm 2 on the Hanan grid (V, E) uses two depth-first searches for a terminal pair (c.f. Section 1.5). A separation set of such a traversal is no bigger than $|E|$ and there are at most $|E|$ iterations of the while loop body, since at least one edge is added in each step. Hence, the while loop takes no more than $\mathcal{O}(n|E|^2)$ operations, which is also sufficient for the reverse pruning. Therefore, the number of operations in Algorithm 2 is polynomial for fixed dimension d. The naive implementation for 2-GMMN enabled us to find solutions and lower bounds for instances up to a size of $n = 128$ (c.f. Figure 1.9), which is considerably more than $n = 7$ with the brute-force approach (c.f. Section 1.2.2). For the instances of those experiments, the reverse pruning phase takes the majority of the execution time. For an output sensitive approach, it is desirable to perform fewer operations only for those elements that remain in the solution $N \subseteq E$. E.g. depth-first searches on the sparser edge set N_i rather than $E_i \cup N_i$.

The remainder of this section describes a speed-up method that invests additional storage to save computation time. We explicitly store the coordinates in spatial embedding of the Hanan grid with the vertices of the graph (c.f. Section 1.5). The edges that are added to $\mathbf{x}$ are stored on a stack and we use the index in this array as a unique tag value for the edges. Moreover, we allow each graph edge to store such a tag value. Using tag values in a graph traversal allows to restrict the search to edges with tag values in a certain range – e.g. edges that were added after step i.

1.8.1 Maintaining Separation Sets

We store two search trees for each pair of vertex terminals $s, t \in V$. Each search tree only needs to store reachability information of nodes that are on a shortest s-t path (c.f. Section 1.5). We describe the search tree and its maintenance under insertions of edges for the search from s to t – maintaining the tree from t to s is done analogously.

Instead of a Boolean reachability flag for a vertex v, the search tree labels vertices with the smallest tag τ such that there exists a shortest path from s to v with edges that have a tag value of at most τ. Initially, no edge has a tag value and the separation set $S_s \subseteq E$ of the search tree contains only edges that are incident to vertex s and on a shortest s-t path (c.f. Section 1.5). Maintaining the search tree, and thereby its separation set, under insertions of an edge e with tag value τ is simple. If $e \notin S_s$, the edge does not extend a shortest path prefix in the search tree. Hence, no update is performed. Otherwise we label the newly reachable vertex v in the search tree of s with τ. Moreover, we continue the depth-first search from s

at vertex v on edges with tag value at most τ. All vertices newly reachable vertices are labeled with τ in the search tree. In the same run, we can drop edges of S_s that are incident to the newly reachable vertices. Eventually, we add new fringe edges to the separation set S_s.

Over all edge insertions, no edge is traversed twice by the depth-first search. Hence, a total of $\mathcal{O}(|E|)$ operations is sufficient for building a search tree that labels the vertices with the smallest tag value under which they are reachable from s.

1.9 Experimental Results

The results and running times of our single-threaded Java implementation were derived on rather old desktop hardware (Intel i5-2500K) running OpenJDK version `1.7.0_151` on a 64Bit Ubuntu 16.04.3 LTS system. The proposed primal-dual strategy with reverse pruning has the advantage of also constructing a lower bound to the optimal cost of a solution. Let $\mathbf{x}_{\mathrm{OPT}}$ and $\tilde{\mathbf{x}}_{\mathrm{OPT}}$ be optimal solutions for the primal ILP and LP respectively. Consider $\mathbf{x}$ and Y as calculated from Algorithm 2. Given Lemma 1.5.1, we have that the actual approximation ratio is bounded:

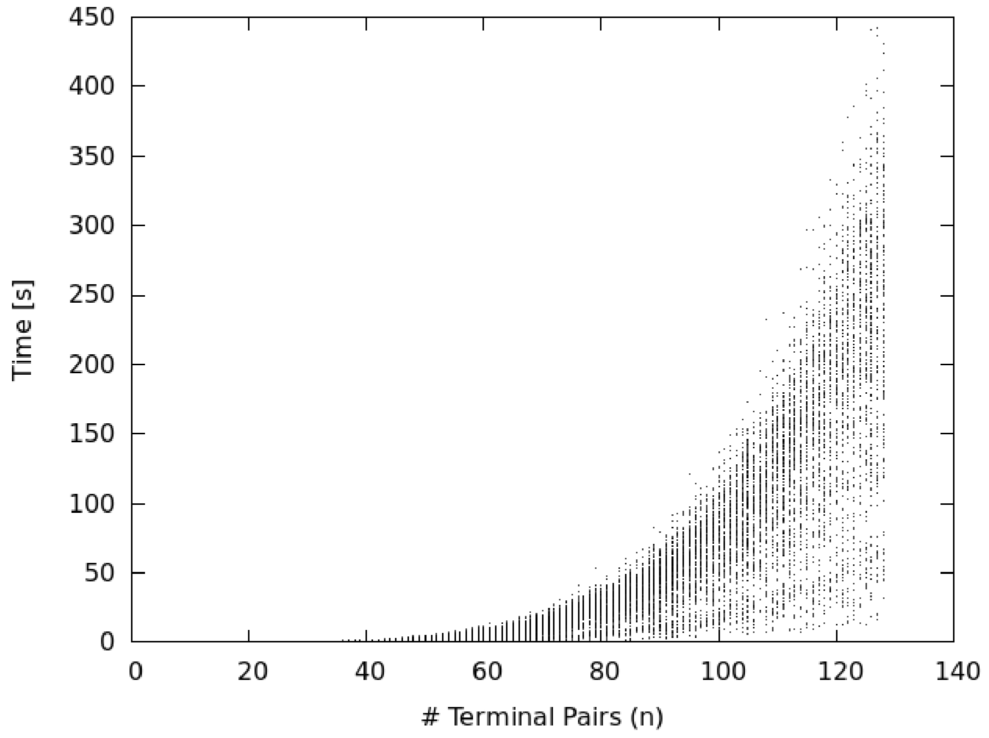
$$\frac{\mathbf{c} \cdot \mathbf{x}}{\mathbf{c} \cdot \mathbf{x}_{\mathrm{OPT}}} \leq \frac{\mathbf{c} \cdot \mathbf{x}}{\mathbf{c} \cdot \tilde{\mathbf{x}}_{\mathrm{OPT}}} \leq \frac{\mathbf{c} \cdot \mathbf{x}}{Y}.$$

The rightmost term – we call it *approximation guarantee* in the remainder of this chapter – is explicitly calculated by Algorithm 2.

Along with the $\mathcal{O}(\log n)$ approximation algorithm for 2-GMMN, Das et al. also provided a family of recursively defined, overlapping instances that show the tightness of the approximation ratio of their algorithm. We evaluated several of those instances and found them all to be solved optimally with a tight lower bound by Algorithm 2 (data not shown).

1.9.1 Randomly Sampled Instances

We evaluated Algorithm 2 on random 2-GMMN instances. To rule out solely good behavior on uniformly chosen points of some square, we sampled different aspect ratios (x-range / y-range) and densities (number of vertices in x / x-range). For a given number of pairs n of a instance, we first choose an aspect ratio $a \in \{1, \dots, 9\}$ uniformly at random. Second we choose a density $d \in \{1n, 2n, \dots, 10n\}$ uniformly at random. Finally we choose n times the two x-coordinates in $\{1, \dots, d \cdot a\}$ and the two y-coordinates in $\{1, \dots, d\}$ uniformly at random. Out of these, we evaluated a total of $19,050$ random instances of several sizes (150 replicates for each $n \in \{2, \dots, 128\}$). Figure 1.10 shows the achieved distribution of the sampling



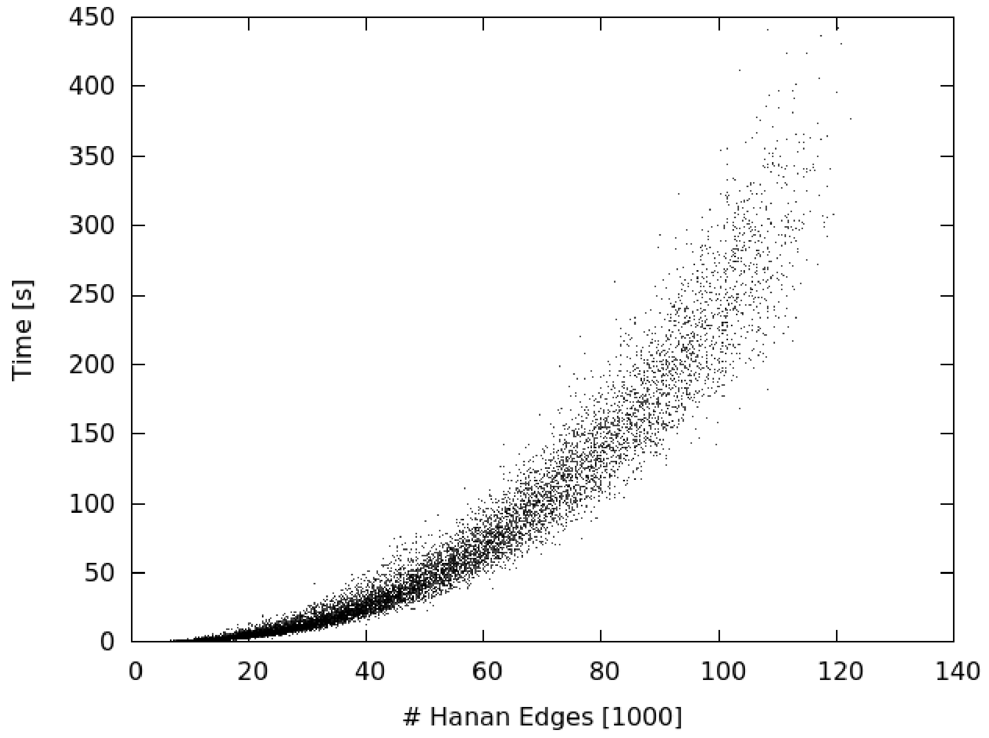


Figure 1.9: Execution times of the primal-dual algorithm in seconds on each of the sampled random instances of 2-GMMN. Time by number of terminal pairs (top) and by number of Hanan grid edges (bottom).

strategy in the parameter space and Figure 1.9 the execution times. See Figure 1.11 for the observed approximation guarantees.

Algorithm 2 seems to find solutions that are optimal and close to optimum in many cases. We found approximation guarantee worse than 2.0 on instances having many terminals and a *very high* degeneracy in the aspect ratio and density at the same time. Using this observation, the worst experimentally observed approximation guarantee on a random 2-GMMN instance ($n = 90$, aspect ratio of $10000/1$, density of $2/100000$) was 2.856.

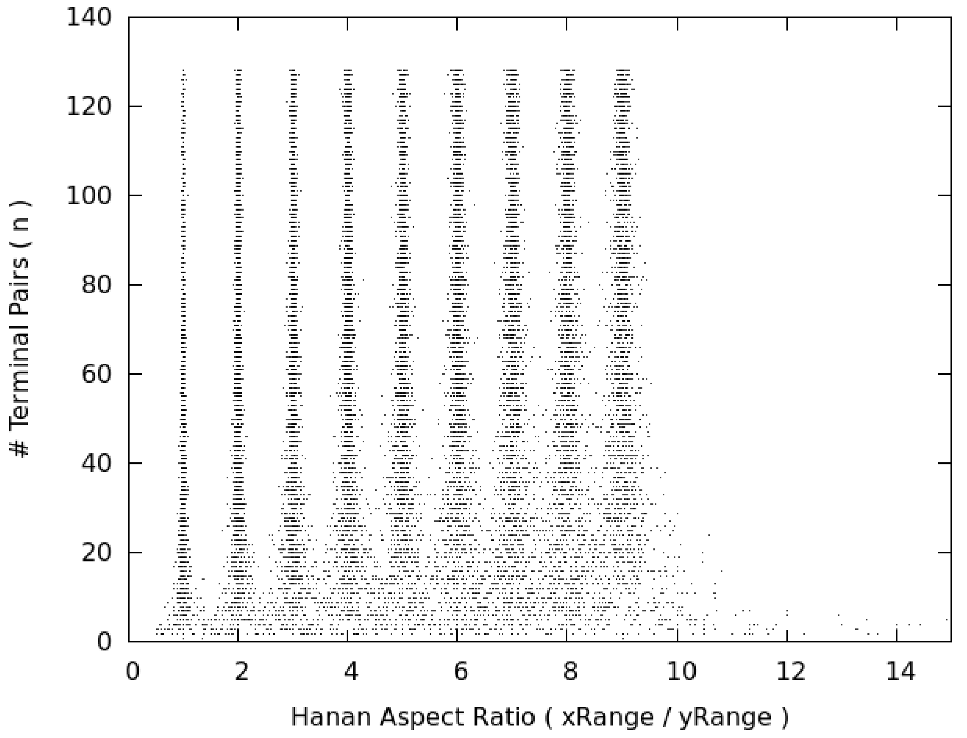
1.10 Summary and Open Problems

We summarize the observations on the polynomial time approximability in this chapter with the following Table 1.1. Note that 2-GMMN is a special case of d-GMMN, which is a special case of Path Cover (c.f. Theorem 1.2.1). Section 1.1 provides an algorithm that achieves an approximation factor constant in the *scale diversity* $\mathcal{D}$ of a 2-GMMN problem. This result always matches the $\mathcal{O}(\log n)$-approximation given in [DFK+17], but is considerably better if the scale diversity is small. For a Path Cover instance, the parameter ω captures the maximum overlap of the shortest path edge sets of vertex pairs. For d-GMMN instances, the maximum degree Δ refers to the geometric intersection graph of a problem instance and ω equals the clique number. For every d-GMMN instance, we have that $\omega \leq \Delta + 1 \leq n$.

Problem	Parameter	Approximability	
2-GMMN	–	within $\mathcal{O}(\log n)$	[DFK+17]
	x- and x/y-separated	within $\mathcal{O}(1)$	[DFK+17]
	scale diversity $\mathcal{D}$	within $\mathcal{O}(\mathcal{D})$	Theorem 1.1.7
d-GMMN	–	not within $(1 + 2 \cdot 10^{-5})$	[MSU09]
	–	within $\mathcal{O}(\log^{d+1} n)$	[DFK+17]
	maximum degree Δ	within $(\Delta + 1)$	Theorem 1.1.3
Path Cover	–	not within $(1 - \delta) \ln \sqrt{n}$ for every $\delta > 0$	Section 1.3
	maximum overlapp ω	within $4\omega^2$,	Theorem 1.7.2

Table 1.1: Summary of polynomial-time approximability. Inapproximability statements rely on the complexity assumtion $\mathbf{P} \neq \mathbf{NP}$.

These polynomial time algorithms have implications on potential $\mathcal{O}(1)$-inapproximability proofs. Such a proof for 2-GMMN *must* use gadgets of many different scales. This is in contrast to the proof in [CGS11]. There the existence of a FPTAS



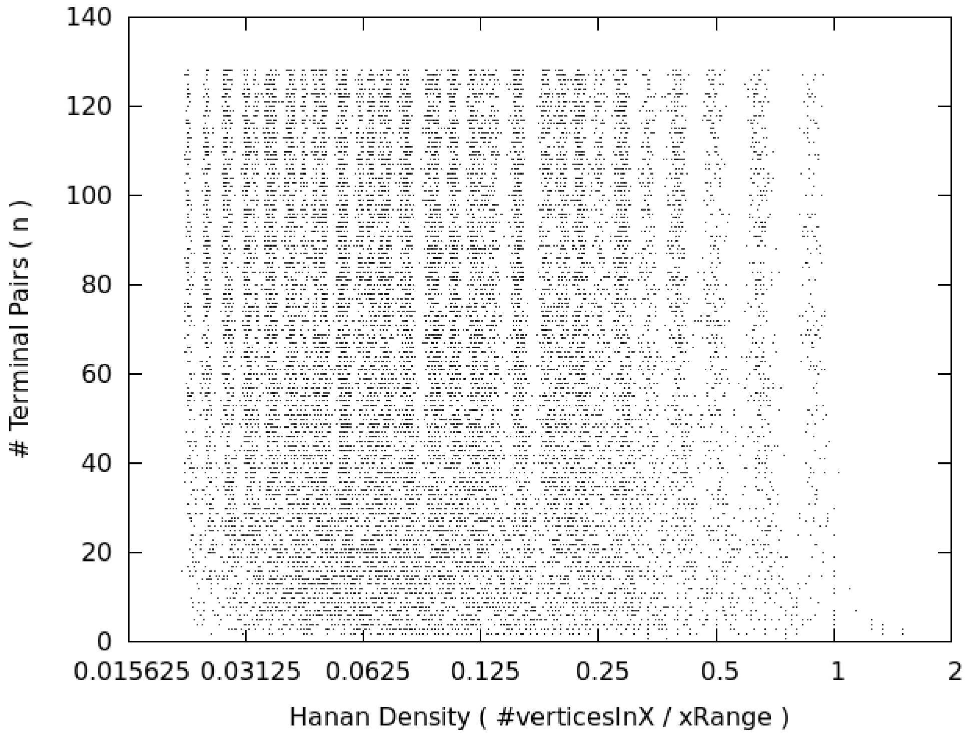
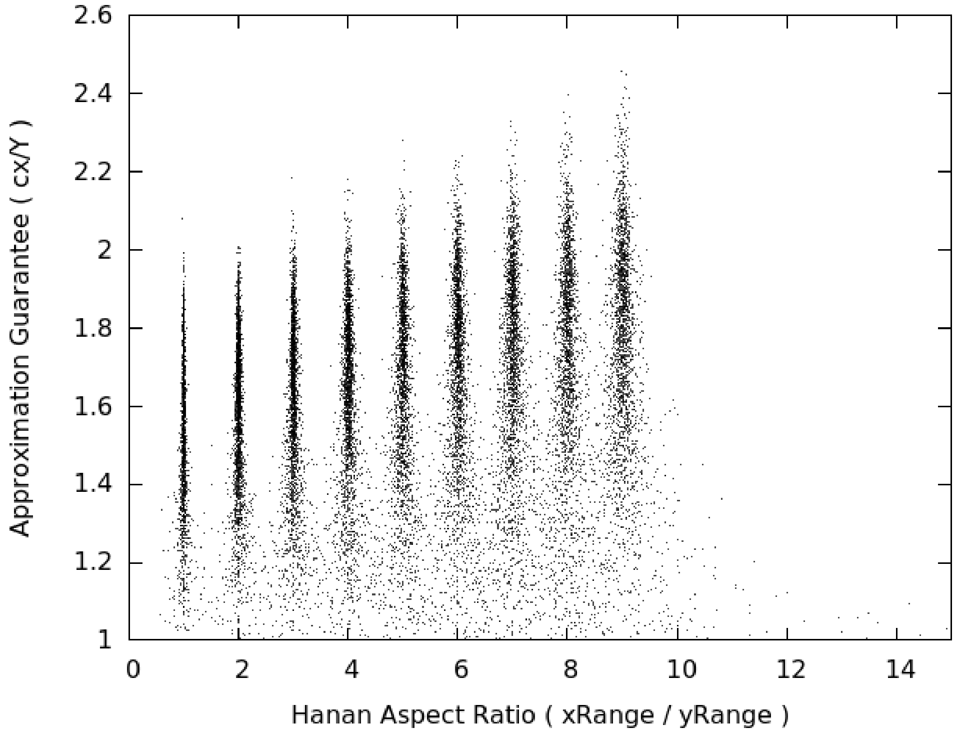


Figure 1.10: Distribution of sampled random instances in the parameter space. Instance size by Hanan aspect ratio (top) and density (bottom).



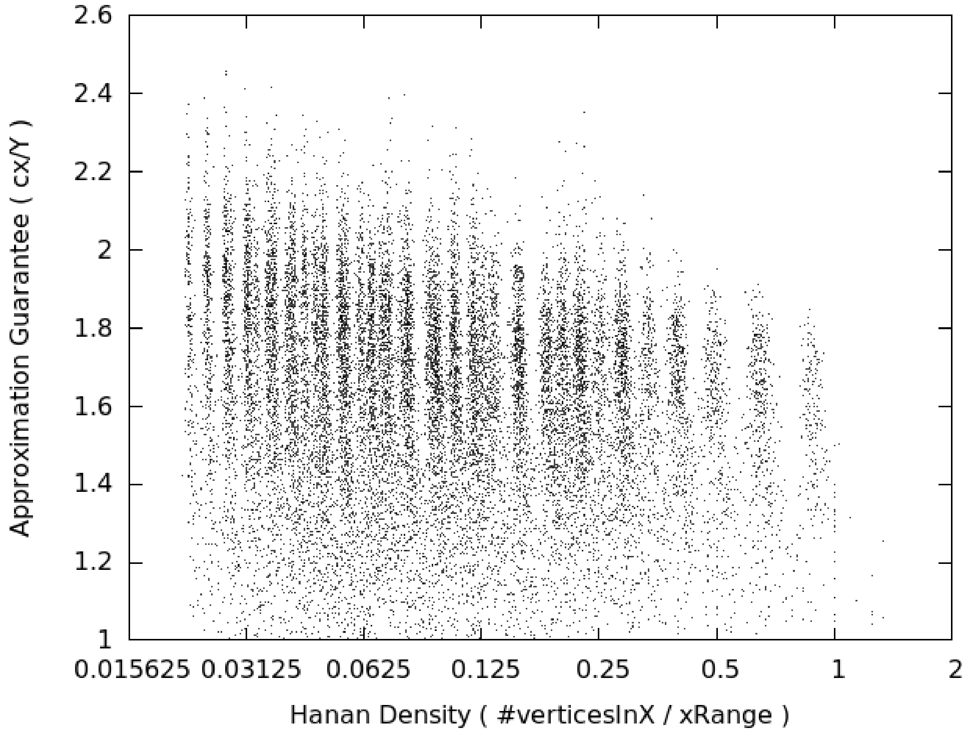


Figure 1.11: Approximation guarantees of sampled random instances by Hanan aspect ratio (top) and density (bottom).

for 2-MMN is disproved (unless $\mathbf{P} = \mathbf{NP}$) by arranging gadgets of about the same shape. Moreover, such a construction *needs* to present a non-constant overlap ω of the boxes associated to the terminal pairs.

On the practical side we have presented a primal-dual algorithm for the more general Path Cover problem in graphs. The algorithm is applicable for d-GMMN and generalizations that require obstacle avoiding for all *or some* of the pairs. The approach performs well on instances of 2-GMMN and produces instance based lower bounds that are close to the cost of the solutions. However, we could only prove a rather weak a priori bound on the approximation ratio of $4\omega^2$ for instances with a maximum overlap of ω. The primal-dual algorithm also solves instances optimal that are worst case instances for [DFK+17].

The $\mathcal{O}(1)$ approximability of 2-GMMN is still unsolved. Given the good practical approximations achievable and the rather specific requirements for $\mathcal{O}(1)$-inapproximability proofs for 2-GMMN – we conjecture that 2-GMMN admits an $\mathcal{O}(1)$ approximation.

Chapter 2

Geometric Map Matching on Heterogeneous Data

In the geometric Map Matching problem one is presented with a sequence of sampled locations, called trace, for which one seeks to find a sequence of movements along directed edges of a graph with straight-line embedding.

The basic data mining problem has many applications in spatial domains but varies on data characteristics. Real world data, like traces or graphs from the OpenStreetMap project [OSM13], does not have homogeneous data quality. Graph details and errors, like missing or misdirected/-placed edges, vary in areas. Each trace has changing sampling density and precision originating from means of travel, measurement errors dependent on locality, different sensors (GPS, GPRS, WLAN, gyroscope, accelerometer) and filter models as well as preprocessing with partially snapping to closest roads of different graphs. Therefore it is unclear how to formalize what a ‘good explanation’ for the spatial measurement is. Especially in the on-line problem setting, in which a data stream is revealed sample-by-sample.

Related Work

There exists a multitude of approaches using additional knowledge of travel means (car, foot, train, etc.) to solve map matching problems for traces with similar characteristics on similar graphs. Survey articles [QON07, Zhe15] describe the challenges and existing approaches and technical report [WWFZ13] compares a variety of different methods on data. There are two conceptually different categories:

The vast majority of methods use hard (e.g. semantic meta-data [AY15]) and soft *numeric parameters* to filter and combine properties of the input data. These numeric parameters are used to balance data characteristics and guide through some selection process [NK09, WZ14]. As it is common to many data mining problems the computation time and quality of the results are mainly determined by the choice of these parameters. Therefore a clear formalized *objective* for the result is missing or determined indirectly by the choice of the parameters. Within this category there are on-line methods that use greedy strategies and global methods that often use dynamic programming. [WWFZ13] compares 10 incremental and 5 global approaches. Eventually additional quality measures are used to compare input data and results. However, careful parameter tuning is *required* which makes unsupervised mining questionable – This is especially problematic with heterogeneous data. Changing data characteristics along and between traces and graphs might require as many as one parameter for each sample and could still fail if the model lacks a parameter accounting for data characteristics. Furthermore, results with bad quality do not guarantee optimality in these additional quality measures. The parameters might just be ill suited. Other approaches leverage homogeneity directly for traces of similar data origin by optimizing a numeric model jointly for a set of input traces [LHK+13].

The other category contains the few *non-parametric* methods. Most prominent approaches clearly define the objective of the problem as minimizing *some* fixed geometric distance for the poly-line of the trace over *all paths* in a spatially embedded graph (c.f. Section 2.2). Often dynamic programming or parameter search is used making these methods computationally more demanding on long traces (c.f. Section 2.1.1). This guards against situations where filters and ill adjusted or learned numeric parameters lead to high values in the geometric quality measure. Therefore these methods are better suited for unsupervised approaches. However, the fundamental mining problem is moved to the question: What is the correct quality measure for the input data? In [BPSW05] the Fréchet distance showed good behavior for rather densely sampled vehicle data on graphs representing road networks. But the results under this geometric similarity measure are *very* sensitive to single outliers and sparse sampled data limiting the usage of poly-line simplifications to speed up calculations [WWFZ13]. Using average or summed Fréchet distances on the other hand suffers if sensor noise is present in some parts of the input when searching for global optimal paths. If the objective asks to globally minimize the similarity measure, algorithms have to go through great length to shave off even small values from an outlier sample. Loosely speaking this effect is similar to over-fitting the flexibility of the metric and increases if data is sampled sparsely.

Some approaches add a numeric model on top of the geometric optimization to overcome heterogeneous data quality [WWFZ13] or restrict geometric optimization on areas provided sensor precision knowledge [WSP06, BBW09] for localization. For sparsely sampled traces of known precision the usage of shortest-paths as connections between candidates showed good results [LZZ+09, EFH+11]. However, the position in the graph closest to a sampled geo-location might well not be the correct position. This moves the question to determining how much 'flexibility' in

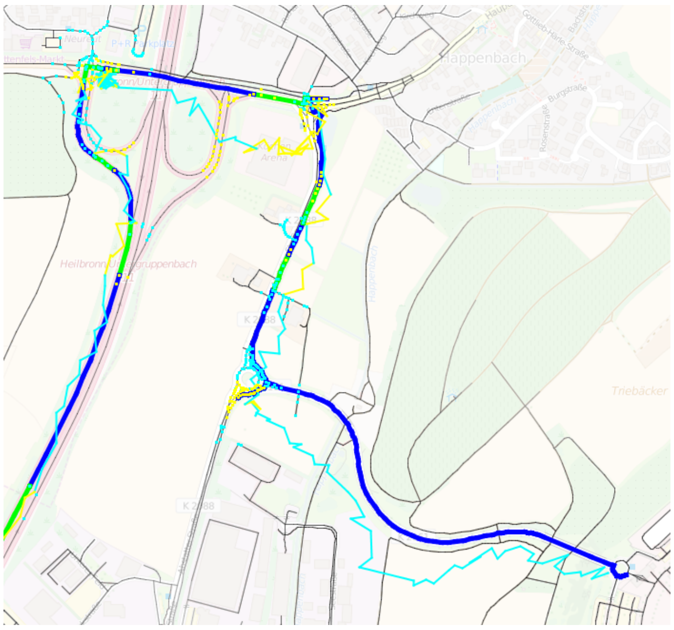


Figure 2.1: OSM trace 1396953 with heterogeneous precision and additional noise (thin line in alternating colors). Graph edges (thin black lines) and dominance decomposition (thick line in alternating colors) are augmented by the result's candidate search space (dots in alternating colors).

the candidate choice is required for each geo-location to achieve a solution that is a 'good' explanation for every part of the trace.

Contribution

We study what quality is achievable for geometric map matching on heterogeneous data. Our concept solely relies on the spatial information of the input and the observation that optimal geometric similarity of parts of the input is limited by the local data quality.

The proposed approach does not focus on filtering or correcting geo-locations of a trace. Using information about the origin of data, sensor type and filtering or error models can clearly improve data quality. However, knowledge on reasonable filter models might often be unknown and different for each trace. We avoid incorporating additional knowledge like hard cut-off filters or soft numeric parameters.

We leverage the graph's edge metric as a tool for localization. This leads to a formalization of a novel objective that is free of numeric parameters and robust in presence of heterogeneous sampling and graph errors: Find a decomposition of the trace into parts along with a sequence of connected paths in the directed graph. Each path must be shortest in the graph's edge metric and minimal in the geometric distance to its respective part of the trace.

The succinctness of the dominance decomposition (DD) objective allows to meet other spatial data mining problems by exchanging graph metric or geometric similarity measure. Computationally, DDs seem more challenging than single objective optimizations. However, DDs allow identification of parts with inherent bad data quality or detail as indication of bad sampling or graph errors. Furthermore, the DD approach provides a simple lossy compression scheme for spatial data with adjustable quality.

We provide a simple algorithm suited for on-line, data stream processing and generic speed-up techniques for exact and approximate calculations. Furthermore, we demonstrate effectiveness and robustness of our methods with experiments on real world datasets with very heterogeneous characteristics. We provide theoretical and empirical comparison with other map matching approaches.

Chapter Outline and Pre-Releases

This chapter reviews well studied geometric similarity measures of curves in Section 2.1. Subsection 2.1.1 describes computational aspects and points out why some measures are theoretically and practically demanding. Section 2.2 describes known extensions to map matching and simplifications, used in practical processing.

Section 2.3 categorizes data characteristics that adversely affect the meaning of global optimization objectives. Section 2.4 derives a new, more locally optimization objective, that is fully formalized in Subsection 2.4.1. Section 2.5 provides an algorithm to solve this objective, along with generic speed-up methods that allow to determine exact and approximate solutions in practice.

Section 2.4.2 provides case studies and extensive comparisons with other optimization approaches for map matching on real world data. Section 2.6 provides details on the empirical evaluation, robustness experiments as well as result quality and time metrics for the algorithm of Section 2.5. Section 2.7 concludes this chapter with a brief discussion on potential extensions and improvements of the new method.

The contents of Sections 2.3-2.7 can also be found in our contribution to the proceedings of the 17th SIAM International Conference on Data Mining [Sey17].

2.1 Geometric Similarity Measures

For two points $q_1, q_2 \in \mathbb{R}^d$ we have the p-norms $\|q_1 - q_2\|_p$ with

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{d} |x_i|^p} \, .$$

and

$$\|x\|_1 \geq \|x\|_2 \geq \ldots \geq \|x\|_\infty$$

for all $x \in \mathbb{R}^d$. A *curve* is a continous mapping $f : [a,b] \to \mathbb{R}^2$ with $a, b \in \mathbb{R}$. Moreover, a curve $P : [0,n] \to \mathbb{R}^2$ is called *polygonal* or *poly-line*, if $n \in \mathbb{N}$ and

$$P(i + \lambda) = (1 - \lambda)P(i) + \lambda P(i+1)$$

for all $i \in \{0, \ldots, n-1\}, \lambda \in [0,1]$. Meaning a poly-line is uniquely determined by a sequence of $n+1$ points with $p_i \neq p_{i+1}$ where points between p_i and p_{i+1} are given by affine combinations. The number of line-segments is denoted by n and

$$l(P) := \sum_{i<n} \|P(i) - P(i+1)\|_2$$

is called length.

A *parametrization* of a poly-line is a continuous mapping $\alpha : [0,1] \to [0,n]$ with $\alpha(0) = 0$ and $\alpha(1) = n$. Moreover, it is called the *arc-length* parameterization, if

$$x = l(P\Big|_{[0,\alpha(x)]})/l(P)$$

for all $x \in [0,1]$. E.g. $P(\alpha(0.5))$ is the half-way point on P.

Let t, t' be poly-lines of n and respectively m line-segments and $\alpha_t, \alpha_{t'}$ their arc-length parameterizations. We define

$$\begin{aligned}
d_A(t,t') &:= && \max_{x\in[0,1]} \left\{ \left\| t\left(\alpha_t\left(x\right)\right) - t'(\alpha_{t'}(x)) \right\|_2 \right\} \\
d_F(t,t') &:= \min_{\substack{\alpha,\beta \text{ monotonous} \\ \text{parameterizations}}} \Bigg\{ && \max_{x\in[0,1]} \left\{ \left\| t\left(\alpha\left(x\right)\right) - t'(\beta(x)) \right\|_2 \right\} \Bigg\} \\
d_{WF}(t,t') &:= \min_{\substack{\alpha,\beta \\ \text{parameterizations}}} \Bigg\{ && \max_{x\in[0,1]} \left\{ \left\| t\left(\alpha\left(x\right)\right) - t'(\beta(x)) \right\|_2 \right\} \Bigg\} \\
d_H(t,t') &:= \min_{\substack{\alpha:[0,1]\to[0,n] \\ \beta:[0,1]\to[0,m]}} \Bigg\{ && \max_{x\in[0,1]} \left\{ \left\| t\left(\alpha\left(x\right)\right) - t'(\beta(x)) \right\|_2 \right\} \Bigg\}
\end{aligned}$$

The distance under arc-length parametrizations d_A and the Hausdorff distance d_H are classical measures of curve similarity. A frequently used illustration of the Fréchet distance d_F is the minimum length leash that is required for a dog and it's owner to traverse their respective poly-lines in a monotonous fashion. We have the following relation between these distance measures for all poly-lines t, t'.

$$d_H(t,t') \leq d_{WF}(t,t') \leq d_F(t,t') \leq d_A(t,t')$$

There are further, *discrete* variants of the Fréchet and the Weak-Fréchet distance that minimize over the parameterizations of sets $\{0,\dots,n\} \to \{0,\dots,n\}$ rather than the intervals. Moreover, there are variants that minimize over discrete or integrated mean values instead of the maximum.

Example 2.1.1. *Consider* $t' = (0,0)(1,0)$ *and* $t = (0,0)(1,0)(0,0)(\frac{3}{4},0)$*. We have* $l(t') = 1,\ l(t) = 11/4$ *and*

$$d_H(t,t') = 0 \quad d_{WF}(t,t') = \frac{1}{4} \quad d_F(t,t') = \frac{1}{2} \quad d_A(t,t') = \frac{7}{11}\ .$$

2.1.1 Computation

Given two line-segments in $\mathbb{R}^2$, the maximal closest-pair distances between their points are also realized with a pair that contains a line-segment's end point. Based on a machine model that provides random access and the arithmetic operations $+, -, *, /, \sqrt[2]{\ }$ and the comparisons $<, =$ on $\mathbb{R}$, one can compute d_A of poly-lines with n and m line-segments within $\mathcal{O}(n+m)$ operations (c.f. Algorithm 3).

Input: $t_0 \dots t_n, t'_0 \dots t'_m$

1. Calculate $l(t)$ and the relative positions in $[0,1]$ for the points t_i on the curve t, and for t' as well.
2. Perform a merge-like sweep over the relative positions while keeping the maximum Euclidean distance between points and the respective interpolations on the other curve.

Algorithm 3: Computation of the Arc-length Distance

A naive approach for calculating d_H computes the perpendicular distances for a point on t to the line-segments of t', and vice versa. This takes $\mathcal{O}(nm)$ operations on such a machine model. In 1997, [BCMS97] proposed a *Scaling Algorithm* for computing d_H in l_p-norms of polygonal curves with $t_i, t'_j \in \{0,\dots,2^k-1\}^2$ and $1 \neq p \neq \infty$ in $\mathcal{O}\big(k(n+m)\big)$ time. The algorithm considers implicit quadtree simplifications of t and t', by rounding to the $i \leq k$ most significand bits: Maintain a set of pairs, of points on t and t', that realize maximal closest-pair distances under scaling from coarse to fine (with subsequent pruning).

For computing the Fréchet distance d_F, there is a known, conditional lower bound on algorithms prohibiting strongly sub-quadratic algorithms. The Strong Exponential Time Hypotheses (SETH) conjectures that there is no $\delta > 0$ such that the k-Satisfiability problem (k-SAT) has an $\mathcal{O}((2-\delta)^N)$ algorithm for all k, where N denotes the number of variables. Assuming this hypothesis, [Bri14] derives a lower bound on algorithms that compute the Fréchet distance. Given a logic formula in conjunctive normal form (CNF) with N variables, M clauses and an $\varepsilon > 0$, the construction provides two curves t, t' with $\mathcal{O}(M \cdot 2^{N/2})$ line-segments with $d_F(t,t') \leq 1$ for satisfiable and $d_F(t,t') > 1 + \varepsilon$ for unsatisfiable formulas. Hence, any algorithm running in $\mathcal{O}(n^{2-\delta})$, where $\delta > 0$ is a small constant, provides an $\mathcal{O}(M^{2-\delta} \cdot (2^{1-\delta/2})^N)$ algorithm for the CNF satisfiability problem.

Alt and Godau [AG95] provide a simple $\mathcal{O}(n^3 \log n)$ algorithm. Their method performs binary search on $\mathcal{O}(n^3)$ distance values paired with solving the decision problem in $\mathcal{O}(n^2)$ operations. This simple approach has the disadvantage, that many, potentially irrelevant, distance values are computed and sorted. The authors achieve a theoretical runtime of $\mathcal{O}(n^2 \log n)$ by using complicated methods for *Parametric Search* that sequentially simulate relevant portions of a parallel run on an AKS sorting network (involving impractically huge constants from expander constructions). The steps to achieve this are much more tedious than the description in [AG95]. Subsequent works [vOV04, vOV05] provide a more practical approach by using Quick Sort for standard Parametric Search, leading to an expected run time of $\mathcal{O}(n^2 \log^2 n)$. While the authors provide more details than [AG95], they do not clearly state polynomials for the Parametric Search. They provide computation times around 2 seconds for random input with 128 line-segments on an 667 MHz Pentium III with 128MB RAM. However, the author's proposed extension package to CGAL is, even after more than ten years, not available to the public. See also [GP13] for a practical approach that uses randomization and Box Sort to meet the asymptotic run time of Cole's Parametric Search, with high probability.

The discrete Fréchet distance, as mentioned above, allows computation with a simple $\mathcal{O}(n^2)$ dynamic program [EM94]. Moreover, there is also a known weakly sub-quadratic algorithm for the discrete Fréchet distance [AAKS14].

2.1.2 Computing the Fréchet Distance

To get a handle on pairs of points that are no further than ε apart, [AG95] defines the *Free Space*

$$F_\varepsilon = \Big\{(p,q) \in [0,n] \times [0,m] \ : \ \big\|t(p) - t'(q)\big\|_2 \leq \varepsilon\Big\} \ .$$

Note that $F_\varepsilon \subseteq F_{\varepsilon'}$ for $\varepsilon \leq \varepsilon'$. The reader may consider $F_\varepsilon\Big((0,3)(8,3),(2,0)(8,6)\Big)$ for $\varepsilon \in \{0,1,2,4\}$ and $F_1\Big((0,1)(2,1),(1,0)(3,0)\Big)$ as illustrations. The Free Space has a natural decomposition into cells, induced from two line segments

$$C_{i,j} = [i, i+1] \times [j, j+1] \ .$$

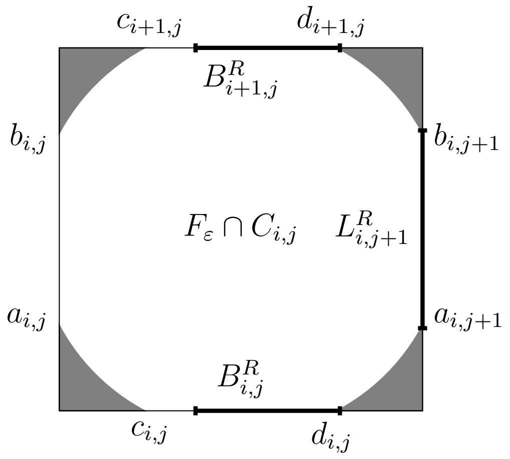
Each $F_\varepsilon \cap C_{i,j}$ is convex, e.g. intersection of the unit square and an ellipse (possibly degenerated to a line). To see that, extend the two affine mappings τ, τ' of the line-segments from their interval domains to whole $\mathbb{R}$. The mapping $f : \mathbb{R}^2 \to \mathbb{R}^2$ with $(r,s) \mapsto \tau(r) - \tau'(s)$ is affine as well. The ε-disk $D_\varepsilon = \{x \in \mathbb{R}^2 \ : \ \|x\|_2 \leq \varepsilon\}$ is convex, and so is

$$F_\varepsilon \cap C_{i,j} = f^{-1}(D_\varepsilon) \ \cap \ [i,i+1] \times [j,j+1]$$

since convexity is preserved under affine mappings and intersections. As a representation of a cell $F_\varepsilon \cap C_{i,j}$, we consider it's (potentially empty) 1D intervals on the left side $L_{i,j} = [a_{i,j}, b_{i,j}]$ and on the bottom side $B_{i,j} = [c_{i,j}, d_{i,j}]$ of the Free Space diagram:

$$\begin{aligned}
a_{i,j} &= \min\{x \in [i,i+1] \ : \ \|t(x) - t'(j)\|_2 \leq \varepsilon\} \\
b_{i,j} &= \max\{x \in [i,i+1] \ : \ \|t(x) - t'(j)\|_2 \leq \varepsilon\} \\
c_{i,j} &= \min\{x \in [j,j+1] \ : \ \|t(i) - t'(x)\|_2 \leq \varepsilon\} \\
d_{i,j} &= \max\{x \in [j,j+1] \ : \ \|t(i) - t'(x)\|_2 \leq \varepsilon\}
\end{aligned}$$

Note that these values can be determined as solutions of two quadratic equations.



Every monotonous curve in F_ε from $(0,0)$ to (n,m) represents two parameterizations, one for t and one for t', with a leash length of no more than ε.

Remark 2.1.2. For the Weak-Fréchet distance d_{WF}, such paths do not need to be monotonous.

Consider aboves t, t' example with $\varepsilon = 1/4$ as illustration. Therefore, we consider this subset of F_ε, which is

$$R_\varepsilon = \Big\{(p,q) \in F_\varepsilon \;:\; \exists \text{ a monotonous curve from } (0,0) \text{ to } (p,q) \text{ in } F_\varepsilon\Big\}\,.$$

This translates to sub-intervals $L^R_{i,j} = L_{i,j} \cap R_\varepsilon$ and $B^R_{i,j} = B_{i,j} \cap R_\varepsilon$ containing the points that are reachable with a monotonous path from $(0,0)$. Their decision algorithm [AG95] is based on the fact

$$d_F(t,t') \leq \varepsilon \iff (n,m) \in L^R_{n-1,m} \iff (n,m) \in B^R_{n,m-1}$$

and that the monotonous reachable top and right intervals of a cell can be computed if left and bottom intervals are already computed. Executing the dynamic program of Algorithm 4 takes no more than $\mathcal{O}(nm)$ operations.

The actual calculation of $d_F\big(t_0 \ldots t_n, t'_0 \ldots t'_m\big)$ is now due to finding the smallest ε such that F_ε contains a monotonous curve. There are three kinds of 'critical' ε values, that potentially change the reachability.

i) $\varepsilon \in \Big\{\|t_0 - t'_0\|_2, \|t_n - t'_m\|_2\Big\}$

ii) ε minimal with $L_{i,j}$ or $B_{i,j}$ becoming non empty

iii) ε minimal with $a_{i,j} = b_{i,k}$ $(j < k)$ or $c_{i,j} = d_{k,j}$ $(i < k)$

```
    Input: $t_0 \dots t_n, t'_0 \dots t'_m$
1   Compute $L_{i,j}$ and $B_{i,j}$ for each $(i,j)$
2   Compute $B^R_{i,0}$ for each $i \in \{0,\dots,n-1\}$
3   Compute $L^R_{0,j}$ for each $j \in \{0,\dots,m-1\}$
4   Compute $L^R_{i,j+1}, B^R_{i+1,j}$ from $L_{i,j+1}, B_{i+1,j}, L^R_{i,j}, B^R_{i,j}$ for each
      $i \in \{0,\dots,n-1\}$ and each $j \in \{0,\dots,m-1\}$
5   return $(n,m) \in L^R_{n-1,m}$
```

Algorithm 4: Decision Algorithm for the Fréchet Distance

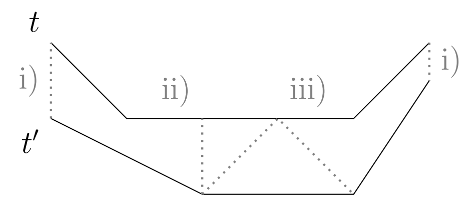


Figure 2.2: Categories of critical values in the Fréchet distance calculation process.

Figure 2.2 shows an illustration of these categories. The $\mathcal{O}(nm)$ values of ii) are the perpendicular distance of a point to a line-segment and the $\mathcal{O}(n^2m + nm^2)$ values of iii) are due the intersection of a bisector (of two points on one poly-line) with a line-segment (on the other poly-line). The algorithm is now standard binary search on the sorted sequence of these values, resulting in a total of $\mathcal{O}\big((n^2m + nm^2)\log nm\big)$ operations.

Theoretical Speedup with Parametric Search

The running time of aboves Binary Search is dominated by sorting the critical values of type iii). The authors achieve an improvement to just $\mathcal{O}(nm \log nm)$ by the expense of a complicated approach with impractical, huge constants. The basic idea is to use Cole's improved version of Megiddo's Parametric Search on a set of input polynomials of degree one. This approach uses AKS sorting networks and was later extended to allow polynomials of low degree as input. The following sketches the major ideas.

Restricting to type i) and ii) values provides a minimal, continuous interval $I = (\varepsilon_1, \varepsilon_2]$ that contains $\varepsilon^* = d_F(t,t')$. Some of the type iii) values might now be in this interval and aboves reachability predicate evaluates to **false** for all such values smaller than ε^*. Parametric Search seeks to find a result interval, containing solely ε^* and no other type iii) values, by means of shrinking the interval around ε^*. In computing the Fréchet distance, the actual value of ε^* is obtainable from the result's interval boundaries. Their approaches seek to exploit monotonicity of the decision problem together with the monotonous increase of the interval boundaries $b_{i,k}, d_{i,j}$ and decrease of the inverval boundaries $a_{i,j}, c_{i,j}$ (functions in ε).

For an $\varepsilon \in I$, the authors consider the set $A(\varepsilon) = \{a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}\}$ of the $\mathcal{O}(nm)$ non-empty Free Space interval boundaries. Sorting such an $A(\varepsilon)$ reveals horizontal $(a_{i,j} = b_{i,k})$ and vertical $(c_{i,j} = d_{k,j})$ passages. Hence, relevant type iii) values in I change the sort order of $A(\varepsilon)$. This seemingly more demanding approach can save unnecessary comparisons in a way that sorting algorithms do.

When aiming for asymptotic bounds, Parametric Search is typically described on a sequential simulation of a parallel sorting algorithm (e.g. AKS sorting networks) involving additionally linear time median calculations and a weighting scheme for the simulation. We describe the more recent, asymptotically weaker but practical approach, that uses QuickSort at two conceptional stages. To provide a simpler but more detailed presentation than [vOV04], we consider the sets

$$\begin{aligned} A_i(\varepsilon) &:= \Big\{a_{i,j}(\varepsilon), b_{i,j}(\varepsilon) : 0 \leq j < m\Big\} && 0 \leq i < n \\ C_j(\varepsilon) &:= \Big\{c_{i,j}(\varepsilon), d_{i,j}(\varepsilon) : 0 \leq i < n\Big\} && 0 \leq j < m \end{aligned}$$

of mappings. Since these mappings are either monotonous increasing or decreasing for $\varepsilon \in I$, two mappings $f, g \in A_i$ have at most one intersection point $r \in I$. Moreover, one derives the order of $f(\varepsilon^*)$ and $g(\varepsilon^*)$ from the order of r and ε^*. The main idea is to simulate the sorting of the sets $A_i(\varepsilon^*), C_j(\varepsilon^*)$, despite ε^* is unknown, to derive subsequently smaller interval bounds.

We start the iterative QuickSort variant on the sub-arrays

$$\Big\{S_i\Big\} = \Big\{A_i(\varepsilon^*) : 0 \leq i < n\Big\} \cup \Big\{C_j(\varepsilon^*) : 0 \leq j < m\Big\}$$

and describe one step. Choose a pivot mapping p_i from each S_i and collect the intersections points of p_i and mappings of S_i within I in a set $\mathcal{B}$, called *batch*. Sort these $\mathcal{O}(nm)$ values:

$$\mathcal{B} = \{r_1, \dots, r_l\}$$

Use Binary Search paired with calls to the (expensive) decision algorithm to find a new, smaller interval $I = [r_i, r_{i+1}]$ that contains ε^* – This takes $\mathcal{O}(nm \cdot \log nm)$ operations. Now, use the new I to actually pivot each S_i into two sub arrays, one with $f(\varepsilon^*) < p_i(\varepsilon^*)$ and one with $f(\varepsilon^*) \geq p_i(\varepsilon^*)$.

The expected number of such pivoting steps is $\mathcal{O}(\log(n+m))$, leading to a total, expected running time of $\mathcal{O}(nm \cdot \log^2 nm)$ for QuickSort based Parametric Search for computing $d_F(t, t')$.

However, exact calculations of intersection points of quadratic solution formulas, which contain square roots, is a problem in practice. Neither the works of [AG95] nor [AERW03] provide a reduction to a sorting problem of simpler, polynomial functions in ε – Tough [vOV04] claims that this is possible in one, short sentence.

2.2 Fréchet Map Matching in Graphs

The described decision algorithm for d_F (c.f. Algorithm 4) allows to fill the dynamic programming table in several sequences – e.g. column-wise from bottom to top. This allows a simple generalization to graphs with straight-line embedding of the nodes.

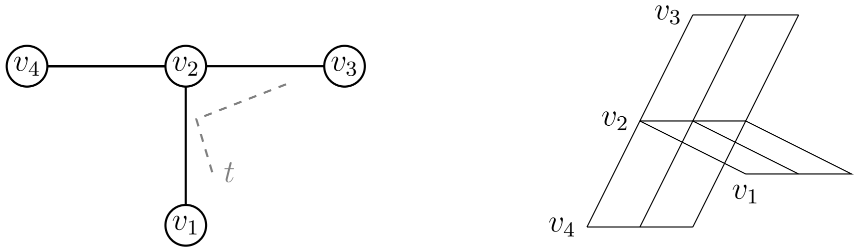


Figure 2.3: Illustration of a poly-line $t = t_1 t_2 t_3$ and a spatially embedded graph graph (left) and the induced Free Space Surface (right).

Consider each edge as the directed line-segment connecting its nodes. The regular Free Space of the edge and a polygonal curve $t = t_1 \dots t_l$ is a single strip of Free Space cells, say from left to right. The *Free Space Surface* is obtained by gluing together all these Free Space strips at the nodes of the graph. One may think of this construction as l copies of the graph, stacked from left-to-right. There are $\mathcal{O}(|E| \cdot l)$ cells in the Free Space Surface and adjacent edges in the graph have adjacent cells in the construction. Moreover, a monotonous curve from *some* edge at the left (the copy of t_1) to some edge at the right (the copy of t_l) identifies, again, two continuous, monotonous parameterizations of t and some directed path in the graph.

Aboves decision algorithm is adapted by filling the dynamic programming table columns, in a sweep-line fashion, from left-to-right. In this setting, the result path is not yet fixed, as is the sequence in which the cells of a column need to be determined. However, one simply updates the cell's $B^R_{i,j}$ values (the intervals stacked above the graphs nodes) in the sequence that the sweep-line intersects their left boundary. Using Fibonacci Heaps in column steps, this decision algorithm takes $\mathcal{O}\Big(l\,(|E| + |V| \log |V|)\Big)$ operations.

The actual optimization is performed analogously to aboves discussion on parametric search, adding a $\mathcal{O}(\log(l|E|))$ factor with the impractical huge constants. For graphs with $|E| \in \mathcal{O}(|V|)$, this method of [AERW03] provides a geometric map matching algorithm with

$$\mathcal{O}\Big(\ l|E|\ \log(l|E|)\ \log|E|\ \Big)$$

operations. However, the simple binary search algorithm takes $\mathcal{O}(l|E|^2 \log(l|E|))$ operations to sort type iii) values on a graph with $|E| > l$.

2.2.1 Weak-Fréchet Map Matching in Undirected Graphs

The Weak-Fréchet distance is an alternative, with simpler decision and computation algorithms [AG95] (c.f. Section 2.2.1). An experimental study on 45 GPS tracks of vehicles (30s sampling, <100 line segments on average) on a road network graph ($14,356$ edges for Athens, Greece) finds that global geometric map matching, based on approximations to Fréchet and Weak-Fréchet distances, provides better results, measuring average Fréchet distance between input and output, than an ad-hoc incremental algorithm [BPSW05]. The authors did not find differences in Fréchet and Weak-Fréchet results on this dataset and undirected graph. However, the Weak-Fréchet objective does not respect edge directions, since non-monotonous parameterizations are used, and the binary search on the decision problem is much slower than their incremental map matching approach. An output sensitive Weak-Fréchet map matching algorithm is presented in [WSP06]. This approach consequently extends ideas from [AG95] to decide and compute the Weak-Fréchet distance based on graph traversal. To avoid storing the whole Free Space Surface of the graph, their *Adaptive Clipping* algorithm uses a fixed, numeric bound to consider only relevant parts of the graph for computation. The authors achieve computation times around 45 seconds for the longest poly-lines (around 200 points) on an Intel Pentium 4 with 3.6GHz and 2GB RAM.

This section provides a simple algorithm formulation that does not require the knowledge of a clipping bound value. A practical approach can use Hashing to store relevant cells of the Free Space Surface.

We consider the graph $G_r = (V_r, E_r)$ that originates from placing nodes on the 4 cell boundaries over the Free Space Surface of $G = (V, E)$ and $t = t_0 \dots t_n$. The node set $V_r = E \times \{0, \dots, n\} \cup V \times \{1, \dots, n\}$ and there are 6 edges per Free Space cell to connect the 4 boundary nodes (c.f. Figure 2.4). The graph G_r has $\mathcal{O}(|E|n)$ nodes and edges. For adjacent $(k, l), (k', l') \in V_r$, we set the edge weight $c(k, l, k', l')$ to the smallest ε such that the two Free Space intervals become non-empty. That is the maximum of the two respective type ii) critical values. See Algorithm 5 for a search of a minimum bottleneck value path in G_r.

For the analysis, we assume $\mathcal{O}(1)$ operations for weight calculation and operations on the hash set `reachable`, that contains pairs of integers which identify nodes in V_r.

The procedure `Q.enqueue` is called at most a total of $\mathcal{O}(n|E|)$ times. Apart from that, the while loop is executed no more than $\mathcal{O}(n|E|)$ times, leading to an total of

$$\mathcal{O}(n|E| \log(n|E|))$$

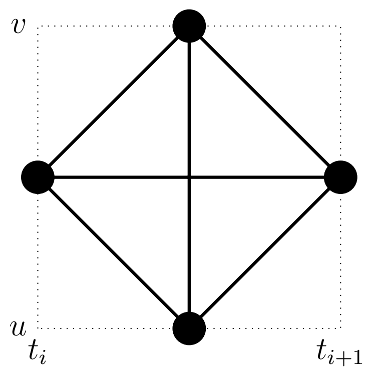


Figure 2.4: Illustration of the overlay graph G_r over one cell of the Free Space Surface of $G = (V, E)$ and $t = t_0 \dots t_n$. The cell originates from an edge $\{u, v\} \in E$ and a line-segment $t_i t_{i+1}$ on t [AG95].

```
   Input: $G = (V, E), t = t_0 \dots t_n$
 1 res := $-\infty$
 2 reachable := $\emptyset$
 3 for $e \in E$ do
 4     Q.add($e$, 0, minDist($e, t_0$))      /* type i) critical values      */
 5 end
 6 Q.makeHeap()
 7 while $Q \neq \emptyset$ do
 8     $(k, l, \varepsilon)$ := Q.extractMin()
 9     res := max($\varepsilon, res$)
10     reachable.add($k, l$)
11     if $(k, l) \in E \times \{n\}$ then
12         return res
13     end
14     for $(k', l') \in$ neighbours$(k, l) \setminus$ reachable do
15         Q.enqueue($k', l', c(k, l, k', l')$)
16     end
17 end
```

Algorithm 5: Weak-Fréchet Dijkstra

operations. Path computation can be achieved by storing additional predecessor nodes.

There are many ways to avoid the $|E|$ elements in the heap initialization for a map-matching query – E.g. restricting to paths that start at a node of G. This allows to find such nodes, on demand, with a precomputed geometric nearest-neighbor-search data structure that enumerates the next closest node. In practice, many graphs allow to simply store and retrieve the next closest edge of G to the point t_0 based on a simple grid data structure.

2.3 Obstacles in Formalizing Objectives

Streams of spatial data present different kinds of data problems [QON07]. Isolated samples have been stored or transmitted that are not even close to the real position (*sporadic errors*). These problems are often successfully overcome with very simple filters ignoring the sample. Some traces use GPS correction maps for the sampling but others have a slight offset to true locations. Geo-locations were rounded to numbers with lower precision. These *systematic errors* are resolvable by smoothing, learning a correction map with very few numeric parameters or allowing some fixed precision radius when candidates for a geo-location are chosen. This work focuses on the robustness against data problems that are heterogeneous along a trace as well as across traces. We distinguish between the following three conceptual categories.

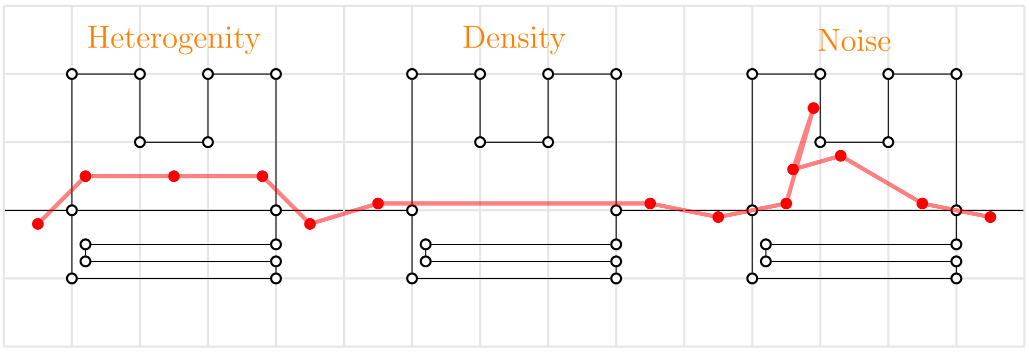


Figure 2.5: Categories of input characteristics in trace (red, $t = t_1 \ldots t_{14}$ from left to right) and graph (black) data. Grid cells (light gray) indicate a distance of 1 in x and y direction.

Heterogeneity Some samples have higher precision than others or some parts of the trace describes movement that can not originate from moves along edges in the graph. See Figure 2.9 for an example. Reasons for that include missing edges, erroneous edge directions, filtering of geo-locations due to local GPS precision, disturbances through glass, steel, reflections, weather, tunnel, trees or clear sky.

Density Discrepancy between the density of geo-location sampling and edges in the graph. E.g.: i) changing travel means (foot, bike, car ...), ii) parts of the graph may be detail rich or iii) geo-locations were sparsified in a preprocessing. See Figure 2.11 for an example.

Noise Sensor noise when there is no movement at all – e.g. stop at a traffic light. Connecting positions on edges that are closest to geo-locations might require moving along edges far away from this part of the trace. See Figure 2.10 for an example.

Figure 2.5 illustrates the three categories with trace $t = t_1 \dots t_{14}$ (red, left to right). The spatially embedded graph in this example consists of three blocks of similar structure. The left block allows traversal via a detour d_l (upper part), a direct path s_l (middle part) and a maze m_l (lower part). The middle block allows traversal via d_m (upper part) and m_m (lower part), while the right block has the paths d_r, s_r and m_r. We consider how paths with globally minimal geometric similarity measures (c.f. Section 2.1) are affected by these data characteristics.

- **Weak-Fréchet Distance:** The path $p = m_l m_m s_r$ provides minimal $d_{WF}(p, t)$ in this graph. This is determined by the Euclidean distance of 1.5 of sample t_{10} to s_r. Therefore, any traversal in the left block and m_m in the middle block are allowed.

- **Fréchet Distance:** The path $p = d_l m_m d_r$ provides minimal $d_F(p, t)$ in this graph. It is determined by the Euclidean distance (≈ 1.86) of a point around the half-way point, between t_6 and t_7, and the lower right corner node of m_m. This is less than 1.9, as provided from the traversal via d_m. This allows any of the s or d traversals for the left and right block.

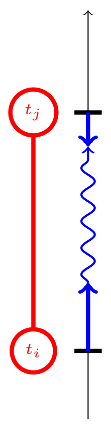
- **Hausdorff Distance:** A path with minimal d_H can essentially start on mid of m_l, traverse m_m and end on mid of s_r. This is determined by the Euclidean distance of 1.5 of sample t_{10} to s_r.

One may well think of additional cases, in which a local data problem affects global optimization of a geometric similarity measure.

2.3.1 Notation

We describe our approach in the setting of geo-locations and graph nodes lying in the plane to simplify notation. In the geometric map matching problem we are provided with an input poly-line $t = t_1 \dots t_n$ of geo-locations $t_i \in \mathbb{R}^2$ that might have heterogeneous precision, noise and varying density. Furthermore, a directed graph $G = (V, E)$ with straight-line embedding $\eta : V \to \mathbb{R}^2$ and an edge weight function $\gamma : E \to \mathbb{N}$ is provided. In all our experiments we set $\gamma(e)$ to the length of the edge e in the embedding, rounded to a certain precision (c.f. Section 2.6). Note that the objective and our methods allow different edge weights for different spatial data mining applications. To simplify notation of the following, we assume that G is strongly connected and shortest paths are unique (e.g. symbolic perturbation of the edge metric γ).

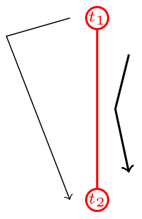
A geo-location $t_i \in \mathbb{R}^2$ has up to $|E|$ *candidates* of the form $(e, p) \in E \times [0, 1]$. Here p is such that the Euclidean distance between t_i and the relative position p on the directed edge e (regarding its node embedding) is minimal. Note that if the edge allows movement in both directions, the candidate positions of both directed edges coincide. Furthermore, the candidate positions on two adjacent edges might coincide in their common graph node. We denote the set of candidate positions of t_i with $C(i)$ and one may think of it as a list of (e, p) pairs, sorted by their Euclidean distance to t_i. Given the *true* candidates c and c' for positions t_i and t_j one can derive a movement along the edges between these two candidates by choosing the shortest path with respect to the edge weights γ. We denote this path with $\mathrm{SP}_\gamma(c, c')$. This is a natural choice for dense *and* sparse sampling [LZZ+09, EFH+11], if the true candidates are known. Such a γ-*shortest path* between candidates in turn has a poly-line originating from the straight-line embedding of the graph. In the following we frequently compare these with the poly-lines of a part $t_i \dots t_j$ with a geometric similarity measure (lower values indicate higher similarity).



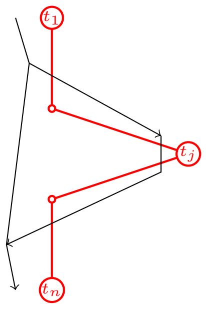
In the remaining sections of this chapter, we frequently use the geometric similarity measure $d = d_A$ as introduced above. Such parameterizations are computationally simple and a pessimistic measure of similarity (c.f. Section 2.1). However, they provide a simple bijective mapping between the two poly-lines which makes them suited for *lossy compression*. If $d(p, t)$ does not exceed a desired quality, one can store the poly-line p rather than t (c.f. Figure 2.17). This is a metric on poly-lines. However, the proposed decomposition approach only needs a geometric similarity measure of poly-lines. Hausdorff or Fréchet distances are less pessimistic choices for d.

2.4 What is a 'good explanation'

Our approach is based on the idea that parts of the spatial measurement have different data quality and we seek to explain the input poly-line with a sequence of γ-shortest paths. Other than global geometric optimization, we leverage the edge metric of the graph as a tool for localization. For a poly-line with n points we have $\mathcal{O}(2^n)$ decompositions into parts, where each part $t_i \dots t_j$ might be explained with any of the $\mathcal{O}(|E|^2)$ paths connecting a $c \in C(i)$ with some $c' \in C(j)$.



For only two geo-locations t_1, t_2 in $\mathbb{R}^2$ the choice of a 'good' matching is simple: Choose $c \in C(1)$ and $c' \in C(2)$ such that $d(\mathrm{SP}_\gamma(c, c'), t_1t_2)$ is minimal. This is a natural choice irrespective of coarse or fine sampling that implicitly defines the required 'flexibility' for the candidate selection of t_1 and t_2. Having a better geometric similarity than the other γ-shortest paths is the motivation for our formal definition of a *dominating* path.



However, lifting this idea to a map matching objective for the entire trace poses a problem. On the one hand, geometric minimal γ-shortest paths of two consecutive pairs of geo-locations are not necessarily connected, but on the other hand, the ground truth movement may consists of several γ-shortest paths leading to rather high $d(p, t_1 \dots t_n)$ values for any γ-shortest paths p between $C(1)$ and $C(n)$.

We address this in our novel map matching objective by requiring connectivity but allow each part of the poly-line $t_1 \dots t_n$ to be explained by a different γ-shortest path. Provided k such decomposition parts and shortest paths, we can think of an explanation as a vector of length $n-1$ listing the k values of the geometric similarity measure d in sequence. There are many ways to formalize an map matching objective based on such a decomposition. For example, minimizing the maximum d value of a part suffers similarly to global geometric optimization, if parts of the input have inherent bad data quality (e.g. a missing graph edge). We aim for an objective that allows evaluation in an on-line, stream processing, fashion but still provides optimality. Consequently, we allow to decline *some* earlier candidate choices, if ignoring them allows a γ-shortest path to provide a better explanation for the suffix, in terms of the geometric similarity measure d.

2.4.1 Dominance Decompositions

Each ground truth movement along the graph edges is a sequence of γ-shortest paths of maximal length. Let g be one of these rather long paths and $t = t_0 \dots t_n$ a perfectly sampled poly-line with $d(g, t) = 0$. Since a splitting of a γ-shortest path results in two γ-shortest paths, the perfect sampling t further refines $g = s_1 s_2 \dots s_n$, where each s_i originates from a rather short γ-shortest path. Now, heterogeneous precision, noise and varying density lead to the fact that each $t_i \in \mathbb{R}^2$ bears a different Euclidean distance to its true position on g. A t_i might now be closer to different candidates in the graph G than to its true position on g.

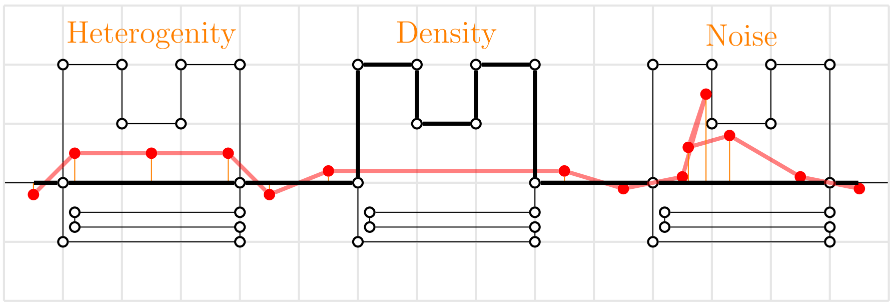
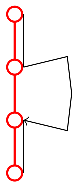


Figure 2.6: Example graph (thin black lines), location samples $t = t_1 \dots t_{14}$ (thick red line) and γ-shortest path decomposition of the ground truth movement $g = s_1 s_2 \dots s_{13}$ (thick black line), as induced by t.

Therefore, if we were forced to extend a fixed path that is a ‘good explanation’ for $t_0 \dots t_1$ to one of the $|E|$ candidates of t_2, all such choices of p could be a ‘poor explanation’ for $t_1 \dots t_2$. If there is a ‘better explanation’ for $t_0 \dots t_2$ we would rather use this γ-shortest path q as an explanation for the entire *part* than explaining $t_1 \dots t_2$ with such a p. (c.f. Figure 2.6).

Note that the sample t_2 does not necessarily need a small Euclidean distance to its ground truth position. It is sufficient that $q = s_1 s_2$ is geometrically more similar to $t_0 \dots t_2$ than a p is to $t_1 \dots t_2$. If the sampling, in light of parts underlying different kinds of errors, provides some samples which make g a better geometric explanation, we can reconstruct the γ-shortest paths of g from t.



Since the graph might have ‘errors’ as well, the existence of a true movement along the directed edges seems problematic for a real world GPS trace. However, we can equally consider this part of the sampled trace as one with bad data quality. Meaning, a ‘good explanation’ for this part has inherent high geometric distance values.

We capture this in the following recursive, formal definition of the *dominating path* for a suffix. Note that dominating candidates $\widehat{C}$ merely describe ends of dominating paths, allowing us to enforce connectivity. Let $C(1), \dots, C(n)$ be the candidates for a poly-line $t = t_1 \dots t_n$.

Definition 2.1. For $n = 1$ all candidates of t_1 are called **dominating candidate** for the trivial 1-suffix and $\widehat{C}(1) = \{1\} \times C(1)$.

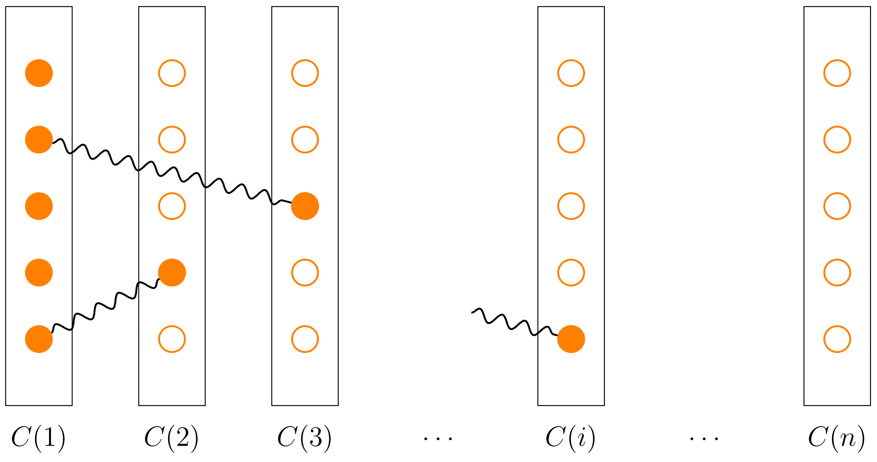


Figure 2.7: Illustration of the dominating candidates in $\widehat{C}(n-1)$ (orange). These are starting points of γ-shortest paths that are considered for the n-suffixes.

For $n > 1$ consider all γ-shortest paths p starting in a dominating candidate c' with $(j, c') \in \widehat{C}(n-1)$ and ending in some $c \in C(n)$. We call c the dominating candidate iff. p minimizes $d(p, t_j \dots t_n)$, resolving ties by choosing higher j. We call p the **dominating path** for the n-suffixes. More formally, let

$$A = \Big\{(j, c', c) : (j, c') \in \widehat{C}(n-1), c \in C(n)\Big\}$$
$$(j, c', c) = \operatorname*{argmin}_{(j,c',c)\in A} d\Big(\,\mathrm{SP}_\gamma(c', c), t_j \dots t_n\Big)\ .$$

We call $p = \mathrm{SP}_\gamma(c', c)$ the dominating path and (n, c) the dominating candidate for the n-suffixes. We set $\widehat{C}(n) = \widehat{C}(n-1) \cup \big\{(n, c)\big\}$.

See Figure 2.7 for an illustration. Note that the dominating path is uniquely defined and has optimal geometric similarity over the suffixes of the poly-line t. One could aim for a more general definition of suffix dominating path by allowing more than *one* dominating candidate for a sample, leading to a Pareto set of suffix dominating paths for a trace. Now we can formally define the computational task of the novel map matching objective.

Given: A trace $t = t_1 \dots t_n$ with $t_i \in \mathbb{R}^2$ and a directed simple graph $G = (V, E)$ with straight-line embedding $\eta : V \to \mathbb{R}^2$ and edge costs $\gamma : E \to \mathbb{N}$.

Find: A decomposition $\{i_1, \dots, i_m\} \subseteq \{2, \dots, n\}$ of t ($i_m = n$) and a connected sequence of γ-shortest paths $p_1 \dots p_m$ such that each p_j dominates the i_j-suffix of t.

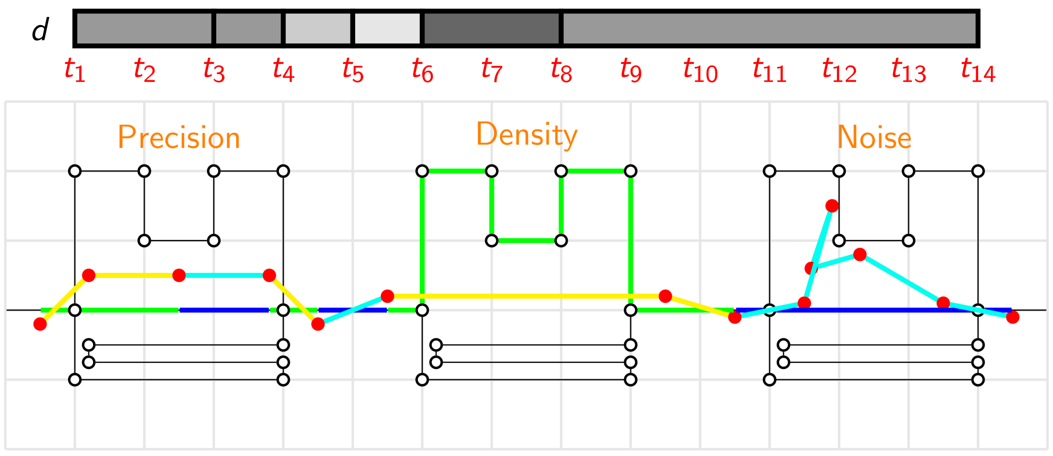

d
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14}
Precision
Density
Noise

Figure 2.8: Example of a dominance decomposition with 6 parts for $t_1 \dots t_{14}$ (red). Alternating colors indicate correspondence between γ-shortest paths (green/blue) and respective parts of the poly-line (yellow/turquoise). Values of the geometric similarity measure d are indicated in shades of gray.

Such a solution is called *dominance decomposition* of t and each path p_i is optimal for the suffix of its respective part. Using the geometric similarity measure $d(,)$, as introduced above, each solution provides a bijection between points along the trace and points along the path. Note that the objective is *not* to globally minimize the maximum error of piece-wise arc-length parameterizations, since we require each factor to be locally minimal. There is always such a decomposition since the graph is strongly connected and the decomposition is unique by definition. For consistency consider an arbitrary path p in G and perfectly sampled geo-locations – e.g. geo-locations of each node are present in the input. Each factor of a γ-shortest path factorization of $p = p_1 \dots p_m$ dominates a suffix with a zero value in the geometric similarity measure. See Figure 2.8 for a conceptual example and Figures 2.1, 2.9, 2.10 and 2.11 for examples on real world data.

This geometric objective is independent of numeric parameters like allowed local error-radii between geo-locations and candidates, relative importance between keeping bearing, speed and other frequently used numeric parameters.

2.4.2 Other Map Matching Objectives

In contrast to globally minimizing a rather flexible geometric distance measure or score, this objective formulates minimal solutions for each part of the trace. Therefore parts of a DD with high yet optimal geometric (dis)similarity values indicate data problems of trace or graph in *this area.* Applications can use this for

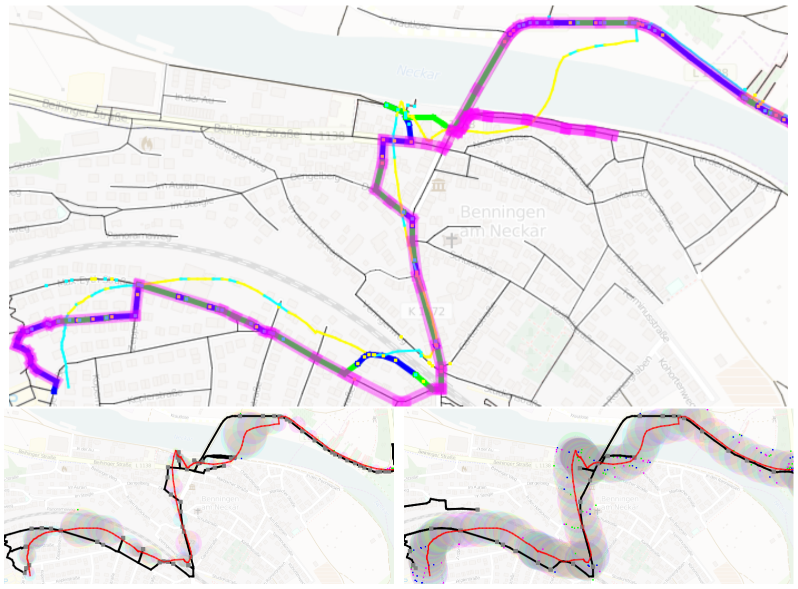


Figure 2.9: Comparison of DD map matching (thick green/blue lines) with HMM [NK09] map matching (thick magenta lines) and SP [EFH+11] (thick black lines) map matching on real world trace data (thin yellow/turqoise or red lines). See Section 2.4.2 for details. OSM trace 1396953 track 16 exhibits *heterogeneous* data quality.

identifying errors in graph or trace data.

SPX map matching [EFH+11] computes a shortest path, under the constraint to traverse a sequence of sets of vertices, each of which contains the vertices that are within a radius of X (in meter) around the respective location measurement. We also provide comparison with [NK09] as provided by the GraphHopper Project [Kar08, Hol] implementation. The method maximizes the probability score of a Hidden Markov Model (HMM) globally. We use the default numeric parameters of this implementation.

Figures 2.9, 2.10 and 2.11 provide case comparisons on data of the OpenStreetMap trace database. DD factors of input poly-lines (thin lines) are plotted with alternating yellow and turquoise and the respective γ-shortest paths (thick lines) with green and blue on top of underlying graph edges (thin black lines). The background shows an underlying rendering from the OSM map in very light opacity. For simple comparison we add the results of HMM as an overlay (very thick magenta line). The results of SP9 and SP61 is given as a thick black line, where the trace (thin red line) is augmented by circles indicating the respective allowed candidate radius, containing at least one graph node, for each location sample. See Section 2.6 for further details.

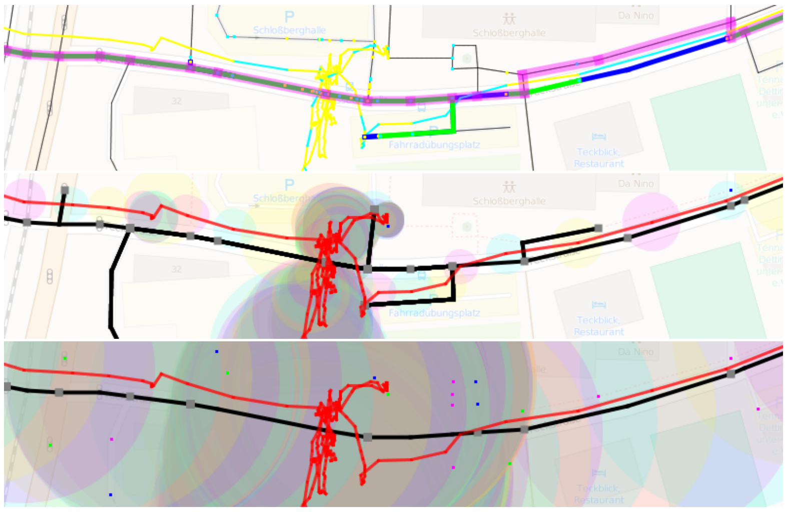


Figure 2.10: Comparison of DD map matching (thick green/blue lines) with HMM [NK09] map matching (thick magenta lines) and SP [EFH+11] (thick black lines) map matching on real world trace data (thin yellow/turqoise or red lines). See Section 2.4.2 for details. OSM trace 537916 track 1 exhibits an intermediate part with *noise*.

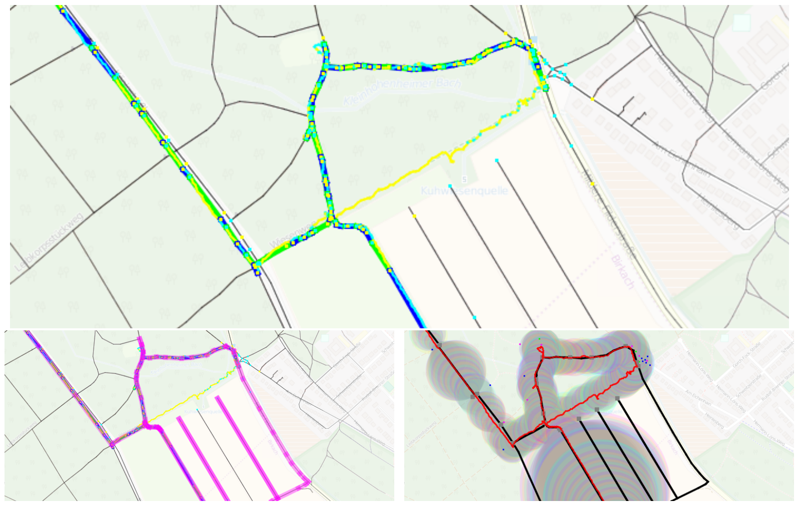


Figure 2.11: Comparison of DD map matching (thick green/blue lines) with HMM [NK09] map matching (thick magenta lines) and SP [EFH+11] (thick black lines) map matching on real world trace data (thin yellow/turqoise or red lines). See Section 2.4.2 for details. OSM trace 658956 track 0 exhibits missing graph edges.

Figure 2.9 exhibits heterogeneous precision. The first parts of the sampled poly-line have low precision. However, precision improves in between and later on. DD identifies intermediate parts with better geometric matches. HMM (very thick magenta line) tends to ignore these parts and provides unreasonable results for the movement over the river's bridge, which is sampled in low precision. SP9 (bottom left) returns several unreasonable detours, just to meet the fixed candidate sets. This is not fully resolved with the parameters of SP61 (bottom right), which ignores intermediate detours of rather good sampling precision.

Figure 2.10 exhibits intermediate sensor noise, after entering a parking lot. DD does not lack to provide the parking lot detour in the explanation and is unaffected by the sensor noise. HMM ignores the noise but the parking lot detour as well. Furthermore, the noise part leads the global optimization of the probability score to favor an unreasonable detour on a rather good sampled earlier part. SP9 provides unreasonable results for the noisy parts and SP61 ignores the parking lot.

Figure 2.11 exhibits missing graph edges. This situation is similar to location measurements and graph edges having different sampling *densities*. DD is only affected in **one** part of the trace and provides high precision matching results later on. HMM and SP61 provide very unreasonable results for this situation.

We conclude the section with an extensive comparison on a heterogeneous, real world dataset (c.f. Table 2.2). Finding a measure to judge the quality of output and input poly-line is equally hard as the map matching problem itself: One would design an algorithm to return optimal results under this measure. Figure 2.12 simply shows the discrete average distances of the geometric similarity measure d, comparing output with input poly-line. HMM was unable to find matchings for $2,128$ traces ('broken sequence' exception), which we excluded from plotting in Figure 2.12.

2.5 Finding Dominance Decompositions

Despite the potentially big sets of γ-shortest paths considered for minimizing the geometric poly-line distance, we can efficiently enumerate relevant candidates using the following simple observation: For two consecutive geo-locations $t, t' \in \mathbb{R}^2$ consider the candidate (e, p) closest to t and (e', p') closest to t'. Their Euclidean distance is $\varepsilon_l = ||t, (e, p)||_2$ and $\varepsilon_l' = ||t', (e', p')||_2$. Let q denote the poly-line of the γ-shortest path connecting (e, p) to (e', p'). The value $\varepsilon_p = d(q, tt')$ is an upper bound on the Euclidean distances of potentially better candidates for t and t' since every arc-length re-parameterization of a path using candidates of higher distance immediately has higher $d(\ , t \dots t')$ values, resulting from an endpoint. Many geometric similarity measures d provide this property. We therefore call $d(\ , \ )$ *compatible* to $|| \ ||_2$.

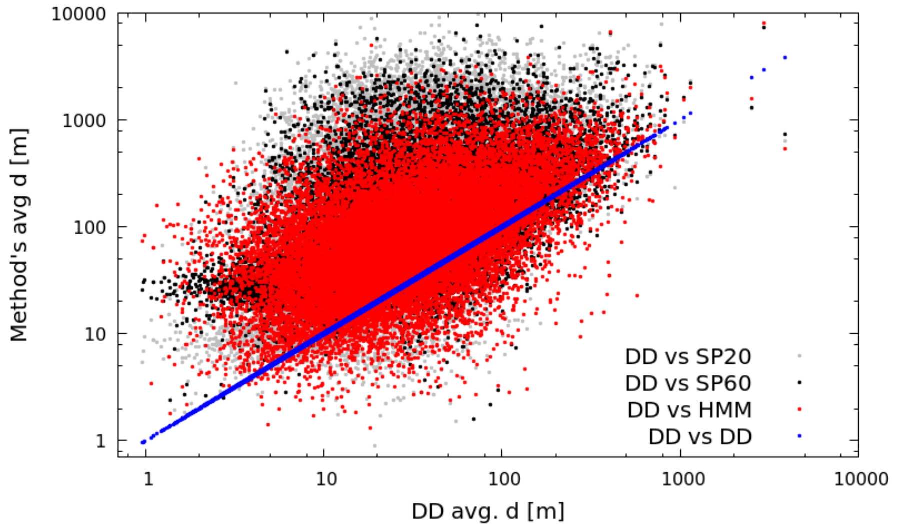


Figure 2.12: Comparison of map matching methods. Each trace is represented with 4 points, where the x-coordinates are the discrete average error of the DD and the y coordinates are discrete averages of similarity measure d for SP20 (gray), SP60 (black), HMM (red) and, again, DD (blue).

This self limiting nature between the 'local errors' ε_l and the 'path errors' ε_p enables a search via local enumeration of candidates with Euclidean distance up to ε_p. Algorithm 6 describes a dynamic programming approach for finding dominance decompositions. Recall that $C(i)$ denotes the list of all candidates for t_i in increasing Euclidean distance. Given the recursive nature of the objective, we keep all intermediate results and finally retrieve the decomposition parts with backtracking. The `dom` array contains the dominating candidate for each sample and the `pre` array contains the sample index of its best predecessor. E.g. the last part of the decomposition is $t_{\text{pre}[n]} \dots t_n$ with the dominating path $\text{SP}_\gamma\Big(\text{dom}[\text{pre}[n]], \text{dom}[n]\Big)$.

2.5.1 Speeding up the calculations

The running time of Algorithm 6 depends on how well the trace is mapable to the graph. Let the complexity parameter $K \in \mathcal{O}(|E|)$ denote the maximal number of relevant candidates of a sample in the input (e.g. iterations of the repeat loop). We use that each of the $(n+1)$ candidate sets can be determined and stored in an min heap in $\mathcal{O}(|E|)$ time. Extracting a candidate with minimal distance costs $\mathcal{O}(\log |E|)$ time. The geometric poly-line distance d allows calculation in linear time and each γ-shortest path has no more than $|E|$ edges. Based on an all-pair shortest path lookup table for G, we can retrieve each γ-shortest path and evaluate

```
Input: $t_0 \dots t_n$ with $t_i \in \mathbb{R}^2$ , $G = (V, E)$, $\gamma : E \to \mathbb{N}$, $\eta : V \to \mathbb{R}^2$ , $d( \ , \ )$
       compatible with $\| \ \|_2$
Vars: dom$[0 : n]$, pre$[1 : n]$
$\varepsilon_l[1 : n] = 0$; $\varepsilon_p[1 : n] = \infty$; dom$[0] = C(0)$
/* search while updating bounds */
for $i = 1 \ \dots \ n$ do
    repeat
        $c = C(i)$.pop()
        $\varepsilon_l[i] = \big\|t_i, c\big\|_2$                /* raise lower bound */
        for $j = i - 1 \ \dots \ 0$; $c' \in$ dom$[j]$ do
            $\delta = d\big(\mathrm{SP}_\gamma(c', c) \ , \ t_j \dots t_i\big)$
            if $\delta < \varepsilon_p[i]$ then
                $\varepsilon_p[i] = \delta$              /* lower upper bound */
                dom$[i] = \{c\}$; pre$[i] = j$
            end
        end
    until $\varepsilon_l[i] \geq \varepsilon_p[i]$;
end
Output: Backtrack dom$[n]$ using pre$[n]$
```

Algorithm 6: Basic Search Algorithm for Dominance Decompositions (c.f. Section 2.5).

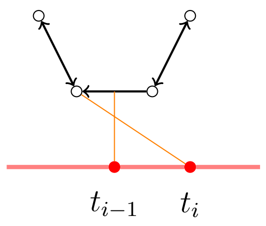


Figure 2.13: Greedy upper bounds for the initialization of $\varepsilon_p[i]$. The dominating candidate of t_{i-1} and a candidate of t_i are indicated with orange lines.

d in $\mathcal{O}(|E| + n)$. This leads to an asymptotic running time of

$$\mathcal{O}\Big(nK\big(\log|E| + (n+K)(|E|+n)\big)\Big).$$

However, the closer the input poly-line t is to a ground truth movement g along edges ($K = 1$ and $d(t, g) = 0$), the closer the running time gets to merely calculating d for consecutive parts of t.

In the following we describe speed-up techniques that allow for different edge weights γ and geometric similarity measures d at query time. Naive approaches introduce a numeric parameter for a Douglas-Peucker poly-line simplification for the input trace, leading to a reduced n. Alternatively, numeric parameters limiting the length of a part in the decomposition or a hard limit for the maximum allowed candidate distance ε_l in the search could be introduced. The latter even compromises the optimality of parts in the resulting decomposition. In the following we describe speed-up techniques that preserve optimality of the results. The techniques base on using the currently best upper bound $\varepsilon_p[i]$ of sample t_i to prune calculations of non-optimal alternatives.

Greedy Upper Bound Each sample has a candidate position on each edge of the graph. Therefore, one can use the dominating candidate (e, p) of t_{i-1} to obtain an initial upper bound on $\varepsilon_p[i]$, without shortest path search. There is either a γ-shortest path from (e, p) to a candidate position of t_i on e or one of the adjacent edges of e. See Figure 2.13 for an illustration.

Shortest Path Pruning via Prefix Moats We traverse the graph edges in inverse direction to calculate all γ-shortest paths in the graph that target the current candidate c originating from any of the dominating predecessors $\mathrm{dom}[j]$ with $\|\,\mathrm{dom}[j], t_j\|_2 < \varepsilon_p[i]$. Furthermore, prior to the dominance search for sample i we tag each graph edge with the minimum $\|\ \|_2$ distance between one of the incident nodes and points on the poly-line $t_0 \dots t_i$. When searching for γ-shortest paths for candidates of t_i we can stop the search on edges with higher tag-values than the currently best upper bound $\varepsilon_p[i]$. Any matching of these paths with a suffix of $t_0 \ldots t_i$ would result in higher $d(\ , \ )$ values of this part. Figures 2.15 and 2.1 illustrate graph edges (thin lines) having lower or higher distance tags with darker and lighter shades of gray, respectively.

Early Exit During Metric Calculation Relative arc-length re-parameterizations of poly-lines can be calculated in linear time - this extends to evaluating $d(\ , \ )$. When calculating δ for a new candidate path it is safe to abort calculations if δ already exceeded the upper bound $\varepsilon_p[i]$.

Douglas-Peucker Simplifications This technique uses simplifications of the input trace to reduce computation time. Douglas-Peuker simplifications recursively divide an input poly-line in parts at a point, where the Hausdorff distance d_H is maximal to the straight line-segment spanning the entire part. The subset of split points with distance at least δ is called a δ simplification of the poly-line.

Since the triangle inequality holds for many geometric similarity measures of poly-lines (c.f. Section 2.1), such simplifications are usable for global geometric map matching: If t' is a simplification of t with small $d_H(t', t)$ and a poly-line p of a path in the underlying graph minimizes $d_H(p, t')$, then we also have

$$d_H(p,t) \leq d_H(p,t') + d_H(t',t) \ .$$

This translates to DD map matching based on the Hausdorff distance as geometric similarity measure, since d_H provides the following concatenation property of poly-lines:

$$d_H(t_1t_2t_3, p_1p_2p_3) \leq \max\left\{d_H(t_1t_2, p_1p_2), d_H(t_2t_3, p_2p_3)\right\}$$

The same concatenation property holds for d_F and d_{WF} (c.f. Section 2.1). Therefore, each part of a dominance decomposition of a simplification translates to a part on the original trace.

More formally, let $t_0 \ldots t_n$ be a poly-line and the indexes $\{s_0, \ldots, s_m\} \subseteq \{0, \ldots, n\}$ a δ simplification ($s_0 = 0$, $s_m = n$) for some $\delta \geq 0$. Then for each poly-line p and each part $t_{s_i}t_{s_{(i+1)}} \ldots t_{s_j}$ with $i \leq j$ of a dominance decomposition, we have

$$d_H\big(p, t_{s_i}t_{s_i+1} \ldots t_{s_j}\big) < \delta + d_H\big(p, t_{s_i}t_{s_{(i+1)}} \ldots t_{s_j}\big).$$

However, d_A does not provide aboves concatenation property in general. We have $d_A\Big(\ (0,0)(1,0)(0,0)(1,0) \ , \ (0,0)(1,0) \ \Big) = 2/3$ and concatenating these poly-lines with the line-segment $(1,0)(4,0)$ leads to an value of at least $4/3$ for d_A.

Nevertheless, using DD map matching based on d_A on a Douglas-Peuker simplification t' of t provides bounds on d_H of the induced decomposition parts in t, since $d_H(p,t') \leq d_A(p,t')$ (c.f. Section 2.1).

2.6 Empirical Evaluation

We implemented the algorithms and speed-up techniques from Section 2.5 in an interactive Java application with GUI. There exist many data structures that allow fast enumeration of the nearest candidates of a geo-location $t_i \in \mathbb{R}^2$. We do not take advantage of these advanced methods since the focus of this work is quality rather than computational time. The implementation uses a simple grid with fixed distances. On average 60.3% of the total running time for one trace is required for the candidate lookups in the grid. The results and running times of our single-threaded Java application were derived on rather old desktop hardware (Intel i5-2500K) running OpenJDK version `1.7.0_111` on a 64Bit Ubuntu 14.04.4 LTS system on kernel `3.13.0-91-generic`. The implementation does not generalize the DD approach using the directed acyclic graph (DAG) of strongly connected components of a graph. We simply use the biggest strongly connected component in the graph for the algorithm. For the edge metric $\gamma : E \to \mathbb{N}$ we used the geodesic distance of the respective nodes rounded to a precision of 100 millimeters. We used the geodesic variant of $d(\ , \ )$ as geometric similarity measure (c.f. Section 2.3.1). Note that the DD approach allows other poly-line distances (e.g. Fréchet) and edge metrics γ for localization.

2.6.1 Real World Graph Data

We use graph data from the OpenStreetMap project [OSM13] as of September 1st, 2016. This is an open data crowd-sourcing effort to create a detail rich map of the world by manually integrating geo-location data provided by users. The quality and details of the map vary strongly in different regions. E.g. the Bavaria download has about twice the size than the whole of China [Geo16]. Evidently, the map varies in how rich urban, sub-urban and nature areas are captured in their details.

Region	# nodes	# edges
Washington State, US	4,490,823	9,189,076
Stuttgart, Germany, EU	2,785,490	5,953,999
London, England, UK	816,557	1,709,438
Saarland, Germany, EU	625,815	1,301,997

We derived these graphs and their spatial embedding by extracting all OSM 'ways' allowing any kind of movement (e.g. foot, bicycle, car ...).

2.6.2 Robustness Experiments

Available datasets with ground-truth typically have homogeneous quality and are unproblematic for many methods. We obtained the baseline results for our exper-

ID	#samples	length	length	#parts	max	avg
1	2356	35.3	35.3	280	45.0	7,0
2	1070	22.6	22.6	90	19.6	5,1
3	1566	22.5	22.5	182	19.9	4,7
4	1177	28.3	28.3	178	39.9	5,6
5	884	9.5	9.4	106	28.1	3,7
6	1017	16.7	16.8	127	17.3	4,5
7	2368	26.3	26.3	212	22.2	4,9
8	1135	21.4	21.3	193	28.6	3,1
9	1543	34.4	34.4	282	19.9	5,1
10	1320	16.7	16.7	160	25.9	7,0

Table 2.1: Baseline DD matchings for the GISCup'12 training dataset. Columns denote length [km] of the input and matched poly-lines, the number of parts in the DD. Judging the derived bijective mapping of the DD between the points on input and output poly-line, the columns max and avg denote the maximum and the discrete average distance [m] of matched points.

iments from a clean dataset with manual, visual inspection. The training dataset of the GISCup'12 [AKRT12] contains 10 spatial poly-lines of $9-35$km car rides with an average sampling density of $10-24$m and high precision in the area of Redmond (Washington, US). Each of the 10 visually obtained baseline match has an average value of the bijective mapping to its input poly-line below $7m$ (c.f. Table 2.1).

We used Douglas-Peucker poly-line simplifications of the input traces to derive traces with varying *sampling densities*. We used maximum allowed Hausdorff distances of $1, 2, 5, 10, 20, 40, 80$ and 100m. The 10 curves in the lower part of Figure 2.14 relate to the right y-axis and show the percentage of surviving samples for each of the 10 traces. The respective curves in the upper part relate to the left y-axis and show the percentage of graph edges in the base line solution that are met by the edges in the DD solution of the simplification. We consider the baseline and the alternate solution paths as discrete strings of edge indexes. The true positive rate is the edge count of longest-common-subsequence and baseline path. Note that this is a rather pessimistic robustness measure – E.g. different lanes in the same direction of a highways are often modeled with different edges in the graph data.

In order to evaluate the stability of DD matching results under *heterogeneous precision* and under *noise*, we introduced a random error for the points along the poly-line. We sample uniformly isotropic error vectors from a 130×130m square.

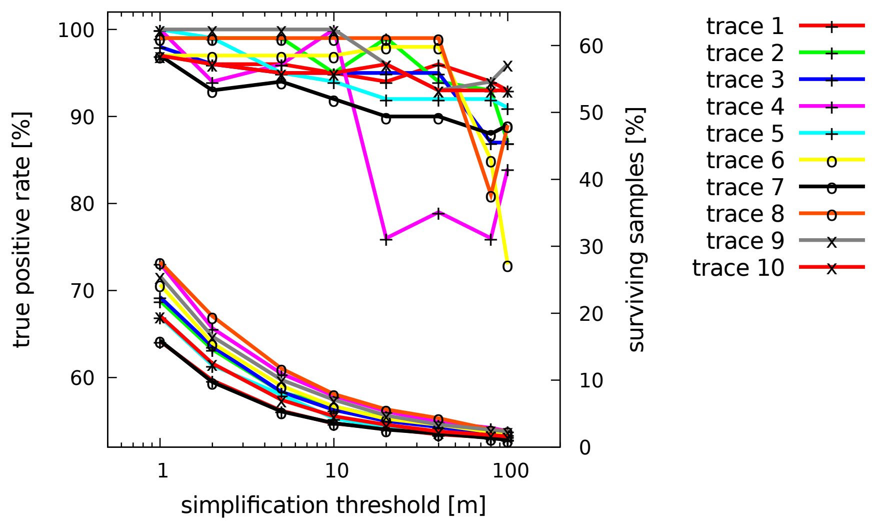


Figure 2.14: Robustness of DD under sampling densities. The lower ten curves relate to the right y-axis whereas the upper ten curves relate to the left y-axis.

In order to simulate heterogeneous data quality along each of the traces, we introduced such error vectors to samples with a certain probability. E.g. a probability of 0.1 means that about 10% of the samples along the poly-line of a trace received this synthetic error offset. See Figure 2.1 for an example with a $35 \times 35m$ square and probability 1.0. Figure 2.15 provides an additional example. Note that adding errors artificially renders a comparison to ground-truth eventually meaningless. That is as soon as the heterogeneous errors are big enough to make an alternative path more 'reasonable'. Figure 2.16 shows the results of these experiments for the 10 traces. The x-axis shows the applied error probabilities. Again, the lower 10 curves relate to the right y-axis and show the geometric poly-line distance $d(\ , \ )$[km] as a measure of (dis)similarity to the baseline. The respective curves in the upper parts relate to the left y-axis. They show the true positive rates achieved of the DD for the disturbed poly-lines compared to baseline paths.

2.6.3 Real World Trace Data

In this section we demonstrate the effectiveness of our methods and speed-up techniques on heterogeneous, real world data. The OpenStreetMap project [OSM13] provides a dataset of all user uploaded data until 9th of April 2013 (257.2 GiB `.gpx` files). We extracted all traces that are fully contained in the boundary polygon of the regions Saarland, London and Stuttgart. We process this highly heterogeneous data sets *without* advanced cleaning methods. We simply required traces to

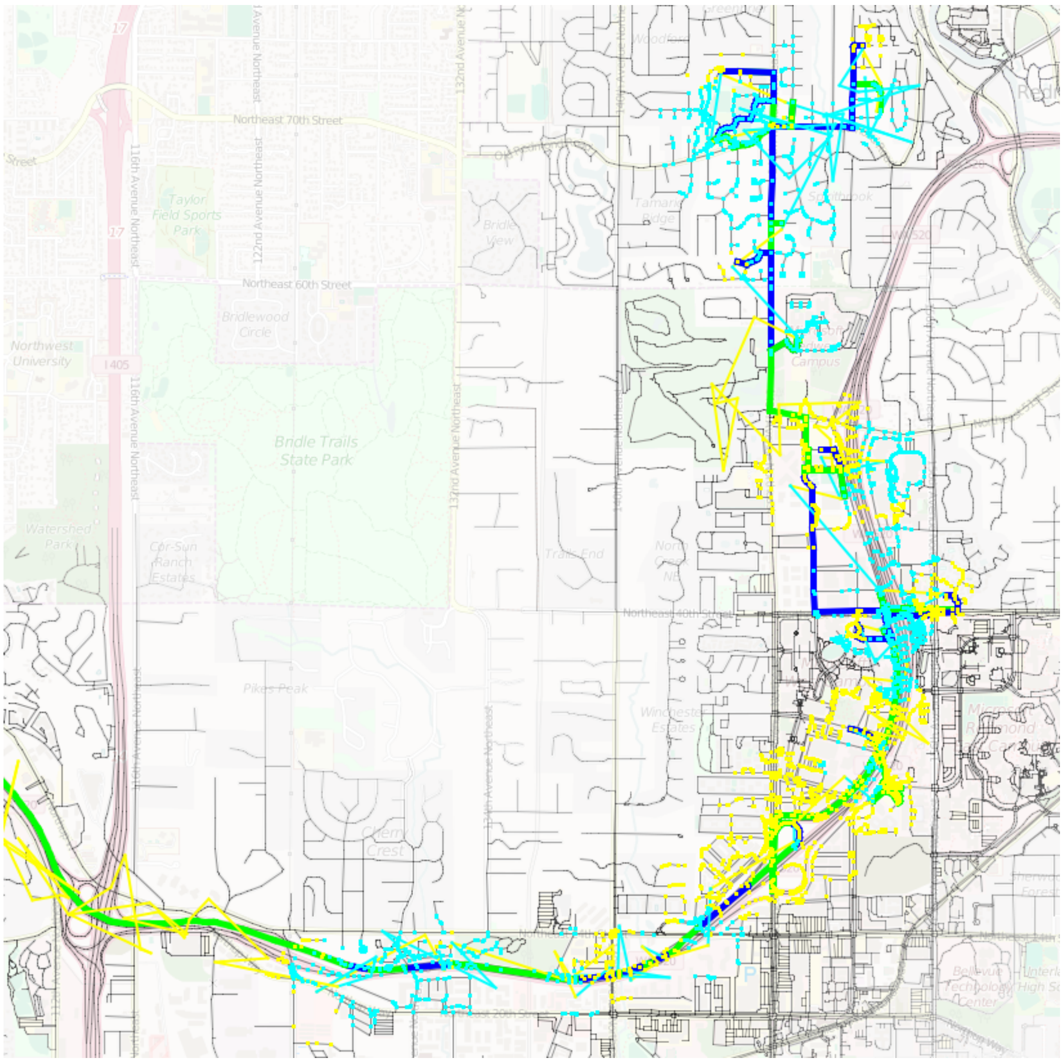


Figure 2.15: Trace 2 of the GISCup'12 training dataset under poly-line simplification and heterogeneous, noisy errors. The disturbed trace is given as fine line in colors alternating between turquoise and yellow. The resulting parts of DD map matching are depicted as thick lines in colors alternating between blue and green. Filled dots indicate the respective candidate search spaces and the opacity of the underlying graph edges (thin black lines) illustrates the prefix moats speed-up technique.

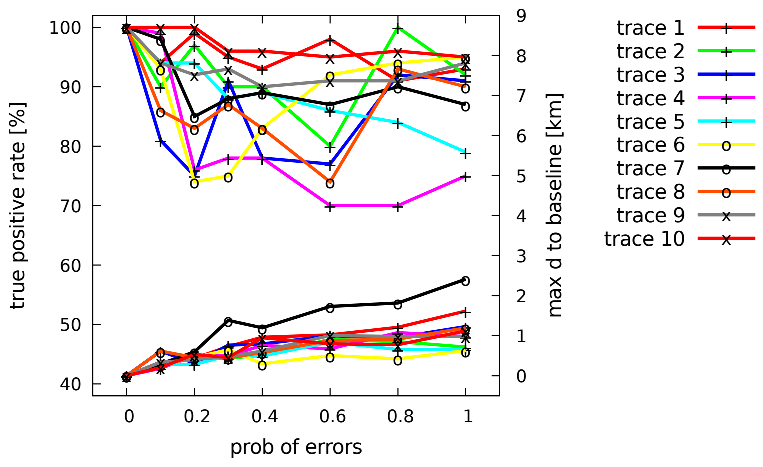


Figure 2.16: Robustness of DD under heterogeneous precision and noise. The lower ten curves relate to the right y-axis whereas the upper ten curves relate to the left y-axis.

Region no. traces	Stuttgart 17, 710	London 4, 375	Saarland 2, 572
no. points [100]	7.5 (13.7)	9.0 (23.4)	9.0 (15.1)
length [km]	9.7 (15.9)	5.1 (7.2)	11.4 (15.1)
density [10/km]	15.3 (24.5)	21.5 (53.6)	14.8 (46.7)
max. spread [km]	4.1 (6.9)	2.2 (3.1)	4.4 (6.2)
max. hop [10m]	11.7 (20.0)	10.5 (18.0)	11.5 (18.4)

Table 2.2: Mean and standard deviation statistics for the 24, 657 spatial poly-lines in the datasets.

have at least $50m$ in length, dropped sampling duplicates and single samples showing a zig-zag outlier pattern with their neighbouring samples. Table 2.2 provides statistics on some data characteristics.

We use all three speed-up techniques and simplify each input trace to a maximum Hausdorff distance of 20m with the Douglas-Peucker Algorithm. See Figure 2.17 (top left) for computation times. We further limit the search of the dominating candidates to a maximum distance of 400m for each sample. This is indicated by the vertical line in Figure 2.17 (top right), showing that only very few of the 24, 657 results are not guaranteed to be optimal. Note than an average value of 10m in the bijective mappings of the geometric similarity measure d (c.f. Section 2.3.1) is a *very* convenient result for such a heterogeneous dataset.

Adaptive Lossy Compression

The simple bijective mapping provided from a DD allows an adjustable lossy compression scheme: The parts of the trace not exceeding some desired quality value are stored as shortest paths rather than geo-locations. See Figure 2.17 (bottom) for the compression rates and average errors obtained by DD map matching.

2.7 Research Directions

In future work, we seek to provide more efficient speed-up techniques. The property of simplifications or fixing a geometric edge weight for the graph seems promising. On the experimental side, we are interested in evaluating the quality of dominance decompositions under different geometric poly-line distances, like Hausdorff or strong, weak and average Fréchet.

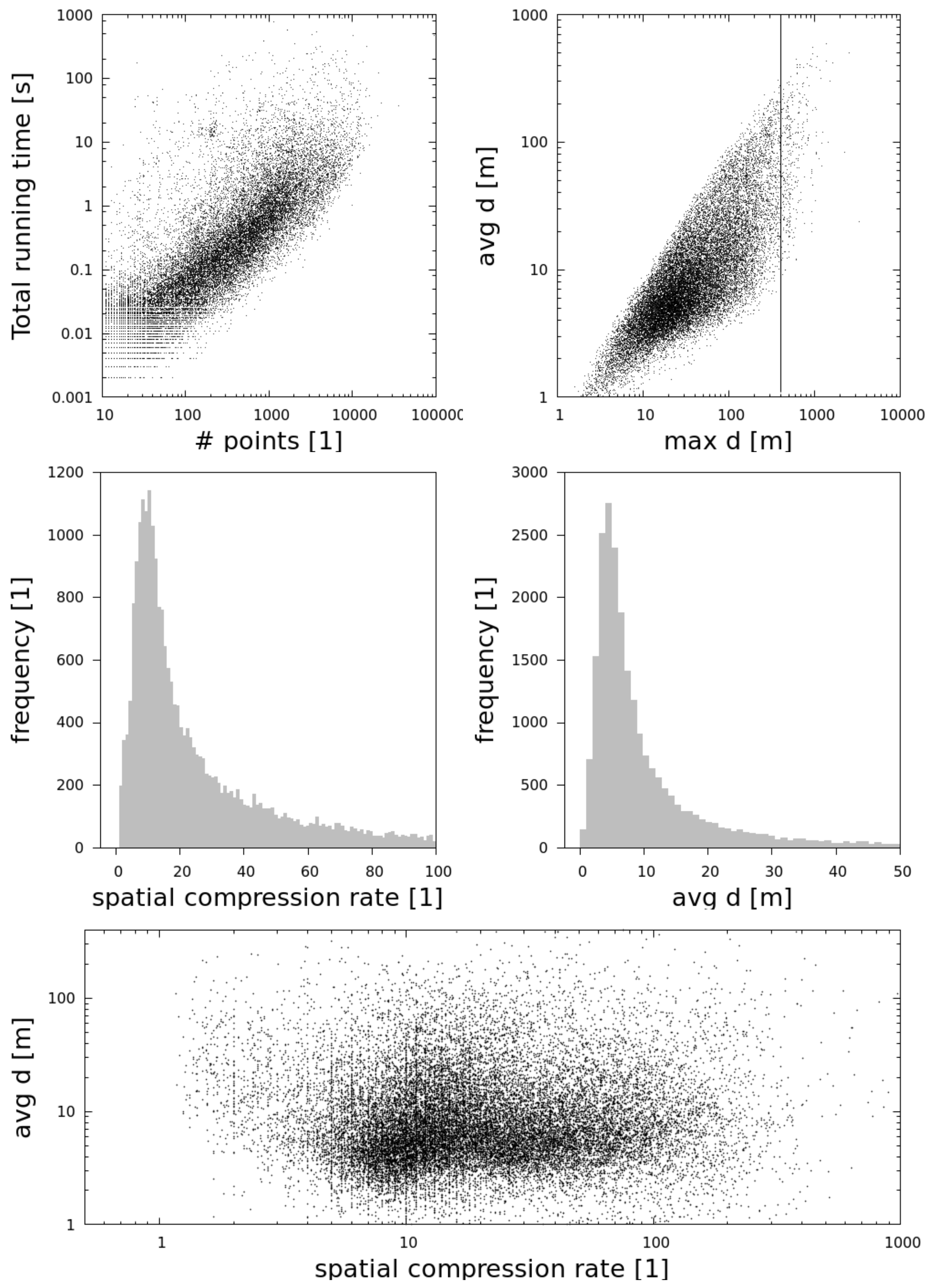


Figure 2.17: Running times (top left), result quality (top right and mid right) and spatial compression rates (mid left and bottom) of Algorithm 6 using the speed-up techniques. Each DD matching result for the 24, 657 traces is depicted as a dot in the scatter plots.

Chapter 3

Rational Points on Unit Spheres

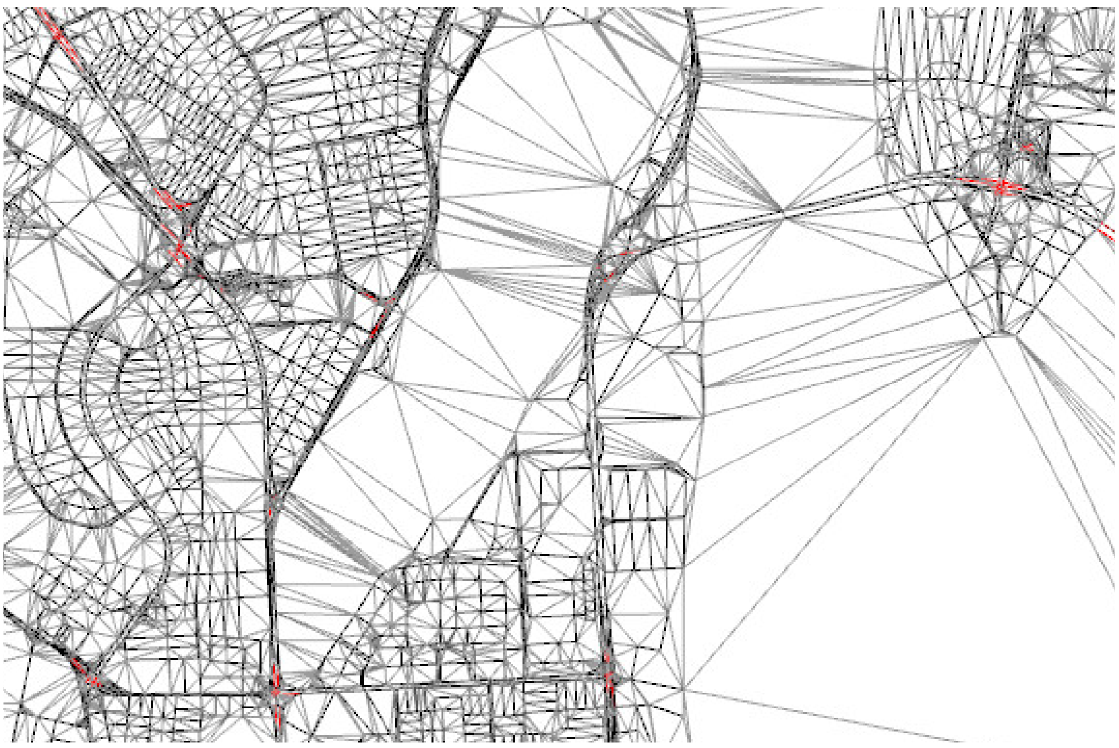


Figure 3.1: Spherical Delaunay triangulation (gray) constrained to contain all line segments (black) of streets in Ecuador. Intersections of constraints are given in red.

3.1 Introduction

Many mathematical sciences use trigonometric functions in symbolic coordinate transformations to simplify fundamental equations of physics or mathematical systems. However, rational numbers are dominating in computer processing as they allow for simple storage as well as fast exact and inexact arithmetics (e.g. GMP [Gt12], IEEE Float, MPFR [FHL+07]). Therefore problems on spherical surfaces often require to scale a point vector, as in choosing a point uniform at random [Mar72], or to evaluate a trigonometric function for a rational angle argument, as in dealing with geo-referenced data (e.g. rational longitude and latitude values).

A classical theoretical barrier is Niven's theorem [Niv85], which states that the sole rational values of sine for rational multiples of π are $0, \pm 1/2$ and ± 1. The well known Chebyshev polynomials have roots at these values, hence give rise to representations for these algebraic numbers. However, arithmetics in a full algebraic number field might well be too demanding for many applications. For products of sine and cosine, working with Euler's formula on the complex unit circle and Chebyshev polynomials would suffice though.

This manifests in problems of *exact* geometrical computations, since standard methodology relies on Cartesian input [LPY05]. Spheres and ellipsoids are common geometric objects and rational solutions to their defining quadratic polynomials are closely related to Diophantine equations of degree 2. The famous Pythagorean Triples are known to identify the rational points on the circle $\mathbb{S}^1$. Moreover, the unit sphere has a dense set of rational points and so do ellipsoids with rational half-axes through scaling. Spherical coordinates are convenient to reference such Cartesians with angle coordinates and *geo-referenced data* denotes points with rational angles. Standard approximations of Cartesians do not necessarily fulfill these equations, therefore subsequent algorithmic results can suffer greatly.

This chapter focuses on finding rational points *exactly on* the unit sphere $\mathbb{S}^{d-1} = \left\{x \in \mathbb{R}^d \,:\, \sum_i x_i^2 = 1\right\}$ with bounded distance to the point $x/\|x\|_2$ – its closest point on $\mathbb{S}^{d-1}$. In this work, $x \in \mathbb{R}^d$ can be given by any finite means that allow to compute a rational approximation to it with arbitrary target precision. Using rational Cartesian approximations for spherical coordinates, as derived from MPFR, is just one example of such a black-box model. Moreover, we are interested in calculating rational points on $\mathbb{S}^d$ with small denominators.

Related Work

Studies on spherical Delaunay triangulations (SDT), using great-circle segments on the sphere $\mathbb{S}^2$, provide common ways to avoid and deal with the point-on-sphere problem in computational geometry. We list the approaches by the categories of Robust Computational Geometry [She97].

The *fragile* approaches [PZ15, JGR+13, Ren97] ignore that the input may not be on $\mathbb{S}^2$ and succeed if the results of all predicate evaluations happen to be correct. Input point arrangements with close proximity or unfortunate locations bring these algorithms to crash, loop or produce erroneous output. The *quasi-*

robust approaches [Bro79, BDH96] weaken the objective and calculate a Delaunay tessellation in d-Simplexes. Lifting to a $d+1$ convex hull problem is achieved by augmenting a rational coordinate from a quadratic form – The augmented point exactly meets the (elliptic) paraboloid equation. However, the output only identifies a SDT if all input points are already on the sphere, otherwise the objectives are distinct. Equally unclear is how to address spherical predicates and spherical constructions. The *robust* approaches [Saa99] use the circle preserving stereographic projection from $\mathbb{S}^2$ to the plane. The perturbation to input, for which the output is correct, can be very large as the projection does not preserve distances. Furthermore, achieving additional predicates and constructions remains unclear. The *stable* approaches provide geometric predicates and constructions for points on $\mathbb{S}^2$ by explicitly storing an algebraic number, originating from scaling an ordinary rational approximation to unit length [dCCLT09]. Algebraic number arithmetics can be avoided for $\mathbb{S}^2$, but exact evaluation relies on specifically tailored predicates [CdCL$^+$10], leaving the implementation of new constructions and predicates open.

Kleinbock and Merrill provide methods to quantify the density of rational points on $\mathbb{S}^d$ [KM15], that extend to other manifolds as well. Recently, Schmutz [Sch08] provided a divide-&-conquer approach on the sphere equation, using Diophantine approximation by continued fractions, to derive points in $\mathbb{Q}^d \cap \mathbb{S}^{d-1}$ for a point on the unit sphere $\mathbb{S}^{d-1}$. The main theorem bounds the denominators in ε-approximations, under the $\| \ \|_\infty$ norm, with $(\sqrt{32}\lceil \log_2 d \rceil / \varepsilon)^{2\lceil \log_2 d \rceil}$. Based on this, rational approximations in the orthogonal group $O(n, \mathbb{R})$ and in the unitary matrix group $U(n, \mathbb{C})$ are found. This is of particular interest for sweep-line algorithms: [CDR92] studies finding a rotation matrix with small rationals for a given rational rotation angle of an 2D arrangement.

Contribution

The strong lower bound on rational approximations to other rational values does not hold for geo-referenced data, considering Niven's theorem. We derive explicit constants to Liouville's lower bound, for a concrete geo-referenced point, that is within a factor 2 of the strong lower bound. Moreover, we prove that floating-point numbers *cannot* represent Cartesian coordinates of points that are exactly on $\mathbb{S}^1$ or $\mathbb{S}^2$.

We describe how the use of rotation symmetry and approximations with fixed-point numbers suffice to improve on the main theorem of [Sch08]. We derive rational points *exactly on* $\mathbb{S}^{d-1}$ with denominators of at most $10(d-1)/\varepsilon^2$ for any $\varepsilon \in \left(0, \frac{1}{8}\right]$. Moreover, our method allows for even smaller denominators based on algorithms for simultaneous Diophantine approximations, though a potentially weaker form of approximation would suffice.

The controlled perturbations provided by our method allow exact geometric algorithms on $\mathbb{S}^d$ to rely on rational rather than algebraic numbers – e.g. enabling convex hull algorithms to efficiently obtain spherical Delaunay triangulations on $\mathbb{S}^d$ and not just Delaunay tessellations. Moreover, the approach allows for inexact but ε-stable geometric constructions – e.g. intersections of Great Circle segments.

We demonstrate the quality and effectiveness of the method on several, including one whole-world sized, point sets. We provide open-source implementations for the method and its application in the case of spherical Delaunay triangulations with intersections of constraints.

Chapter Outline and Pre-Releases

Section 3.2 reviews results on Diophantine approximation and establishes an explicit lower bound on a point with rational angle coordinates. Section 3.3.1 clarifies that floating-point numbers cannot represent Cartesian coordinates of points on unit spheres, for certain dimensions. Section 3.3.2 describes the approximation method and Sections 3.3.3 and 3.3.4 provide bounds on quality and denominator size. Section 3.4 discusses implementation issues and Section 3.5 the experimental results. Section 3.6 closes the chapter with directions for future work.

The method of Section 3.3.2 found application in our contribution to the proceedings of the 19th Workshop on Algorithm Engineering and Experiments [BBF+17]. Apart from the tighter analysis in Section 3.3.3, the results of this chapter are contained in our contribution to the proceedings of the International Symposium on Symbolic and Algebraic Computation [BS17a].

3.2 Definitions and Tools

The 2nd *Chebyshev polynomials* U_n of degree n are in $\mathbb{Z}[X]$, given their recursive definition:

$$U_0(x) = 1 \qquad U_1(x) = 2x$$
$$U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x) \ .$$

It is well known [Riv74], that the n roots of U_n are exactly the values

$$\{ \ \cos\left(\pi k / \left(n+1\right)\right) \ : \ k = 1, \ldots, n \ \} \ .$$

Hence the polynomials U_n give rise to algebraic representations for cosine values of rational multiples of π. This is particularly useful in conjunction with classic results on *Diophantine approximations*, that are known since 1844 [Lio51]:

Theorem 3.2.1 (Liouville's Lower Bound)**.** *For any algebraic $\alpha \in \mathbb{R}$ of degree $n \geq 2$, there is a positive constant $c(\alpha) > 0$ such that*

$$\left|\alpha - \frac{p}{q}\right| \geq \frac{c(\alpha)}{q^n}$$

for any $p \in \mathbb{Z}$ and $q \in \mathbb{N}$.

This nice proof was translated from the German wikibooks Project – thanks to the anonymous authors. See [Lio51] for the original proof in French language.

Proof. Let $\alpha \in \mathbb{R}$ be algebraic of degree n and root of the corresponding polynomial $f(X) \in \mathbb{Z}[X]$ of degree n, meaning

$$f(\alpha) = a_0 + a_1\alpha + \cdots + a_n\alpha^n = 0$$

with $a_0, \ldots, a_n \in \mathbb{Z}$ and $a_n \neq 0$. Polynomial division with the linear factor $X - \alpha$ in the ring $\mathbb{C}[X]$ provides

$$f(X) = (X - \alpha) \cdot g(X). \tag{3.1}$$

Note that the polynomial $g(X)$ has algebraic coefficients and is not necessarily in $\mathbb{Z}[X]$. However, the mapping $\mathbb{R} \to \mathbb{C}$, $t \mapsto g(t)$ is continuous, by means of real numbers $c_1 > 0$, $c_2 > 0$ with

$$|g(x)| \leq c_1 \tag{3.2}$$

for $|\alpha - x| < c_2$. Since $n < \infty$, we can assume w.l.o.g. that no additional roots are in this neighborhood of α, meaning

$$f(x) \neq 0 \tag{3.3}$$

for $|\alpha - x| < c_2$ and $x \neq \alpha$.

Claim: The statement of the Theorem holds for $c := \min\left\{c_2, \frac{1}{c_1}\right\}$.

Suppose there are $p, q \in \mathbb{Z}$, $q > 0$ with

$$\left|\alpha - \frac{p}{q}\right| < \frac{c}{q^n} \quad . \tag{3.4}$$

We show that his implies $\alpha = \frac{p}{q}$. From (3.4), we immediately derive

$$\left|\alpha - \frac{p}{q}\right| < c \leq c_2 \ , \tag{3.5}$$

leading (3.2) to imply $\left|g(\frac{p}{q})\right| \leq c_1$. We derive, from (3.1) and again (3.4), that

$$\left|f\left(\frac{p}{q}\right)\right| = \left|\frac{p}{q} - \alpha\right| \cdot \left|g\left(\frac{p}{q}\right)\right| < \frac{c}{q^n} \cdot c_1 \leq \frac{1}{q^n},$$

meaning $\left|q^n \cdot f\left(\frac{p}{q}\right)\right| < 1$.

However $q^n \cdot f\left(\frac{p}{q}\right) = a_0 q^n + a_1 p q^{n-1} + \cdots + a_n p^n \in \mathbb{Z}$ and its absolute value is smaller than 1, hence has to be 0. Moreover, $f(\frac{p}{q}) = 0$ and (3.5) with (3.3) imply $\alpha = \frac{p}{q}$, which closes the argument. □

Apart from this lower bound on rational approximations, there is another important folklore result on the existence of simultaneous Diophantine approximations. Such approximations have surprisingly small errors, despite their rather small common denominator.

Theorem 3.2.2 (Dirichlet's Upper Bound). *Let $N \in \mathbb{N}$ and $\alpha \in \mathbb{R}^d$ with $0 \leq \alpha_i \leq 1$. There are integers $p \in \mathbb{Z}^d$, $q \in \mathbb{Z}$ with $1 \leq q \leq N$ and*

$$\left|\alpha_i - \tfrac{p_i}{q}\right| \leq \frac{1}{q\sqrt[d]{N}}\ .$$

The folklore proof bases on Dirichlet's famous Pigeonhole Principle.
See proofwiki.org or Chapter 11.12 in [HW54].

Proof. We consider the partition of $[0,1]^d$ in N^d regular d-cubes of length $L = \sqrt[d]{N}$. We further define a sequence of points $(a^{(j)})_{j=1,\ldots,N^d+1} \in [0,1]^d$ with $a^{(j)} := j \cdot \alpha - \lfloor j \cdot \alpha \rfloor$ (component wise operations). There are indices $k > l$ such that the points $a^{(k)}$ and $a^{(l)}$ are contained in the same d-cube. We have the (component wise) inequalities

$$-\frac{1}{L} < a^{(k)} - a^{(l)} < \frac{1}{L}$$
$$-\frac{1}{L} < k\alpha - \lfloor k\alpha \rfloor - l\alpha + \lfloor l\alpha \rfloor < \frac{1}{L}$$
$$-\frac{1}{L} < (k-l)\alpha - (\lfloor k\alpha \rfloor - \lfloor l\alpha \rfloor) < \frac{1}{L}$$

Setting $q = k - l$ and $p_i = \lfloor k\alpha_i \rfloor - \lfloor l\alpha_i \rfloor$ provides integers as required. □

For $d = 1$, the continued fraction (equivalently the Euclidean) algorithm is famous [HW54] for finding approximations with $|\alpha - p/q| \leq 1/2q^2$. This spurred the field of number theory to study generalizations of the continued fraction algorithm that come close to Dirichlet's upper bound, but avoid brute-force calculations. Some more recent methods are discussed in Section 3.3.4.

Our approach uses the *Stereographic Projection* in $\mathbb{R}^d$. Let $p = (0, \ldots, 0, 1) \in \mathbb{R}^d$ be the fixed point for the projection τ, mapping all points of a ray from p to the intersection with the hyperplane $x_d = 0$.

$$\begin{aligned}\tau : \mathbb{R}^d \setminus (\mathbb{R}^{d-1} \times \{1\}) &\to \mathbb{R}^{d-1}\\ x &\mapsto \left(\frac{x_1}{1-x_d}\ , \ \ldots\ , \ \frac{x_{d-1}}{1-x_d}\right)\end{aligned}$$

The surjective mapping τ is injective as well, when restricted to the domain $\mathbb{S}^{d-1} \setminus \{p\}$. We further define the mapping σ, which is

$$\begin{aligned}\sigma : \mathbb{R}^{d-1} &\to \mathbb{R}^d \setminus \{p\}\\ x &\mapsto \left(\ \frac{2x_1}{1+S^2}\ , \ \ldots\ , \ \frac{2x_{d-1}}{1+S^2}\ , \ \frac{-1+S^2}{1+S^2}\ \right)\end{aligned}$$

where $S^2 = \sum_{j=1}^{d-1} x_j^2$. We have $\operatorname{img}\sigma \subseteq \mathbb{S}^{d-1}$, since

$$\|\sigma(x)\|_2^2 = \frac{(-1+S^2)^2 + \sum_{i=1}^{d-1}(2x_i)^2}{(1+S^2)^2} = 1 \quad .$$

Furthermore, $x = \tau \circ \sigma(x)$ for all $x \in \mathbb{R}^{d-1}$, since

$$(\tau \circ \sigma)_i\,(x) = \frac{\frac{2x_i}{1+S^2}}{1 - \frac{-1+S^2}{1+S^2}} = \frac{2x_i}{1+S^2+1-S^2} = x_i$$

holds for all $1 \le i < d$. Hence, σ and τ are inverse mappings. Note that images of rational points remain rational in both mappings, establishing a bijection between rational points in $\mathbb{R}^{d-1}$ and $\mathbb{S}^{d-1}$.

3.2.1 Lower Bounds and Instances for Geo-referenced Data on $\mathbb{S}^d$

It is well known in Diophantine approximation that rational numbers have algebraic degree 1 and are hard (in the following qualitative sense) to approximate with other rational numbers. The following folklore observation is an analog to Liouville's lower bound.

Observation 1. For rational numbers $\frac{a}{b} \neq \frac{p}{q}$, we have

$$\left|\frac{a}{b} - \frac{p}{q}\right| = \left|\frac{aq-bp}{bq}\right| \ge \frac{1}{bq}$$

If $q < b$, we have a lower bound of $1/q^2$ for rational approximations to $\frac{a}{b}$ with denominators up to q. Pythagorean triples $(x, y, z) \in \mathbb{N}^3$ provide such rational points on $\mathbb{S}^1$, since $(x/z)^2 + (y/z)^2 = 1$. We have a lower bound of $1/z^2$ for approximations with denominators $q < z$. See Section 3.3.4 for rational points on $\mathbb{S}^d$ with the same denominator property.

The situation might look different when dealing with geo-referenced data (rational angle arguments) only. However, using Chebyshev's polynomials in conjunction with Liouville's lower bound (c.f. Theorem 3.2.1) allows to derive explicit constants for Diophantine approximations of $\cos(108°)$.

Given spherical coordinates, the first coordinate of a point on $\mathbb{S}^d$ might well have algebraic values of $r_i = \cos(\frac{i}{5}\pi)$ for $i \in \{1, 2, 3, 4\}$.

$$
\begin{aligned}
(r_1, r_2, r_3, r_4) &= \left(\frac{1+\sqrt{5}}{4}, \frac{-1+\sqrt{5}}{4}, \frac{1-\sqrt{5}}{4}, \frac{-1-\sqrt{5}}{4}\right) \\
&\approx (+0.8090, +0.3090, -0.3090, -0.8090)
\end{aligned}
$$

Over $\mathbb{Z}[X]$, the polynomial $U_4(x) = 16x^4 - 12x^2 + 1$ has the irreducible factors

$$
U_4(x) = \underbrace{(4x^2 - 2x + 1)}_{=:f(x)}(4x^2 + 2x - 1)
$$

Since r_1 and r_3 are the roots of the polynomial f, they have algebraic degree $n = 2$.

Using Liouville's lower bound for r_3, we have for all $\frac{p}{q} \in \mathbb{Q}$

$$
\left| r_3 - \frac{p}{q} \right| \geq \frac{\min\{c_2, \frac{1}{c_1}\}}{q^n} \quad ,
$$

with constants c_1 and c_2 according to the proof of Liouville's Theorem [Lio51]. The constants $c_1, c_2 > 0$ exist, since the polynomial division of f with the linear factor $(x - r_3)$ results in the continuous function $g(x) = (x - r_1)$. For $c_2 = 1/2 < \sqrt{5}/2$, the interval $I := [r_3 - c_2, r_3 + c_2] \subseteq \mathbb{R}$ is sufficiently small to exclude different roots of f and the inequality

$$
\max_{x \in I} |g(x)| = \max_{x \in I} |x - r_1| < c_1
$$

is met with a generous choice of $c_1 = 2$. This leads to an explicit lower bound on the approximation error to r_3 with denominators q of

$$
\left| \cos(108°) - \frac{p}{q} \right| \geq \frac{1}{2 \cdot q^2} \; .
$$

3.3 Results

Apart from integers, contemporary computing hardware heavily relies on *floating point numbers.* These are triplets (s, m, e) with $s \in \{0,1\}$, $m \in \{0,\dots,2^l - 1\}$ and $e \in \{-2^{k-1}+1,\dots,2^{k-1}-1\}$. The IEEE standard for Float is $(l,k) = (23,8)$ and $(52,11)$ for Double. The rational number described by such a triplet is

$$\operatorname{val}(s,m,e) = (-1)^s \cdot \begin{cases} \frac{2^l + m}{2^l} 2^e & e > 0 \\ \frac{2^l + m}{2^l} \frac{1}{2^{|e|}} & e < 0 \\ \frac{0 + m}{2^l} \frac{1}{2^{2^{k-1}-2}} & e = 0 \end{cases}$$

where the latter case describes 'denormalized' numbers. In each case, the uncanceled rational value has some power of 2 as the denominator. Since powers of two are the sole divisors of a 2^i, the denominator of the canceled rational has to be a power of 2, too. Hence, rational values representable by floating point numbers are a subset of the following set P and fixed-point binary numbers are a subset of P_i:

$$\begin{aligned} \operatorname{img}\operatorname{val} \subseteq \left\{\frac{z}{2^i} : i \in \mathbb{N}, z \in \mathbb{Z}, z \text{ odd}\right\} &= P \\ \left\{\frac{z}{2^i} : z \in \mathbb{Z}\right\} = P_i &\subseteq P \quad . \end{aligned}$$

3.3.1 Floating Point Numbers are Insufficient

Fix-point and floating-point arithmetics of modern CPUs work within a subset of rational numbers, in which the denominator is some power of two and the result of each arithmetic operation is 'rounded'.

Theorem 3.3.1. *There are only* 4 *floating point numbers on* $\mathbb{S}^1$ *and* 6 *on* $\mathbb{S}^2$.

Proof. We show $\mathbb{S}^{d-1} \cap P^d \not\subseteq \{-1,0,1\}^d$ implies $d \geq 4$. Suppose there is a non-trivial $p \in \mathbb{S}^{d-1} \cap P^d$ with d minimal. Let $x_i/2^{e_i}$ denote the canceled fraction of its i-th coordinate. We have that all $x_i \neq 0$, x_i are odd numbers and all $e_i > 0$ (since p is not one of the $2d$ poles and d is minimal).

W.l.o.g. $e_1 \leq e_2 \leq \ldots \leq e_d$. We rewrite the sphere equation $1 = \sum_{j=1}^d (x_i/2^{e_i})^2$ to

$$x_1^2 = 4^{e_1} - \sum_{j=2}^{d} 4^{e_1 - e_j} x_j^2 \quad .$$

For an odd integer y, we have $y^2 = (2k+1)^2 = 4(k^2+k)+1$, leading to the congruence

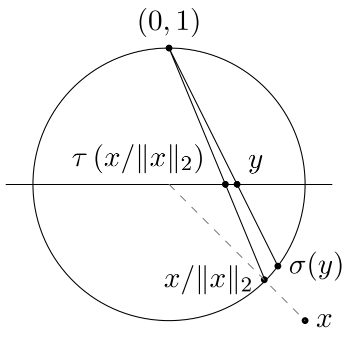
$$1 \equiv 0 - \sum_{j=2}^{d} \chi_{e_1}(e_j) \qquad \bmod 4 \ .$$

Where the characteristic function $\chi_{e_1}(e_j)$ is 1 for $e_1 = e_j$ and 0 otherwise. For $d \in \{2,3\}$ the right hand side can only have values of $0, -1$ or -2, a contradiction. □

Note that Theorem 3.3.1 translates to spheres with other radii through scaling. Suppose a sphere in $\mathbb{R}^3$ of radius 2^j has a non-trivial solution $y \in P^3$, then $y/2^j \in P^3$ and would be on $\mathbb{S}^2$, too.

3.3.2 Snapping to Rational Points

We now describe how to compute a good rational approximation *exactly on* the unit sphere $\mathbb{S}^{d-1}$. The input point $x \in \mathbb{R}^d$ can be given by any finite means that allows to compute rational approximations of arbitrary target precision – E.g. rational approximations of Cartesians for spherical coordinates. For the input x, we denote its closest point on $\mathbb{S}^{d-1}$ with $x/\|x\|_2$. The stereographic projection τ and its inverse mapping σ provide $\sigma(\tau(x/\|x\|_2)) = x/\|x\|_2$, since the argument is on $\mathbb{S}^{d-1}$. Instead of determining the value of τ exactly, we calculate an approximation $y \in \mathbb{Q}^d$ and finally evaluate $\sigma(y)$ under exact, rational arithmetics. Hence, the result $\sigma(y)$ is exactly on $\mathbb{S}^{d-1}$.



The stereographic projection does not preserve distances, leaving it open to bound the approximation error and the size of the resulting denominators. We use the *rotation symmetry* of the sphere to limit the stretching of σ (c.f. Lemma 3.3.3): For a non-zero point $x \in \mathbb{R}^d$ we can assume that $i = d$ maximizes $|x_i|$ and $x_d < 0$, otherwise we change the standard orthonormal basis by swapping dimension i and d and using a negative sign for dimension d. Note that such rotations do not change the actual coordinate values. To keep the *size of denominators* in $\sigma(y)$ small, we use fixed-point arithmetics to determine $y \in \mathbb{Q}^{d-1}$ (c.f. Lemma 3.3.5).

See Algorithm 7 for a precise description. Note that the rational point y in statement 2 solely needs to meet the target approximation in the individual coordinates for

$$\tau_i(x/\|x\|_2) = \frac{x_i}{\|x\|_2 - x_d} \quad .$$

```
In: $x \in \mathbb{R}^d, \quad \varepsilon \in \left(0, \frac{1}{8}\right]$
  1. Assert $x_d = \min_i -|x_i|$
  2. Choose $y \in \mathbb{Q}^{d-1}$ with $|y_i - \tau_i\left(x/\|x\|_2\right)| \le \frac{\varepsilon}{2\sqrt{d-1}}$
  3. Return $\sigma(y) \in \mathbb{Q}^d$.
```

Algorithm 7: PointToSphere

Generally, this can be determined with methods of 'approximate expression evaluation' to our target precision [LPY05]. If x is an approximation to a geo-referenced point, this denominator is well conditioned for calculations with multi-precision floating-point arithmetics [BFS98, FHL+07]. Using exact rational arithmetics for statement 3, we obtain rational Cartesian coordinates *on* the unit sphere.

Observation 2. For $d > 1$ and $x \in \mathbb{S}^{d-1}$ with $x_d = \min_i -|x_i|$, we have

$$\|\tau(x)\|_2 \le \sqrt{\frac{\sqrt{d}-1}{\sqrt{d}+1}} < 1 \quad .$$

Proof. Using $x_d = \min_i -|x_i|$ and $\sum_i x_i^2 = 1$, we have the bounds $1/d \le x_d^2 \le 1$ and

$$\|\tau(x)\|_2^2 = \frac{\sum_{i=1}^{d-1} x_i^2}{(1-x_d)^2} = \frac{1-x_d^2}{(1-x_d)^2} = \frac{1+x_d}{1-x_d} \le \frac{1-1/\sqrt{d}}{1+1/\sqrt{d}} \; .$$

Where the latter term is in $(0,1)$ for any d. □

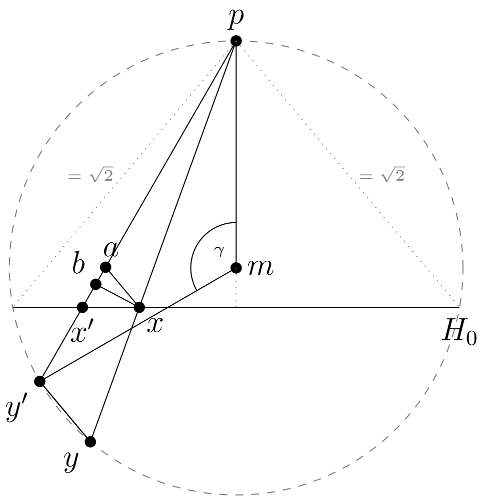
Hence the $(d-1)$-ball $\mathbf{B}_1^{d-1} = \{x \in \mathbb{R}^d \;:\; \|x\|_2 \le 1\}$ contains $\tau(x)$.

3.3.3 Approximation Quality

See [BS17a] for an earlier version of this work with a weaker, but elementary, analysis.

We consider the problem in the 2D hyperplane $H_{pyy'}$, defined by two points $y = \sigma(x)$, $y' = \sigma(x')$ on $\mathbb{S}^{d-1}$ and the projection pole $p \in \mathbb{R}^d$. Given the rotation step in Algorithm 7, the projection plane $H_0 = \{x \in \mathbb{R}^d : x_d = 0\}$ separates p and y, y' in $\mathbb{R}^d$ and in $H_{pyy'}$. Since each $q \in H_0 \cap S^{d-1}$ has $\|q - p\|_2 = \sqrt{2}$ (consider $\overline{pq}$ in H_{0pq}), the circumcircle C of p, y and y' contains exactly two of these points. Hence, the line of $H_{pyy'} \cap H_0$ is orthogonal to the circumcircle's diameter through

p. Moreover, the circles diameter is in $[\sqrt{2}, 2]$. We denote with x the point that is closer to p in $H_{pyy'}$, meaning $\|x\|_2 \leq \|x'\|_2$. Note that x' and x can be on the same or opposite circumcircle halves.



In this section we denote with $B = \overline{bx}$ the perpendicular from x on $\overline{py'}$, $E = \overline{xx'}$, $L = \overline{yy'}$ and $L_x = \overline{xa}$ its triangle scaled version meeting x. Note that B and L_x are above H_0, hence above E.

Lemma 3.3.2. *For $x, x' \in \mathbf{B}_1^{d-1}$ with $\|x\|_2 \leq \|x'\|_2$, we have*

$$\frac{\|p - x\|_2}{\|p - \sigma(x)\|_2} \|\sigma(x) - \sigma(x')\|_2 \leq \|x - x'\|_2 \ .$$

Proof. We show $L_x \leq E$ by proofing $\alpha \leq \beta$ for the two angles

$$\begin{aligned} \beta &:= \measuredangle bxx' \\ \alpha &:= \measuredangle axb \ . \end{aligned}$$

The inner angle sum of $\triangle xab$ with a supplementary angle argument and triangle scaling provide $\measuredangle py'y = 90° + \alpha$. Let m denote the center of C. Since $\overline{pm}$ is orthogonal on H_0 and $\measuredangle bx'x = 90° - \beta$, we have $\measuredangle mpx' = \beta$. In the isosceles triangle $\triangle py'm$, the central angle $\gamma = 180° - 2\beta$. Fixing arc $\overline{py'}$ on C for the inscribed angle theorem provides $\measuredangle y'yp = \gamma/2$.

The inner angle sum of $\triangle pyy'$ states

$$\begin{aligned} 0 \leq \measuredangle y'py &= 180° - \measuredangle y'yp - \measuredangle py'y \\ &= 180° - \measuredangle y'yp - (90° + \alpha) \\ &= 180° - \gamma/2 - (90° + \alpha) \\ &= -\alpha + \beta \ . \end{aligned}$$

□

Lemma 3.3.3. *For $x, x' \in \mathbf{B}_1^{d-1}$, we have*

$$\left\|\sigma(x) - \sigma(x')\right\|_2 \leq 2 \; \|x - x'\|_2.$$

Proof. Using Lemma 3.3.2, we have $L_x \leq E$ and the statement follows via triangle scaling:

$$L = L_x \; \overline{py} \; / \; \overline{px} \leq 2L_x \leq 2E \; ,$$

since $\overline{px} \geq 1$ and $\overline{py} \leq 2$. □

This statement is tight, considering the two points $x = 0$ and $x' = \left(\frac{\varepsilon}{\sqrt{d-1}}, \ldots, \frac{\varepsilon}{\sqrt{d-1}}\right)$. We have $\|x - x'\|_2 = \varepsilon$ and $\|\sigma(x) - \sigma(x')\|_2 = 2\frac{1}{\sqrt{1+\varepsilon^2}}\varepsilon$.

Theorem 3.3.4. *Algorithm 7 calculates an ε-approximation exactly on the unit sphere.*

Proof. Let $x^* = x/\|x\|_2$ and $\sigma(y)$ denote the result. Given the rotation, x^* holds for Observation 2. Hence, we can use Lemma 3.3.3 to derive

$$\begin{aligned}
\|\sigma(y) - x^*\|_\infty &= \|\sigma(y) - \sigma(\tau(x^*))\|_\infty \\
&\leq \|\sigma(y) - \sigma(\tau(x^*))\|_2 \\
&\leq 2\|y - \tau(x^*)\|_2 \\
&\leq 2\sqrt{(d-1)\frac{\varepsilon^2}{4(d-1)}} = \varepsilon
\end{aligned}$$

as upper bound on the approximation error. □

This analysis is rather tight, as demonstrated by the red curve and points in Figure 3.2.

3.3.4 Denominator Sizes

We now describe a relation between rational images of σ and the lowest common multiple of denominators of its rational pre-images. This leads to several strategies for achieving small denominators in the results of Algorithm 7.

Lemma 3.3.5 (Size of images under σ)**.** *Let $x \in \mathbb{Q}^{d-1} \cap \mathbf{B}_1^{d-1}$ with $x_i = p_i/q_i$ and $Q = lcm(q_1, \ldots, q_{d-1})$ be the lowest common multiple, then*

$$\sigma_k(x) = \frac{n_k}{m}$$

with integers $n_i, m \in \{-2Q^2, \ldots, 2Q^2\}$ for all $1 \leq k \leq d$.

Proof. Let $q'_i \in \{1, \ldots, Q\}$ such that $q'_i \cdot q_i = Q$ for all i. Since the formula of σ is similar in all but the last dimension, we describe the following two cases. For $k = d$, we have

$$\sigma_k(x) = \frac{-1 + \sum_{i=1}^{d-1} p_i^2/q_i^2}{1 + \sum_{i=1}^{d-1} p_i^2/q_i^2} = \frac{-Q^2 + \sum_{i=1}^{d-1} {q'_i}^2 p_i^2}{Q^2 + \sum_{i=1}^{d-1} {q'_i}^2 p_i^2} =: \frac{n_k}{m}$$

Using the bound $x \in \mathbf{B}_1^{d-1}$, we have $0 \leq \sum_{i=1}^{d-1} {q'_i}^2 p_i^2 \leq Q^2$ and we derive for n_k and m

$$|n_k| = \left| - Q^2 + \sum_{i=1}^{d-1} {q'_i}^2 p_i^2 \right| \leq Q^2$$

$$m = Q^2 + \sum_{i=1}^{d-1} {q'_i}^2 p_i^2 \leq 2Q^2$$

For $k < d$, we have

$$\begin{aligned}\sigma_k(x) &= \frac{2p_k/q_k}{1 + \sum_{i=1}^{d-1} p_i^2/q_i^2} \\ &= \frac{Q^2 \cdot 2p_k/q_k}{Q^2 + \sum_{i=1}^{d-1} {q'_i}^2 p_i^2} \\ &= \frac{Qq'_k \cdot 2p_k}{Q^2 + \sum_{i=1}^{d-1} {q'_i}^2 p_i^2} =: \frac{n_k}{m}\end{aligned}$$

Using the bound $x \in \mathbf{B}_1^{d-1}$, we have that each $|p_i| \leq q_i$ and this bounds $|n_k| = Qq'_k \cdot 2|p_k| \leq 2Q^2$. We already discussed the bound on m in the first case. □

Note that we apply this lemma in practice with fixed-point binary numbers $p_i/q_i \in P_s$. Meaning all $q_i = 2^s = Q$ for some significant size s.

Theorem 3.3.6. *Denominators in ε-approximations of Algorithm 7 are at most*

$$\frac{10(d-1)}{\varepsilon^2}\,.$$

Proof. Using standard multi-precision floating point arithmetics allows to derive rational values y, with denominators that are $Q = \lceil \frac{2\sqrt{d-1}}{\varepsilon} \rceil$. Using $\varepsilon \leq 1/8$ and Lemma 3.3.5 bounds the size of the denominators in images σ with

$$\begin{aligned}2Q^2 &\leq 2\left(1 + \frac{2\sqrt{d-1}}{\varepsilon}\right)^2 \\ &= \frac{2}{\varepsilon^2}\left(\underbrace{\varepsilon^2 + \varepsilon 4\sqrt{d-1}}_{\leq (d-1)} + 4(d-1)\right)\,.\end{aligned}$$

□

For certain dimensions and in practice (c.f. Section 3.5.1), we can improve on the simple usage of fixed-point binary numbers. For $\mathbb{S}^1$ we can rely on the continued fraction algorithm to derive rational approximations of $\alpha = \tau(x/\|x\|_2)$ with $|\alpha - p/q| \leq 1/2q^2$. Using this in Algorithm 7 leads to approximations with $\varepsilon = 1/q^2$ on the circle $\mathbb{S}^1$ with denominators of at most $2q^2$.

Note that for $\mathbb{S}^d$ with $d \geq 2$ one can rely on algorithms for simultaneous Diophantine approximations (c.f. Theorem 3.2.2) to keep the lowest common multiple Q in Lemma 3.3.5 small. Note that it might well be *simpler* to find Diophantine approximations with small Q.

There have been many approaches to find generalizations of the continued fraction algorithm for $d > 1$. One of the first approaches is the Jacobi-Perron algorithm, which is rather simple to implement [Tac94] (c.f. Section 3.5.1). More advanced approaches [PPB17] rely on the LLL-algorithm for lattice basis reduction [LLL82]. For $d = 2$ there is an algorithm to compute all Dirichlet Approximations [JKP79], which we find hard to oversee given its extensive presentation. Moreover, their experimental comparison shows that the Jacobi-Perron algorithm is practically well suited for $d = 2$.

We close this section with a transfer result of Theorem 3.2.2 with our Theorem 3.3.4 and Lemma 3.3.5.

Corollary 3.3.7. *Let $x \in \mathbb{S}^{d-1}$ and $N \in \mathbb{N}$. There is $p \in \mathbb{Z}^{d-1}$ and $q \in \{1, \ldots, N\}$ with*

$$\left\| x - \sigma\left(\frac{1}{q}p\right) \right\|_\infty \leq \frac{2\sqrt{d-1}}{q \sqrt[d-1]{N}}$$

and all denominators of $\sigma\left(\frac{1}{q}p\right)$ are at most $2q^2$.

This existence statement allows for brute-force computations. However, we just use it for comparisons in Section 3.5.1.

3.4 Implementation

Apart from [CdCL+10] for $\mathbb{S}^2$, most implementations of spherical Delaunay triangulations are not ‘stable’. Approaches based on d-dimensional convex hull algorithms produce only a tessellation for input not exactly *on* $\mathbb{S}^{d-1}$ (c.f. Section 3.1).

Few available implementations allow dynamic point or constraint insertion and deletion – not even in the planar case of $\mathbb{R}^2$. The ‘Computational Geometry Algorithms Library’ (CGAL [Pro15]) is, to our knowledge, the sole implementation providing dynamic insertions/deletions of points *and* constraint line segments in $\mathbb{R}^2$.

With [BS17c], we provide open-source implementations of Algorithm 7 for $\mathbb{S}^d$. In [BS17b], we provide an implementation for spherical Delaunay triangulations on $\mathbb{S}^2$ with ε-stable constructions of intersection points of constraint line-segments (c.f. Section 3.4.2).

3.4.1 RATional Sphere Snapping for $\mathbb{S}^d$

Libratss is a C++ library which implements Algorithm 7, based on the open-source GMP library for exact rational arithmetics [Gt12] and the GNU 'Multiple Precision Floating-Point Reliably'(MPFR) library [FHL+07]. The implementation allows both, input of Cartesian coordinates of arbitrary dimension and spherical coordinates of $\mathbb{S}^2$. Note that this implementation allows geometric algorithms, as for d-dimensional convex hull, to rely on rational input points that are *exactly* on $\mathbb{S}^{d-1}$. In light of the discussion on the denominator sizes in Section 3.3.4, we provide two additional strategies to fixed-point snapping, as analyzed in Theorem 3.3.4. We implemented the Continued Fraction Algorithm to derive rational ε-approximations with small denominators and the Jacobi-Perron algorithm for $\mathbb{S}^2$. The library interface also allows to automatically chose the approximation method which results in smaller denominators, approximation errors or other objectives, like byte-size.

3.4.2 Incremental Constrained Delaunay Triangulation on $\mathbb{S}^2$

Libdts2 implements an adapter for the *dynamic constraint* Delaunay triangulation in the Euclidean plane $\mathbb{R}^2$ of CGAL. Since this implementation requires an initial outer face, we introduce a small triangle, that only contains the north-pole, to allow subsequent insertions of points and constraints. For points *exactly* on the unit sphere, the predicate **'is A in the circumcircle of B, C and D'** reduces to the well studied predicate **'is A above the plane through B, C and D'**. The implementation overloads all predicate functions accordingly and uses Algorithm 7 for the construction of rational points on the sphere for intersections of Great Circle segments.

Reductions of Spherical Predicates to Cartesian Orientation Predicates

We first describe a reduction from the spherical predicates to well studied Cartesian predicates.

Lemma 3.4.1 (Great Circle Orientation Predicate)**.** *Let $p_1, p_2 \in \mathbb{S}^2$ with $p_1 \neq p_2$ and P the plane containing p_1, p_2 and the origin $(0, 0, 0)$ and C be the Great Circle* *through p_1 and p_2. For $q \in \mathbb{S}^2$ we have*

$$\begin{aligned} q \textit{ left-of } P &\iff q \textit{ left-of } C \\ q \in P &\iff q \in C \\ q \textit{ right-of } P &\iff q \textit{ right-of } C \end{aligned}$$

Proof. $\mathbb{S}^2 \cap P = C$ and $\mathbb{S}^2 = L \cup C \cup R$. □

Lemma 3.4.2 (Circumsphere Predicate)**.** *Let P denote the plane through non-identical points $p_1, p_2, p_3 \in \mathbb{S}^2$ and the half space containing the origin $(0, 0, 0)$ is called 'below P'. We further call $S_{123} \subseteq \mathbb{R}^3$ the closed volume of the sphere with p_1, p_2, p_3 and the origin on its surface. For a point $q \in \mathbb{S}^2$ we have*

$$\begin{aligned} q \textit{ above } P &\iff q \in S_{123} \setminus \partial S_{123} \\ q \in P &\iff q \in \partial S_{123} \\ q \textit{ below } P &\iff q \notin S_{123} \end{aligned}$$

Proof. P is uniquely determined because three different points on the unit sphere are not co-linear. Since S_{123} and $\mathbb{S}^2$ are spheres, their cuts with P are circles in P and the two circles are identical as they contain p_1, p_2 and p_3 on their boundary. This circle C has a radius of at most 1 and partitions the points of the unit sphere into three sets

$$\mathbb{S}^2 = A \cup C \cup B$$

where $A \subseteq S_{123} \not\supseteq B$. If C is a Great Circle we resolve ambiguity for 'above' and the center of S_{123} by choosing the open half spaces that first contain $(0, 0, 1)$, then $(0, 1, 0)$ and eventually $(1, 0, 0)$. We have $\mathbb{S}^2 \neq \partial S_{123}$, since the origin is a fourth point on S_{123} and $q \in P$ iff. $q \in C$ iff. $q \in \partial S_{123}$. Therefore it is sufficient to show $q \in A \iff q$ above P. To this end we consider the convex volume of the unit sphere $S \subseteq \mathbb{R}^3$ and $D = S \cap S_{123}$. Note that ∂D contains C and A. Since the cut with the closed half-space of the plane P cuts a convex body into at most three parts and $C \subseteq P$, we have that all of A is 'above' P. □

ε-stable geometric constructions

Any means of geometric construction that allows to approximate a certain point, can be used as input for Algorithm 7 – e.g. the intersection of Great Circle segments. Consider two intersecting segments of rational points on $\mathbb{S}^2$. The two planes, containing the segments and the origin as a third point, intersect in a straight line. Each (rational) point on this line can be used as input for our method, as they identify the two intersection points on the sphere. Using such input for Algorithm 7 allows simple schemes to derive stable geometric constructions of rational points on $\mathbb{S}^d$ within a distance of ε to the target point.

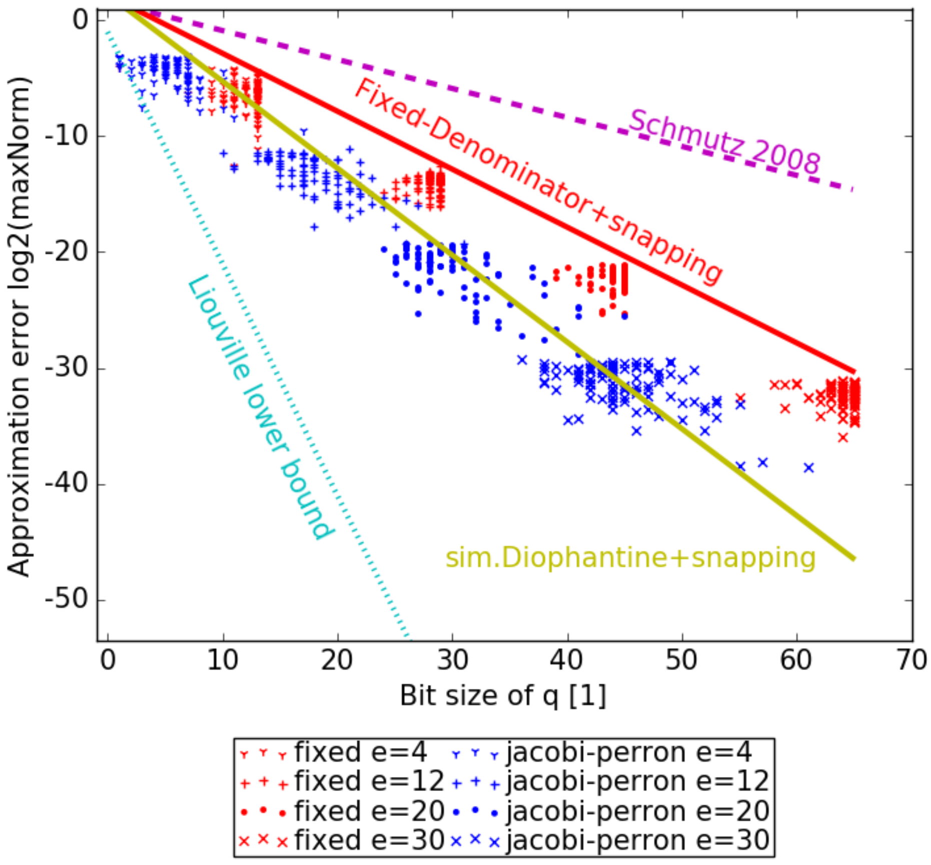


Figure 3.2: Approximation quality and denominator size of 100 random points on $\mathbb{S}^2$ for various levels of target precision e and approximation strategies (red, blue) of Algorithm 7. Theoretic bounds are indicated with lines.

3.5 Experiments

We used real world and synthetic data for our experiments. geo-referenced data was sampled from regional extracts from the OpenStreetMap project [OSM17], as of January 26th, 2017. Random Cartesian coordinates of points on $\mathbb{S}^d$ were created with the uniform generator 2 of [Mar72]. All benchmarks were conducted on a single core of an Intel Xeon E5-2650v4. Peak memory usage and time were measured using the `time` utility.

3.5.1 Approximation Quality and Size

We experimentally analyze the actual approximation error in results of Algorithm 7 for several levels of ε using the MPFR library. In this section e denotes the significands required in statement 2 of Algorithm 7 for the required result precision ε. This is

$$e = \left\lceil -\log_2\left(\frac{\varepsilon}{2\sqrt{d-1}}\right)\right\rceil \quad .$$

We simply setup the MPFR data types with significand sizes up to 1024 Bits, and conducted our experiments on much lower levels of e. This allows us to derive some 'measure' of the actual approximation errors of our method.

We analyzed the approximation errors δ and denominator bit-sizes q for 100 random points on $\mathbb{S}^2$. Figure 3.2 compares the results of our algorithm under several levels of target precision e and strategies for statement 2 in our method. The magenta line indicates the quality and size of the approach in [Sch08]. The red line indicates the bounds of our Theorems 3.3.4 and 3.3.6 on the fixed-point strategy, while the yellow line indicates the bound of Corollary 3.3.7. Note that results using the Jacobi-Perron strategy (blue dots) allows our method to further improve on the fixed-point strategy (red dots). Note that we use Liouville's lower bound as statement on the approximability of a worst-case point. There might well be points of higher algebraic degree that allow better approximations (c.f. Section 3.2.1).

Table 3.1 exhibits average approximation errors δ, denominator bit-sizes q and the computation time t of our method for millions of points. Synthetic data sets have several dimensions, while the real world data sets have dimension 3. For $\mathbb{S}^2$, we provide comparison of the fixed-point strategy (fx) with the Jacobi-Perron strategy (jp) of our method. Using $e = 31$ is sufficient to obtain results *exactly on* $\mathbb{S}^2$ with a δ of less than 1cm, relative to a sphere with radius of the earth. This is enough for most applications dealing with spatial data *and* allows storage within the word size of contemporary computing hardware. This allows practical applications on $\mathbb{S}^2$ to store 4 integer long values for the 3 numerators and the common denominator (c.f. Lemma 3.3.5) occupying 32 Bytes. Note that storing 3 double values occupies 24 Bytes but *cannot* represent Cartesian coordinates *exactly on* the sphere.

3.5.2 Constrained Delaunay Triangulation with Intersection Constructions

A Constrained Delaunay Triangulation of a point set contains required line-segments as edges, but is as close to the Delaunay triangulation as possible [Che87]. We used *very large* street networks of several regions from the OpenStreetMap project for points and constraint edges – E.g. each line-segment of a street is an edge in the result triangulation. Since $\sim 0.5\%$ of the line-segments in these data sets intersect, we approximated the intersection points using $e = 31$ for Algorithm 7.

		Germany	Planet	u.a.r $\mathbb{S}^2$	u.a.r $\mathbb{S}^9$	u.a.r $\mathbb{S}^{99}$
dimension		3	3	3	10	100
size [10^3]		2,579.6	3,702.4	1,000.0	1,000.0	100.0
e=23						
	$\delta[m]$	0.7	0.7	0.7	1.0	3.2
fx	q [1]	46.0	46.0	46.0	46.0	46.0
	t [μs]	17	16	16	117	546
	$\delta[m]$	0.4	0.4	0.5	-	-
jp	q [1]	33.6	34.2	34.1	-	-
	t [μs]	63	57	58	-	-
e=31						
	$\delta[m]$	2.7e-3	2.6e-3	2.8e-3	4.0e-3	12.6e-3
fx	q [1]	62.0	62.0	62.0	62.0	62.0
	t [μs]	17	16	17	118	554
	$\delta[m]$	1.7e-3	1.7e-3	1.8e-3	-	-
jp	q [1]	45.2	45.8	45.8	-	-
	t [μs]	77	72	73	-	-
e=53						
	$\delta[m]$	6.3e-10	6.2e-10	6.6e-10	9.6e-10	30.1e-10
fx	q [1]	106.0	106.0	106.0	106.0	106.0
	t [μs]	16	16	17	118	548
	$\delta[m]$	3.9e-10	3.9e-10	4.3e-10	-	-
jp	q [1]	77.2	77.8	77.7	-	-
	t [μs]	118	111	112	-	-
e=113						
	$\delta[m]$	5.5e-28	5.4e-28	5.7e-28	8.3e-28	26.1e-28
fx	q [1]	226.0	226.0	226.0	226.0	226.0
	t [μs]	19	19	19	126	617
	$\delta[m]$	3.4e-28	3.4e-28	3.7e-28	-	-
jp	q [1]	164.5	165.1	165.1	-	-
	t [μs]	219	218	220	-	-

Table 3.1: Mean-values of approximation error δ [m], denominator bit-size q [1] and computation time t [μs] for synthetic and real-world point sets for various dimensions and levels of target precision e. The Jacobi-Perron strategy is denoted by 'jp' and the fixed-point strategy by 'fx'.

	Saarland	Germany	Europe	Planet
Input				
Segments $[10^6]$	0.32	25.75	222.92	668.61
Output				
Vertices $[10^6]$	0.29	24.45	213.01	634.42
Edges $[10^6]$	0.87	73.37	639.04	$1,903.27$
Faces $[10^6]$	0.58	48.91	426.03	$1,268.84$
Resource usage				
Time [h:m]	$<$ 0:01	0:05	0:49	5:08
Memory [GiB]	$<$ 0.4	27.7	243.2	724.1

Table 3.2: Time and memory usage to compute spherical Delaunay triangulations for OpenStreetMap data sets.

Table 3.2 exhibits total running time, peak memory usage and the result sizes of our `libdts2` implementation. Small data sets like Saarland and Germany allow quick calculation on a recent workstation computer. See Figure 3.1 for the Ecuador dataset. Note that the current implementation has a storage overhead for each point, as we keep the results of the GMP library rather than truncating to integers of architectures word size. Computing the triangulation for the planet data set was only possible on rather powerful hardware with at least 768 gigabytes of memory taking less than a quarter of a day.

3.6 Research Directions

From a practical point of view, it is of great interest to bound the storage size of denominators to a maximum of 64Bits – the word size of current computing architectures. We seek to improve our (already satisfactory) results by using advanced algorithms for simultaneous approximation, like the LLL-algorithm or the Dirichlet approximation algorithm for $\mathbb{S}^2$.

For the theoretical part, we are interested if finding simultaneous rational approximations with small lowest common multiple of the denominators is simpler than finding Dirichlet approximations. We are also interested in generalizing the method to provide rational approximations with small *absolute* errors on ellipsoids with rational semi-principal axes – e.g. the geographic WGS84 ellipsoid.

Bibliography

[AAKS14] Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the Discrete Fréchet Distance in Subquadratic Time. *SIAM J. Comput.*, 43(2):429–449, 2014.

[AERW03] Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. *J. Algorithms*, 49(2):262–283, 2003.

[AG60] E. Asplund and B. Grünbaum. On a coloring problem. *Math. Scand.*, 8:181–188, 1960.

[AG95] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995.

[AKRT12] Mohamed H. Ali, John Krumm, Travis Rautman, and Ankur Teredesai. GIS Cup 2012. In *Proc. ACM Conference SIGSPATIAL GIS*, GIS'12, pages 597–600, 2012.

[AY15] Heba Aly and Moustafa Youssef. semMatch: road semantics-based accurate map matching for challenging positioning data. In *Proc. ACM Conference SIGSPATIAL GIS*, GIS'15, pages 5:1–5:10, 2015.

[BBF$^+$17] Daniel Bahrdt, Michael Becher, Stefan Funke, Filip Krumpe, André Nusser, Martin Seybold, and Sabine Storandt. Growing Balls in $\mathbb{R}^d$. In *Proc. of the 19th Workshop on Algorithm Engineering and Experiments*, ALENEX'17, pages 247–258, 2017.

[BBW09] Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, SODA'09, pages 645–654, 2009.

[BCMS97] E. Belogay, Carlos Cabrelli, Ursula Molter, and Ron Shonkwiler. Calculating the Hausdorff Distance Between Curves. *Inf. Process. Lett.*, 64(1):17–22, 1997.

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[BFS98] Christoph Burnikel, Stefan Funke, and Michael Seel. Exact Geometric Predicates Using Cascaded Computation. In *Proc. of Sympos. on Comput. Geom.*, SoCG'98, 1998.

[BH98] Vasco Brattka and Peter Hertling. Feasible Real Random Access Machines. *J. Complexity*, 14(4):490–526, 1998.

[BPSW05] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. Conference on Very Large Data Bases*, VLDB'05, pages 853–864, 2005.

[Bri14] Karl Bringmann. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *Proc. of the 55th IEEE Annual Symposium on Foundations of Computer Science*, FOCS'14, pages 661–670, 2014.

[Bro79] Kevin Q. Brown. Voronoi Diagrams from Convex Hulls. *Inf. Process. Lett.*, 9(5):223–228, 1979.

[BS17a] Daniel Bahrdt and Martin P. Seybold. Rational Points on the Unit Sphere: Approximation Complexity and Practical Constructions. In *Proc. of International Symposium on Symbolic and Algebraic Computation*, ISSAC'17, 2017.

[BS17b] Daniel Bahrdt and Martin P. Seybold. Libdts2 library on GitHub. `www.github.com/fmi-alg/libdts2`, 2017.

[BS17c] Daniel Bahrdt and Martin P. Seybold. Libratss library on GitHub. `www.github.com/fmi-alg/libratss`, 2017.

[BSS89] Lenore Blum, Mike Shub, and Steve Smale. On a Theory of Computation and Complexity Over the Real Numbers: NP-Completeness, Recursive Functions and Universal Machines. *Bulletin (New Series) of the American Mathematical Society*, 21:1–46, 1989.

[Bur65] J.P. Burling. *On coloring problems of families of prototypes.* PhD thesis, University of Colorado, 1965.

[CdCL$^+$10] Manuel Caroli, Pedro M. M. de Castro, Sébastien Loriot, Olivier Rouiller, Monique Teillaud, and Camille Wormser. Robust and Efficient Delaunay Triangulations of Points on Or Close to a Sphere. In *Proc. of Sympos. on Exp. Alg.*, pages 462–473, 2010.

[CDR92] J. Canny, B.R. Donald, and E.K. Ressler. A rational rotation method for robust geometric algorithms. In *Proc. of Sympos. on Comput. Geom.*, SoCG'92, 1992.

[CGS11] Francis Y. L. Chin, Zeyu Guo, and He Sun. Minimum Manhattan Network is NP-Complete. *Discrete & Computational Geometry*, 45(4):701–722, 2011.

[Che87] L. P. Chew. Constrained Delaunay Triangulations. In *Proc. of Sympos. on Comput. Geom.*, SoCG'87, pages 215–222, 1987.

[CNV08] Victor Chepoi, Karim Nouioua, and Yann Vaxès. A rounding algorithm for approximating minimum Manhattan networks. *Theor. Comput. Sci.*, 390(1):56–69, 2008.

[dCCLT09] Pedro Machado Manhães de Castro, Frédéric Cazals, Sébastien Loriot, and Monique Teillaud. Design of the CGAL 3D Spherical Kernel and application to arrangements of circles on a sphere. *Comput. Geom.*, 42(6-7):536–550, 2009.

[DFK$^+$17] Aparna Das, Krzysztof Fleszar, Stephen Kobourov, Joachim Spoerhase, Sankar Veeramoni, and Alexander Wolff. Approximating the Generalized Minimum Manhattan Network Problem. *Algorithmica*, 2017.

[Die10] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics, Volume 173. Springer-Verlag, Heidelberg, 4th edition, July 2010.

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proc. of the 46th annual ACM Symposium on Theory Of Computing* , STOC, pages 624–633, 2014.

[EFH$^+$11] J. Eisner, S. Funke, A. Herbst, A. Spillner, and S. Storandt. Algorithms for Matching and Predicting Trajectories. In *Proc. ACM-SIAM Conference Algorithm Engineering and Experiments*, ALENEX'11, pages 84–95, 2011.

[EM94] Thomas Eiter and Heikki Mannila. Computing Discrete Fréchet Distance. Technical report, Christian Doppler Laboratory for Expert Systems, TU Vienna, 1994.

[Eng10] Birgit Engels. The Transitive Minimum Manhattan Subnetwork Problem in 3 dimensions. *Discrete Applied Mathematics*, 158(4):298–307, 2010.

[Fei98] Uriel Feige. A Threshold of Ln N for Approximating Set Cover. *J. ACM*, 45(4):634–652, July 1998.

[FHL$^+$07] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. MPFR: A Multiple-precision Binary Floating-point Library with Correct Rounding. *ACM Trans. Math. Softw.*, 33(2), June 2007.

[FS14] Stefan Funke and Martin P. Seybold. The Generalized Minimum Manhattan Network Problem (GMMN) – Scale-Diversity Aware Approximation and a Primal-Dual Algorithm. In *Proc. of the 26th Canadian Conference on Computational Geometry*, CCCG'14, 2014.

[Geo16] Geofabrik. Regional Abstracts of the OSM Map. `http://www.geofabrik.de`, 2016.

[GGP$^+$94] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, É. Tardos, and D. P. Williamson. Improved Approximation Algorithms for Network Design Problems. In *Proc. of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'94, pages 223–232, 1994.

[GLN01] Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Approximating a Minimum Manhattan Network. *Nord. J. Comput.*, 8(2):219–232, 2001.

[GP13] Michael T. Goodrich and Pawel Pszona. Cole's Parametric Search Technique Made Practical. In *Proc. of the 25th Canadian Conference on Computational Geometry*, CCCG'13, 2013.

[Gt12] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition, 2012. `http://gmplib.org/`.

[GW95] Michel X. Goemans and David P. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[Hol] Stefan Holder. GitHub Project HMM-Lib.

[HW54] Godfrey H. Hardy and Edward M. Wright. *An introduction to the theory of numbers.* Oxford University Press., 1954.

[Jai01] Kamal Jain. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. *Combinatorica*, 21(1):39–60, 2001.

[JGR$^+$13] D. W. Jacobsen, M. Gunzburger, T. Ringler, J. Burkardt, and J. Peterson. Parallel algorithms for planar and spherical Delaunay construction with an application to centroidal Voronoi tessellations. *Geosci. Model Dev.*, 6(4), 2013.

[JKP79] W. Jurkat, W. Kratz, and A. Peyerimhoff. On best two-dimensional Dirichlet-approximations and their algorithmic calculation. *Mathematische Annalen*, 244(1), 1979.

[Kar08] Peter Karich. GitHub Project GraphHopper Map Matching, Snapshot 0.8 (2016-09-08).

[KKRW10] Bastian Katz, Marcus Krug, Ignaz Rutter, and Alexander Wolff. Manhattan-Geodesic Embedding of Planar Graphs. In *Proc. 17th Internat. Sympos. Graph Drawing* , GD'09, pages 207–218, 2010.

[KM15] Dmitry Kleinbock and Keith Merrill. Rational approximation on spheres. *Israel Journal of Mathematics*, 209(1):293–322, 2015.

[LHK$^+$13] Y. Li, Q. Huang, M. Kerber, L. Zhang, and L. Guibas. Large-scale Joint Map Matching of GPS Traces. In *Proc. ACM Conference SIGSPATIAL GIS*, GIS'13, pages 214–223, 2013.

[Lio51] Joseph Liouville. Sur des classes très-étendues de quantités dont la valeur n'est ni algébrique, ni même réductible à des irrationalles algébriques. *J. Math. pures et app.*, 16:133–142, 1851.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *MATH. ANN*, 261:515–534, 1982.

[LPY05] Chen Li, Sylvain Pion, and Chee-Keng Yap. Recent progress in exact geometric computation. *J. Log. Algebr. Program.*, 64(1):85–111, 2005.

[LR00] Bing Lu and Lu Ruan. Polynomial Time Approximation Scheme for the Rectilinear Steiner Arborescence Problem. *J. Comb. Optim.*, 4(3):357–363, 2000.

[LZZ$^+$09] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate GPS trajectories. In *Proc. ACM Conference SIGSPATIAL GIS*, GIS'09, pages 352–361, 2009.

[Mar72] G. Marsaglia. Choosing a point from the surface of a sphere. *The Annals of Mathematical Statistics*, 43(2), 1972.

[MSU09] Xavier Muñoz, Sebastian Seibert, and Walter Unger. The Minimal Manhattan Network Problem in Three Dimensions. In Sandip Das and Ryuhei Uehara, editors, *WALCOM*, volume 5431 of *Lecture Notes in Computer Science*, pages 369–380. Springer, 2009.

[Niv85] Ivan Niven. *Irrational Numbers*, volume 11. Math. Assoc. of America, 1 edition, 1985.

[NK09] Paul Newson and John Krumm. Hidden Markov map matching through noise and sparseness. In *Proc. ACM Conference SIGSPATIAL GIS*, GIS'09, pages 336–343, 2009.

[OSM13] The OpenStreetMap Project OSM. GPS Traces. `http://planet.openstreetmap.org/gps/`, 2013.

[OSM17] The OpenStreetMap Project OSM. Map Data. `www.openstreetmap.org`, 2017.

[PKK$^+$13] Arkadiusz Pawlik, Jakub Kozik, Tomasz Krawczyk, Michał Lasoń, Piotr Micek, William T. Trotter, and Bartosz Walczak. Triangle-Free Geometric Intersection Graphs with Large Chromatic Number. *Discrete & Computational Geometry*, 50(3):714–726, Oct 2013.

[PPB17] Attila Pethő, Michael E. Pohst, and Csanád Bertók. On multidimensional Diophantine approximation of algebraic numbers. *J. Number Theory*, 171, 2017.

[Pro15] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.7 edition, 2015.

[PZ15] Florian Prill and Günther Zängl. A Compact Parallel Algorithm for Spherical Delaunay Triangulations. In *Proc. Conf. on Parallel Processing and Appl. Math.*, 2015.

[QON07] Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007.

[Ren97] Robert J. Renka. Algorithm772 STRIPACK Delaunay Triangulation and Voronoi Diagram on the Surface of a Sphere. *Trans. Math. Softw.*, 23(3), 1997.

[Riv74] Theodore J. Rivlin. *The Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory.* Tracts in Pure & Applied Mathematics. Wiley, 1974.

[RSHS92] Sailesh K. Rao, P. Sadayappan, Frank K. Hwang, and Peter W. Shor. The Rectilinear Steiner Arborescence Problem. *Algorithmica*, 7(2&3):277–288, 1992.

[Saa99] Alan Saalfeld. Delaunay Triangulations and Stereographic Projections. *Cartography and Geographic Information Science*, 26(4):289–296, 1999.

[Sch79] Arnold Schönhage. *On the power of random access machines*, pages 520–529. Springer Berlin Heidelberg, Berlin, Heidelberg, 1979.

[Sch08] Eric Schmutz. Rational points on the unit sphere. *Central European Journal of Mathematics*, 6(3), 2008.

[Sch15] Michael Schnizler. The Generalized Minimum Manhattan Network Problem. Master's thesis, University of Stuttgart, 2015.

[Sey17] Martin P. Seybold. Robust Map Matching for Heterogeneous Data via Dominance Decompositions. In *Proc. of the 2017 SIAM International Conference on Data Mining*, SDM'17, pages 813–821, 2017.

[She97] Jonathan Richard Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18(3):305–368, 1997.

[Sny92] Timothy Law Snyder. On the Exact Location of Steiner Points in General Dimension. *SIAM J. Comput.*, 21(1):163–180, 1992.

[SS05] Weiping Shi and Chen Su. The Rectilinear Steiner Arborescence Problem Is NP-Complete. *SIAM J. Comput.*, 35(3):729–740, 2005.

[Tac94] Hiroko Tachii. On Jacobi-Perron algorithm. In *Proc. Japan Acad. Ser. A Math. Sci.*, volume 70, pages 317–322. The Japan Academy, 1994.

[Vaz03] Vijay V. Vazirani. *Approximation algorithms.* Springer, 2003.

[vOV04] René van Oostrum and Remco C. Veltkamp. Parametric search made practical. *Comput. Geom.*, 28(2-3):75–88, 2004.

[vOV05] René van Oostrum and Remco C. Veltkamp. Parametric Search Package: Tutorial, 2005.

[Wil02] David P. Williamson. The primal-dual method for approximation algorithms. *Math. Program.*, 91(3):447–478, 2002.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.

[WSP06] Carola Wenk, Randall Salas, and Dieter Pfoser. Addressing the Need for Map-Matching Speed: Localizing Globalb Curve-Matching Algorithms. In *Proc. Conference on Scientific and Statistical Database Management*, SSDBM'06, pages 379–388, 2006.

[WWFZ13] Hong Wei, Yin Wang, George Forman, and Yanmin Zhu. Map Matching by Frechet Distance and Global Weight Optimization. Technical report sjtucstr201302001, Department of Computer Science and Engineering, Shanghai Jiao Tong University, 2013.

[WZ14] Guanfeng Wang and Roger Zimmermann. Eddy: an error-bounded delay-bounded real-time map matching algorithm using HMM and online Viterbi decoder. In *Proc. ACM Conference SIGSPATIAL GIS*, GIS'14, pages 33–42, 2014.

[Zac01] Martin Zachariasen. A catalog of Hanan grid problems. *Networks*, 38(2):76–83, 2001.

[Zhe15] Yu Zheng. Trajectory Data Mining: An Overview. *ACM Trans. Intell. Syst. Technol.*, 6(3):29:1–29:41, 2015.