Algorithms for Optimal Transport and Wasserstein Distances

Dissertation
for the award of the degree
"Doctor rerum naturalium" (Dr. rer. nat.)
of the Georg-August-Universität Göttingen
within the doctoral program "Mathematical Sciences"
of the Georg-August University School of Science
(GAUSS)

submitted by
Jörn Schrieber
from Hamburg

Göttingen, 2019

Thesis Committee

Prof. Dr. Dominic Schuhmacher, Institute for Mathematical Stochastics

Prof. Dr. Anita Schöbel, Institute for Numerical and Applied Mathematics

Members of the Examination Board

Reviewer:

Prof. Dr. Dominic Schuhmacher, Institute for Mathematical Stochastics

Second Reviewer:

Prof. Dr. Anita Schöbel, Institute for Numerical and Applied Mathematics

Further Members of the Examination Board

Prof. Dr. Axel Munk, Institute for Mathematical Stochastics

Prof. Dr. Stephan Huckemann, Institute for Mathematical Stochastics

Prof. Dr. Gerlind Plonka-Hoch, Institute for Numerical and Applied Mathematics

Prof. Dr. Russell Luke, Institute for Numerical and Applied Mathematics

Date of the oral examination

14 February, 2019

Acknowledgements

I would like to express my gratitude towards everyone, who supported me during my work on this thesis.

First and foremost, thanks to my supervisors Dominic Schuhmacher and Anita Schöbel, who took the time for discussions on a regular basis and provided me with their unique perspectives, insights on the matter and invaluable advice. This dedication is not taken for granted and greatly appreciated!

I wish to thank everyone else I had the pleasure to collaborate with on research projects: Carsten Gottschlich, Max Sommerfeld, Axel Munk and Yoav Zemel from the Institute for Mathematical Stochastics. Further thanks to Carsten Gottschlich and Christian Böhm for their aid in resolving countless technical difficulties.

I would also like to thank my colleagues from the Institute for Mathematical Stochastics, who all contribute to the friendly and cooperative atmosphere at the institute, and the workgroup for optimization at the Institute for Numerical and Applied Mathematics, who despite my spatial distance always welcomed me to seminars and workshops.

Furthermore, thanks to Khwam Tabougua Trevor for helping me with the creation of the plots in Chapter 5 during his time as a student assistant.

I gratefully acknowledge funding by the German Research Foundation (DFG) as part of the Research Training Group 2088 “Discovering Structure in Complex Data: Statistics Meets Optimization and Inverse Problems” during the time from October 2015 to September 2018.

Preface

Optimal Transport and *Wasserstein Distance* are closely related terms that do not only have a long history in the mathematical literature, but also have seen a resurgence in recent years, particularly in the context of the many applications they are used in, which span a variety of scientific fields including - but not limited to - imaging, statistics and machine learning. Due to drastic increases in data volume and a high demand for Wasserstein distance computation, the development of more efficient algorithms in the domain of optimal transport increased in priority and the advancement picked up pace quickly.

This thesis is dedicated to algorithms for solving the optimal transport problem and computing Wasserstein distances. After an introduction to the field of optimal transport, there will be an overview of the application areas as well as a summary of the most important methods for computation with a focus on the discrete optimal transport problem. This is followed by a presentation of a benchmark for discrete optimal transport together with a performance test of a selection of algorithms on this data set. Afterwards, two new approaches are introduced: a probabilistic approximation method for Wasserstein distances using subsampling and a clustering method, which aims to generalize multiscale methods to discrete optimal transport problems, including instances with a non-metric cost function.

Collaborations and Contributions

While this thesis is largely based on the author's original research, it would not have been possible without the work of other people. In particular, parts

of this thesis have been previously published as papers and are the results of collaborations.

Chapters 1 and 2 serve as a general introduction into optimal transport and an overview over popular algorithms, respectively, and is the author's compilation of work done by other researchers.

Chapter 3 is a collaboration between Dominic Schuhmacher, Carsten Gottschlich and the author of this thesis and was published as the paper *DOTmark - A Benchmark for Discrete Optimal Transport* [78] in IEEE Access. Conceptualization and creation of the benchmark itself was largely done by Dominic Schuhmacher, while the author was responsible for the performance test, including the implementation of the shielding method, execution and analysis of the results.

The content of Chapter 4 is available as the preprint *Optimal Transport: Fast Probabilistic Approximation with Exact Solvers* [90], which is joint work with Max Sommerfeld, Yoav Zemel, Axel Munk. While the author and Max Sommerfeld contributed in equal parts to the simulations and analysis in Section 4.4, Max Sommerfeld and Axel Munk conceptualized the subsampling scheme and proved the theoretical results with several improvements contributed by Yoav Zemel. The proofs of the results in Section 4.3 are omitted in this thesis and they can be found in the appendix of the paper [90]. Due to this collaboration the content of this chapter is also contained in Max Sommerfeld's doctoral thesis [88].

Chapter 5 is based on the author's own research.

Contents

Chapter 1

Introduction

Between the different mathematical fields, the subject of optimal transport is uniquely positioned. The history of its theory goes back to the 18th century, when the French mathematician Gaspard Monge published his work [59] on a novel problem formulation at the boundary between analysis and probability theory, which would later become known as the *Monge formulation* of optimal transport. Two centuries later, Kantorovich revised this subject with his more general statement of the problem [46], which laid the foundation for optimal transport to be as widely applied as it is today and the *Kantorovich formulation* is primarily considered in modern applications. In the discrete case, both formulations have deep connections to well studied optimization problems. For example, the assignment problem is a special case of the Monge formulation and the Kantorovich problem can be written as a minimum cost flow problem on a complete bipartite network.

Conceptually, this problem revolves around the idea of efficient rearrangement and operates on probability measures. What is rearranged in a particular application can vastly differ - from physical goods, people or particles to more abstract concepts, like greyscale values of pixels in images or the classification result of a neural network - as long as it can be cast in the form of a measure. It comes as no surprise that potential applications are plenty and consequently optimal transport research has thrived in recent years. The rich theoretical background in this field is more and more supported by the development and

analysis of modern methods to tackle optimal transport problems.

However, in many subject areas, the application of optimal transport or Wasserstein distances is still held back by the large computational burden. Despite the ability to cast optimal transport into the form of a very simple linear program, the input sizes often vastly exceed the boundaries of viability of even the latest linear programming methods with high runtime and memory consumption. This was somewhat mitigated by advances on regularized optimal transport and the introduction of the *Sinkhorn scaling* algorithm to optimal transport [20], which gave it another boost to viability in applications, as it is a simple and fast, albeit not necessarily precise, algorithm.

1.1 Mathematical Setup

The theory of optimal transport has been explored thoroughly since its inception with connections drawn to analysis and probability theory. There is great and extensive literature for anyone who wants to engage deeply with the theory, in particular, a book written by Fields Medal holder Cédric Villani [96]. More recently, two books have been published with a higher emphasis on applied mathematics - one in 2015 by Filippo Santambrogio [74] giving insight into the theory from a numerical point of view and a new open book by Gabriel Peyré ad Marco Cuturi from 2018 [65], which gives a comprehensive overview of numerical methods. Since this thesis is focused on algorithms and applications for optimal transport, we keep the mathematical introduction succinct. For further background see the literature mentioned and the references therein.

Transport Maps and Couplings In what follows, we consider optimal transport between two separable and complete metric spaces $\mathcal{X}$ and $\mathcal{Y}$, equipped with Borel σ-algebras, and probability measures μ on $\mathcal{X}$ and ν on $\mathcal{Y}$. Most of the time throughout this thesis we have $\mathcal{X} = \mathcal{Y}$ and only consider Euclidean spaces. A *transport map* (or *Monge map*) is a measurable mapping $T\colon \mathcal{X} \to \mathcal{Y}$ that transports the mass of μ onto ν, that is, $T_{\#}\mu = \nu$, where $T_{\#}\mu$ denotes the *pushforward* of μ under T. In other words, for each

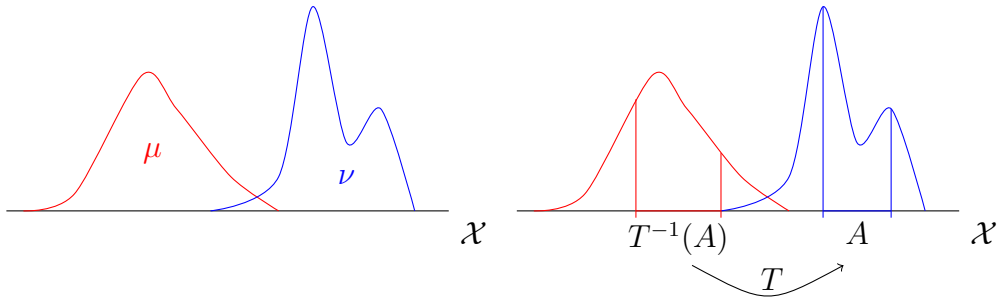


Figure 1.1: The left picture shows two measures μ and ν on $\mathcal{X}$ given as densities. The right depicts a measurable subset $A \subseteq \mathcal{X}$ and its inverse image under a transport map T. The push forward condition requires, that $\nu(A)$ and $\mu(T^{-1}(A))$ are equal for any measurable subset A.

measurable set $A \subseteq \mathcal{Y}$, $\mu(T^{-1}(A)) = \nu(A)$.

A different notion of rearrangement is the *transference plan* or *coupling* between measures. This is a measure π on the product space $\mathcal{X} \times \mathcal{Y}$, whose marginals are μ and ν on $\mathcal{X}$ and $\mathcal{Y}$, respectively, that is, $\pi(A \times \mathcal{Y}) = \mu(A)$ and $\pi(\mathcal{X} \times B) = \nu(B)$ for any measurable $A \subseteq \mathcal{X}, B \subseteq \mathcal{Y}$. This condition can be expressed in terms of pushforwards of projections: $pr_{\mathcal{X}\#}\pi = \mu$ and $pr_{\mathcal{Y}\#}\pi = \nu$, where $pr_{\mathcal{X}}$ is the projection onto $\mathcal{X}$ and $pr_{\mathcal{Y}}$ the projection onto $\mathcal{Y}$. We denote the set of all couplings between μ and ν as $\Pi(\mu, \nu)$.

In the case $\mathcal{X} = \mathcal{Y}$, if we think of measures on $\mathcal{X}$ as "configurations of mass" or "piles of sand" on $\mathcal{X}$, then a transport map T can be seen as a shift of the mass that transforms one configuration (μ) into the other (ν). This is ensured by the pushforward condition, as the amount of "sand" $\nu(A)$ in a set A after the shift has to match with the amount $\mu(T^{-1}(A))$ before the shift (see Figure 1.1). Similarly, a transference plan contains information of origins and destinations of mass relocations. For measurable sets $A, B \subseteq \mathcal{X}$, $\pi(A \times B)$ is the amount of "sand", that is picked up at A and placed at B.

It is important to note that couplings are the more general concept than transport maps, as each transport map T defines a coupling as the unique measure π_T that satisfies $\pi_T(A \times B) = \mu(A \cap T^{-1}(B))$, but not every coupling defines a transport map (see Figure 1.2). In fact, in many cases where μ is a discrete measure a transport map cannot exist (for example, if ν is absolutely

Figure 1.2: Two very simple discrete examples. The mass of μ is indicated in red, ν in blue. On the left, the transport can be expressed either as a Monge map or a coupling. In the right example, however, since the mass from one location is split among multiple destinations, this transport cannot be written as a Monge map. Since it is the only possible coupling, the Monge problem is infeasible in this simple case.

continuous with respect to Lebesgue measure in $\mathbb{R}^D$), but there always exists a coupling, since the product measure $\mu \otimes \nu$ is a trivial example in any case.

Remark. When $\mathcal{X} = \mathcal{Y}$, and in particular for discrete optimal transport, we often consider two subsets X and Y of $\mathcal{X}$ with $\text{supp}(\mu) \subseteq X$ and $\text{supp}(\nu) \subseteq Y$. It suffices to look at restrictions of transport maps T to X and of couplings π to $X \times Y$. Also, the measures μ and ν are not technically required to be probability measures - it suffices to assume finite measures with $\mu(\mathcal{X}) = \nu(\mathcal{X})$.

Optimal Transport Formulations Now that we have an idea of the concepts of transportation, we have a look at the efficiency of transportation, which is indicated by a cost function $c\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$. For two points $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, $c(x, y)$ is the cost of transporting mass from location x to location y. For optimal transport on a single metric space $(\mathcal{X}, d)$, the cost function is often $c = d^p$ for $p \geq 1$, which means transportation cost is simply a power of the distance between source and target locations. However, a close connection between cost function and distance is not necessary and many other cost functions are possible.

With a cost function c we can now define the cost of transport maps and couplings. The cost of a transport map T is the integral of the cost between

source and target locations under T with respect to μ,

$$\int_{\mathcal{X}} c(x, T(x))\, d\mu(x).$$

Trying to find the feasible transport map with the least cost is formulated in the *Monge formulation* of optimal transport (MOT), as follows:

$$\text{(MOT)} \quad \min_{T} \int_{\mathcal{X}} c(x, T(x))\, d\mu(x), \quad \text{s.t. } T_{\#}\mu = \nu$$

This is the first version of an optimal transportation problem. It is mostly considered in the context of continuous measures, since it is not feasible in many discrete cases. The problem formulation that operates on couplings, the *Kantorovich formulation*, is always feasible and since couplings are more general than transport maps, the Kantorovich formulation can be viewed as a relaxation of the Monge formulation.

The cost of a transference plan π is defined as the integral of the cost function on the product space with respect to π,

$$\int_{\mathcal{X}\times\mathcal{Y}} c(x, y)\, d\pi(x, y),$$

and consequently, the Kantorovich formulation of the optimal transport (KOT) problem is to find the feasible coupling with the least cost,

$$\text{(KOT)} \quad \min_{\pi} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y)\, d\pi(x, y) \quad \text{s.t. } \pi \in \Pi(\mu, \nu).$$

This is a (potentially infinite-dimensional) linear problem and thus generally expected to be easier than the Monge problem. It always admits an optimal solution, as long as the cost function $c\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$ is lower semi-continuous (see for example [74, Theorem 1.7]).

We mainly consider KOT throughout this work, although we generally put an emphasis on discrete measures, which allows us to restate it as a finite linear program.

Wasserstein Distance One very important aspect of optimal transport is that it gives rise to the Wasserstein distance [95]. Essentially, this concept allows us to lift the ground metric d on a complete, separable metric space $\mathcal{X}$ to a distance between probability measures on $\mathcal{X}$, which is the core reason it is useful in many theoretical contexts and applications. It has several different names, such as earth mover's distance [71], Mallows distance [55], Monge-Kantorovich-Rubinstein distance or similar ([47], [52]), depending on the field in which it is used.

Simply put, the p-Wasserstein distance between two probability measures μ and ν on a complete, separable metric space $(\mathcal{X}, d)$ for $p \geq 1$ is the p-th root of the optimal transport cost above, with respect to the cost function $c = d^p$:

$$W_p(\mu, \nu) = \left(\min_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} d^p(x, y)\, d\pi(x, y) \right)^{\frac{1}{p}}$$

W_p defines a metric on the space of probability measures over $\mathcal{X}$ with finite moment of order p, that is, the set

$$P_p(\mathcal{X}) = \left\{ \mu \in P(\mathcal{X}) \colon \int_{\mathcal{X}} d^p(x_0, x) d\mu(x) < \infty \right\},$$

where $P(\mathcal{X})$ is the set of probability measures on $\mathcal{X}$ and $x_0 \in \mathcal{X}$ is an arbitrary element [96]. A proof that W_p satisfies the metric axioms can be found for example in [96, Chapter 6]. The fact that the Wasserstein distance W_p incorporates the ground distance d on $\mathcal{X}$ is very attractive both in theoretical and practical contexts, since large distances between mass locations of μ and ν on $\mathcal{X}$ are reflected in a higher Wasserstein distance.

Discrete Optimal Transport Depending on the measures μ and ν the optimal transport problem can take different forms. Usually, one differentiates between three types of optimal transport problems:

- *Continuous optimal transport* - both measures are continuous.
- *Semi-discrete optimal transport* - one of the measures is continuous, the other one is discrete.

- *Discrete optimal transport* - both measures are discrete.

These three problem types require different, carefully tailored methods, and algorithms suited to solve one of these types usually do not transfer easily to other types. However, it is possible to reformulate a problem as a different type. For example, if both measures are continuous, one can discretize one or both to obtain the (semi-)discrete optimal transport problem and utilize (semi-)discrete methods for solving it. Similarly, discrete measures can be made continuous by interpreting a Dirac mass as uniformly distributed over a small area. Both introduce an error that is controllable, for example by the diameter of the area a Dirac mass is blurred on.

In this work, we mostly focus on algorithms for the Kantorovich formulation of finite, and hence discrete, optimal transport, although we also have a look at important methods for the semi-discrete case and the entropically regularized optimal transport. In the discrete case KOT can be written as a finite linear program. To this end, we consider measures μ and ν, which are finite sums of Diracs,

$$\mu = \sum_{i=1}^{n} \mu_i \delta_{x_i} \quad \text{and} \quad \nu = \sum_{j=1}^{m} \nu_j \delta_{y_j},$$

where $n, m \in \mathbb{N}$, all $\mu_i, \nu_j \in \mathbb{R}_+$ and $x_i, y_j \in \mathcal{X}$ for all i, j, and require that

$$\sum_{i=1}^{n} \mu_i = \sum_{j=1}^{m} \nu_j.$$

Further, we define the sets $X = \operatorname{supp}(\mu)$ and $Y = \operatorname{supp}(\nu)$ (sometimes the elements in X are called *sources* and the elements of Y are called *sinks*). Any cost function $c\colon X \times Y \to \mathbb{R}_+$ only has finitely many values and as such, we can interpret it as a *cost matrix* $C = (c_{i,j}) \in \mathbb{R}^{n \times m}$, where $c_{i,j} = c(x_i, y_j)$.

Any transference plan π is finitely supported on $X \times Y$ and we write it as $\pi_{i,j} = \pi(x_i, y_j)$ for $i = 1, \ldots, n$ and $j = 1, \ldots m$. With that the marginality condition $\pi \in \Pi(\mu, \nu)$ is simply that the rows of π written as a matrix $(\pi_{i,j})_{i,j}$ sum up to the values μ_i and the columns of π sum up to the values ν_j. The

objective function, the cost of the transference plan, is a linear function in π,

$$\sum_{x\in X}\sum_{y\in Y} c(x,y)\pi(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}\pi_{i,j}.$$

We obtain the following linear programming formulation with variables $\pi_{i,j}$:

$$\begin{aligned} \text{(DOT)} \qquad \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}\pi_{i,j} \\ \text{subject to} \quad & \sum_{j=1}^{m} \pi_{i,j} = \mu_i \;\; \forall i = 1,\dots,n \\ & \sum_{i=1}^{n} \pi_{i,j} = \nu_j \;\; \forall j = 1,\dots,m \\ & \pi_{i,j} \geq 0 \qquad \forall i,j \end{aligned}$$

This formulation is known in the linear programming literature as the transportation problem [54]. It is a special case of the minimum cost network problem and contains itself the assignment problem as the special case with $n = m$ and $\mu_i = \nu_j = 1$ for all i, j.

1.2 Applications

Recently, Wasserstein distances particularly have gained a lot of attention in many fields mainly due to the ability to meaningfully incorporate the ground distance of a space into the distance between measures. This often makes the Wasserstein distance the preferable option over different distances between measures, such as for example the total-variation distance. It also means the Wasserstein metric, and by extension optimal transport, is applied in many different contexts. We outline some of the main fields, where these concepts are applied. Again, this is by no means an exhaustive list and there are many other important applications.

Imaging This is a broad field with high importance to many scientific areas (biology, medicine and others) and contains itself a plethora of subfields and

problems. Optimal transport distances are considered in a lot of different imaging contexts, mainly as a way to gauge the difference between two images. Greyscale images, for example, are easily interpreted as discrete measures through the greyscale values of the pixels. Since the ground distance on the pixel grid is incorporated in the Wasserstein distance, it reflects well what humans perceive as similar or different. This was shown by Ruber and others [71], who successfully applied the earth mover's (Wasserstein) distance to image retrieval.

Since then, both continuous and discrete optimal transport has been applied to different imaging problems. Some are demonstrated in [63], including image interpolation along transport maps [42], adaptive color transfer [67] and color image segmentation [68]. There are applications in medical imaging [73], computer vision [34, 61], shape matching [40], image classification [99] and on more specific problems, such as the detection of phishing web pages [31], measuring plant color differences [48] or three-part image decomposition [93, 52, 17], which itself can be applied to fingerprint segmentation [94].

Probability and Statistics In probability theory, the Wasserstein metric is a useful tool for approximation, convergence of measures and more. Two select applications are perturbation for Markov chains with applications to Marcov chain Monte Carlo (MCMC) methods [72] and Poisson process approximation [79].

Mallows distance is an alternative term for the Wasserstein metric, that is often used in the context of mathematical statistics. It can give rise to theoretical techniques, for example used in proofs of central limit and similar theorems [45, 24]. However, it also appears in practical applications like goodness-of-fit testing [60, 29]. Other important concepts in the realm of optimal transport in statistics are empirical optimal transport distances [89, 92, 60] and Wasserstein barycenters [1], which are for instance used for Bayesian inference [91].

Machine Learning Artificial intelligence, deep learning, neural networks – those are fields, which have gained wide-spread traction over the last few

years. Although neural networks have been a well-known concept for over a century now and machine learning was conceptualized over fifty years ago, the hardware technology has only recently become powerful enough for machine learning techniques to bear fruit in many applications. The Wasserstein distance is considered as a loss function, for example in multi-label learning [30], since it can incorporate a ground distance on the label space in an intuitive way. It can occur in altered forms, such as smoothed Wasserstein loss used for dictionary learning [70] or optimal spectral transport for musical unmixing [26]. Generative adversarial networks (GAN) [37] are immensely popular in unsupervised learning and the Wasserstein metric has been adopted to this concept as well [3, 16].

Economics One simple example for optimal transport, that is often used to illustrate the problem, but is also considered in the economics literature, is optimal assignment [50]. It has been explored in different economic contexts such as the labor market [83] or marriage market [6, 19]. Different economic models often require adjustments in the formulation of the transportation problem. For example, introducing unobserved heterogeneity into the matching model leads to a regularized variant of optimal transport [33]. Equilibrium transport is considered in the context of adding taxes to the matching market [25]. For more in-depth information on the matter, we refer to a book by Alfred Galichon published in 2016, *Optimal Transport Methods in Economics* [32], which gives broad insight into the synergy between the two fields.

Other applications besides matching include the quantification of risk measures [69] and martingale optimal transport, which is essentially the usual optimal transport problem with the added constraint that the transport plan is a martingale. It is applied in mathematical finance, for example in robust hedging [23].

1.3 Motivation and Organization of this Work

In this section we outline how the remainder of the thesis is organized. Chapters 3 and 4 are part of the publications [78] and [90], respectively,

and the overview of these chapters we present here contains parts of the introductions of the mentioned papers.

As we have shown above, optimal transport is considered in many different fields and modern applications. Due to the massive volume of available data in many of these applications there is a high demand for increasingly efficient methods to tackle optimal transport problems on a large scale. While considerable advancement has been achieved, as many new and improved algorithms have been developed in the last few years, this is an ongoing research objective.

In Chapter 2 we give an overview of some of the most important algorithms to tackle optimal transport problems. The selection presented is not exhaustive, but rather focused on the most relevant methods in the world of modern optimal transport, which include linear programming approaches, multiscale methods and entropically regularized optimal transport, also known as the Sinkhorn method.

While many of these methods undoubtedly mean substantial progress compared to what was available a decade ago, it is nearly impossible from the current literature to figure out how various of these methods compare to one another and which method is most suitable for a given task. This is mainly due to the fact that only for a few of the modern methods user-friendly code is publicly available. What is more, many articles that introduce new methods compare their computational performance only on a restricted set of self-generated ad-hoc examples and typically demonstrate improved performance only in comparison to some classical method or to a simplified version that does not use the novelty introduced.

This is why we introduce the DOTmark in Chapter 3, which stands for *discrete optimal transport benchmark*. It is a collection of greyscale images divided into ten different classes, including various mass distributions, shapes, microscopy images, as well as classic test images used in image processing. Its main purpose is to serve as open and neutral test data for optimal transport, so that methods can be compared in a more meaningful way. Together with this data set we give a first test of a selection of algorithms on the DOTmark and report the results. See Chapter 3 for a more thorough description of the

benchmark and the results of the test.

In this test we focus on precise, singlescale algorithms and it becomes apparent that those methods are not very suitable for optimal transport solutions on high resolution images, as 128×128 pixel images are already excluded from the test due to prohibitively high runtimes. Thus, multiscale methods and approximate methods are increasingly important. In Chapter 4 we introduce a simple subsampling scheme, which proves to be a powerful tool in approximating Wasserstein and other optimal transport distances. It can use exact methods as back-end solvers, provides us with non-asymptotic approximation guarantees in the Wasserstein case and by changing the sample size it can be tuned toward higher accuracy or lower runtimes as desired. We show that this scheme performs very convincingly on instances from the DOTmark delivering an approximation with a relative error of less than 5% in about 1% of the runtime. The theoretic results imply that the approximation quality is independent of the size of the full problem in the low-dimensional case, which includes many examples, such as 2D-imaging.

In the recent past, several multiscale approaches for optimal transport have been introduced. The idea is to reduce the number of sources and sinks by aggregating them to clusters and define an instance of optimal transport on the clusters in resemblance of the original problem. The solution process on the coarsened instance is much faster and by propagating a solution to the finer scale, a good solution to the original problem can typically be obtained efficiently. The coarsening and propagation can be iterated to create multiple scales of the problem. See Chapter 2 for a more detailed explanation of the ideas behind multiscale methods. Ad-hoc tests suggest that multiscale methods typically lead to a significant speedup compared to the singlescale version of an algorithm. However, those methods rely on the ability to easily create coarser versions of the problems at hand. For example, in images we can simply lower the resolution in order to get a coarser version. Similarly, when the optimal transport instance is comprised of point clouds in a Euclidean space there are clustering methods, which can be used to aggregate points in accordance to the distances between them. But other problems, such as assignment problems, typically do not have cost matrices, which are defined

through proximity in a metric space. In these cases, the multiscale approaches mentioned above cannot be applied.

We aim to change this unfortunate circumstance by introducing *cost-based clustering* in Chapter 5. This is an approach to cluster sources and sinks into subsets in a meaningful way only based on the data we have available in any discrete optimal transport instance: the measures μ and ν and the cost matrix C. We show that any clustering provides us with a rigorous error bound on the orginal optimal transport cost after solving the problem on the coarse scale and we set up the *cost-based clustering problem* as to find the clustering with the best error bound. Unfortunately, this problem is NP-hard, thus finding the best clustering is not a feasible strategy in the context of multiscale methods for optimal transport. However, we provide several heuristics to this problem and show that clusterings of reasonable quality can be found in a reasonable runtime. This lays the foundation for the application of multiscale methods to general discrete optimal transport problem. Moreover, we show that applying these data-driven algorithms to geometric problems can yield interesting clusterings typical geometric clustering methods cannot find.

Although all necessary preliminaries for Chapters 3, 4 and 5 are introduced in the first two chapters of this thesis, the specific problem setting is still stated in each of the later three chapters. While this may lead to repetition for readers of the whole thesis, this is done to clarify the notation and setting, which is slightly different in each chapter. Furthermore, the chapters end up being more self-contained.

Chapter 2

Algorithms for Optimal Transport

Several methods to tackle optimal transport problems have been known for a long time. Many of the algorithms that are used in modern applications, however, were developed over the last decade. Gabriel Peyré and Marco Cuturi released an open book in 2018, *Computational Optimal Transport* [65], in which a majority of the relevant algorithms for optimal transport is discussed. It provides a comprehensive overview of the different types of methods and applications.

In light of this publication, we intend to keep this chapter concise and focused on a summary of the most important methods. This includes the algorithms tested in Chapter 3, multiscale methods and others, which are particularly relevant in the context of this thesis. Furthermore, we give a rather heuristic description of the methods and since their correctness is well established, we refrain from discussing this in detail.

The methods discussed in Chapter 3 were also briefly introduced in the paper about the DOTmark [78]. Parts of the descriptions in this chapter are taken from this publication.

2.1 Linear Programming

Obviously, since the Kantorovich formulation in the discrete setting is a finite linear program, standard algorithms, such as the simplex method, are applicable. However, this formulation has a very particular form as a special case of the minimum cost flow network problem, which makes it possible to use more specialized algorithms to solve it. We start by restating the program:

$$\begin{aligned} \text{(DOT)} \qquad \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}\pi_{i,j} \\ \text{subject to} \quad & \sum_{j=1}^{m} \pi_{i,j} = \mu_i \;\; \forall i = 1,\dots,n \\ & \sum_{i=1}^{n} \pi_{i,j} = \nu_j \;\; \forall j = 1,\dots,m \\ & \pi_{i,j} \geq 0 \qquad \forall i,j \end{aligned}$$

Here, $c_{i,j}$ are the elements of the cost matrix $C \in \mathbb{R}_+^{n\times m}$ and the measures μ and ν satisfy $\sum_i \mu_i = \sum_j \nu_j$ in order for the problem to be feasible. The dual formulation is given as follows:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} u_i\mu_i + \sum_{j=1}^{m} v_j\nu_j \\ \text{subject to} \quad & u_i + v_j \leq c_{i,j} \qquad \forall i,j \end{aligned}$$

The dual variables u_i and v_j are also called *potentials*. This dual formulation is utilized in most of the linear programming methods. The complementary slackness condition is

$$(c_{i,j} - u_i - v_j)\cdot \pi_{i,j} = 0 \quad \forall i,j.$$

This means, whenever we find a primal feasible solution π and a dual feasible solution (u,v) satisfying the above condition, the solutions must be optimal.

We have a look at well-established methods, such as the Hungarian method for assignment problems, the auction algorithm and the transportation simplex, as well as more modern modifications, such as the shortlist method and the shielding neighborhood method.

Hungarian Method This method goes back to Kuhn in 1955 [51], who developed it based on the results of two Hungarian mathematicians and dubbed it the Hungarian method. It is an algorithm specialized on assignment problems, that is, the special case of the transportation problem with $n = m$ and $\mu_i = \nu_j = 1$ for all i, j. Thus, one source is assigned to precisely one sink (you can think of workers assigned to jobs) in such a way that the total cost of the assignment is minimized. The algorithm is a primal-dual method at heart and more generalized primal-dual algorithms were derived from this method later. It is described in more detail for example in [5, Section 10.7], including a proof of its convergence.

The algorithm begins by defining a dual feasible solution $(\hat{u}, \hat{v})$ via

$$\hat{u}_i := \min_{j=1,\dots,n} c_{i,j} \quad \forall\, i = 1, \dots, n$$

and

$$\hat{v}_j := \min_{i=1,\dots,n} c_{i,j} - \hat{u}_i \quad \forall\, j = 1, \dots, n.$$

The reduced cost matrix $\hat{C}$ is defined by $\hat{c}_{i,j} := c_{i,j} - \hat{u}_i - \hat{v}_j$. Now, if we can find an assignment π such that each non-zero entry of π corresponds to a zero entry of $\hat{C}$, by complementary slackness this assignment is optimal.

If not, we look at the minimum number of rows and columns necessary to cover all zero entries in $\hat{C}$, fix one such cover and define the sets of uncovered rows and columns as S_r and S_c, respectively. Now we find the minimal uncovered entry in $\hat{C}$:

$$c_0 := \min_{\substack{i \in S_r \\ j \in S_c}} \hat{c}_{i,j} > 0.$$

We obtain a new feasible dual solution $(\bar{u}, \bar{v})$ by setting $\bar{u}_i = \hat{u}_i + c_0$ for $i \in S_r$, $\bar{u}_i = \hat{u}_i$ for $i \notin S_r$, $\bar{v}_j = \hat{v}_j$ for $j \in S_c$ and $\bar{v}_j = \hat{v}_j - c_0$ for $j \notin S_c$ and in turn a new reduced cost matrix $\bar{C}$. This process is iterated until the complementary

slackness conditions are satisfied and we get an optimal assignment through the zero entries of the reduced cost matrix.

Auction Algorithm This is an algorithm, which was originally introduced by Bertsekas in 1981 to solve the assignment problem [9]. It has been adapted to the general discrete optimal transport problem [11]. We describe the version for assignment problems as in [65].

We simply set $X = \{1, \dots, n\}$ here. Note that any feasible basic assignment π is the permutation matrix of a permutation $\sigma\colon X \to X$ and vice versa. With this connection we can rewrite the complementary slackness conditions as

$$u_i + v_{\sigma(i)} = c_{i,\sigma(i)} \quad \forall\, i \in X.$$

Also, if we fix the dual variables v, the best possible completion to a feasible dual solution (u, v) can be achieved by defining $u_i = \min_j c_{i,j} - v_j$ for all i. This is also called a *C^t-transform* (see [65] for details). Consequently, if we have v and σ such that

$$c_{i,\sigma(i)} - v_{\sigma(i)} = \min_j c_{i,j} - v_j \quad \forall\, i \in X,$$

then σ is an optimal assignment.

The algorithm maintains partial assignments consisting of a subset $S \subseteq X$, an injective assignment map $\xi\colon S \hookrightarrow X$, and a dual vector v. It is initialized with an empty assignment $S = \emptyset$ and $v = 0$ and iteratively adds elements to S until $S = X$. For now, we fix $\varepsilon > 0$. In one iteration of the algorithm, choose $i \in X \setminus S$ and search for the first- and second-best indices in the C^t-transform, that is,

$$j_i^{(1)} \in \operatorname*{argmin}_j c_{i,j} - v_j \quad \text{and} \quad j_i^{(2)} \in \operatorname*{argmin}_{j \neq j_i^{(1)}} c_{i,j} - v_j.$$

Then, v is updated by subtracting the value $(c_{i,j_i^{(2)}} - v_{j_i^{(2)}}) - (c_{i,j_i^{(1)}} - v_{j_i^{(1)}}) + \varepsilon$ from $v_{j_i^{(1)}}$. S and ξ are updated to reflect that i is now assigned to $j_i^{(1)}$. That means i is added to S and $\xi(i)$ is set to $j_i^{(1)}$. If $j_i^{(1)}$ was assigned before (if

there was $i' \in S$ with $\xi(i') = j_i^{(1)}$), that assignment is removed. This way, the number of assigned objects $|S|$ never decreases and one entry of v decreases by at least ε in each iteration. Further, the algorithm maintains ε-*complementary slackness* throughout the iterations [65, Proposition 3.7], that is

$$c_{i,\xi(i)} - v_{\xi(i)} \leq \varepsilon + \min_j c_{i,j} - v_j \quad \forall\, i \in S.$$

Eventually, all objects are assigned ($S = X$). For large ε bigger steps can be taken in each iteration, but smaller ε keep the assignments closer to satisfying the complementary slackness conditions. By changing ε throughout the iterations, finite convergence to an optimal assignment can be achieved. This is called ε-scaling [11].

Transportation Simplex This is a specialized version of the network simplex and described in detail for example in [54]. Like other simplex variants, the transportation simplex has two phases: one phase to construct an initial basic feasible solution π and another phase to improve this solution to optimality. Typically, the majority of time is spent in the second phase, as an initial solution to optimal transport is easily obtained.

There are quite a few different ways to construct an initial basic feasible solution. See [39] for a selection of methods suited for this task or [81] for a thorough analysis of the performance of primal and dual heuristics. The method we point out here, the modified row minimum rule, has a universally solid performance both in runtime and quality of the solution constructed. We iterate over all source locations (rows) $x_i \in X$ that still have mass left, choose for each source the available target c_j with the least cost, and include it in the solution by setting $\pi_{i,j}$ to the largest possible value. This process is repeated until all sources are depleted. The solution we obtain is automatically basic.

If we interpret the source and target locations as nodes in a graph and draw arcs for every possible transport, we get a complete bipartite graph. Every non-degenerate basic feasible solution can now be represented by a spanning tree in this graph by choosing all the arcs belonging to active transports in our solution. Given a basic feasible solution, a simplex step is performed as

follows:

- A new variable (a transport $\pi_{i,j}$) is selected to enter the basis.
- This creates a cycle in the previous tree, which is then identified.
- The maximal amount of mass possible is shifted along this cycle, that is, alternately added to and subtracted from consecutive transports.
- A variable that has become zero in the process is removed from the basis.

When searching for a new basic variable, there are again many options, but we focus on the following row minimum strategy due to performance reasons: We use the current solution to compute the values of dual variables u_i and v_j via the complementary slackness conditions that $u_i + v_j = c_{i,j}$, whenever $\pi_{i,j} > 0$. This is a linear system of $n + m - 1$ equations with $n + m$ variables, which can be solved via backwards substitution after setting $u_1 = 0$. Then we scan the non-basic variables $\pi_{i,j}$ row by row and compute the reduced costs $r_{i,j} = c_{i,j} - u_i - v_j$. If we encounter a variable with negative reduced costs, we stop at the end of that row and choose the variable with the least reduced cost encountered thus far as a new basic variable. If no candidates are found among all rows, the current solution is optimal and we stop.

The second phase of the transportation simplex is very similar to the network simplex. The only difference is that we always have a complete bipartite network in the optimal transport problem. In phase one, however, this structure is very beneficial and allows for easy construction of a feasible solution, whereas in the more general network case the introduction of artificial variables is often necessary. More details on the network simplex can also be found in [54].

Shortlist Method At its core, the shortlist method is a variant of the transportation simplex. It comes with three parameters:

- A parameter s for the shortlist length.

- Parameters p and k that control how many variables are searched to find a new basic variable.

Before optimization starts, a shortlist is created for every source, consisting of the s targets with least transport costs, ordered by cost. The basic feasible solution is again constructed by a modified row minimum rule, where the shortlists are prioritized. After that, the solution is improved by simplex steps similar to the transportation simplex, but the search is limited to the shortlists. The lists are scanned, until either k variables with negative reduced costs are found, or p percent of the shortlists have been searched. Then the candidate with least reduced cost is chosen to enter the basis. If no improvement can be achieved within the shortlists, the last solution is improved to global optimality by the same simplex steps as in the transportation simplex. For further details see [39].

Shielding Neighborhood Method The shielding neighborhood method, or shortcut method, was introduced by Schmitzer in [76]. Its main idea is to solve a sequence of sparse (i.e. restricted) optimal transport instances instead of the dense (full) problem. The algorithm is proposed as a multiscale method, but we focus on the singlescale variant, which basically works in the same way:

A *neighborhood* is a small subset $N \subseteq X \times Y$ of the product space and imposes a restricted instance of the problem by only considering transport variables π_{ij} such that $(x_i, y_j) \in N$. Due to the significant reduction in the amount of variables, this instance is much faster to solve.

The idea behind shielding neighborhoods is the so-called *shielding condition*, which ensures that for $(x_i, y_j) \notin N$ a *shortcut* exists. For a given coupling π with $\operatorname{supp}(\pi) \subseteq N$ a shortcut for a pair $(x_1, y_n) \in X \times Y$ is a sequence of active transports, $((x_2, y_2), \ldots, (x_{n-1}, y_{n-1}))$ in $\operatorname{supp}(\pi)$ with $(x_i, y_{i+1}) \in N$ for all $i = 1, \ldots, n-1$, such that

$$c(x_1, y_n) \geq c(x_1, y_2) + \sum_{i=2}^{n-1} \left(c(x_i, y_{i+1}) - c(x_i, y_i) \right).$$

This ensures that the dual constraint corresponding to (x_1, y_n) is satisfied [76, Proposition 3.2].

The shortcuts are not constructed explicitly. Rather, their existence follows from the shielding condition between $x_A \in X$ and $y_B \in Y$: $(x_s, y_s) \in \text{supp}(\pi)$ *shields* x_A from y_B if

$$c(x_A, y_B) - c(x_s, y_B) > c(x_A, y_s) - c(x_s, y_s).$$

A subset $N \subseteq X \times Y$ is a shielding neighborhood for π if $\text{supp}(\pi) \subseteq N$ and every pair $(x, y) \in X \times Y$ is either included in N or there is an active transport $(x_s, y_s) \in \text{supp}(\pi)$ with $(x_A, y_s) \in N$, which shields x from y. The main result exploited by the algorithm is that if π is an optimal coupling with respect to N and N is a shielding neighborhood for π, then π is optimal for the original instance [76, Corollary 3.10].

Starting with a basic feasible solution generated with the modified row minimum rule a shielding neighborhood for that solution is constructed as described in [76]. The algorithm then alternates between optimizing the sparse instance of the problem and generating a new shielding neighborhood for the current solution. If a solution is optimal for two successive shielding neighborhoods, it is globally optimal and the algorithm stops.

2.2 AHA Method

The acronym AHA stands for Aurenhammer, Hoffmann and Aronov, who showed in their seminal paper [4] that the transport problem with squared Euclidean cost is equivalent to an unrestricted continuous minimization problem for a certain convex objective function Φ. The algorithm exploiting this fact is introduced in [57] as a multiscale method, but as in the case of the shielding neighborhood method the singlescale variant works essentially in the same way.

This method applies to the semi-discrete case in $\mathbb{R}^D$ with the squared Euclidean distance $d(x, y) = \|x - y\|^2$. We assume that μ is absolutely continuous with respect to Lebesgue measure and ν is a finite sum of Diracs,

$\nu = \sum_{j=1}^{n} \nu_j \delta_{y_j}$. The key observation utilized by the AHA method is that any power diagram (a.k.a. Laguerre tessellation) governed by the support points $y_1, \ldots, y_n$ of ν and arbitrary weights $w_1, \ldots, w_n \in \mathbb{R}$ characterizes an optimal transport plan from μ to *some* measure living on these support points. By minimizing the function Φ in the weights $w_1, \ldots, w_n$ we can find a power diagram that defines an optimal transport to *the correct* measure ν.

The power diagram is defined as the following decomposition of $\mathbb{R}^D$: Each point y_j has a weight w_j. The cell belonging to y_j with respect to the weight vector $w \in \mathbb{R}^n$ is the set

$$\mathrm{Pow}_j(w) = \left\{ x \in \mathbb{R}^D \colon \|x - y_j\|^2 - w_j \leq \|x - y_i\|^2 - w_i \text{ for all } i = 1, \ldots, n \right\}.$$

The intersection of two adjacent cells is a part of the hyperplane, where the above holds with equality, thus a null set with respect to μ (see Figure 2.1 for a two-dimensional example). Geometrically, the intersection hyperplanes can be roughly characterized as follows: If we draw spheres around the points y_j, whose radii are the square roots of the weights w_j, the hyperplane between two cells contains the intersection of the spheres if it is nonempty. If the two spheres touch, the hyperplane is the common tangent plane at the intersection point. If the two spheres are disjoint, the hyperplane is the set of points in $\mathbb{R}^D$, whose distances to the points of tangency of the two spheres are equal. Figure 2.1 shows the circles around the points.

The transport map T_w, that maps an element $x \in \mathbb{R}^D$ to the point y_j such that $x \in \mathrm{Pow}_j(w)$ is well-defined μ-almost everywhere. It is a known result that T_w is an optimal transport map between μ and $T_{w\#}\mu$ [57]. Therefore, we need to find a weight vector w, so that $T_{w\#}\mu = \nu$. As it turns out [57, Theorem 2], a weight vector w satisfies $T_{w\#}\mu = \nu$ if and only if it is a global minimizer of the convex function

$$\Phi(w) = \sum_{j=1}^{n} \left(\nu_j \cdot w_j - \int_{\mathrm{Pow}_j(w)} \left(\|x - y_j\|^2 - w_j \right) d\mu(x) \right).$$

Evaluating Φ at a given weight vector $w \in \mathbb{R}^n$ involves computation of the power diagram for w and a rather simple integration procedure over each

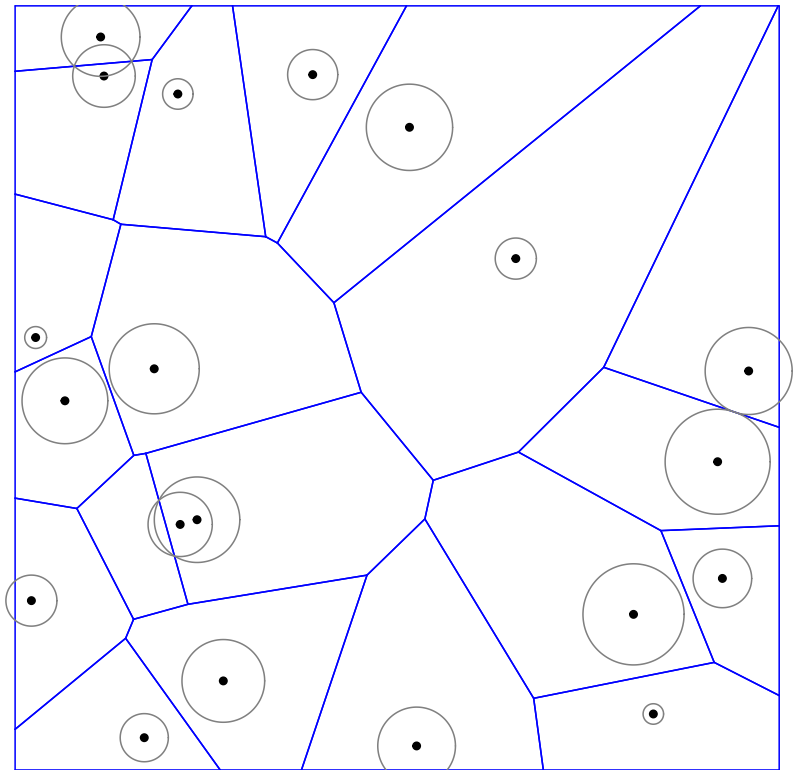


Figure 2.1: Example of a power diagram in $\mathbb{R}^2$. The black dots are the twenty support points y_j of ν. The radii of the grey circles around the points are the square roots of the weights w_j. The cells $\mathrm{Pow}_j(w)$ are constrained by the blue line segments. Each power diagram defines a Monge map μ-almost everywhere as the piecewise constant map, which maps the interior of each cell $\mathrm{Pow}_j(w)$ onto y_j. This example also shows that the points y_j are not necessarily contained in their own cells.

power cell. The gradient of Φ is accessible, and its i-th component is in fact just the difference between ν_i and the μ-mass transported to this point under the current power diagram. In [57] it is shown that using the L-BFGS-B algorithm with Moré–Thuente type line search, a quasi-Newton method in which the inverse Hessian is estimated, outperforms gradient descent methods. Second order algorithms such as the Newton method are problematic, since they require exact computation of the Hessian. A damped Newton algorithm was recently proposed and analyzed [49]. Moreover, the method can be adapted to the case of the non-squared Euclidean distance ($p = 1$) [41]. The main challenge here is that the boundaries of the cells are no longer hyperplanes, which makes the integration over the power cells, and thus the evaluation of the function Φ, more expensive.

2.3 Entropically Regularized Optimal Transport

Although the idea to regularize optimal transport problems by adding an entropic penalty term goes back to 1969 [97], only recently has this apporach gained massive traction in computational optimal transport, as it was discovered by Cuturi in 2013 that entropically regularized optimal transport problems can be very efficiently solved by a simple matrix scaling algorithm [20]. This algorithm is somtimes called Sinkhorn scaling or Sinkhorn-Knopp-Algorithm and was introduced as early as 1967 [86]. It is one of the key reasons why the entropy is used as a regularization term instead of other options, and why regularized optimal transport is covered as an algorithm in this chapter. We loosely follow the explanations given in [65], but keep them more concise.

As usual, we cover the discrete case of the Kantorovich problem. For a coupling π we consider the entropic regularization term

$$H(\pi) := -\sum_{i,j} \pi_{i,j} \left(\log(\pi_{i,j}) - 1\right).$$

For a regularization parameter $\lambda > 0$ the entropically regularized optimal transport problem is

$$\min_{\pi \in \Pi(\mu,\nu)} \sum_{i,j} c_{i,j}\pi_{i,j} - \lambda H(\pi).$$

Since the entropy is a strictly concave function, this results in the regularized transport problem being a strictly convex problem. Consequently, it admits a unique solution π^*_λ for any $\lambda > 0$. This solution always has full support, in sharp contrast to the basic solutions to the original problem obtained through linear programming, which are sparse.

If λ is small, the regularized problem is close to the original, while the convexity is stronger for large λ (in fact, the objective function is λ-strongly convex). This is reflected in the limiting behaviour of π^*_λ, since

$$\lim_{\lambda \to 0} \pi^*_\lambda = \pi^*,$$

where π^* is the optimal solution of the original transport problem with the highest entropy, and

$$\lim_{\lambda \to \infty} \pi^*_\lambda = \mu \otimes \nu,$$

since the product measure is the coupling that maximizes the entropy.

In the Wasserstein case $c = d^p$, we can view the results of the regularized problem as a slightly different optimal transport distance

$$W_{p,\lambda}(\mu,\nu) = \left(\min_{\pi \in \Pi(\mu,\nu)} \sum_{i,j} d^p(x_i, y_j)\pi_{i,j} - \lambda H(\pi) \right)^{\frac{1}{p}},$$

which can be seen as an approximation to the Wasserstein distance, especially for small λ, since $W_{p,\lambda}(\mu,\nu) \to W_p(\mu,\nu)$ as $\lambda \to 0$.

The Sinkhorn Scaling Algorithm For a fixed value $\lambda > 0$, the *Gibbs kernel* associated to the cost matrix C is $K \in \mathbb{R}^{n \times m}$, where

$$K_{i,j} = e^{-\frac{c_{i,j}}{\lambda}}, \quad i = 1, \ldots, n,\ j = 1, \ldots, m.$$

The main fact, that is exploited by the algorithm is that the unique solution π^*_λ has the form $\pi^*_{\lambda,i,j} = u_i K_{i,j} v_j$ with vectors $u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$ akin to dual vectors in the linear program [65, Proposition 4.3]. Since π^*_λ has to be a feasible coupling, we need to find vectors u and v, such that $\pi^*_\lambda \in \Pi(\mu, \nu)$, that is, u and v satisfy the constraints

$$\operatorname{diag}(u) K v = \mu \quad \text{and} \quad \operatorname{diag}(v) K^t u = \nu,$$

where μ and ν are interpreted as mass vectors and $\operatorname{diag}(u)$ is the n by n matrix, which has the entries of u on the diagonal and zero elsewhere.

These are nonlinear constraints in u and v. The general strategy to satisfy these constraints is to start with an arbitrary positive vector (e.g. $v^{(0)} = (1, \ldots, 1)^t$) and update u and v alternatingly to satisfy the constraints. This is achieved by the iterations

$$u^{(k+1)} := \frac{\mu}{K v^{(k)}} \quad \text{and} \quad v^{(k+1)} := \frac{\nu}{K^t u^{(k+1)}},$$

where the division is performed entry-wise. At its core, this is an alternating projections method: The current solution is projected alternatingly onto the sets of solutions (u, v) satisfying the source constraints $\operatorname{diag}(u) K v = \mu$ and the sink constraints $\operatorname{diag}(v) K^t u = \nu$. The iterates of transportation matrices $\pi^{(k)}_\lambda = \operatorname{diag}(u^{(k)}) K \operatorname{diag}(v^{(k)})$ converge to the unique optimal solution π^*_λ as $k \to \infty$ [65, Theorem 4.2].

This algorithm has a very good runtime performance in practice and is thus used in many applications where large-scale computation or approximation of optimal transport distances is necessary, for example in computing Wasserstein barycenters for shape interpolation [87]. However, it has its own issues, in particular with numerical stability. As mentioned before, the approximation of the Wasserstein distance through the regularized version is better the smaller the regularization parameter λ. There are two problems with choosing a small λ in this algorithm:

- A small λ leads to a problem with weaker convexity. Consequently, the convergence of $\pi^{(k)}_\lambda \to \pi^*_\lambda$ is slower.

- Entries of the Gibbs kernel K approach zero quickly for small λ. This leads to numerically unstable iterations of the matrix scaling, which is especially problematic when K approaches or falls below the machine precision threshold. This means λ cannot be chosen arbitrarily small.

There are, however, workarounds for these problems, such as λ-scaling or log-domain stabilization [77]. Another very recently proposed variant of Sinkhorn scaling only updates one greedily selected row or column in each step and is consequently called Greenkhorn [2].

2.4 Multiscale Methods

Multiscale approaches to tackle optimal transport problems go back at least to 2010 for the semi-discrete transportation problem [15]. Independently, Mérigot integrated a mulstiscale scheme into the AHA method in 2011 [57]. There, the discrete measure ν is coarsened in order to decrease the dimension of the convex optimization problem that needs to be solved. For discrete optimal transport different methods have been proposed by Oberman and Ruan in 2015 [62] and later by Schmitzer [76], as well as Gerber and Maggioni [35], among others. We describe the general ideas behind the multiscale approach with the discrete case in mind and summarize the concepts presented in the aforementioned papers.

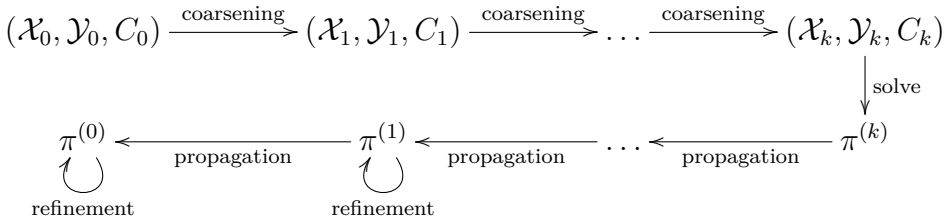
Consider the (Kantorovich) discrete optimal transport setting with two measures μ and ν with finite supports $X = \text{supp}(\mu)$, $Y = \text{supp}(\nu)$ and a cost function $c\colon X \times Y \to \mathbb{R}_+$ written as a cost matrix C. Essentially, multiscale methods can be divided into three steps:

- *Coarsening*: From the original instance, a finite sequence of successively coarser instances is created. This requires the construction of hierarchical partitions for the sets X and Y and the definition of a cost function (matrix) on each scale.

- *Propagation*: Assuming we have a feasible transport plan $\pi^{(i)}$ on a certain scale i, propagation methods construct a feasible plan $\pi^{(i-1)}$ on the finer scale $i-1$.

- *Refinement*: A transport plan is iteratively improved towards optimality. Refinement strategies usually operate on a single scale.

The three terms are chosen according to [35]. Multiscale methods as a whole work roughly as depicted in this diagram, where $(\mathcal{X}_i, \mathcal{Y}_i, C_i)$ denotes an optimal transport instance on scale i and $\pi^{(i)}$ is a coupling for that instance:



We now discuss the three processes in more detail.

Coarsening There are two constructions that have to be taken care of for the coarsening of an optimal transport instance: hierarchical partitions for X and Y and the construction of a cost matrix for each scale. If we fix the number k of scales, a hierarchical partition for $X = \{x_1, \dots, x_n\}$ is a sequence $\mathcal{X}_0, \mathcal{X}_1, \dots, \mathcal{X}_k$ of systems of subsets of X, such that

i) $\mathcal{X}_0 = \{\{x_1\}, \{x_2\}, \dots, \{x_n\}\}$.

ii) $\mathcal{X}_i$ is a partition of X for all $i = 0, \dots, k$.

iii) For each $i = 0, \dots, k-1$ and each subset $S \in \mathcal{X}_i$, there is a set $T \in \mathcal{X}_{i+1}$, such that $S \subseteq T$.

These conditions ensure that our partitions of X are nested in such a way that they become successively coarser. While it is not technically required that the partitions get strictly coarser, as $\mathcal{X}_i = \mathcal{X}_0$ for all i would satisfy the above conditions, identical successive partitions are pointless in practice.

Given a measure $\mu = \sum_{i=1}^n \mu_i \delta_{x_i}$ on X a hierarchical partition $\mathcal{X}_0, \dots, \mathcal{X}_k$ defines a sequence of measures $\mu^{(i)}$ in a natural way: For $i = 0, \dots, k$, we define

$$\mu^{(i)} = \sum_{S \in \mathcal{X}_i} \mu_S^{(i)} \delta_S, \quad \text{where} \quad \mu_S^{(i)} = \mu(S) = \sum_{\substack{i=1 \\ x_i \in S}}^{n} \mu_i.$$

This means, we simply sum up the masses of the elements in each subset of X.

Constructing a cost matrix C_i on scale i is not as straight forward as constructing the measures. Depending on the surrounding space of X and Y and the original cost matrix different options are available:

- If X and Y are supported on regular grids in $\mathbb{R}^D$ with $c(x, y) = \|x-y\|^p$, a simple coarsening strategy is to aggregate 2^D adjoining points to one. When operating on two-dimensional images, for example, this means decreasing the resolution from $N \times M$ to $N/2 \times M/2$ pixels. The new cost matrix is defined by the values $\|\bar{x} - \bar{y}\|^p$, where $\bar{x}$ and $\bar{y}$ are the respective center points of the 2^D grid points.

- If $X, Y \subseteq \mathbb{R}^D$ are point clouds with $c(x, y) = \|x - y\|^p$, the cost matrix between two subsets can be defined as $\|\bar{x} - \bar{y}\|$, where $\bar{x}$ and $\bar{y}$ are (weighted) medians or centers of their respective subsets.

- Generally, for subsets $S \subseteq X$ and $T \subseteq Y$ one can define the cost between S and T as the cost between two selected representatives, that is, $c(\bar{x}, \bar{y})$ with $\bar{x} \in S$ and $\bar{y} \in T$.

- Alternatively, the cost between S and T can be set as the minimum, median, center or mean of the set $\{c(x, y) \colon x \in S, y \in T\} \subseteq \mathbb{R}_+$.

See [76] or [35] for more options and details.

Given partitions $\mathcal{X}_i, \mathcal{Y}_i$ and a cost matrix C_i on the subsets, together with the natural measures $\mu^{(i)}$ and $\nu^{(i)}$ this defines the optimal transport instance on scale i:

$$\begin{aligned}
\min \quad & \sum_{S \in \mathcal{X}_i} \sum_{T \in \mathcal{Y}_i} C_i(S, T) \cdot \pi^{(i)}_{S,T} \\
\text{subject to} \quad & \sum_{T \in \mathcal{Y}_i} \pi^{(i)}_{S,T} = \mu^{(i)}_S && \forall S \in \mathcal{X}_i \\
& \sum_{S \in \mathcal{X}_i} \pi^{(i)}_{S,T} = \nu^{(i)}_T && \forall T \in \mathcal{Y}_i \\
& \pi^{(i)}_{S,T} \geq 0 && \forall S \in \mathcal{X}_i, T \in \mathcal{Y}_i
\end{aligned}$$

The number of variables in this instance is $|\mathcal{X}_i||\mathcal{Y}_i|$ instead of nm. Therefore, we obtain a sequence of successively coarser instances with fewer and fewer variables.

Propagation The way a coupling $\pi^{(i+1)}$ is propagated to $\pi^{(i)}$ on scale i again depends on the type of problem. Sometimes, it is not the coupling itself, that is propagated, but rather a *neighborhood* N. This is a subset of the set of variables (or equivalently, a subset of $\mathcal{X}_i \times \mathcal{Y}_i$) with the goal that an optimal coupling for the finer scale i is likely supported on this neighborhood. From a neighborhood N a coupling supported on N can be constructed, for example via a network simplex method.

We describe several options from the aforementioned papers, based on neighborhoods or otherwise:

- A simple neighborhood propagation method is to define a neighborhood on scale i through the support of the coupling $\pi^{(i+1)}$ on scale $i+1$ via

$$N := \left\{(S,T) \in \mathcal{X}_i \times \mathcal{Y}_i \colon \exists (S',T') \in \operatorname{supp}(\pi^{(i+1)}) \text{ with } S \subseteq S', T \subseteq T'\right\}.$$

- *Spatial neighborhood growth* [62]: a similar method on grids. The support of $\pi^{(i+1)}$ is extended by the four adjacent grid points in $\mathcal{X}_i$ and $\mathcal{Y}_i$. Then, the neighborhood is propagated as above.

- *Shielding neighborhood propagation* [76]: The shielding method maintains a shielding neighborhood on each scale and this neighborhood is propagated as above.

- *Capacity constraint propagation* [35]: By adding randomized capacity constraints on the active transports of a coupling $\pi^{(i+1)}$ a larger support is forced, including also the *second-best* transportation arcs. This support is propagated as above.

- We introduce another propagation technique in Section 5.3.3, which directly propagates a coupling $\pi^{(i+1)}$ to a coupling $\pi^{(i)}$ on scale i.

Refinement Refinement strategies mostly employ iterations of optimal transport methods, such as iterations of the transportation simplex, in order to improve the coupling $\pi^{(i)}$. If the propagation was done via a neighborhood, this neighborhood can also be used to restrict the number of variables in the refinement process. Another strategy is *potential refinement* [35], where the potentials (dual solution) of a coupling are improved. If the transport problem on scale i is an assignment problem, then the primal-dual iterations of the Hungarian method can be used as a refinement strategy.

Chapter 3

DOTmark: A Benchmark for Discrete Optimal Transport

This chapter is largely identical to the paper *DOTmark - A Benchmark for Discrete Optimal Transport* [78], which is a collaboration with Dominic Schuhmacher and Carsten Gottschlich. Its purpose is twofold. First, we propose a collection of real and simulated images, the DOTmark, that is designed to span a wide range of different mass distributions and serves as a benchmark for testing optimal transport algorithms. The data can be downloaded at `www.stochastik.math.uni-goettingen.de/DOTmark/`. We invite other researchers to use this benchmark, report their results, and thus help building a more transparent picture of the suitability of different methods for various tasks.

The second purpose is to provide a survey and a performance test based on the DOTmark for a cross section of established methods. Since not much code is freely available, we have used previous implementations of our own (done to the best of our knowledge) of various methods, added an implementation of the recently proposed shielding neighborhood method [76] and let them compete against each other. This also allows us to draw conclusions about the behavior of different methods on different types of input data. In order to make this comparison as meaningful as possible, we restricted ourselves to using only singlescale methods and the squared Euclidean distance as a cost

function. We hope this comparison will provide a first spark for a healthy competition of various methods in the public discussion.

3.1 Brief Theoretical Background

For the present context it is sufficient to restrict ourselves to optimal transport on $\mathbb{R}^d$. Let X, Y be subsets of $\mathbb{R}^d$ and let μ and ν be probability measures on X and Y, respectively. In this paper we will always have $X = Y$, but using different notation for domain and target space makes definitions easier to grasp.

A *transport map* T is any (measurable) map $X \to Y$ that transforms the measure μ into the measure ν. More precisely it satisfies $\mu(T^{-1}(B)) = \nu(B)$ for every measurable $B \subset Y$. A *transference plan* is a measure π on $X \times Y$ with marginals $\pi(\cdot \times Y) = \mu$ and $\pi(X \times \cdot) = \nu$. The set of transference plans from μ to ν is denoted by $\Pi(\mu, \nu)$. Any transport map T from μ to ν defines a transference plan π_T from μ to ν as the unique measure satisfying $\pi_T(A \times B) = \mu(A \cap T^{-1}(B))$ for all measurable $A \subset X$ and $B \subset Y$. Not every transference plan π can be represented in this way, because transference plans allow mass from one site $x \in X$ to be split between multiple destinations, which is not possible under a transport map. Figure 3.1 shows such an example.

We assume that the cost of transporting a unit mass from $x \in X$ to $y \in Y$ is $c_p(x, y) = \|x - y\|^p$ for some $p \geq 1$. The minimum cost for transferring μ to ν is then given by

$$C_p(\mu, \nu) = \min_{\pi \in \Pi(\mu,\nu)} \int_{X \times Y} \|x - y\|^p \, d\pi(x, y). \tag{3.1}$$

Taking the p-th root, we obtain the *Wasserstein metric* W_p. More precisely we have

$$W_p(\mu, \nu) = C_p(\mu, \nu)^{1/p}$$

for any measures μ and ν that satisfy $\int_X \|x\|^p \, d\mu(x) < \infty$ and $\int_Y \|y\|^p \, d\nu(y) < \infty$. In order to evaluate the Wasserstein metric, we need to find an optimal

solution to (3.1), i.e., a minimizing transference plan π. This problem is often referred to as the Kantorovich formulation of optimal transport. Note that by Theorem 4.1 in [96] a minimizing π always exists. However, it neither has to be unique nor representable in terms of an optimal transport map.

Often one would like to compare data sets that are available as images from a certain source, e.g. real photography, astronomical imagery, or microscopy data. We may think of such images as discrete measures on a grid. For example, the first two panels in Figure 3.1 show tiny clippings from STED microscopy images of mitochondrial networks. A question of interest might be whether both images stem from the same part of the network, which can in principle be answered by finding an optimal transference plan (third panel in Figure 3.1) and computing the Wasserstein distance. Note that this coarse resolution is not representative for a serious analysis, but was only chosen for illustrative purposes.

Even if the measures we would like to consider are more general probability measures, we can always approximate them (weakly) by a discrete probability measure, e.g. by considering the empirical distribution of a sample from the general measure or based on a more sophisticated quantization scheme. Lemma 8.3 in [12] characterizes when optimal costs are approximated in this way (e.g. always if X and Y are compact). Theorem 5.20 in [96] and the subsequent discussion give sufficient conditions about the approximation of optimal transference plans.

Assume now that we have discrete measures of the form $\mu = \sum_{i=1}^{m} \mu_i \delta_{x_i}$ and $\nu = \sum_{j=1}^{n} \nu_j \delta_{y_j}$ and write $c_{ij} = \|x_i - y_j\|^p$. In what follows, we always have $m = n$, and $(x_i)_{1 \leq i \leq m} = (y_j)_{1 \leq j \leq n}$ form a regular square grid in $\mathbb{R}^2$, but since it is more intuitive, we keep different notation for source locations and target locations. Let π_{ij} be the amount of mass transported from x_i to y_j.

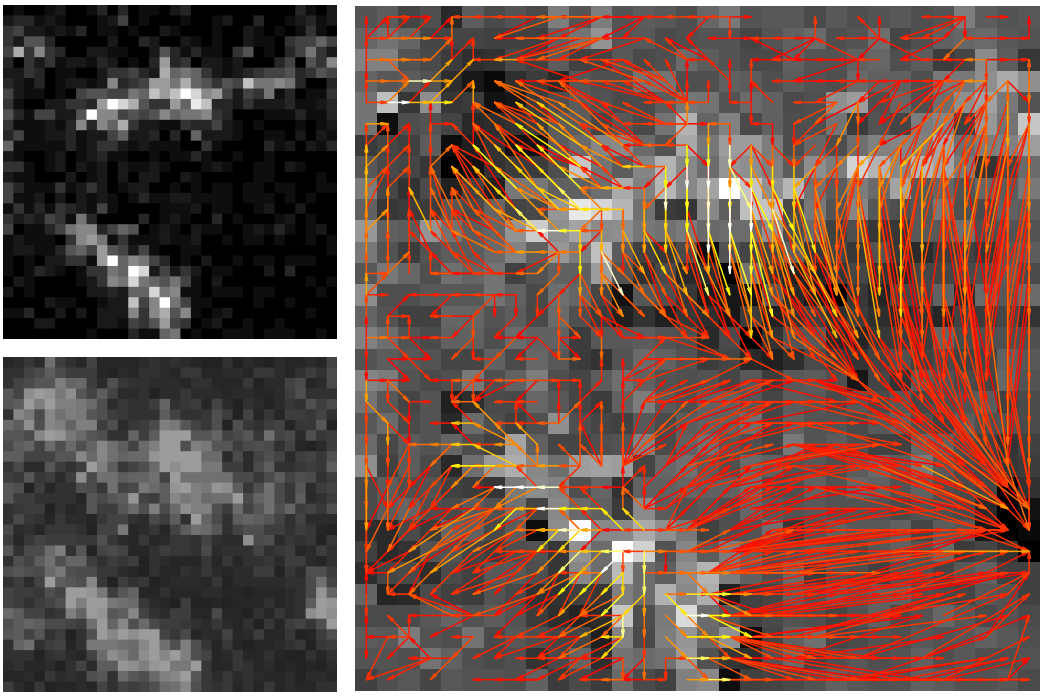


Figure 3.1: *Left panels:* Two tiny clippings A (top) and B (bottom) from STED microscopy images of mitochondrial networks. *Right panel:* The difference $A - B$ of the first two panels with an optimal transference plan for $p = 2$ superimposed. Arrows show from where to where mass is transported in the optimal transference plan. The colors indicate the amount of mass from dark red (small) to bright yellow (large). Since mass from individual sites is split (indicated by several arrows leaving the site), this transference plan cannot be represented by a transport map.

Then, the problem (3.1) can be rewritten as a linear program:

$$\begin{aligned} \text{(DOT)} \qquad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\pi_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} \pi_{ij} = \mu_i \ \forall i = 1, \dots, m \\ & \sum_{i=1}^{m} \pi_{ij} = \nu_j \ \forall j = 1, \dots, n \\ & \pi_{ij} \geq 0 \end{aligned}$$

This is the classic transportation problem from linear programming. Efficient

ways of solving this problem for small to medium sized (m and) n have been known since the middle of the last century. However, in the context of modern optimal transport problems it has become necessary to solve such problems efficiently at a scale where (m and) n are many thousands or even tens of thousands and more. Currently, this cannot be done with the classical algorithms and requires utilizing the geometry of the problem in one way or the other.

3.2 Benchmark

Our philosophy in compiling this benchmark was to represent a wide range of theoretically different structures, while incorporating typical images that are used in praxis and/or have been used for previous performance tests in the literature. We refer to it as DOTmark, where DOT stands for discrete optimal transport.

The benchmark consists of 10 classes of 10 different images (in what follows sometimes called mass distributions or measures), each of which is available at the 5 different resolutions from 32×32 to 512×512 (in doubling steps per dimension). This allows for a total of 45 computations of Wasserstein distances between two images for any one class at any fixed resolution. Table 3.1 gives an overview of how the classes were created. Classes 1–7 are random simulations of scenarios based on various probability distributions. Images at different resolutions are generated independently from each other but according to the same laws. Classes 8–10 were obtained by ad-hoc choices of simple geometric shapes, classic test images and images of mitochondria acquired using STED super-resolution microscopy [43, 44, 98]. For geometric shapes and classic test images the various resolutions available are coarsenings of a single image. For the microscopy images different clippings of various sizes have been selected from larger images to obtain the various resolutions.

We shifted and scaled the pixel values for all classes and randomly re-distributed a small percentage of the mass in order to achieve non-negative integer values at each pixel with an average of 10^5. We chose integer values to make the benchmark (directly) accessible to a wide range of algorithms

#	Name	Description
1	WhiteNoise	i.i.d. uniformly distributed values in $[0, 1]$ at each pixel
2	GRFrough	GRF with $\sigma^2 = 1$, $\nu = 0.25$, $\gamma = 0.05$
3	GRFmoderate	GRF with $\sigma^2 = 1$, $\nu = 1$, $\gamma = 0.15$
4	GRFsmooth	GRF with $\sigma^2 = 1$, $\nu = 2.5$, $\gamma = 0.3$
5	LogGRF	exp-function of a GRF with $\sigma^2 = 1$, $\nu = 0.5$, $\gamma = 0.4$
6	LogitGRF	Logistic function of a GRF with $\sigma^2 = 4$, $\nu = 4.5$, $\gamma = 0.1$
7	CauchyDensity	Bivariate Cauchy density with random center and a varying scale ellipse
8	Shapes	An ad-hoc choice of simple geometric shapes
9	ClassicImages	Standard greyscale test images used in image processing
10	Microscopy	Clippings from STED microscopy images of mitochondria

Table 3.1: The 10 classes in the DOTmark with details about their creation. GRF stands for Gaussian random field. For technical details and the meaning of the parameters see text.

and to be able to verify correctness of the optimal transport cost precisely, at least in the case $p = 2$, where integer transportation costs between grid points may be assumed.

Figure 3.2 shows the first image of each of the classes 1–6 along with average histogram over all members of the class. Figure 3.3 shows the complete collection of images in classes 7–10.

We provide some more details on how classes 2–6 are generated. The images are simulated from stationary centered Gaussian random fields (GRF) on $[0, 1]^2$ with Matérn covariance function $k := k_{\sigma^2,\nu,\gamma} : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$,

$$k_{\sigma^2,\nu,\gamma}(x, y) = \sigma^2 \frac{2^{\nu-1}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|x - y\|}{\gamma} \right)^{\nu} K_\nu \Big(\sqrt{2\nu} \|x - y\| / \gamma \Big),$$

where K_ν is the modified Bessel function of the second kind of order ν. In brief this means that the pixel values are distributed according to a

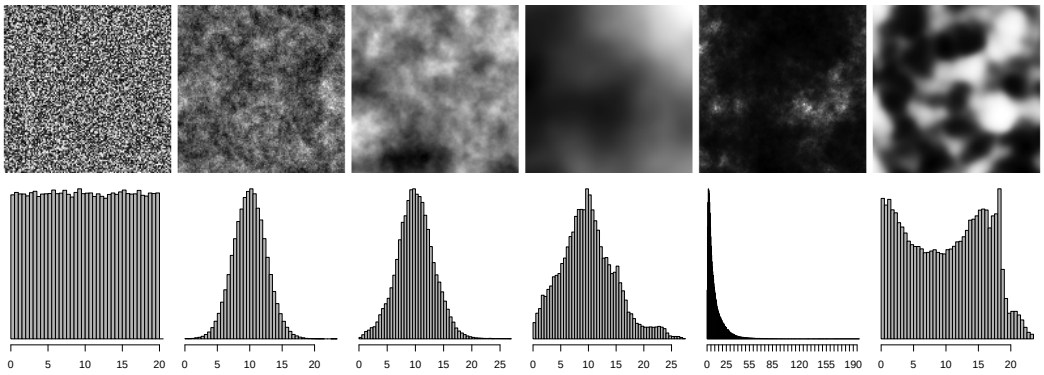


Figure 3.2: *Top row:* One representative at resolution 128×128 for each of the completely randomly generated classes 1–6. *Bottom row:* Average histograms of all images at resolution 128×128 in classes 1–6. A regular bin width of 5000 was used. The annotated pixel values are in multiples of 10^4.

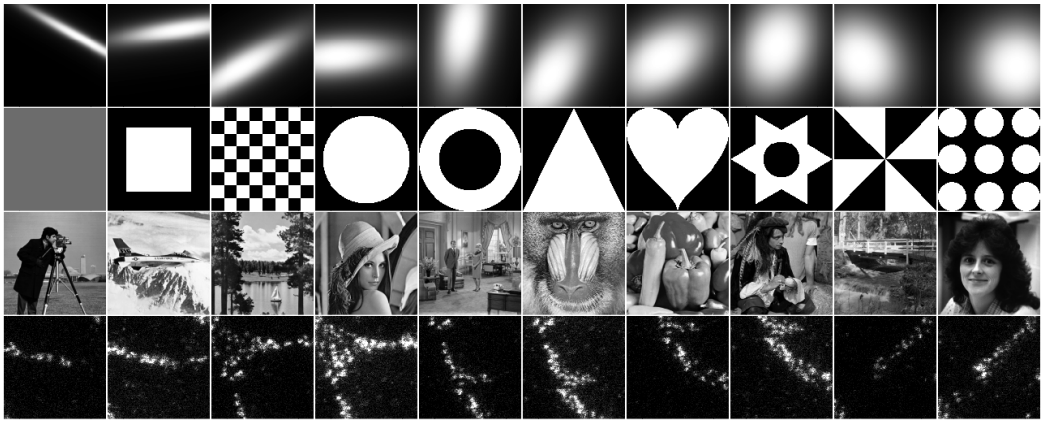


Figure 3.3: The images in the classes 7–10 at resolution 128×128.

multivariate normal distribution with mean vector zero and covariance matrix $(k(x_i, x_j))_{1 \le i,j \le m}$, where x_i, $1 \le i \le m$, is an enumeration of the pixel centers. The Matérn covariance function is a popular choice in spatial statistics. Its significance comes from the fact that in addition to having parameters $\sigma^2 > 0$ for the variance and $\gamma > 0$ for the range of the covariance, it also has a parameter $\nu > 0$ that allows to control the regularity of the random image created from very rough (ν small) to very smooth (ν large). Accordingly, classes 2–4 go from very rough with short range dependence to quite smooth with long range dependence. Class 5 is rough with long range dependence, which is hard to see from Figure 3.2 because of the exponential function

applied to the pixel values. Class 6 is very smooth with medium range dependence and the logistic function $\psi : \mathbb{R} \to [0,1]$, $\psi(x) = e^x \big/ (1 + e^x)$ was applied to the pixel values. See [36] for more theoretical details about Gaussian random fields. Our simulations were performed using the R package `RandomFields`; see [75] and [66].

Histograms 4 and 6 deviate quite a bit from the theoretical histograms expected due to the rescaling and redistribution of mass that we apply in order to obtain mass distributions that are non-negative, integer-valued, and have an average of 10^5. Also, due to long range dependence and smoothness, histogram 4 is based on a sample of much smaller effective size from the normal distribution than histograms 2 and 3.

On the whole we consider this a reasonable and versatile benchmark for many (planar and typically grid-based) optimal transport algorithms. It covers a wide selection of types of mass distributions whose comparisons are useful for theoretical or practical investigations. Similar types have been considered in the literature before. Gottschlich and Schuhmacher [39] have considered a sparse version of the WhiteNoise class. Schmitzer [76] considered sums of randomly scaled and positioned Gaussian densities (sometimes filtered by discontinuous masks), which is a somewhat different type of random function generation along the lines of our classes 2–7. Further random or deterministic functions and geometric shapes were considered by Benamou, Carlier, Cuturi, Froese, Nenna, Oberman, Peyré, Ruan and others; see e.g. [8], [7], [62]. The use of real greyscale images as in class 9, but also color images, is abundant in the computer science literature (e.g. [71], [64]), where optimal transport is typically considered on some feature space. Mérigot [57] illustrated and tested his algorithm on greyscale images directly. Biological imagery has been used in [38] (fingerprints in feature space), [35] (brain MRIs) and elsewhere.

Another example that is frequently used, in particular in the statistics and machine learning literature, are images from the MNIST handwritten digit database. Due to the low resolution of these images we do not consider them, but we might include other images of handwritten text in later revisions of the benchmark.

All in all, we see the current benchmark as a first stable version. We are

happy to adapt and extend future versions based on feedback from other researchers.

3.3 Tested Methods

In what follows, we list the methods that we have tested on the DOTmark. Due to the large number of suggested methods, which unfortunately is not well reflected in the number of user-friendly implementations available, some restrictions had to be made. We chose methods with a good track record, such as the AHA method, as well as some new and promising methods, such as the shielding method. In order to make our comparison as meaningful as possible, we abstained from using methods of a purely approximative nature, such as Sinkhorn scaling [20].

Multiscale versions exist for all tested methods. Sometimes these are tailor-made, like for the AHA and the shielding methods (see [57] and [76]). Sometimes these are just relatively simple but efficient procedures exploiting the fact that all mass distributions considered live on a square grid in $\mathbb{R}^2$. Such a simple strategy may be as follows: First solve the transport problem on coarsened images (e.g. obtained by adding up the pixel values in contiguous squares of four pixels); then propagate the obtained transport plan in a suitable way so that a feasible transport plan for the finer images is obtained; finally solve the original (fine) problem using this transport plan as a starting value.

In our experience this simple strategy already results in an improvement by a factor of 2 to 5 in the transportation simplex at resolution 64×64. A more elaborate, efficient, but not entirely rigorous alternative was proposed in [62]. Since the precise variant and implementation of a multiscale method may distort competition and distract from the merit of an algorithm as such, we decided not to use *any* multiscale methods for this first comparison.

All used methods are described in Chapter 2. For more information see the respective sections and references therein.

Transportation Simplex One of the classical optimal transport methods we test in this paper is the transportation simplex. We use the modified row minimum rule to determine a basic feasible solution and in the simplex steps we use the row minimum strategy to find a variable to include into the basis. These methods were chosen due to their consistently good performance. In the test we use our implementation of the transportation simplex in Java.

Shortlist Method Just as with the transportation simplex, we use our Java implementation of the shortlist method for the benchmark. Many routines are shared with the transportation simplex. We use the default parameters presented in [39]. They were chosen with regard to the problems considered in that paper – a version of the class WhiteNoise with the Euclidean distance as cost function ($p = 1$) and irregular source and target locations.

Shielding Neighborhood Method For the internal sparse instances we follow the recommendation of the paper introducing this method [76] and use the network simplex solver of CPLEX with a warm start of the previously optimal basis. The Java API is used to create the models and we implemented the shielding neighborhood generation routine in Java as well. One call of the CPLEX solver for a single sparse instance is what we refer to as one iteration of the shielding method in the remainder of this paper.

Schmitzer published his own code of the method on his website[1]. But since we encountered difficulties in getting it to run on our server and since we already had our own well-working implementation of this method in place, we decided to use the latter in our test. As all the other methods it was implemented to the best of our knowledge. Since the majority of the runtime is occupied by the internal CPLEX solver, we expect our code to have a similar runtime as the code provided by Schmitzer.

Additionally, we tested the shielding method with an adapted implementation of the transportation simplex as internal solver, using the same pivot strategy and simplex step routines as before.

[1] `https://www.ceremade.dauphine.fr/~schmitzer/`

AHA Method Our concrete implementation is largely based on [57], but as with other algorithms we only use the singlescale version for comparability and we follow that paper in using the L-BFGS-B algorithm with Moré–Thuente type line search. Since we use a continuous optimization method, we typically cannot reach a weight vector w_* where the gradient of Φ is exactly zero (and hence the image measure of μ is exactly ν), but have to stop when its length is still slightly positive. We thus commit a small controllable error, which we refer to as *precision error*.

In order to make the algorithm applicable to the fully discrete situation we study in the other algorithms, we turn the first image into an absolutely continuous measure μ by interpreting pixel values as masses uniformly distributed over the squared areas represented by the pixels, rather than centered at grid points. Compared to the other methods this leads to slightly different results, a discrepancy we refer to as *blurring error*.[2]

Unlike for the other methods we use an implementation that is written mainly in C with some minimal R overhead. The construction of power diagrams was reimplemented in C based on ideas from the CGAL Regular_triangulation_2 package and other sources. For the L-BFGS-B algorithm we used the implementation in the R function `optim`.

Solvers For representatives of LP solvers, we used CPLEX and Gurobi. For both we modeled the optimal transport problem as an LP and used the default parameters. This is denoted CPLEX-Def and Gurobi-Def, respectively. Additionally, we tested the network simplex solver CPLEX provides (CPLEX-NWS). Gurobi does apparently not come with a network solver, but as the Gurobi documentation page for methods[3] recommends the dual simplex for memory intensive models, we included it in our tests (Gurobi-DS). All models were set up using the Java APIs of the solvers.

[2]Note that the term "error" is subjective. We might as well declare that we want to solve the semidiscrete problem, in which case all the other methods commit a "concentration error".

[3]`http://www.gurobi.com/documentation/6.5/refman/method.html#parameter:Method`

3.4 Computational Results

All our tests were performed using a single core on a Linux server (AMD Opteron Processor 6140 from 2011 with 2.6 GHz). Note that much better absolute runtimes can be achieved when using modern CPU hardware. For many of the algorithms considerable further improvements are possible by multithreaded implementations that use multiple CPU cores simultaneously.

In our experiments we placed the main emphasis on ensuring that the commercial solvers and all of our own implementations were run under the same conditions. In particular, they were all restricted to use only one of 32 available cores, which was realized by the Linux kernel feature cgroups.

Pairing any two of the 10 images in each of the 10 classes gives 45 transport problems ("instances") per class, yielding 450 instances in total. These were all solved at resolutions 32×32 and 64×64 by each of the described methods using the squared Euclidean metric as cost function. All optimal transport costs returned were checked for correctness. The AHA method is the only procedure, where we cannot expect precisely correct results due to the errors described in Section 3.3. These errors are reported in Subsection 3.4.2. All other errors were zero.

3.4.1 Runtimes

The runtimes of the tests are listed in Table 3.2 for 32×32 and Table 3.3 for 64×64, respectively, averaged over all 45 instances in one class. The average over all classes can be found in the bottom row under 'Overall'. The fastest algorithm for each class is highlighted in bold. Additionally, boxplots for four selected methods are given in Figure 3.4.

As the numbers show, the shielding neighborhood method is clearly the fastest algorithm for 32×32 instances among the methods tested. It takes hardly more than half the time on average compared to the transportation simplex, the shortlist method and the AHA method. The default solvers of CPLEX and Gurobi perform particularly badly. It is remarkable, however, that the network simplex solver of CPLEX outperforms the default solvers and the Gurobi dual simplex by a huge margin. The performance is still

Class	TPS	SHL	Shielding		AHA	CPLEX		Gurobi	
			CPX	TPS		Def	NWS	Def	DS
WhNoise	1.58	1.38	**0.67**	1.3	3.28	29.8	5.76	8.1	50.5
GRFrgh	2.16	1.98	**1.08**	2.3	3.19	43.0	5.94	9.0	50.7
GRFmod	3.45	3.80	**1.86**	6.5	3.17	77.3	6.14	21.1	49.8
GRFsmth	4.23	5.46	**2.66**	10.0	4.39	101.9	6.15	36.0	50.4
LogGRF	5.18	6.40	**3.00**	10.8	6.80	119.9	6.23	49.5	51.7
LogitGRF	4.50	5.26	**2.40**	7.9	8.49	98.4	6.33	31.0	51.7
Cauchy	4.46	6.06	**3.62**	12.7	3.76	140.4	6.09	54.1	49.3
Shapes	1.06	1.07	**0.92**	8.9	1.27	8.9	1.22	5.2	9.1
Classic	3.33	3.26	**1.58**	5.2	2.01	68.5	6.05	18.0	49.6
Micro	2.34	3.02	**1.66**	9.4	3.14	35.2	2.74	20.7	22.1
Overall	3.23	3.77	**1.94**	7.5	3.95	72.3	5.26	25.3	43.5

Table 3.2: Average runtimes on 32×32 instances in seconds. The columns represent the methods tested: transportation simplex (TPS), shortlist method (SHL), shielding neighborhood method with CPLEX (CPX) and TPS as internal solvers, AHA method, CPLEX with default (Def) parameters and the network simplex solver (NWS), as well as Gurobi with default parameters and the dual simplex solver (DS).

somewhat worse than our implementations of the transportation simplex and the shortlist method.

At resolution 64×64 we see a similar picture with some exceptions. The shielding method (with CPLEX as internal solver) is even further ahead of most other algorithms, but at the same time the AHA method, which seems to be scaling much better than the linear programming approaches, has gained even more and in fact shows the best times now for many of the classes. However, keep in mind that the results of this method are not exactly correct and the timing varies according to the stopping criterion one applies (see the next subsection).

The CPLEX network solver is with the exception of the classes WhiteNoise and GRFrough quite a bit faster at resolution 64×64 than our implementations of the transportation simplex and the shortlist methods. In the two classes Shapes and Microscopy, which have many zeros in the images, it is even competitive with the shielding and AHA methods. Overall, it shows the most

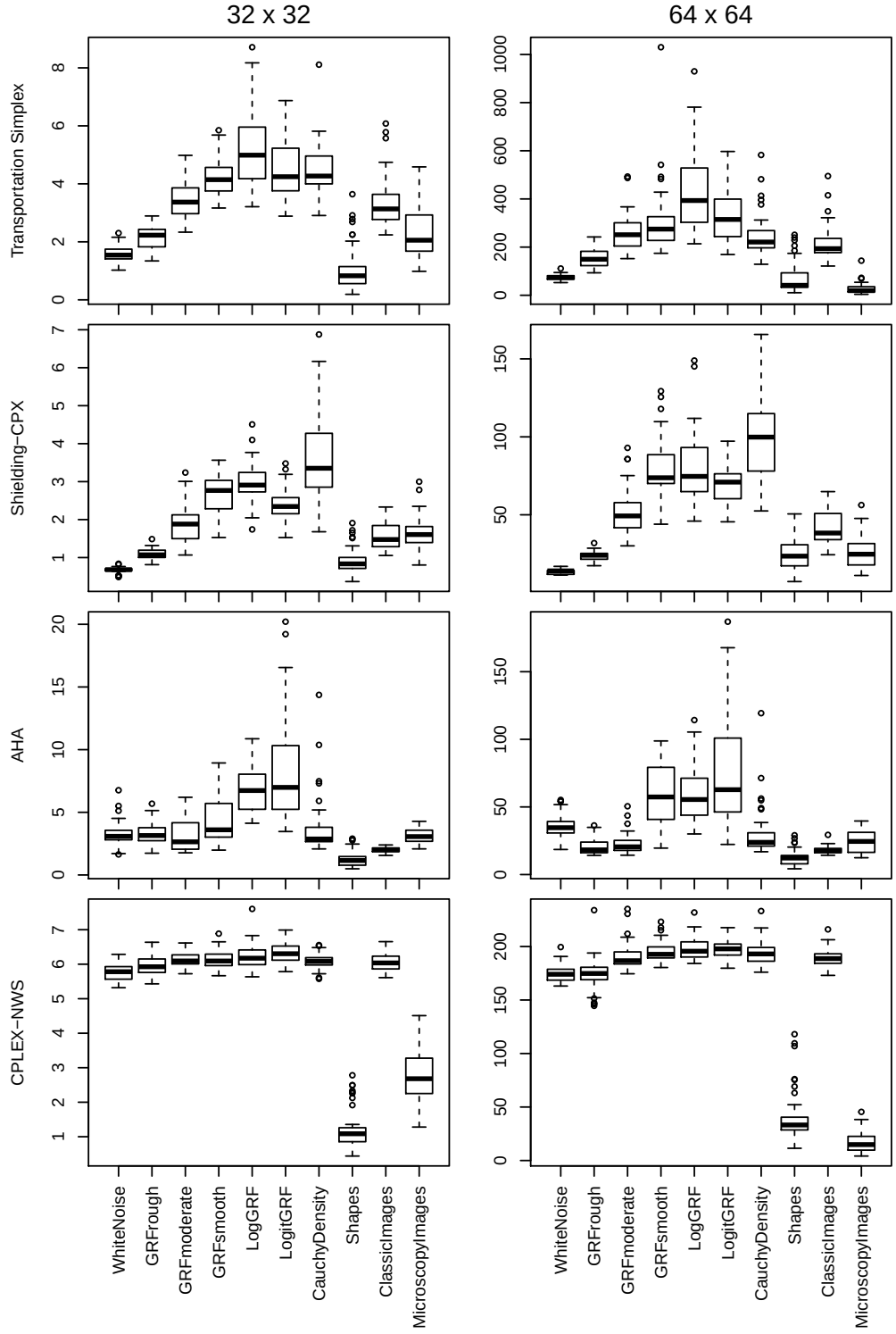


Figure 3.4: Boxplots of the runtimes of selected methods in seconds. Every box represents 45 computed instances. *Left:* 32×32. *Right*: 64×64.

Class	TPS	SHL	Shielding		AHA	CPLEX		Gurobi	
			CPX	TPS		Def	NWS	Def	DS
WhNoise	74	56	**13**	65	36	2057	174	311	1657
GRFrgh	153	110	24	153	**20**	3216	174	473	1659
GRFmod	261	306	51	469	**23**	4592	190	1971	1634
GRFsmth	306	468	80	778	**57**	5621	195	3723	1590
LogGRF	439	531	79	756	**59**	7156	198	4628	1687
LogitGRF	333	362	**69**	552	77	6024	198	2294	1637
Cauchy	245	397	97	968	**30**	6336	195	4461	1435
Shapes	73	73	25	893	**12**	885	40	302	219
Classic	218	246	41	418	**18**	6298	189	1551	1546
Micro	28	37	26	824	24	352	**18**	179	114
Overall	213	258	51	588	**36**	4254	157	1989	1318

Table 3.3: Average runtimes on 64×64 instances in seconds. See the caption above for details.

consistent performance both across the various benchmark classes (if effective size of the problem is taken into account) and within each class; see the last row in Figure 3.4.

In contrast, the other methods show a much stronger sensitivity with respect to the class considered. Especially for classes 4–7 and to some extent also for class 3, we see higher average computation times and in particular a much wider spread of times including several outliers in Figure 3.4.

3.4.2 Errors of the AHA Method

As described in Section 3.3, the AHA method does not solve the problems we consider here with full accuracy, but makes a blurring error by interpreting pixel values of the source measure μ as uniformly distributed over small squares and a precision error by tackling a continuous minimization problem which for numerical reasons cannot be solved exactly.

In Table 3.4 we report the precision error (PE) in terms of the mass in the probability measure μ that is wrongly allocated, as well as the relative

Wasserstein error (RWE) made with the AHA method, i.e.

$$\frac{W_2^{\mathrm{AHA}}(\mu, \tilde{\nu}) - W_2^{\mathrm{TSP}}(\mu, \nu)}{W_2^{\mathrm{TSP}}(\mu, \nu)},$$

where AHA and TSP denote the methods used and $\tilde{\nu}$ denotes the second marginal of the transference plan returned by AHA.

We can see that the precision error is reasonably small. What is more, if we assume that we would have to reroute the wrongly allocated mass roughly by a distance that corresponds to the true Wasserstein distance between μ and ν (this is reasonable in the sense that it is roughly the same order of magnitude as relevant distances measured in the image), we can compare the precision errors to the relative Wasserstein errors and see that the former play only a minor role. Consequently, the RWE is mainly due to the blurring effect. This is corroborated further by the fact that the RWE for the 64×64 resolution is considerably smaller than for 32×32.

Instance Class	PE 32	PE 64	RWE 32	RWE 64
WhiteNoise	0.3098e-05	0.3400e-05	0.031224	0.027232
GRFrough	0.4261e-05	0.6327e-05	0.018957	0.008058
GRFmoderate	0.9840e-05	1.8471e-05	0.003965	0.001075
GRFsmooth	2.4910e-05	4.0698e-05	0.001303	0.000349
LogGRF	4.6378e-05	5.6221e-05	0.000556	0.000230
LogitGRF	1.6833e-05	3.7144e-05	0.002008	0.000682
CauchyDensity	3.9273e-05	8.2546e-05	0.000988	0.000289
Shapes	1.3435e-05	3.3610e-05	0.003192	0.000994
ClassicImages	0.7797e-05	1.8311e-05	0.005119	0.001512
Microscopy	2.7111e-05	6.1215e-05	0.001243	0.000412
Overall	1.9294e-05	3.5794e-05	0.006855	0.004083

Table 3.4: Average precision error (PE) and relative Wasserstein error (RWE) over the ten classes. See text for details.

3.4.3 Iterations of the Shielding Method

The shielding method solves the optimal transport problem via a sequence of restricted instances. Here we have a look on how many iterations of these

instances are necessary.

On the scale 32×32 the average number of iterations varies between 3.9 for WhiteNoise and 16.6 for CauchyDensity. The numbers for scale 64×64 are higher (4.6 through 28.5), but show similar behavior otherwise. Figure 3.5 presents scatterplots of the runtime against the number of iterations.

For most classes (green points) we observe a linear scaling of the runtime with the number of iterations. This means that the runtime in each iteration is roughly the same over these classes and we may conclude, since CPLEX has runtimes that scale consistently with model size, that the neighborhood sizes remain more or less constant as well. The only exceptions are the

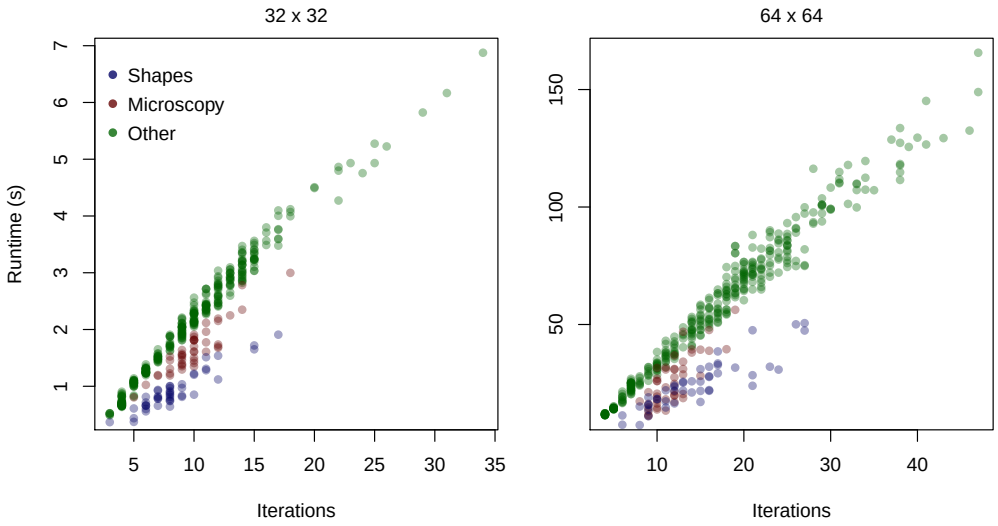


Figure 3.5: Scatterplots for Shielding-CPX showing the runtimes against the number of iterations for the classes Shapes (blue), Microscopy (red) and the other classes combined (green). Every data point represents one of the 450 computed instances. *Left:* 32×32. *Right:* 64×64.

classes Shapes (blue) and Microscopy (red), where the runtimes are lower than expected from the number of iterations. This can be attributed to the internal solver, which benefits from the lower effective dimension that comes from the zero mass pixels occurring in these two classes. The difference is not as significant as for the global CPLEX network simplex runtimes, since the dimension has already been reduced by the construction of the shielding neighborhoods.

3.5 Discussion

If we look at the different classes, we note that the solvers perform much better on the classes where the number of pixels with mass zero is large (Microscopy and Shapes). This is because they seem to benefit particularly from the reduction of the effective dimension of the problem. The runtimes are very consistent across the other classes, which allows the conclusion that the solvers can only exploit the mathematical structure of the model, but are unable to use geometric features of the input data to their advantage.

Other linear programming methods benefit from the lower effective dimension as well, although the difference is not as significant. However, these methods seem to be comparatively faster on classes where most of the transports are rather short (rough structure, such as GRFrough or WhiteNoise), and slower on classes with longer transports (smooth structure, such as GRFsmooth or LogGRF). This can be explained by the initial solution routine, which is shared across many of the tested methods. The greedily selected initial transport plan, which favors short transports, is more likely to be close to optimal in short range classes than in long range classes. The shortlist method, which performs another greedy step in addition when searching for new basis variables within the shortlists, benefits particularly from short transports in the solutions, but not as much as in the sparse examples considered in [39] with the Euclidean metric as cost function.

The runtimes of the AHA method are relatively consistent. They are only considerably shorter for the class Shapes. This may be due to the fact that in these instances there are only a few different mass values in the images and the mass is uniformly distributed over large areas.

Interestingly, a comparison of the transportation simplex and the CPLEX network simplex reveals that the performance of the transportation simplex is better on 32×32 instances, while at resolution 64×64 the opposite is the case. This can be explained by looking at the two methods at hand. Although details of the CPLEX network simplex solver are not known, it is safe to assume that the simplex steps are implemented very efficiently. On the other hand, the transportation simplex has the advantage of an easily

obtainable good initial solution, whereas in the network simplex a preceding simplex phase is necessary. This makes the TPS perform better on smaller instances. On the higher resolution our results suggest that the advantage of a strong initial solution is not as influential as the efficient simplex steps.

Considering the small disparity between the transportation simplex and the CPLEX network simplex runtimes, the difference in performance between the two internal solvers for the shielding neighborhood method is surprisingly large. This is due to the fact that initial solutions to the interior models are available in the shielding method and thus the first phase of the network simplex is not necessary. This is why the initial advantage of the transportation simplex disappears and hence using CPLEX as the internal solver yields much lower runtimes.

Another observation worth mentioning is that the runtimes, and therefore numbers of iterations, of the shielding method for the randomly created classes 1–6 agree very well with the ranges of dependence within the data of the classes. The class with the lowest runtimes, WhiteNoise, has no dependence at all, whereas the classes with long range dependences, GRFsmooth and LogGRF, belong to the more difficult classes for the shielding method to solve. That seems to indicate that constructing small shielding neighborhoods prevents larger updates to the current solution per iteration, which might be required in these instances. This may also contribute to the comparatively long and inconsistent runtimes on the class CauchyDensity.

Based on this first comparison of singlescale methods, we give the following **recommendations** for large discrete transport problems (on grids):

1. If you have to be precise and have access to the IBM CPLEX software, use the shielding method in combination with CPLEX.

2. If you have to be reasonably precise but can afford a small controllable error, use the AHA method. This is especially advisable if you require a very high resolution for the mass distribution at the source, as this comes for the AHA method at virtually no extra cost.

3. Both methods are not widely available nor typically very efficient for costs other than the squared Euclidean metric. So, for other costs direct use of a conventional simplex algorithm or the shortlist method may be preferable.

4. If you use a solver directly, be very careful which one you use and that you call the most appropriate function. Especially when using CPLEX, make sure that you use the network simplex solver and that you set up the input as a network structure. If you can, solve the model with a warm start.

We emphasize again that the absolute runtimes given in Tables 3.2 and 3.3 should not be taken at face value and that actual computations on modern CPUs are typically much faster. While the relative comparison presented here is justified to the best of our knowledge, it allows only limited conclusions about the performance of multiscale variants and multithreaded implementations of the different methods.

3.6 Outlook

By providing this benchmark we hope to improve the comparability of different methods for solving discrete optimal transport problems. Contributions or suggestions for extending the benchmark are welcome. In particular we plan to include data sets concentrated on more general grid structures and especially with irregular support points if there is enough public interest.

The R package `transport` [80] offers user-friendly implementations that are mostly written in C of three of the methods presented here (transportation simplex, shortlist and the AHA method). It will be updated in the near future to include additional state-of-the-art methods.

Solving transport problems exactly for larger images (e.g. with 128, 256 or 512 pixels in each dimension) is still computationally very demanding, even for state-of-the-art methods. Efficient solutions of such large problems could pave the way for a new class of algorithms in image processing. In

the area of computer vision and image processing, important applications include image enhancement, denoising, inpainting, feature extraction and compression. In one subdomain of image processing, these challenges are approached by decomposing images into two or three parts [93], e.g. a cartoon component, which contains piecewise constant or piecewise smooth parts, a texture component, which captures oscillating patterns, and a noise component, which contains small scale objects (corresponding to high frequency parts in the Fourier domain). After the decomposition step, the texture component can be utilized for applications such as fingerprint segmentation [94]. Image decompositions are obtained by formulating and solving minimization problems that impose suitable norms on the respective components. The total variation (TV) norm is commonly used for the cartoon component and the G-norm [58] for the texture component. Recent works by Brauer and Lorenz [17] and by Lellmann *et al.* [52] connect the G-norm to solutions of transport problems. Typically, the minimization problems described above are solved iteratively and are computationally expensive. It is conceivable to formulate transport norms for image decomposition, which would require to solve a large transport problem in each iteration. Thus, efficient algorithms for optimal transport are a prerequisite for future research in this direction.

Chapter 4

Probabilistic Approximation with Exact Solvers

Except for minor adjustments, this chapter is identical to the publication *Optimal Transport: Fast Probabilistic Approximation with Exact Solvers* [90], which is joint work with Max Sommerfeld, Yoav Zemel and Axel Munk, and many parts of this paper also appear in Max Sommerfeld's doctoral thesis [88]. The proofs of the theoretical results in Section 4.3 are omitted here. They can be found in the appendix of the paper [90].

4.1 Introduction

Optimal transport distances compare probability measures by incorporating a suitable ground distance on the underlying space, typically driven by the specific application, e.g. Euclidean distance. This often makes it preferable to competing distances such as total-variation or χ^2-distances, which are oblivious to any metric or similarity structure on the ground space. Note that total variation is the Wasserstein distance with respect to the trivial metric, which usually does not carry the geometry of the underlying ground space. In this setting, optimal transport distances have a clear and intuitive interpretation as the amount of 'work' required to transport one probability distribution onto the other. This notion is typically well-aligned with human

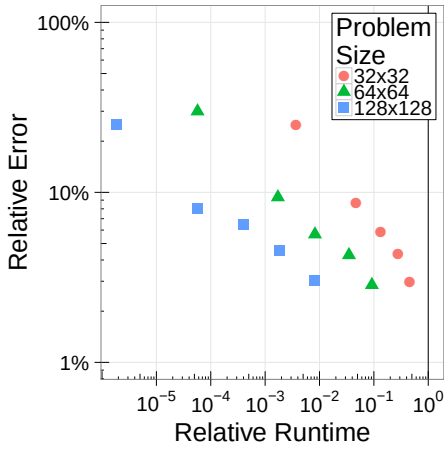


Figure 4.1: Relative error and relative runtime compared to the exact computation of the proposed scheme. Optimal transport distances and its approximations were computed between images of different sizes (32×32, 64×64, 128×128). Each point represents a specific parameter choice in the scheme and is a mean over different problem instances, solvers and cost exponents. For the relative runtimes the geometric mean is reported. For details on the parameters see Figure 4.2.

perception of similarity [71].

4.1.1 Computation

The outstanding theoretical and practical performance of optimal transport distances is contrasted by its excessive computational cost. For example, optimal transport distances can be computed with an auction algorithm [10]. For two probability measures supported on N points this algorithm has a worst-case run time of $\mathcal{O}(N^3 \log N)$. Other methods like the transportation simplex have sub-cubic empirical average runtime (compare [39]), but exponential worst-case runtimes.

Therefore, many attempts have been made to design improved algorithms. We give some selective references: Ling and Okada [53] proposed a specialized algorithm for L_1-ground distance and $\mathcal{X}$ a regular grid and report an empirical runtime of $\mathcal{O}(N^2)$. Gottschlich and Schuhmacher [39] improved existing general purpose algorithms by initializing with a greedy heuristic. Their

Shortlist algorithm achieves an empirical average runtime of the order $\mathcal{O}(N^{5/2})$. Schmitzer [76] solves the optimal transport problem by solving a sequence of sparse problems. The theoretical runtime of his algorithm is not known, but it exhibits excellent performance on two-dimensional grids (see [78] or Chapter 3).

Despite these efforts, many practically relevant problems remain well outside the scope of available algorithms. See [78] or Chaper 3 for an overview and a numerical comparison of state-of-the-art algorithms for discrete optimal transport. This is true in particular for two- or three-dimensional images and spatio temporal imaging, which constitute an important area of potential applications. Here, N is the number of pixels or voxels and is typically of size 10^5 to 10^7. Naturally, this problem is aggravated when many distances have to be computed as is the case for Wasserstein barycenters [1, 21], which have become an important use case.

To bypass the computational bottleneck, many surrogates for optimal transport distances that are more amenable to fast computation have been proposed. Shirdhonkar and Jacobs [84] proposed to use an equivalent distance based on wavelets that can be computed in linear time but cannot be calibrated to approximate the Wasserstein distance with arbitrary accuracy. Pele and Werman [64] threshold the ground distance to reduce the complexity of the underlying linear program, obtaining a lower bound for the exact distance. Cuturi [20] altered the optimization problem by adding an entropic penalty term in order to use faster and more stable algorithms, see also [2]. Bonneel and others [14] consider the 1-D Wasserstein distances of radial projections of the original measures, exploiting the fact that, in one dimension, computing the Wasserstein distance amounts to sorting the point masses and hence has quasi-linear computation time.

4.1.2 Contribution

We do *not* propose a new algorithm to solve the optimal transport problem. Instead, we propose a simple probabilistic scheme as a meta-algorithm that can use any algorithm (e.g. those mentioned above) solving finitely supported

optimal transport problems as a black-box back-end and gives a random but fast approximation of the exact distance. This scheme

a) is extremely easy to implement, to parallelize and to tune towards higher accuracy or shorter computation time as desired;

b) can be used with any algorithm for transportation problems as a back-end, including general LP solvers, specialized network solvers and algorithms using entropic penalization [20];

c) comes with theoretical non-asymptotic guarantees for the approximation error of the Wasserstein distance — in particular, this error is independent of the size of the original problem in many important cases, including images;

d) works well in practice. For example, the Wasserstein distance between two 128^2-pixel images can typically be approximated with a relative error of less than 5% in only 1% of the time required for exact computation.

4.2 Problem and Algorithm

Although our meta-algorithm is applicable to exact solvers for any optimal transport distance between probability measures, for example the Sinkhorn distance [20], the theory we present here concerns the Kantorovich transport distance [46], often also denoted as *Wasserstein distance*.

Wasserstein Distance Consider a fixed finite space $\mathcal{X} = \{x_1, \dots, x_N\}$ with a metric $d : \mathcal{X} \times \mathcal{X} \to [0, \infty)$. Every probability measure on $\mathcal{X}$ is given by a vector $\boldsymbol{r}$ in

$$\mathcal{P}_{\mathcal{X}} = \left\{ \boldsymbol{r} = (r_x)_{x \in \mathcal{X}} \in \mathbb{R}^{\mathcal{X}}_{\geq 0} : \sum_{x \in \mathcal{X}} r_x = 1 \right\},$$

via $P_{\boldsymbol{r}}(\{x\}) = r_x$. We will not distinguish between the vector $\boldsymbol{r}$ and the measure it defines. For $p \geq 1$, the *p-th Wasserstein distance* between two

probability measures $\boldsymbol{r}, \boldsymbol{s} \in \mathcal{P}_{\mathcal{X}}$ is defined as

$$W_p(\boldsymbol{r}, \boldsymbol{s}) = \left(\min_{\boldsymbol{w} \in \Pi(\boldsymbol{r}, \boldsymbol{s})} \sum_{x, x' \in \mathcal{X}} d^p(x, x') w_{x,x'} \right)^{1/p}, \tag{4.1}$$

where $\Pi(\boldsymbol{r}, \boldsymbol{s})$ is the set of all probability measures on $\mathcal{X} \times \mathcal{X}$ with marginal distributions $\boldsymbol{r}$ and $\boldsymbol{s}$, respectively. The minimization in (4.1) can be written as a linear program

$$\min \sum_{x,x' \in \mathcal{X}} w_{x,x'} d^p(x, x') \quad \textbf{s.t.} \quad \sum_{x' \in \mathcal{X}} w_{x,x'} = r_x, \quad \sum_{x \in \mathcal{X}} w_{x,x'} = s_{x'}, \quad w_{x,x'} \geq 0, \tag{4.2}$$

with N^2 variables $w_{x,x'}$ and $2N$ constraints, where the weights $d^p(x, x')$ are known and have been precalculated.

Approximating the Wasserstein Distance The idea of the proposed algorithm is to replace a probability measure $\boldsymbol{r} \in \mathcal{P}(\mathcal{X})$ with an empirical measure $\hat{\boldsymbol{r}}_S$ based on i.i.d. picks $X_1, \ldots, X_S \sim \boldsymbol{r}$ for some integer S:

$$\hat{r}_{S,x} = \frac{1}{S} \# \{k : X_k = x\}, \quad x \in \mathcal{X}. \tag{4.3}$$

Likewise, replace $\boldsymbol{s}$ with $\hat{\boldsymbol{s}}_S$. Then, use the *empirical optimal transport distance* (EOT) $W_p(\hat{\boldsymbol{r}}_S, \hat{\boldsymbol{s}}_S)$ as a random approximation of $W_p(\boldsymbol{r}, \boldsymbol{s})$.

Algorithm 1: Statistical approximation of $W_p(\boldsymbol{r}, \boldsymbol{s})$

1: **Input:** Probability measures $\boldsymbol{r}, \boldsymbol{s} \in \mathcal{P}_{\mathcal{X}}$, sample size S and number of repetitions B
2: **for** $i = 1 \ldots B$ **do**
3: Sample i.i.d. $X_1, \ldots, X_S \sim \boldsymbol{r}$ and independently $Y_1, \ldots, Y_S \sim \boldsymbol{s}$
4: $\hat{r}_{S,x} \leftarrow \# \{k : X_k = x\} / S$ **for all** $x \in \mathcal{X}$
5: $\hat{s}_{S,x} \leftarrow \# \{k : Y_k = x\} / S$ **for all** $x \in \mathcal{X}$
6: Compute $\hat{W}^{(i)} \leftarrow W_p(\hat{\boldsymbol{r}}_S, \hat{\boldsymbol{s}}_S)$
7: **end for**
8: **Return:** $\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) \leftarrow B^{-1} \sum_{i=1}^{B} \hat{W}^{(i)}$

In each of the B iterations in Algorithm 1, the Wasserstein distance

between two sets of S point masses has to be computed. For the exact Wasserstein distance, two measures on N points need to be compared. If we take for example the super-cubic runtime of the auction algorithm as a basis, Algorithm 1 has worst-case runtime

$$\mathcal{O}(BS^3 \log S)$$

compared to $\mathcal{O}(N^3 \log N)$ for the exact distance. This means a dramatic reduction of computation time if S (and B) are small compared to N.

The application of Algorithm 1 to other optimal transport distances is straightforward. One can simply replace $W_p(\hat{\boldsymbol{r}}_S, \hat{\boldsymbol{s}}_S)$ with the desired distance, e.g. the Sinkhorn distance [20], see also our numerical experiments below. Further, the algorithm can be applied to non-discrete instances as long as we can sample from the measures. However, the theoretical results below only apply to the EOT on a finite ground space $\mathcal{X}$.

4.3 Theoretical Results

We give general non-asymptotic guarantees for the quality of the approximation $\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) = B^{-1} \sum_{i=1}^{B} W_p(\boldsymbol{r}_{S,i}, \boldsymbol{s}_{S,i})$ (where $\boldsymbol{r}_{S,i}$ are independent empirical measures of size S from $\boldsymbol{r}$; see Algorithm 1) in terms of the expected L_1-error. That is, we give bounds of the form

$$E\left[\left|\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) - W_p(\boldsymbol{r}, \boldsymbol{s})\right|\right] \leq g(S, \mathcal{X}, p), \tag{4.4}$$

for some function g. We are particularly interested in the dependence of the bound on the size N of $\mathcal{X}$ and on the sample size S as this determines how the number of sampling points S (and hence the computational effort of Algorithm 1) must be increased for increasing problem size N in order to retain (on average) a certain approximation quality. In a second step, we obtain deviation inequalities for $\hat{W}^{(S)}(\boldsymbol{r}, \boldsymbol{s})$ via concentration of measure techniques.

Related Work The question of the convergence of empirical measures to the true measure in expected Wasserstein distance has been considered in detail by Boissard and Le Gouic[13] and Fournier and Guillin [28]. The case of the underlying measures being different (that is, the convergence of $EW_p(\hat{\boldsymbol{r}}_S, \hat{\boldsymbol{s}}_S)$ to $W_p(\boldsymbol{r}, \boldsymbol{s})$ when $\boldsymbol{r} \neq \boldsymbol{s}$) has not been considered to the best of our knowledge. Theorem 4.3.1 is reminiscent of the main result of [13]. However, we give a result here, which is explicitly tailored to finite spaces and makes explicit the dependence of the constants on the size N of the underlying set $\mathcal{X}$. In fact, when we consider finite spaces $\mathcal{X}$ which are subsets of $\mathbb{R}^D$ later in Theorem 4.3.4, we will see that in contrast to the results of [13], the rate of convergence (in S) does not change when the dimension gets large, but rather the dependence of the constants on N changes. This is a valuable insight as our main concern here is how the subsample size S (driving the computational cost) must be chosen when N grows in order to retain a certain approximation quality.

4.3.1 Expected Absolute Error

Recall that, for $\delta > 0$ the *covering number* $\mathcal{N}(\mathcal{X}, \delta)$ of $\mathcal{X}$ is defined as the minimal number of closed balls with radius δ and centers in $\mathcal{X}$ that is needed to cover $\mathcal{X}$. Note that in contrast to continuous spaces, $\mathcal{N}(\mathcal{X}, \delta)$ is bounded by N for all $\delta > 0$.

Theorem 4.3.1. *Let $\hat{\boldsymbol{r}}_S$ be the empirical measure obtained from i.i.d. samples $X_1, \dots, X_S \sim \boldsymbol{r}$, then*

$$E\left[W_p^p(\hat{\boldsymbol{r}}_S, \boldsymbol{r})\right] \leq \mathcal{E}_q/\sqrt{S}, \tag{4.5}$$

where the constant $\mathcal{E}_q := \mathcal{E}_q(\mathcal{X}, p)$ is given by

$$\mathcal{E}_q = 2^{p-1} q^{2p} (\operatorname{diam}(\mathcal{X}))^p \left(q^{-(l_{\max}+1)p} \sqrt{N} + \sum_{l=0}^{l_{\max}} q^{-lp} \sqrt{\mathcal{N}(\mathcal{X}, q^{-l} \operatorname{diam}(\mathcal{X}))} \right) \tag{4.6}$$

for any $2 \leq q \in \mathbb{N}$ and $l_{\max} \in \mathbb{N}$.

Remark 4.3.2. Since Theorem 4.3.1 holds for any integer $q \geq 2$ and $l_{\max} \in \mathbb{N}$, they can be chosen freely to minimize the constant $\mathcal{E}_q$. In the proof they appear as the branching number and depth of a spanning tree that is constructed on $\mathcal{X}$ (see appendix).

Based on Theorem 4.3.1, we can formulate a bound for the mean approximation error of Algorithm 1. A mean squared error version is given below, in Theorem 4.3.8.

Theorem 4.3.3. *Let $\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s})$ be as in Algorithm 1 for any choice of $B \in \mathbb{N}$. Then for every integer $q \geq 2$*

$$E\left[\left|\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) - W_p(\boldsymbol{r}, \boldsymbol{s})\right|\right] \leq 2\mathcal{E}_q^{1/p} S^{-1/(2p)}. \tag{4.7}$$

Proof. The statement is an immediate consequence of the reverse triangle inequality for the Wasserstein distance, Jensen's inequality and Theorem 4.3.1,

$$\begin{aligned} E\left[\left|\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) - W_p(\boldsymbol{r}, \boldsymbol{s})\right|\right] &\leq E\left[W_p(\hat{\boldsymbol{r}}_S, \boldsymbol{r}) + W_p(\hat{\boldsymbol{s}}_S, \boldsymbol{s})\right] \\ &\leq E\left[W_p^p(\hat{\boldsymbol{r}}_S, \boldsymbol{r})\right]^{1/p} + E\left[W_p^p(\hat{\boldsymbol{s}}_S, \boldsymbol{s})\right]^{1/p} \leq 2\mathcal{E}_q^{1/p}/S^{1/(2p)}. \end{aligned}$$

□

Measures on Euclidean Space While the constant $\mathcal{E}_q$ in Theorem 4.3.1 may be difficult to compute or estimate in general, we give explicit bounds in the case when $\mathcal{X}$ is a finite subset of a Euclidean space. They exhibit the dependence of the approximation error on $N = |\mathcal{X}|$. In particular, it comprises the case when the measures represent two- or more dimensional images.

Theorem 4.3.4. *Let $\mathcal{X}$ be a finite subset of $\mathbb{R}^D$ with the usual Euclidean metric. Then,*

$$\mathcal{E}_2 \leq \min(2^{D/2}, 2^p) \min(2^{D/2} D^{D/4}, D^{p/2}) 2^{3p} (\operatorname{diam}(\mathcal{X}))^p \cdot C_{D,p}(N),$$

where $N = |\mathcal{X}|$ *and*

$$C_{D,p}(N) = \begin{cases} 3 & \text{if } D/2 - p < 0, \\ 1 + \frac{1}{2p} \log_2 N & \text{if } D/2 - p = 0, \\ 1 + 2N^{\frac{1}{2}(1-\frac{2p}{D})} & \text{if } D/2 - p > 0. \end{cases} \tag{4.8}$$

One can obtain bounds for $\mathcal{E}_q$, $q > 2$ (see the proof), but the choice $q = 2$ leads to the smallest bound. In particular, we have for the most important cases $p = 1, 2$:

Corollary 4.3.5. *Under the conditions of Theorem 4.3.4,*

$$p = 1 \quad \Longrightarrow \quad \mathcal{E}_2 \leq 8D^{1/2} \operatorname{diam}(\mathcal{X}) \cdot \begin{cases} 3 \cdot 2^{D/2} & \text{if } D < 2, \\ 2 + \log_2 N & \text{if } D = 2, \\ 2 + 4N^{\frac{1}{2}(1-\frac{2}{D})} & \text{if } D > 2. \end{cases}$$

$$p = 2 \quad \Longrightarrow \quad \mathcal{E}_2 \leq 64D(\operatorname{diam}(\mathcal{X}))^2 \cdot \begin{cases} 3 \cdot 2^{D/2} & \text{if } D < 4, \\ 4 + \log_2 N & \text{if } D = 4, \\ 4 + 8N^{\frac{1}{2}(1-\frac{4}{D})} & \text{if } D > 4. \end{cases}$$

Theorem 4.3.4 gives control over the error made by the approximation $\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s})$ of $W_p(\boldsymbol{r}, \boldsymbol{s})$. Of particular interest is the behavior of this error as N gets large (e.g. for high resolution images). We distinguish three cases. In the *low-dimensional case* $p' = D/2 - p < 0$, we have $C_{D,p}(N) = \mathcal{O}(1)$ and the approximation error is $\mathcal{O}(S^{-\frac{1}{2p}})$ independent of the size of the image. In the *critical case* $p' = 0$ the approximation error is no longer independent of N but is of order $\mathcal{O}\left(\log(N) S^{-\frac{1}{2p}}\right)$. Finally, in the *high-dimensional case* the dependence on N becomes stronger with an approximation error of order

$$\mathcal{O}\left(\left(\frac{N^{(1-\frac{2p}{D})}}{S}\right)^{\frac{1}{2p}}\right).$$

In all cases one can choose $S = o(N)$ while still guaranteeing vanishing approximation error for $N \to \infty$. In practice, this means that S can typically

be chosen (much) smaller than N to obtain a good approximation of the Wasserstein distance. In particular, this implies that for low-dimensional applications with two or three dimensional histograms (for example greyscale images, where N corresponds to the number of pixels / voxels and $\boldsymbol{r}, \boldsymbol{s}$ correspond to the grey value distribution after normalization), the approximation error is essentially not affected by the size of the problem when p is not too small, e.g. $p = 2$.

While the three cases in Theorem 4.3.4 resemble those given by Boissard and Le Gouic [13], the rate of convergence in S as seen in Theorem 4.3.1 is $\mathcal{O}(S^{-1/2})$, regardless of the dimension of the underlying space $\mathcal{X}$. The constant depends on D, however, roughly at the polynomial rate $D^{p/2}$ and through $C_{D,p}(N)$. It is also worth mentioning that by considering the dual transport problem, one can invoke the framework of Shalev-Shwartz and others [82], particularly Theorem 7. However, the dependence on S and N and the constants are not easily accessible from that paper.

Remark 4.3.6. The results presented here extend to the case where $\mathcal{X}$ is a bounded, countable subset of $\mathbb{R}^D$. However, $\mathcal{E}_q$ can only be bounded in the low-dimensional case ($D/2 - p < 0$) due to Theorem 4.3.4 and is infinite otherwise.

4.3.2 Concentration Bounds

Based on the bounds for the expected approximation error we now give non-asymptotic guarantees for the approximation error in the form of deviation bounds using standard concentration of measure techniques.

Theorem 4.3.7. *If $\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s})$ is obtained from Algorithm 1, then for every $z \geq 0$*

$$P\left[|\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) - W_p(\boldsymbol{r}, \boldsymbol{s})| \geq z + \frac{2\mathcal{E}_q^{1/p}}{S^{1/2p}}\right] \leq 2\exp\left(-\frac{SBz^{2p}}{8\ \mathrm{diam}(\mathcal{X})^{2p}}\right). \quad (4.9)$$

Note that while the mean approximation quality $2\mathcal{E}_q^{1/p}/S^{1/(2p)}$ only depends on the subsample size S, the stochastic variability (see the right-hand side

term in (4.9)) depends on the product SB. This means that the repetition number B cannot decrease the expected error, but it decreases the magnitude of fluctuation around it.

From these concentration bounds we can obtain a mean squared error version of Theorem 4.3.3:

Theorem 4.3.8. *Let $\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s})$ be as in Algorithm 1 for any choice of $B \in \mathbb{N}$. Then for every integer $q \geq 2$ the mean squared error of the EOT can be bounded as*

$$E\left[\left|\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) - W_p(\boldsymbol{r}, \boldsymbol{s})\right|^2\right] \leq 18\mathcal{E}_q^{2/p} S^{-1/p} = \mathcal{O}(S^{-1/p}).$$

Remark 4.3.9. The power 2 can be replaced by any $\alpha \leq 2p$ with rate $S^{-\alpha/(2p)}$, as can be seen from a straightforward modification of the first lines of the proof.

For example, in view of Theorem 4.3.4, when $\mathcal{X}$ is a finite subset of a $\mathbb{R}^D$ with the (optimal) choice $q = 2$, we obtain

$$E\left[\left|\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) - W_p(\boldsymbol{r}, \boldsymbol{s})\right|^2\right] \leq 3^2 2^9 D C_{D,p}^{2/p}(N)[\operatorname{diam}(\mathcal{X})]^2 S^{-1/p}.$$

with the constant $C_{D,p}(N)$ given in (4.8). Thus, we qualitatively observe the same dependence on N as in Theorem 4.3.4, e.g. the mean squared error is independent of N when $D < 2p$.

4.4 Simulations

This section covers the numerical findings of the simulations. Runtimes and returned values of Algorithm 1 for each back-end solver are reported in relation to the results of that solver on the original problem. Four different solvers were tested.

4.4.1 Simulation Setup

The setup of our simulations is identical to that of [78] and Chapter 3. One single core of a Linux server (AMD Opteron Processor 6140 from 2011 with

2.6 GHz) was used. The original and subsampled instances were run under the same conditions.

Three of the four methods featured in this simulation are exact linear programming solvers. The transportation simplex is a modified version of the network simplex solver tailored towards optimal transport problems. Details can be found for example in [54]. The shortlist method [39] is a modification of the transportation simplex, that performs an additional greedy step to quickly find a good initial solution. The parameters were chosen as the default parameters described in that paper. The third method is the network simplex solver of CPLEX (`www.ibm.com/software/commerce/optimization/cplex-optimizer/`). For the transportation simplex and the shortlist method the implementations provided in the R package *transport* [80] were used. The models for the CPLEX solver were created and solved via the R package *Rcplex* [18].

Additionally, the Sinkhorn scaling algorithm [20] was tested in our simulation. This method computes an entropy regularized optimal transport distance. The regularization parameter was chosen according to the heuristic in [20]. Note that the Sinkhorn distance is not covered by the theoretical results from Section 4.3. The errors reported for the Sinkhorn scaling are relative to the values returned by the algorithm on the full problems, which themselves differ from the actual Wasserstein distances.

The instances of optimal transport considered here are discrete instances of two different types: regular grids in two dimensions, that means images in various resolutions, as well as point clouds in $[0,1]^D$ with dimensions $D = 2$, 3 and 4. For the image case, from the DOTmark, which contains images of various types intended to be used as optimal transport instances in the form of two-dimensional histograms, three instances were chosen: two images of each of the classes White Noise, Cauchy Density, and Classic Images, which are then treated in the three resolutions 32×32, 64×64 and 128×128. Images are interpreted as finitely supported measures. The mass of a pixel is given by the greyscale value and the support of the measure is the grid $\{1, \dots, R\} \times \{1, \dots, R\}$ for an image with resolution $R \times R$.

In the White Noise class the greyscale values of the pixels are independent

of each other, the Cauchy Density images show bivariate Cauchy densities with random centers and varying scale ellipses, while Classic Images contains greyscale test images. See [78] or Chapter 3 for further details on the different image classes and example images. The instances were chosen to cover different types of images, while still allowing for the simulation of a large variety of parameters for subsampling.

The point cloud type instances were created as follows: The support points of the measures are independently, uniformly distributed on $[0, 1]^D$. The number of points N was chosen 32^2, 64^2 and 128^2 in order to match the size of the grid-based instances. For each choice of D and N, three instances were generated with regards to the three images types used in the grid based case. Two measures on the points are drawn from the Dirichlet distribution with all parameters equal to one. That means, the masses on different points are independent of each other, similar to the white noise images. To create point cloud versions of the Cauchy Density and Classic Images classes the greyscale values of the same images were used to get the mass values for the support points. In three and four dimensions, the product measure of the images with their sum of columns and with themselves, respectively, was used.

All original instances were solved by each back-end solver in each resolution for the values $p = 1$, $p = 2$, and $p = 3$ in order to be compared to the approximative results for the subsamples in terms of runtime and accuracy, except for CPLEX, where the 128×128 instances could not be solved due to memory limitations. Algorithm 1 was applied to each of these instances with parameters $S \in \{100, 500, 1000, 2000, 4000\}$ and $B \in \{1, 2, 5\}$. For every combination of instance and parameters, the subsampling algorithm was run 5 times in order to mitigate the randomness of the results.

Since the linear programming solvers had a very similar performance on the grid-based instances (see below), only one of them - the transportation simplex - was tested on the point cloud instances.

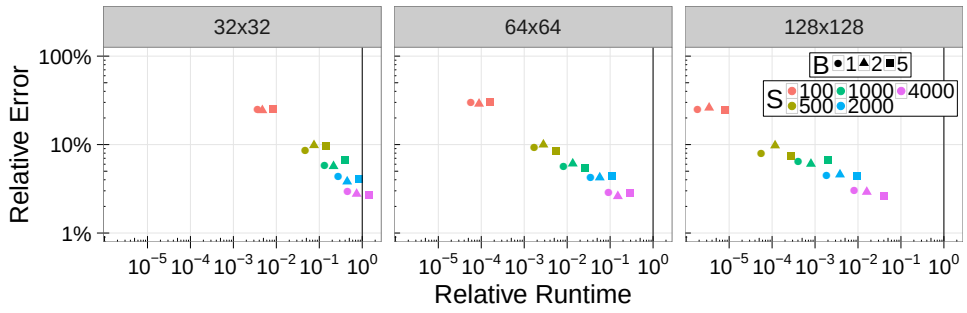


Figure 4.2: Relative errors $|\hat{W}_p^{(S)}(\boldsymbol{r}, \boldsymbol{s}) - W_p(\boldsymbol{r}, \boldsymbol{s})|/W_p(\boldsymbol{r}, \boldsymbol{s})$ vs. relative runtimes $\hat{t}/t$ for different parameters S and B and different problem sizes for images. $\hat{t}$ is the runtime of Algorithm 1 and t is the runtime of the respective back-end solver without subsampling.

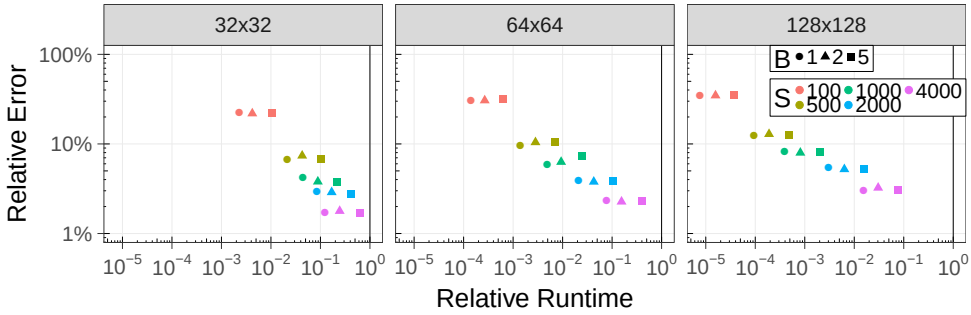


Figure 4.3: Relative errors vs. relative runtimes for different parameters S and B and different problem sizes for point clouds. The number of support points matches the number of pixels in the images.

4.4.2 Computational Results

As mentioned before, all results of Algorithm 1 are relative to the results of the methods applied to the original problems. We are mainly interested in the reduction in runtime and accuracy of the returned values. Many important results can be observed in Figure 4.2 and 4.3. The points in the diagram represent averages over the different methods, instances, and multiple tries, but are separated in resolution and choices of the parameters S and B in Algorithm 1.

For images we observe a decrease in relative runtimes with higher resolution, while the average relative error is independent of the image resolution. In the point cloud case, however, the relative error increases slightly with the instance size. The number S of sampled points seems to considerably affect the relative error. An increase of the number of points results in more accurate values, with average relative errors as low as about 3% for $S = 4000$, while still maintaining a speedup of two orders of magnitude on 128×128 images. Lower sample sizes yield higher average errors, but also lower runtimes. With $S = 500$ the runtime is reduced by over four orders of magnitude with an average relative error of less than 10%. As to be expected, runtime increases linearly with the number of repetitions B. However, the impact on the relative errors is rather inconsistent. This is due to the fact, that the costs returned by the subsampling algorithm are often overestimated, therefore averaging over multiple tries does not yield improvements (see Figure 4.4). This means that in order to increase the accuracy of the algorithm it is advisable to keep $B = 1$ and instead increase the sample size S. However, increasing B can be useful to lower the variability of the results.

On the contrary, there is a big difference in accuracy between the image classes. While Algorithm 1 has consistently low relative errors on the Cauchy Density images, the exact optimal costs for White Noise images cannot be approximated as reliably. The relative errors fluctuate more and are generally much higher, as one can see from Figure 4.5 (left). In images with smooth structures and regular features the subsamples are able to capture that structure and therefore deliver a more precise representation of the images

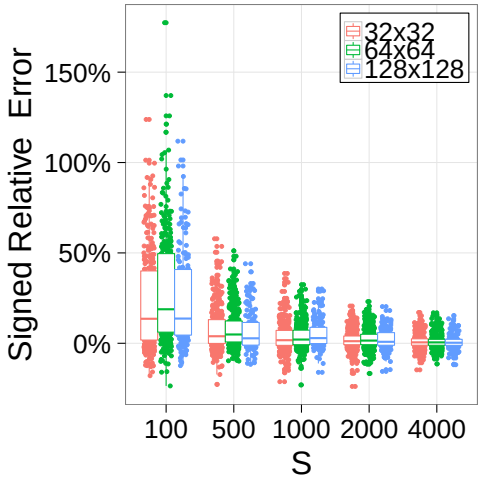


Figure 4.4: The signed relative approximation errors $\left(\hat{W}_p^{(S)}(\boldsymbol{r},\boldsymbol{s}) - W_p(\boldsymbol{r},\boldsymbol{s})\right)/W_p(\boldsymbol{r},\boldsymbol{s})$ show that the approximation overestimates the exact distance for small S but the bias vanishes for larger S.

and a more precise value. This is not possible in images that are very irregular or noisy, such as the White Noise images, which have no structure to begin with. The Classic Images contain both regular structures and more irregular regions. Therefore, their relative errors are slightly higher than in the Cauchy Density cases. The algorithm has a similar performance on the point cloud instances, that are modelled after the Cauchy Density and Classic Images classes, while the Dirichlet instances have a more desirable accuracy compared to the White Noise images, as seen in Figure 4.5 (right).

There are no significant differences in performance between the different back-end solvers for the Wasserstein distance. As Figure 4.6 shows, accuracy seems to be better for the Sinkhorn distance compared to the other three solvers which report the exact Wasserstein distance.

In the results of the point cloud instances we can observe the influence of the value $p' = (D/2) - p$ on the scaling of the relative error with the instance size N for constant sample size ($S = 4000$). This is shown in Figure 4.7. We observe an increase of the relative error with p', as expected from the theory. However, we are not able to clearly distinguish between the three cases $p' < 0$, $p' = 0$ and $p' > 0$. This might be due to the relatively small instance sizes

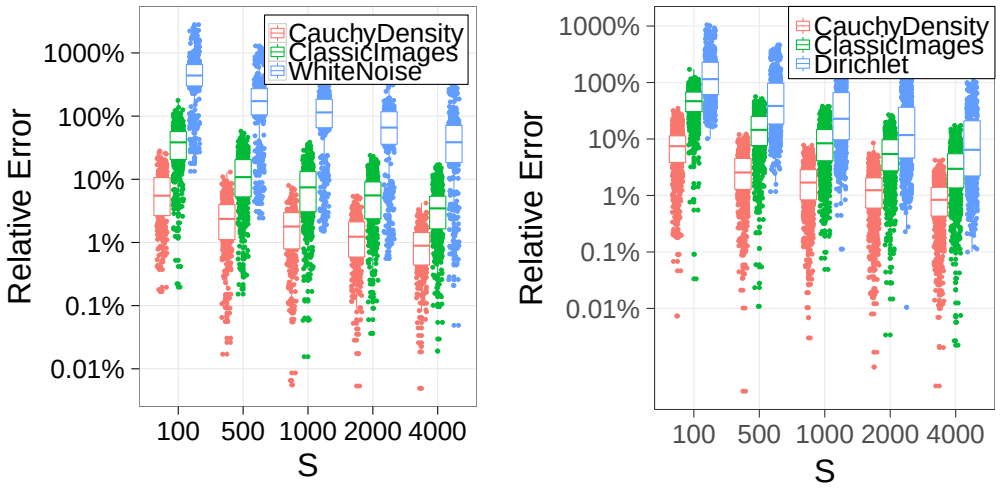


Figure 4.5: A comparison of the relative errors for different image classes (left) and point cloud instance classes (right).

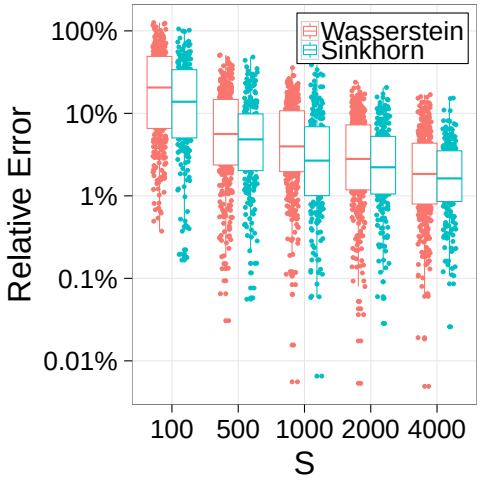


Figure 4.6: A comparison between the approximations of the Wasserstein and Sinkhorn distances.

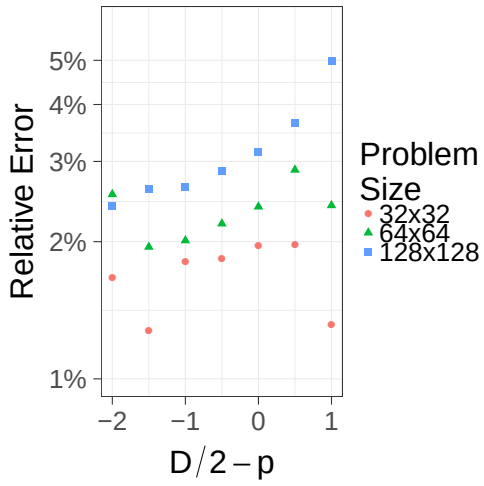


Figure 4.7: A comparison of the mean relative errors in the point cloud instances with sample size $S = 4000$ for different values of $p' = (D/2) - p$.

N in the experiments. While we see that the relative errors are independent of N in the image case (compare Figure 4.2), for the point clouds N has an influence on the accuracy that depends on p'.

4.5 Discussion

As our simulations demonstrate, subsampling is a simple, yet powerful tool to obtain good approximations to Wasserstein distances with only a small fraction of required runtime and memory. It is especially remarkable that in the case of two-dimensional images for a fixed amount of subsampled points, and therefore a fixed amount of time and memory, the relative error is independent of the resolution/size of the images. Based on these results, we expect the subsampling algorithm to return similarly precise results with even higher resolutions of the images it is applied to, while the effort to obtain them stays the same. Even in point cloud instances the relative error only scales mildly with the original input size N and is dependent on the value p'.

The numerical results (Figure 4.2) show an inverse polynomial decrease of the approximation error with S, in accordance with the theoretical results. As we see little dependence on the cost exponent p we speculate that the rate

$O(S^{-1/2p})$ might be improved upon. In fact, recent results on the asymptotic distribution of the empirical Wasserstein distance would suggest an $\mathcal{O}(S^{-1/2})$ rate [89].

When applying the algorithm, it is important to note that the quality of the returned values depends on the structure of the data. In very irregular instances it is necessary to increase the sample size in order to obtain similarly precise results, while in regular structures a small sample size suffices.

Our scheme allows the parameters to be easily tuned towards faster runtimes or more precise results, as desired. Increases and decreases of the sample size S are recommended to influence the performance in either direction, while the parameter B should only be increased if a particularly low variability of the estimate is required or if the repetitions can be computed in parallel. Otherwise, the higher runtime should be spent with a higher sample size (compare Figure 4.2).

The scheme presented here can readily be applied to other optimal transport distances, as long as a solver is available, as we demonstrated with the Sinkhorn distance [20]. Empirically, we can report good performance in this case, suggesting that entropically regularized distances might be even more amenable to subsampling approximation than the Wasserstein distance itself. Extending the theoretical results to this case would require an analysis of the mean speed of convergence of empirical Sinkhorn distances, which is an interesting task for future research.

All in all, subsampling proves to be a general, powerful and versatile tool that can be used with virtually any optimal transport solver as back-end and has both theoretical approximation error guarantees, and a convincing performance in practice. It is a challenge to extend this method in a way which is specifically tailored to the geometry of the underlying space $\mathcal{X}$, which may result in further improvements.

Chapter 5

Cost-Based Clustering: A General Multiscale Approach to Discrete Optimal Transport

5.1 Introduction

As previously discussed, discrete optimal transport problems are considered in many applications (see Section 1.2 for an overview). Despite the availability of general purpose algorithms, such as the transportation simplex, due to large-scale instances more modern and efficient methods are required. Several, sometimes very recently introduced algorithms are described in Chapter 2. We summarize a number of strategies employed by these methods to push the boundaries of viability of large-scale optimal transport:

- *Performance optimization*: Through code optimization, parallelization or simply the usage of more modern hardware, the runtimes of methods can be lowered. Commercial solvers are more and more efficient and performance enhancing variations, such as the shortlist method [39] compared to the transportation simplex, have been introduced.

- *Exploitation of geometric structures*: The transportation simplex and similar algorithms only make use of values of the cost function as

coefficients, but are oblivious of any underlying geometric structure. For Wasserstein distance computation in Euclidean spaces in particular, this means missed potential. Methods making use of the geometry include the shielding method [76], the AHA method [57] and others.

- *Approximative methods*: If it is not necessary to compute an exact solution or the exact optimal transport cost, approximative methods can be applied. Options include primal or dual heuristics [81], the entropically regularized optimal transport [20] or randomized approximation via subsampling [90] (see Chapter 4).

- *Multiscale methods*: These are approaches to reduce the size of the problem (coarsening), solve the smaller instance more quickly and derive from this a solution to the original problem (propagation). There are different strategies for the coarsening, the propagation and refinement of solutions. See Section 2.4 including the references for a more thorough introduction to multiscale methods.

Despite these advancements, some exceedingly difficult problems remain. If approximations are not an option, the instance at hand does not have an exploitable geometric structure and standard methods – however optimized – are still too time- or memory-consuming, the last option to rely on are multiscale approaches. We give the following prototypical example of such a problem:

Example 5.1.1. In economics, optimal transport problems arise in matching models, such as labor market matching. Workers with different properties must be matched with jobs with different requirements, such that the weighted total benefit is maximized. Each pair of worker and job carries a benefit value, similar to cost values between sources and sinks in the regular optimal transport problems. However, workers and jobs cannot easily be embedded into a metric space in a way that expresses the benefit values in terms of the distances. This makes it impossible to apply many of the best performing strategies. Labor market and other matching problems are brought up for example in [32].

Adapting multiscale methods to cases like this, however, is not an easy task. The main problem is that many coarsening strategies rely on geometric structure as well, for example, a grid. Popular clustering methods, which can be considered for coarsening, such as k-means clustering and similar, also require point clouds in a Euclidean space. Therefore, it is not guaranteed that these algorithms can be applied in general. We introduce a clustering method in this chapter that has no requirements on the cost function in order to achieve the highest level of generality and make multiscale methods available for any optimal transport problem. We name the new method *cost-based clustering*. While it is driven by available data instead of geometry, this approach also offers advantages in some Euclidean situations, particularly in high dimensions, and we highlight examples in Section 5.5.

Discrete Optimal Transport In this chapter we focus on discrete optimal transport. The problem statement is the same as given in Section 1.1. We identify measures μ on $X = \{x_1, \dots x_N\}$ and ν on $Y = \{y_1, \dots y_M\}$ with mass vectors, say $\mu_i = \mu(x_i)$ for all $i = 1, \dots, N$ and $\nu_j = \nu(y_j)$ for all $j = 1, \dots, M$. Similarly, we identify the cost function $c\colon X \times Y \to \mathbb{R}_+$ with a cost matrix $C = (c_{i,j})_{\substack{1\le i\le N\\ 1\le j\le M}} \in \mathbb{R}_+^{N\times M}$ with $c_{i,j} = c(x_i, y_j)$ for all i, j and view couplings π as matrices $\pi \in \mathbb{R}_+^{N\times M}$ as well.

We consider the following linear program with variables $\pi_{i,j}$:

$$
\begin{aligned}
\text{(DOT)} \qquad \min \quad & \langle C,\pi\rangle \\
\text{subject to} \quad & \sum_{j=1}^{M} \pi_{i,j} = \mu_i \ \forall i = 1, \dots, N \\
& \sum_{i=1}^{N} \pi_{i,j} = \nu_j \ \forall j = 1, \dots, M \\
& \pi_{i,j} \ge 0
\end{aligned}
$$

Here, $\langle\cdot,\cdot\rangle$ between matrices stands for the Frobenius inner product. In what follows, inequalities between matrices are interpreted as componentwise inequalities. Although the notation is mostly identical to the setup in Sec-

tion 1.1, note that the numbers of elements in X and Y are denoted with capital letters N and M here. An instance of DOT is entirely characterized by μ, ν and C. It is feasible if and only if $\mu(X) = \nu(Y)$ and we always assume that this is the case.

We introduce a multiscale approach to this general setting. See Section 2.4 and references therein for a description of the main ideas behind multiscale methods. In what follows, the focus lies on cost-based clustering as a way to make a coarsening method available in general. We also introduce a propagation algorithm which makes use of the way the instance was coarsened. While multiscale methods usually operate on more than two scales, we only consider one coarsening step here, since the coarsening and propagation techniques can be iterated. This means, we need to find partitions for the sets X and Y, as well as a meaningful cost matrix on the subsets.

5.2 The Cost-Based Clustering Problem

Finding partitions of X and Y that permit a good representation of the original problem in the multiscale scheme in the general setting is not an easy task in general, because we might not have a geometrical structure or a metric space; and therefore, classical clustering algorithms cannot be applied. In fact, we need to find clusterings based only on the measures μ and ν, as well as the cost matrix C.

However, for a given clustering we can show bounds on the absolute deviation of the optimal transport cost of the clustered problem from the original if a suitable cost matrix between clusters is chosen. Based on this result, we can define the cost-based clustering problem as follows: Find the clusterings of X and Y into n and m subsets, respectively, which minimize a suitable cost deviation bound. We state the problem first and prove the bounds afterwards.

Definition 5.2.1. Let $X = \{x_1, \dots, x_N\}$ and $Y = \{y_1, \dots, y_M\}$, μ and ν be measures on X and Y, respectively, and C a cost matrix. For $n, m \in \mathbb{N}$ and partitions $\mathcal{X} = \{X_1, \dots, X_n\}$ and $\mathcal{Y} = \{Y_1, \dots, Y_m\}$ of X and Y into n and

m subsets, respectively, we define

- the *minimum cost matrix* $C^{min}(\mathcal{X},\mathcal{Y}) \in \mathbb{R}_+^{n\times m}$ by

$$c^{min}_{k,l}(\mathcal{X},\mathcal{Y}) := \min_{\substack{x_i\in X_k\\ y_j\in Y_l}} c_{i,j} \text{ for all } k=1,\dots,n \text{ and } l=1,\dots,m,$$

- the *maximum cost matrix* $C^{max}(\mathcal{X},\mathcal{Y}) \in \mathbb{R}_+^{n\times m}$ by

$$c^{max}_{k,l}(\mathcal{X},\mathcal{Y}) := \max_{\substack{x_i\in X_k\\ y_j\in Y_l}} c_{i,j} \text{ for all } k=1,\dots,n \text{ and } l=1,\dots,m,$$

- and the *gap matrix* $G(\mathcal{X},\mathcal{Y}) := C^{max}(\mathcal{X},\mathcal{Y}) - C^{min}(\mathcal{X},\mathcal{Y})$ with entries $g_{k,l}(\mathcal{X},\mathcal{Y})$.

With this we define four *gap objective functions*:

$$G_{row}(\mathcal{X},\mathcal{Y}) := \sum_{k=1}^{n}\left(\mu(X_k)\cdot \max_{l=1,\dots,m} g_{k,l}(\mathcal{X},\mathcal{Y})\right),$$

$$G_{col}(\mathcal{X},\mathcal{Y}) := \sum_{l=1}^{m}\left(\nu(Y_l)\cdot \max_{k=1,\dots,n} g_{k,l}(\mathcal{X},\mathcal{Y})\right),$$

$$G_{min}(\mathcal{X},\mathcal{Y}) := \sum_{k=1}^{n}\sum_{l=1}^{m}\min\{\mu(X_k),\nu(Y_l)\}\cdot g_{k,l}(\mathcal{X},\mathcal{Y}),$$

$$G_{prod}(\mathcal{X},\mathcal{Y}) := \sum_{k=1}^{n}\sum_{l=1}^{m}\mu(X_k)\cdot\nu(Y_l)\cdot g_{k,l}(\mathcal{X},\mathcal{Y}).$$

When they are clear from the context, we leave out the arguments $(\mathcal{X},\mathcal{Y})$ of the matrices and functions above. We can now define the cost-based clustering problem.

Definition 5.2.2. Let X and Y be finite sets with measures μ and ν and a cost matrix C as above and let $n,m \in \mathbb{N}$. The *cost-based clustering problem* (CBC) with respect to G_* is to find clusterings $\mathcal{X}$ for X and $\mathcal{Y}$ for Y minimizing $G_*(\mathcal{X},\mathcal{Y})$, where G_* stands for one of the above gap objective functions $G_{row}, G_{col}, G_{min}$ and G_{prod}.

Usually, clusterings are meant to convey the similarity of points or elements in the same subset in some sense. Since we have no direct notion of similarity or dissimilarity between points in X in this general setting of optimal transport, we need to measure similarity with respect to the costs to the points in Y. Given two clusters X_k and Y_l, the gap matrix entry $g_{k,l}$ represents how different the points in X_k are with respect to the costs to the points in Y_l and vice versa. Therefore, we generally prefer small entries in the gap matrix.

The four objective functions are different weightings of the gap matrix. G_{row}, G_{col} and G_{min} have an additional meaning as deviation bounds, whereas G_{prod} can lead to interesting clusterings in practice (see Section 5.4.2 for results). Every function defines one version of CBC. Unfortunately, finding the optimal clustering for any of the objective functions is hard.

Theorem 5.2.3. *CBC is NP-hard for any* $G_* \in \{G_{row}, G_{col}, G_{min}, G_{prod}\}$.

We postpone the proof of this theorem to Section 5.7.

5.2.1 Deviation Bounds of Clusterings

As mentioned, three of the gap objective values for a clustering, and hence their minimum, serve as a bound for the absolute deviation of the optimal cost value on the clusters from the desired optimal cost value if the cost matrix between clusters is chosen suitably.

Theorem 5.2.4. *Let* $\mathcal{X} = \{X_1, \dots, X_n\}$ *and* $\mathcal{Y} = \{Y_1, \dots, Y_m\}$ *be clusterings of* X *and* Y*, respectively. Let* $\bar{C} \in \mathbb{R}_+^{n \times m}$*, such that*

$$C^{min}(\mathcal{X}, \mathcal{Y}) \leq \bar{C} \leq C^{max}(\mathcal{X}, \mathcal{Y}).$$

Let $\bar{c}^*$ *be the optimal objective value of the optimal transport problem on the clusters defined by* $\mathcal{X}$*,* $\mathcal{Y}$ *and* $\bar{C}$ *and let* c^* *be the optimal cost value of the original problem. Then*

$$|\bar{c}^* - c^*| \leq \min\{G_{row}(\mathcal{X}, \mathcal{Y}), G_{col}(\mathcal{X}, \mathcal{Y}), G_{min}(\mathcal{X}, \mathcal{Y})\}.$$

Before we go into the proof, we introduce some notation. Let π^* be an optimal coupling for the original problem and let $\bar{\pi}^*$ and π^*_{min} be optimal couplings on the clusters with respect to $\bar{C}$ and C^{min}, respectively. We define $\tilde{C}^{min} \in \mathbb{R}_+^{N\times M}$ by $\tilde{c}^{min}_{i,j} := c^{min}_{k,l}$ for the k and l, such that $x_i \in X_k$ and $y_j \in Y_l$. Analogously, we define $\tilde{C}^{max} \in \mathbb{R}_+^{N\times M}$. Since $c^{min}_{k,l} \leq c_{i,j} \leq c^{max}_{k,l}$ for all $x_i \in X_k$, $y_j \in Y_l$, we have $\tilde{C}^{min} \leq C \leq \tilde{C}^{max}$. We denote an optimal coupling for the original problem, with respect to $\tilde{C}^{min}$ instead of C as $\tilde{\pi}^*_{min}$. Then we have the following lower and upper bounds on c^* and $\bar{c}^*$:

Lemma 5.2.5. *In the situation of Theorem 5.2.4 we have*

i) $c^*_{min} \leq c^* \leq \langle C^{max}, \pi^*_{min}\rangle$ *and*

ii) $c^*_{min} \leq \bar{c}^* \leq \langle C^{max}, \pi^*_{min}\rangle$,

where c^*_{min} *is the optimal cost value of the clustered problem with respect to* C^{min}.

Proof. The optimal transport instance on the clusters with respect to C^{min} is equivalent to the instance on X and Y with respect to $\tilde{C}^{min}$ in the sense that every feasible (and hence every optimal) solution to the former can be trivially propagated to a feasible (or optimal) solution to the latter with the same cost value. Vice versa, through aggregating transports according to the clustering every solution on the finer scale has a natural counterpart solution on the clusters with the same cost. Therefore, $\langle C^{min}, \pi^*_{min}\rangle = \langle \tilde{C}^{min}, \tilde{\pi}^*_{min}\rangle$ and similarly $\langle \tilde{C}^{max}, \tilde{\pi}^*_{min}\rangle = \langle C^{max}, \pi^*_{min}\rangle$. Since the problem with respect to $\tilde{C}^{min}$ is a relaxation of the original problem,

$$\begin{aligned} c^*_{min} &= \langle C^{min}, \pi^*_{min}\rangle = \langle \tilde{C}^{min}, \tilde{\pi}^*_{min}\rangle \leq \langle \tilde{C}^{min}, \pi^*\rangle \leq \langle C, \pi^*\rangle = c^* \\ &= \langle C, \pi^*\rangle \leq \langle C, \tilde{\pi}^*_{min}\rangle \leq \langle \tilde{C}^{max}, \tilde{\pi}^*_{min}\rangle = \langle C^{max}, \pi^*_{min}\rangle. \end{aligned}$$

The bounds on $\bar{c}^*$ follow from

$$c^*_{min} = \langle C^{min}, \pi^*_{min}\rangle \leq \langle C^{min}, \bar{\pi}^*\rangle \leq \langle \bar{C}, \bar{\pi}^*\rangle = \bar{c}^* \leq \langle \bar{C}, \pi^*_{min}\rangle \leq \langle C^{max}, \pi^*_{min}\rangle.$$

□

Proof of Theorem 5.2.4. It follows immediately from Lemma 5.2.5 that

$$|\bar{c}^* - c^*| \leq \langle C^{max}, \pi^*_{min} \rangle - c^*_{min} = \langle C^{max} - C^{min}, \pi^*_{min} \rangle = \langle G, \pi^*_{min} \rangle.$$

We denote the components of π^*_{min} as $\pi^*_{k,l}$ for $k = 1, \ldots, n$ and $l = 1, \ldots, m$. Then we have

$$\begin{aligned}
\langle G, \pi^*_{min} \rangle &= \sum_{k=1}^{n} \sum_{l=1}^{m} g_{k,l} \pi^*_{k,l} \leq \sum_{k=1}^{n} \max_{l=1,\ldots,m} g_{k,l} \cdot \sum_{l=1}^{m} \pi^*_{k,l} \\
&= \sum_{k=1}^{n} \mu(X_k) \cdot \max_{l=1,\ldots,m} g_{k,l} = G_{row}, \\
\langle G, \pi^*_{min} \rangle &= \sum_{l=1}^{m} \sum_{k=1}^{n} g_{k,l} \pi^*_{k,l} \leq \sum_{l=1}^{m} \max_{k=1,\ldots,n} g_{k,l} \cdot \sum_{k=1}^{n} \pi^*_{k,l} \\
&= \sum_{l=1}^{m} \nu(Y_l) \cdot \max_{k=1,\ldots,n} g_{k,l} = G_{col}, \\
\langle G, \pi^*_{min} \rangle &= \sum_{k=1}^{n} \sum_{l=1}^{m} g_{k,l} \pi^*_{k,l} \leq \sum_{k=1}^{n} \sum_{l=1}^{m} \min\{\mu(X_k), \nu(Y_l)\} \cdot g_{k,l} = G_{min}
\end{aligned}$$

and hence

$$|\bar{c}^* - c^*| \leq \min\{G_{row}, G_{col}, G_{min}\}.$$

□

Theorem 5.2.4 gives us rigorous bounds for the deviation of a clustered solution with respect to a suitable cost matrix $\bar{C}$ from the original for a given clustering. The same is true for Lemma 5.2.5, since it gives us concrete upper and lower bounds on c^*. However, in order to obtain these, π^*_{min} has to be known, that means an additional optimal transport instance on the clusters has to be solved to optimality. This makes the bounds from Lemma 5.2.5 impractical to use as an objective funciton for the clustering process.

When a clustering is given and $\bar{c}^*$ is computed for some $\bar{C}$ between C^{min} and C^{max} we can give an interval of possible values for c^*, namely

$$c^* \in [\bar{c}^* - \gamma, \bar{c}^* + \gamma],$$

where $\gamma = \min\{G_{row}, G_{col}, G_{min}\}$ for the clustering at hand. If in addition we solve the instance on the clusters with respect to C^{min} and have access to π^*_{min} we can compute the lower and upper bounds in Lemma 5.2.5 directly and obtain the interval

$$c^* \in [c^*_{min}, \langle C^{max}, \pi^*_{min} \rangle] ,$$

which has a length of $\langle G, \pi^*_{min} \rangle \leq \gamma$ instead of 2γ. This means there is always the option to cut the interval of potential cost values in half by computing π^*_{min}. Another option is to choose $\bar{C} = C^{min}$, which accomplishes both at the same time. However, the approximation $\bar{c}^*$, which is then a lower bound, will probably be less precise. Compare Section 5.4.2 for how the choice of $\bar{C}$ has an influence on the difference between $\bar{c}^*$ and c^*.

Another important aspect is that if the cost-based clustering is used as a setup for a multiscale scheme and a feasible solution for the finer scale (here on X and Y) is constructed via the propagation method presented in Section 5.3.3, we obtain an additional upper bound, which can in practice be significantly tighter than the upper bounds provided here (compare Section 5.4.2). Therefore, in that case the value of additionally computing π^*_{min} might be diminished.

5.2.2 Comparison to Other Clustering Objectives and Deviation Bounds

As mentioned already, the main difference between the cost-based clustering and other clustering problems is that we operate on two sets X and Y with a cost function $c\colon X \times Y \to \mathbb{R}_+$ instead of a distance function $d\colon X \times X \to \mathbb{R}_+$ on only one space X. However, even if the optimal transport instance at hand takes place in $\mathbb{R}^D$ for $D \in \mathbb{N}$ and $c(x, y) = \|x - y\|^p$ as cost function, the objective functions are different to other clustering problems, such as k-means clustering. This means while there are established algorithms computing clusterings, they might not yield clusterings well suited for our purpose of minimizing the error bounds.

For example, most clustering algorithms aggregate points, which are close together. While closeby points likely have similar distances to points in the other set, the same might be true for points which are far from each other. The following example illustrates an extreme case of this effect achieved by symmetry.

Example 5.2.6. Suppose we have two sets with four points each as shown in Figure 5.1, left, both equipped with uniform distributions, and $c(x_i, y_j) = \|x_i - y_j\|^2$ for all x_i, y_j. Now suppose we want to find two clusters for X, while the trivial clustering $\mathcal{Y} = \{\{y_1\}, \{y_2\}, \{y_3\}, \{y_4\}\}$ of Y is fixed. Most proximity-based clustering algorithms find the clusters $\{x_1, x_2\}$ and $\{x_3, x_4\}$, as indicated by the ellipses in Figure 5.1, right, which we would agree with judging by geometric intuition. However, because of the symmetry, the cost matrix

$$C = \begin{pmatrix} 2 & 1 & 2 & 5 \\ 5 & 2 & 1 & 2 \\ 2 & 1 & 2 & 5 \\ 5 & 2 & 1 & 2 \end{pmatrix}$$

has two pairs of identical rows, the first and third, as well as the second and forth row. This means the clustering $\{x_1, x_3\}$ and $\{x_2, x_4\}$ has a gap matrix with only zero entries, which makes it the optimal clustering with respect to the objective functions in the cost-based clustering problem. In fact, since the objective value is zero, the optimal transport instance on the clusters has the same optimal cost as the original instance, since no information is lost due to the clustering.

Of course, this is a small example only constructed for the purpose of showing this effect and it is reliant on some kind of (approximative) symmetry. Usually, geometric clustering algorithms produce clusterings, which are reasonable in terms of their objective values of the gap functions in CBC. There are some situations, however, in which a symmetry effect is in place. For example, if X and Y are concentrated on lower dimensional affine subspaces A and B of $\mathbb{R}^D$, which are orthogonal to each other, optimal clusterings have concentric spherical clusters around the intersection point

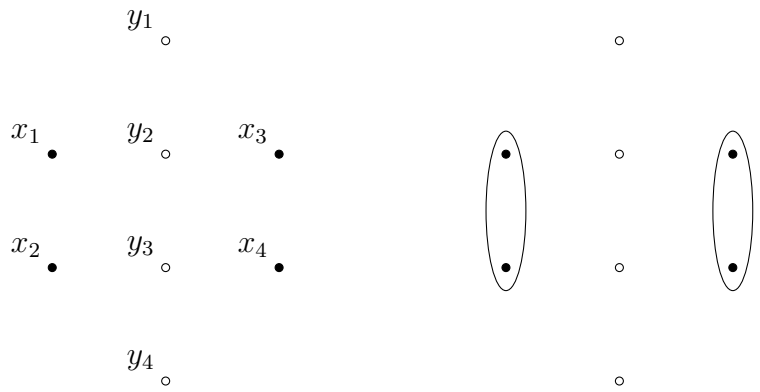


Figure 5.1: This example illustrates that aggregating close points can lead to a suboptimal clustering with respect to the objective functions of the cost-based clustering problem.

between A and B. Such clusterings are typically not found by traditional geometric clustering algorithms. This is discussed further in Section 5.5.

A different error bound is given by the Wasserstein metric, if we assume we have a metric space (X, d) with probability measures μ and ν with finite support in X and $c(x, y) = d^p(x, y)$. The optimal transport cost c^* between μ and ν is the p-th power of the Wasserstein distance between μ and ν,

$$c^* = W_p^p(\mu, \nu).$$

Any clusterings of the supports of μ and ν together with representatives in X for each cluster define approximative probability measures $\hat{\mu}$ and $\hat{\nu}$ with

$$\bar{c}^* = W_p^p(\hat{\mu}, \hat{\nu}).$$

Due to the Wasserstein metric triangle inequality, we have

$$|W_p(\mu, \nu) - W_p(\hat{\mu}, \hat{\nu})| \leq W_p(\mu, \hat{\mu}) + W_p(\nu, \hat{\nu}),$$

and hence

$$|c^{*\frac{1}{p}} - \bar{c}^{*\frac{1}{p}}| \leq W_p(\mu, \hat{\mu}) + W_p(\nu, \hat{\nu}).$$

The distances $W_p(\mu, \hat{\mu})$ are often easily computed. Let, for example,

$\operatorname{supp}(\mu) = \{x_1, \dots, x_N\} \subseteq X$ with clusters $X_1, \dots, X_n$ and representatives $\hat{x}_1, \dots \hat{x}_n \in X$. If each point x_i is contained in the cluster X_k with the closest representative with respect to d, then

$$W_p^p(\mu, \hat{\mu}) = \sum_{k=1}^{n} \sum_{x_i \in X_k} \mu(x_i) d^p(x_i, \hat{x}_k),$$

which is the case for the k-means algorithm or the standard grid-based aggregation for multiscale schemes on images. Note that this is an error bound between the p-th roots of the two optimal objective values, while the gap value given by the cost-based clustering provides an error bound on the absolute deviation of the optimal cost values. The next example shows a comparison of the two bounds in the grid-based case.

Example 5.2.7. Let $X = Y = \{1, \dots, R\}^2 \subseteq \mathbb{R}^2$ for some even number $R \in 2\mathbb{N}$ with $d(x, y) = \|x - y\|$ and consider the clustering $\mathcal{X} = \{X_1, \dots, X_n\}$ with $n = R^2/4$, where four adjoining points are aggregated and the representatives $\hat{x}_k$ of the clusters are the centers of the four clustered points. This defines $\hat{\mu}$ and $\hat{\nu}$ on $\mathcal{X}$ for probability measures μ and ν on X as detailed in Section 2.4 under the paragraph *Coarsening*. Since $d^p(x_i, \hat{x}_k) = 2^{-\frac{p}{2}}$ for all X_k and $x_i \in X_k$, we have

$$|c^{*\frac{1}{p}} - \bar{c}^{*\frac{1}{p}}| \leq W_p(\mu, \hat{\mu}) + W_p(\nu, \hat{\nu}) = 2 \cdot \left(2^{-\frac{p}{2}}\right)^{\frac{1}{p}} = \sqrt{2}.$$

The error bounds of the cost-based clustering are more difficult to compute and often depend on μ and ν, since the values in the gap matrix range between $d^p((1,1),(2,2)) = 2^{\frac{p}{2}}$ and

$$\begin{aligned} &d^p((1,1),(R,R)) - d^p((2,2),(R-1,R-1)) \\ &= \left((R-1)\sqrt{2}\right)^p - \left((R-3)\sqrt{2}\right)^p = 2^{\frac{p}{2}} \cdot \left((R-1)^p - (R-3)^p\right). \end{aligned}$$

For $p = 1$, however, we can compute G_{row} and G_{col} directly. For every cluster X_k there exists another cluster X_l, which is diagonally aligned (assuming $R \geq 4$), such that the difference between the maximal and the minimal distances is precisely twice the diameter of the clusters, thus $g_{k,l} = 2\sqrt{2}$.

This is in line with the more general maximal gap matrix entry given above. Therefore,

$$G_{row} = G_{col} = 2\sqrt{2}.$$

G_{min} depends on μ and ν with the best case being $\sqrt{2}$ and the worst case being $2\sqrt{2}$, depending on how the masses of μ and ν are distributed. In any case,

$$|c^* - \bar{c}^*| \leq W_1(\mu, \hat{\mu}) + W_1(\nu, \hat{\nu}) \leq G_{min} \leq G_{row} = G_{col},$$

meaning that the Wasserstein bound is the tightest of the bounds, G_{row} and G_{col} are the least tight and G_{min} is in between depending on μ and ν. For $p > 1$ these bounds cannot be compared directly, since the Wasserstein bound applies to $|c^{*\frac{1}{p}} - \bar{c}^{*\frac{1}{p}}|$, whereas the deviation bound from the cost-based clustering applies to $|c^* - \bar{c}^*|$.

5.3 Clustering Algorithms

Despite being an NP-hard problem, there are some fast polynomial algorithms producing clusterings that are maybe not optimal with respect to the objective function, but reasonable approximate solutions to the cost-based clustering problem. The same is true for most clustering problems, since they are typically NP-hard and tackled via heuristic methods, such as k-means, k-center and similar geometric algorithms, via hierarchical clusterings, such as agglomerative and divisive clustering, and via other methods.

When adapting established clustering methods to our problem, we need to pay attention to a couple of features in particular:

- We cannot rely on a metric or distance function between elements in X, since we only have a function measuring the costs between elements in X and Y. This immediately rules out a lot of the geometric methods like k-means clustering. While there are of course optimal transport instances in metric or Euclidean spaces that can be handled with these clustering methods, this work is focused on the general case.

- Even in the case of optimal transport in a Euclidean space it is not

ensured that geometric clustering methods such as k-means lead to clusterings, which are preferable with respect to the objective functions of the cost-based clustering problem. This is highlighted in Example 5.2.6 and more examples can be found in Section 5.5.

- Runtime has to be a major focus of the clustering methods. Since the general discrete optimal transport problem on N elements can be solved in $\mathcal{O}(N^3 \log(N))$ time with the auction algorithm [10] and sub-cubic empirical runtimes are possible with a specialized transportation simplex method [39], it is important to keep the clustering runtimes low, as cost-based clustering is meant to accelerate the solution process in a multiscale scheme.

5.3.1 Agglomerative Clustering

One of the heuristic clustering methods that can be adapted to the cost-based clustering problem is the agglomerative clustering. It is a hierarchical bottom-up clustering approach, where every element starts as one cluster and in each step two current clusters are selected to be merged until the desired number of clusters is reached. The selection happens greedily with respect to a linkage criterion, that is usually tied to an objective function or a dissimilarity function between the elements.

Agglomerative clustering methods go back to Florek et al. in 1951 [27], who suggested a nearest-neighbor rule, which is now known as single-linkage. Later, more efficient clustering algorithms for single-linkage [85] and complete-linkage [22] were developed in 1973 and 1977, respectively. The distance between clusters is evaluated by the shortest distance between elements in single-linkage, and by the farthest distance between elements in complete-linkage.

Unfortunately, those and many other linkage criteria require a distance or dissimilarity function $d\colon X \times X \to \mathbb{R}_+$, also called dissimilarity coefficient, whereas we only have access to a cost (or dissimilarity) function $c\colon X \times Y \to \mathbb{R}_+$ between elements in X and Y. This is why we suggest linkage criteria based on the objective functions of the cost-based clustering problem $G_{row}, G_{col}, G_{min}$

and G_{prod}.

Starting with $\mathcal{X} = \{\{x_1\}, \ldots, \{x_N\}\}$ and $\mathcal{Y} = \{\{y_1\}, \ldots, \{y_M\}\}$, the gap matrix is the zero matrix. Whenever we fuse two clusters of $\mathcal{X}$ (or $\mathcal{Y}$) we eliminate one row (column) of the gap matrix and another row (column) increases its values. This leads to an increase of the gap objective function values, that depends on the choice $G_* \in \{G_{row}, G_{col}, G_{min}, G_{prod}\}$. We can compute this increase in advance in order to choose the two clusters to fuse, which lead to the least increase of the function. This process is iterated until the desired number of clusters is reached. The two sets X and Y are clustered in succession. Since this linkage criterion depends on the choice of G_*, it has to be chosen beforehand.

In each step of clustering X (and likewise when clustering Y with different notation), assuming N current clusters, the fusion matrix is an $\mathbb{R}_+^{N \times N}$ matrix F, such that $f_{k,l}$ is the increase in the chosen objective function if X_k and X_l are fused. Since it is symmetric with diagonal zero, we only need to keep track of the upper triangular part. In most of the cases, if we fuse two clusters X_k and X_l, the other entries do not change, and we only have to compute the entries for the new cluster $X_k \cup X_l$. This leads to Algorithm 2 below. However, if we cluster X with respect to G_{col} or Y with respect to G_{row}, due to the column-wise (or row-wise) maximum of the gap matrix in the objective function, all of the values in the fusion matrix might change after the aggregation of two clusters. This means we either have to recalculate the entire fusion matrix after each step (Algorithm 3) or accept that the remaining values in the matrix are inaccurate (Algorithm 2). Another option is to recalculate the matrix, whenever the number of clusters is halved, or similar. This leads to variants of Algorithms 2 and 3.

Some details on the implementation:

- In addition to G we always keep track of C^{min} and C^{max} during the clustering. The reason is simply that they are needed to compute the fusion values $f_{i,j}$ and it would be inefficient to recalculate the required values each time. For G_{row} (G_{col}) we also keep track of the row-wise (column-wise) maximum of G, that is, $G_{k,\cdot} = \max_l g_{k,l}$ or $G_{\cdot,l} = \max_k g_{k,l}$.

Algorithm 2: Agglomerative cost-based clustering

Input : $X, Y, \mu, \nu, C, n, m, G_*$
Output : Clusterings $\mathcal{X}, \mathcal{Y}$ with gap matrix G
Initialize
$\mathcal{X} \leftarrow \{\{x_1\}, \dots \{x_N\}\}, \mathcal{Y} \leftarrow \{\{y_1\}, \dots \{y_M\}\}, G \leftarrow \mathbf{0} \in \mathbb{R}^{N \times M}$
Initialize F ($N \times N$ fusion matrix for $\mathcal{X}$) by computing $f_{i,j}$ for
$i = 1, \dots, N-1, j = i+1, \dots, N$ with respect to G_*, for other (i, j):
$f_{i,j} \leftarrow \infty$
while $|\mathcal{X}| > n$ **do**
  $(k', l') \leftarrow \operatorname{argmin}_{k,l} f_{k,l}$
  $X_{k'} \leftarrow X_{k'} \cup X_{l'}$
  $X_{l'} \leftarrow \emptyset$ (remove from $\mathcal{X}$)
  Update G, μ and F
end
Initialize F ($M \times M$ fusion matrix for $\mathcal{Y}$)
while $|\mathcal{Y}| > m$ **do**
  $(k', l') \leftarrow \operatorname{argmin}_{k,l} f_{k,l}$
  $Y_{k'} \leftarrow Y_{k'} \cup Y_{l'}$
  $Y_{l'} \leftarrow \emptyset$ (remove from $\mathcal{Y}$)
  Update G, ν and F
end

Algorithm 3: Agglomerative cost-based clustering with recalculation of the fusion matrix (only relevant for G_{row} and G_{col})

Input : $X, Y, \mu, \nu, C, n, m, G_*$
Output : Clusterings $\mathcal{X}, \mathcal{Y}$ with gap matrix G
if $G_* = G_{col}$ **then**
  Perform this algorithm for $Y, X, \nu, \mu, C^t, m, n, G_{row}$
end
else
  Initialize
  $\mathcal{X} \leftarrow \{\{x_1\}, \dots \{x_N\}\}, \mathcal{Y} \leftarrow \{\{y_1\}, \dots \{y_M\}\}, G \leftarrow \mathbf{0} \in \mathbb{R}^{N \times M}$
  Fuse the clusters of X as in Algorithm 2
  for $K = M, \dots, m+1$ **do**
    Calculate the $K \times K$ fusion matrix F for $\mathcal{Y}$
    $(k', l') \leftarrow \operatorname{argmin}_{k,l} f_{k,l}$
    $Y_{k'} \leftarrow Y_{k'} \cup Y_{l'}$
    $Y_{l'} \leftarrow \emptyset$ (remove from $\mathcal{Y}$)
    Update G and ν
  end
end

- When updating F in Algorithm 2 after the fusion of X_i and X_j, we remove rows and columns i and j from F, then add a new row and column for the new cluster $X_i \cup X_j$.

- We compute $f_{i,j}$ for two clusters X_i, X_j depending on G_* as follows

  – for G_{row}:

$$f_{i,j} := \mu(X_i \cup X_j) \cdot \max_{l=1,\dots,M} \left\{ \max\{c_{i,l}^{max}, c_{j,l}^{max}\} - \min\{c_{i,l}^{min}, c_{j,l}^{min}\} \right\} - \mu(X_i) \cdot G_{i,\cdot} - \mu(X_j) \cdot G_{j,\cdot}$$

  – for G_{col}:

$$f_{i,j} := \sum_{l=1}^{M} \nu(Y_l) \cdot \Big(\max \left\{ \max\{c_{i,l}^{max}, c_{j,l}^{max}\} - \min\{c_{i,l}^{min}, c_{j,l}^{min}\}, G_{\cdot,l} \right\} - G_{\cdot,l} \Big)$$

– for G_{min}:

$$f_{i,j} := \sum_{l=1}^{M} \Big(\min\{\mu(X_i \cup X_j), \nu(Y_l)\} \cdot \Big(\max\{c_{i,l}^{max}, c_{j,l}^{max}\} - \min\{c_{i,l}^{min}, c_{j,l}^{min}\}\Big) - \min\{\mu(X_i), \nu(Y_l)\} \cdot g_{i,l} - \min\{\mu(X_j), \nu(Y_l)\} \cdot g_{j,l}\Big)$$

– for G_{prod}:

$$f_{i,j} := \sum_{l=1}^{M} \Big(\nu(Y_l) \cdot \Big(\mu(X_i \cup X_j) \cdot \Big(\max\{c_{i,l}^{max}, c_{j,l}^{max}\} - \min\{c_{i,l}^{min}, c_{j,l}^{min}\}\Big) - \mu(X_i) \cdot g_{i,l} - \mu(X_j) \cdot g_{j,l}\Big)\Big)$$

- In the update steps of G, when X_i and X_j (with $i < j$) are fused, we remove row j and update row i of G. The same for columns in the clustering process of Y.
- We update μ by $\mu_i := \mu_i + \mu_j$ and remove entry j. The same for ν.
- Usually, Algorithm 2 is used for clustering. For the linkage criteria G_{row} and G_{col} a Boolean option `accel` indicates, whether Algorithm 2 (`accel` = `true`) or Algorithm 3 (`accel` = `false`) is used.
- The clustering for (X, Y, μ, ν, C, n, m, G_{row}) via the agglomerative clustering method is the same as the clustering for (Y, X, ν, μ, C^t, m, n, G_{col}). This allows us to always do the unproblematic clustering first in Algorithm 3, so that the clustering where the entire matrix is recalculated in each step is performed with an already reduced number of clusters for the other set. For $G_* \in \{G_{min}, G_{prod}\}$ the clustering for (X, Y, μ, ν, C, n, m, G_*) is the same as the clustering for (Y, X, ν, μ, C^t, m, n, G_*), since G_{min} and G_{prod} are both symmetric in X and Y.

In order to analyze the runtime of the agglomerative clustering method, we focus on the clustering of X first and observe that each computation of

$f_{i,j}$ is done in $\mathcal{O}(M)$ time, independent from the choice of G_*. In Algorithm 2 the fusion matrix F is initialized with N^2 computations of $f_{i,j}$. That means the initialization step is done in $\mathcal{O}(MN^2)$ time. Within the while loop, for $K = |\mathcal{X}|$, choosing the argmin is done in $\mathcal{O}(K^2)$ time and in the update of F, one row is computed, which are K computations of $f_{i,j}$. The other operations are insignificant in runtime. The values of K range from N to $n+1$, hence the runtime of the while loop, and consequentially of the whole clustering of X, is $\sum_{K=n+1}^{N}(K^2 + MK) = \mathcal{O}(N^3 + MN^2)$.

The clustering of Y has only one difference, namely the computation of one value $f_{i,j}$ takes $\mathcal{O}(n)$ instead of $\mathcal{O}(N)$ time, since $|\mathcal{X}|$ has previously been reduced to n. The rest still holds with N and M interchanged, thus the runtime of the clustering of Y is $\mathcal{O}(M^3 + nM^2)$. As a whole, Algorithm 2 has an $\mathcal{O}(N^3 + MN^2 + nM^2 + M^3)$ runtime, thus is cubic in $\max\{M, N\}$.

For Algorithm 3 let us fix the linkage criterion G_{row} (for G_{col} we obtain the same with X and Y exchanged). The first clustering (i.e. of X) is the same as before, thus done in $\mathcal{O}(N^3 + MN^2)$ time. The clustering of Y requires the fusion matrix F to be computed within the loop, so the runtime is $\sum_{K=m+1}^{M} K^2 n = \mathcal{O}(M^3 n)$, which yields $\mathcal{O}(N^3 + MN^2 + nM^3)$ in total. For $N = M$ and $n = m$, for example, this is $\mathcal{O}(N^3 n)$, which is already prohibitively slow for the cost-based clustering as a multiscale method for optimal transport, since the original problem can already be solved in $\mathcal{O}(N^3 \log(N))$ time in the worst case and in order to remain under this order, one would have to choose $n = o(\log(N))$. Thankfully, it is not necessary at all to use Algorithm 3 for G_{min} and G_{prod} and there are alternatives to it for G_{row} and G_{col} as well, albeit with less accurate fusion matrices.

5.3.2 Clustering with Random Representatives

As the name suggests, clustering with random representatives is a probabilistic clustering method, which combines the idea of subsampling to get sparse subsets of X and Y with the idea of cluster representatives, which are present for instance in the k-means clustering algorithm.

Generally, this approach has a focus on shorter runtimes compared to the

agglomerative clustering. The aim here is to get a clustering by only looking at each element once and deciding which cluster would be the best fit for that element instead of evaluating the outcomes of all potential cluster fusions. This also means we do not minimize with respect to one of the gap objective functions, but instead try to cluster elements with similar row or columns in the cost matrix. This assignment criterion generally leads to smaller entries of the gap matrix and consequently to smaller gap objective values. Algorithm 4 shows the details of this method, which works as follows:

- First, we sample n elements $\{\hat{x}_1, \ldots, \hat{x}_n\}$ of X from the distribution μ and in the same way m elements $\{\hat{y}_1, \ldots, \hat{y}_m\}$ of Y from ν. These act as representatives for the n and m clusters, respectively.

- Then, we look at each element $x_i \in X$ and decide which cluster this element should be assigned to. To that end, with the parameter $q \geq 1$ we compute the q-th power of the ν-weighted l_q-distances from the i-th row in the cost matrix to each of the representatives' rows (compare Algorithm 4). The cluster whose representative has the lowest value is chosen for this element. The same is then repeated with each element $y_i \in Y$ and the columns of the cost matrix.

- Since the sampling of the representatives is random, the clustering outcome varies if we run the algorithm several times. In order to mitigate the randomness one can repeat this clustering process and then pick the clustering with the least gap value (that is, $\min\{G_{row}, G_{col}, G_{prod}\}$). This is done via a parameter $K \in \mathbb{N}$, which indicates K repetitions.

- In each assignment step of an element in X to a cluster we compute n values, which are sums of M summands each. In order to further decrease the run time, we add the option to only evaluate the differences of rows of the cost matrix with respect to the columns belonging to the representatives $\hat{y}_1, \ldots, \hat{y}_m$. This decreases the number of summands from M to m, which can mean a significant speedup, if m is small compared to M. However, since a large portion of the cost matrix is effectively ignored, this can lead to less precise results depending on

the initial sampling of representatives. This option is included via a Boolean parameter `accel`.

As long as we insist on incorporating the gap matrix into the assignment criterion, there is not much room to further decrease the runtime of the accelerated version of this method. One option would be to choose the subset of S elements in Y (or the subset of representatives of Y) with the largest mass with respect to ν and only take the entries of these columns, when assigning points of X to their clusters. This would change the effort to compute one value a_j in Algorithm 4 from $\mathcal{O}(M)$ or $\mathcal{O}(m)$ to $\mathcal{O}(S)$, which is an improvement, if S is small enough. Similarly, if we do have access to a distance function d on X, that is computable in $\mathcal{O}(1)$ time, we can use this by choosing $a_j = d(x_i, \hat{x}_j)$. However, if d is not connected to c in any way, this does not necessarily yield good clusterings with respect to the gap matrix.

In the general non-accelerated case, Algorithm 4 has a runtime of

$$\mathcal{O}(KNM \cdot \max\{n, m\}),$$

while the accelerated version has a runtime of

$$\mathcal{O}(Knm \cdot \max\{N, M\}).$$

If K is fixed and $N = M$, both are at most cubic in N and hence below the order $\mathcal{O}(N^3 \log(N))$ of the original problem, since $n = \mathcal{O}(N)$ and $m = \mathcal{O}(M)$. Moreover, if for example $n = m = \mathcal{O}(\sqrt{N})$, the accelerated algorithm has a quadratic runtime in N, while the non-accelerated algorithm terminates in $\mathcal{O}(N^{\frac{5}{2}})$ time.

5.3.3 Propagation

We propose a general propagation technique, which takes a feasible coarse coupling $\bar{\pi}$ on $\mathcal{X}$ and $\mathcal{Y}$ and outputs a feasible coupling π to the original problem. It works in the general discrete optimal transport setting, does not rely on any structure of the cost matrix and is independent of how the

Algorithm 4: Cost-based clustering with random representatives

Input : $X, Y, \mu, \nu, C, n, m, K, q, accel$
Output : Clusterings $\mathcal{X}, \mathcal{Y}$ with gap matrix G
for $k = 1, \ldots, K$ **do**
  Initialize $\mathcal{X}^{(k)} \leftarrow \{X_1^{(k)}, \ldots X_n^{(k)}\}, \mathcal{Y}^{(k)} \leftarrow \{Y_1^{(k)}, \ldots Y_m^{(k)}\}$, with $X_i^{(k)} = Y_j^{(k)} = \emptyset$
  Sample $\{\hat{x}_1, \ldots \hat{x}_n\} \subseteq X$ from μ without replacement
  Sample $\{\hat{y}_1, \ldots \hat{y}_m\} \subseteq Y$ from ν without replacement
  for $i = 1, \ldots, N$ **do**
    for $j = 1, \ldots, n$ **do**
      if *accel* **then**
        $a_j \leftarrow \sum_{t=1}^{m} \nu(\hat{y}_t) \cdot |c(x_i, \hat{y}_t) - c(\hat{x}_j, \hat{y}_t)|^q$
      else
        $a_j \leftarrow \sum_{t=1}^{M} \nu(y_t) \cdot |c(x_i, y_t) - c(\hat{x}_j, y_t)|^q$
      end
    end
    $l \leftarrow \operatorname{argmin}_j a_j$
    $X_l^{(k)} \leftarrow X_l^{(k)} \cup \{x_i\}$
  end
  for $i = 1, \ldots, M$ **do**
    for $j = 1, \ldots, m$ **do**
      if *accel* **then**
        $a_j \leftarrow \sum_{t=1}^{n} \mu(\hat{x}_t) \cdot |c(\hat{x}_t, y_i) - c(\hat{x}_t, \hat{y}_j)|^q$
      else
        $a_j \leftarrow \sum_{t=1}^{N} \mu(x_t) \cdot |c(x_t, y_i) - c(x_t, \hat{y}_j)|^q$
      end
    end
    $l \leftarrow \operatorname{argmin}_j a_j$
    $Y_l^{(k)} \leftarrow Y_l^{(k)} \cup \{y_i\}$
  end
end
$(\mathcal{X}, \mathcal{Y}) \leftarrow \operatorname{argmin}_{(\mathcal{X}^{(1)}, \mathcal{Y}^{(1)}), \ldots, (\mathcal{X}^{(K)}, \mathcal{Y}^{(K)})} \min\{G_{row}, G_{col}, G_{min}\}$
$G \leftarrow$ gap matrix with respect to $\mathcal{X}$ and $\mathcal{Y}$

clustering was obtained. The only necessary input are the original instance, the coupling $\bar{\pi}$ and the clustering itself with the gap matrix.

At its core this method is a greedy algorithm akin to the construction of feasible solutions of the transportation simplex method. The mass of every active transport of $\bar{\pi}$ is distributed among the elements in the respective clusters. This allows us to only look at isolated submatrices of the cost matrix in order to keep the runtime low. Within the submatrices the mass is propagated by a submatrix-minimum-rule which each time chooses the available transport with the least cost. The details can be found in Algorithm 5.

Algorithm 5: Propagation algorithm for the construction of a feasible solution to the original problem based on a solution on the clusters

Input : $\mathcal{X}, \mathcal{Y}, \mu, \nu, C, G, \bar{\pi}$
Output : π
$L \leftarrow$ list of transport indices $\{(k,l) \colon \bar{\pi}_{k,l} > 0\}$ in descending order of values $g_{k,l}$
for $(k,l) \in L$ *in order* **do**
  while $\bar{\pi}_{k,l} > 0$ **do**
    $(i', j') \leftarrow \operatorname{argmin}_{(i,j)} \{c_{i,j} \colon x_i \in X_k, y_j \in Y_l, \mu_i > 0, \nu_j > 0\}$
    $\gamma \leftarrow \min\{\mu_{i'}, \nu_{j'}, \bar{\pi}_{k,l}\}$
    $\pi_{i',j'} \leftarrow \gamma$
    $\bar{\pi}_{k,l} \leftarrow \bar{\pi}_{k,l} - \gamma$, $\mu_{i'} \leftarrow \mu_{i'} - \gamma$, $\nu_{j'} \leftarrow \nu_{j'} - \gamma$
  end
end

The reason why the non-zero transports are sorted by the values in the gap matrix is the following: If we look at one particular transport $\bar{\pi}_{k,l} > 0$ between clusters and fix the coupling π between the elements in these clusters, this might limit the options when propagating other active transports on the same clusters, that is, $\bar{\pi}_{k,l'}$ for some $l' \neq l$ or $\bar{\pi}_{k',l}$ for some $k' \neq k$, since some of the sources (μ_i with $x_i \in X_k$) or sinks (ν_j with $y_j \in Y_l$) might already be exhausted. In order to mitigate this limitation, we propagate those active transports first where the difference between the best possible choice (i.e., $c_{k,l}^{min}$) and the worst possible choice (i.e., $c_{k,l}^{max}$) is the largest, hence the non-zero transports with the largest gap matrix entries.

It is apparent that Algorithm 5 outputs a feasible coupling π if the initial coupling $\bar{\pi}$ was feasible on the clusters. The quality of π with respect to the original optimal transport problem, however, is highly dependent on the quality of $\bar{\pi}$, as well as the quality of the clustering itself. For example, if $\mathcal{X}$ and $\mathcal{Y}$ constitute a perfect clustering, that is, all entries of the gap matrix are zero, and $\bar{\pi}$ is an optimal solution to the optimal transport problem on the clusters with $\bar{C} = C^{min} = C^{max}$, then π is automatically an optimal coupling. This is not guaranteed in any other case, however.

Note that while the solution is always feasible, it might not be a *basic* feasible solution, since we look at non-zero transports on the clusters separately, and that means we might not deplete a source or sink when depleting $\bar{\pi}_{k,l}$ (in Algorithm 5 this is the case $\gamma < \mu_{i'}$ and $\gamma < \nu_{j'}$ in the while-loop). If it is necessary to obtain a basic solution in order to apply a refinement strategy, for example a simplex method, the algorithm can be extended to account for that. If there are cycles in the matrix π they include one of the elements where neither source nor sink is depleted, therefore by keeping track of all of these entries, all cycles can easily be detected and eliminated.

5.4 Simulations

The clustering algorithms described in the previous section have been implemented in *R* and tested for their performance in runtime and quality of the resulting clusterings measured by the gap objective functions. The propagation algorithm was implemented as well and used to compare the resulting feasible solutions to the original problems of the different clusterings through the original cost function. In this section, we describe in detail under which conditions and on which instances these tests were conducted and their results.

5.4.1 Simulation Setup

In order to be able to test many combinations of parameters of the clustering algorithms, we restricted ourselves to three instances of optimal transport.

The three instances were taken from the DOTmark: images 1 and 2 from the three classes *Cauchy Density*, *Classic Images* and *Shapes*, respectively, in 32×32 resolution, that is, $X = Y = \{1, \dots, 32\}^2$ with $c(x_i, y_j) = \|x_i - y_j\|^2$. Descriptions of the different classes and the DOTmark altogether can be found in [78] and Chapter 3.

This choice allows us to look at instances with structually different measures μ and ν, while they are still supported on a grid, so that we can compare the results of the clustering methods above to the standard coarsening of images used in classic multiscale methods. The runtimes are the main reason why we only conduct the test on 32×32 instances. Of course, finding clusterings via the methods described above is possible in larger instances. However, especially Algorithm 3 already takes a long time on 64×64 instances, which is why we decided against testing all combinations of parameters on larger instances in this simulation study.

For the agglomerative clustering, Algorithm 2 was executed when G_* was chosen as either G_{min} or G_{prod}. For G_{row} and G_{col} the option `accel` indicates, whether Algorithm 2 (`accel = true`) or Algorithm 3 was used (`accel = false`). The number of clusters n was chosen as 16, 64 or 264 in order to match the number of clusters we get through the grid-based coarsening and we always set $m = n$.

The same instances and numbers of clusters were used for clustering with random representatives. Additionally, Algorithm 4 was tested both with `accel = true` and `accel = false`. The power parameter q was chosen as 1, 2 and 3, while the number of repetitions K was either 1, 2 or 5.

Independent of the clustering method, the optimal transport problem on the clusters was solved by the transportation simplex provided by the *R* package *transport* [80], since this is a robust and well-understood method, whose performance does not depend on geometric structure. For the cost matrix $\bar{C}$, we tried three different options, all of which satisfy $C^{min} \leq \bar{C} \leq C^{max}$:

- *min*: $\bar{C} := C^{min}$
- *median*: $\bar{c}_{k,l}$ is defined as the median of the set $\{c_{i,j} \colon x_i \in X_k, y_j \in Y_l\}$

- *center*: $\bar{C} := 1/2 \cdot (C^{max} - C^{min})$

All computations were run on a Linux laptop, on a single kernel of an Intel Core i5-5300 CPU with 2.3 GHz.

5.4.2 Computational Results

Before we compare the clustering methods, we analyze the effect of the parameter choices on the runtime and quality of the resulting clusterings both for the agglomerative clustering and the clustering with random representatives. Since the computed instances are images from the DOTmark, we can also compare our results to the standard multiscale procedure of coarsening images and observe the deviation bounds, as well as the quality of the feasible solutions obtained by the propagation algorithm described in Section 5.3.3.

Since the instances are computed on the grid $\{1, \dots, 32\}^2$ with high total mass values $\mu(X)$ and $\nu(Y)$, the optimal cost values of the original instances are roughly of order 10^9. Accordingly, the deviation bounds shown in this section also have very high values.

Parameter Choices for the Agglomerative Clustering First, we look at the different linkage criteria in the agglomerative clustering, that is, the choice of $G_* \in \{G_{row}, G_{col}, G_{min}, G_{prod}\}$ in Figures 5.2 and 5.3. In Figure 5.2 we observe the clustering runtime of Algorithm 2 dependent on the linkage criterion and the number of clusters. The runtimes with respect to G_{min} are far higher than for any other linkage criterion. The other linkage criteria show similar runtimes. While G_{row} appears to have slightly higher runtimes than G_{col}, this can only be attributed to the order of X and Y in the computed instances, since the clustering of (X, Y) with respect to G_{row} is the same as the clustering of (Y, X) with respect to G_{col}.

One can also observe, that the runtime slightly increases with the number of clusters n. This is surprising at first glance, as more fusion steps are necessary to achieve a smaller number of clusters. However, the two sets X and Y are clustered in succession, which means that for a smaller n, although the first clustering takes slightly longer, the second clustering is performed

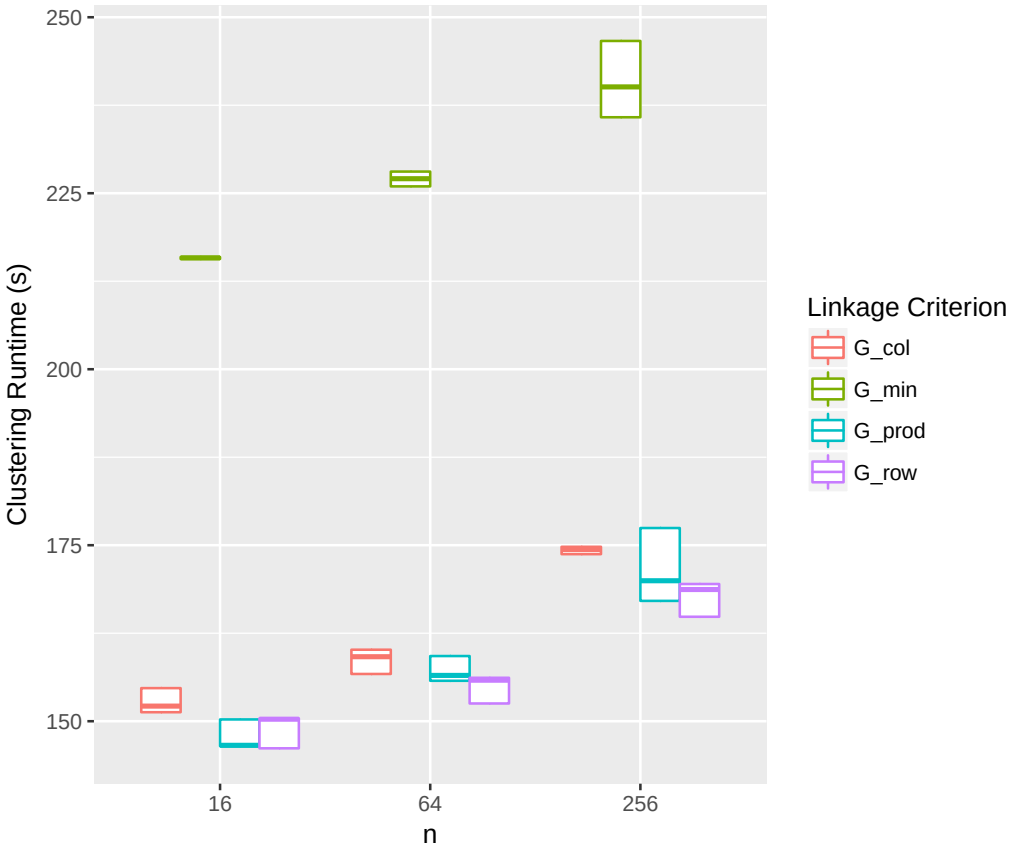


Figure 5.2: Boxplot for the runtime of the agglomerative clustering (Algorithm 2) depending on the number of clusters n and the linkage criterion

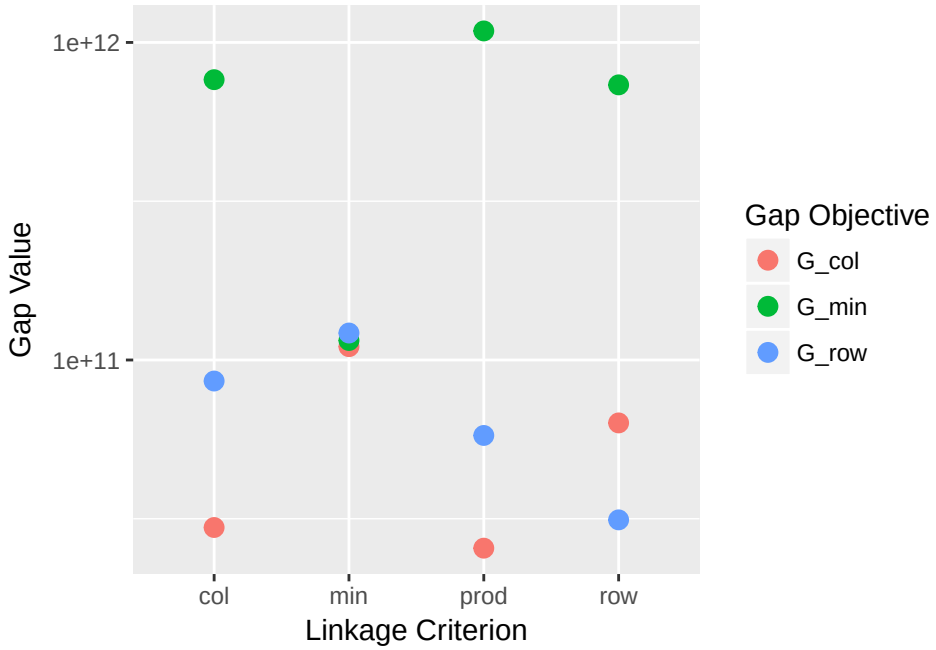


Figure 5.3: This plot shows the mean values of the objective functions $G_{row}, G_{col}, G_{min}$ over multiple clusterings for $n = 256$ on a logarithmic scale. They have been computed with respect to the linkage criterion indicated in the legend.

with respect to fewer clusters, which results in a more significant speedup. This effect is in line with the runtime analysis in Section 5.3.1.

Next, we look at the values of the different gap objective functions relative to the linkage criterion for $n = 256$ (Figure 5.3). Note that G_{prod} is not presented in this figure, since it is not a deviation bound. As expected, if we try to minimize G_{row} or G_{col} in the algorithm, this function has the smallest value and the other one is higher. G_{min}, however, behaves very differently. As a linkage criterion it leads to clusterings that have moderately high gap values with respect to all of the three functions, while none of them, and therefore neither their minimum, are small enough to compete with the other linkage criteria. Also, the function values of G_{min} of the clusterings obtained by the other linkage criteria are much higher (by about one order of magnitude). This means, neither does G_{min} work well as a linkage criterion in the agglomerative clustering, nor are the function values indicative of the quality of a given

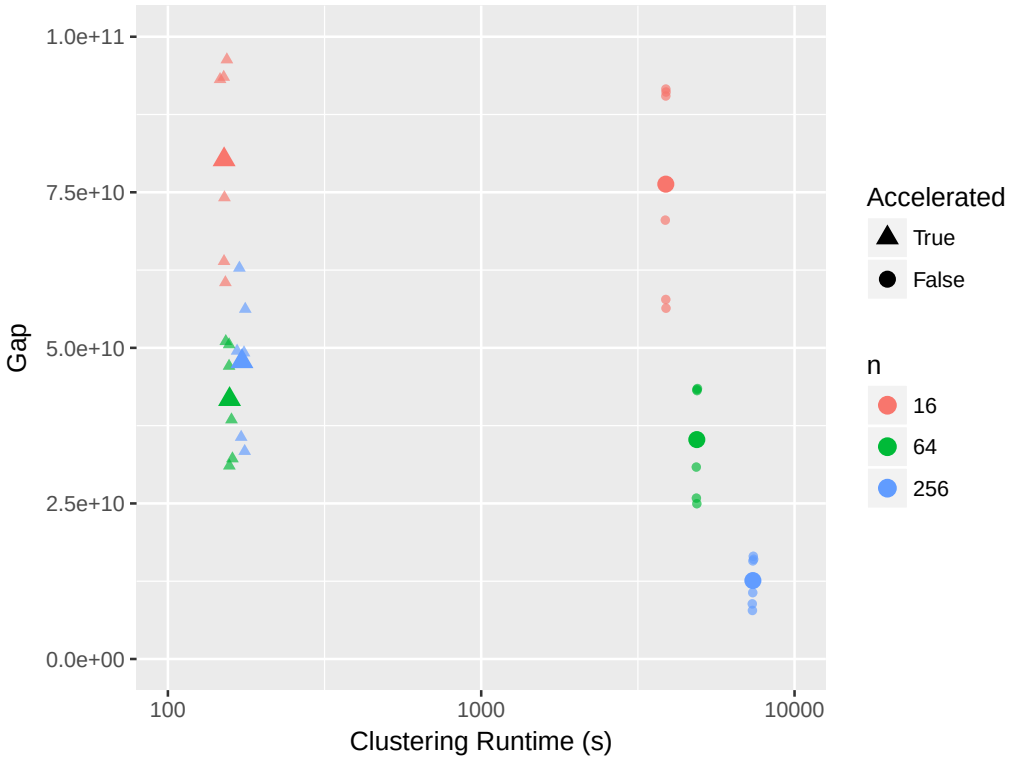


Figure 5.4: Runtime and gap value results for the agglomerative clustering method comparing the accelerated Algorithm 2 with the non-accelerated Algorithm 3 for the linkage criteria G_{row} and G_{col} and various choices for n. *Gap* denotes the deviation bound $\min\{G_{row}, G_{col}, G_{min}\}$. The runtimes are shown on a log-scale.

clustering with respect to the deviation bound $\gamma = \min\{G_{row}, G_{col}, G_{min}\}$. The linkage criterion G_{prod} on the other hand has a similar performance in both runtime and gap values compared to G_{row} and G_{col}. Although the values for G_{col} are smaller than for G_{row} for this criterion, this can only be a result of the order of the two images in the instances, as well as the order of clustering, since G_{prod} is a symmetric function in X and Y.

Figure 5.4 shows the effect of choosing between the accelerated Algorithm 2 and the slower Algorithm 3, when constructing a clustering for the criteria G_{row} and G_{col}. Recall that this only makes a difference for the clustering of the second set (X with respect to G_{col} or Y with respect to G_{row}). However, this has an enormous impact on the runtime in practice. Algorithm 2 runs

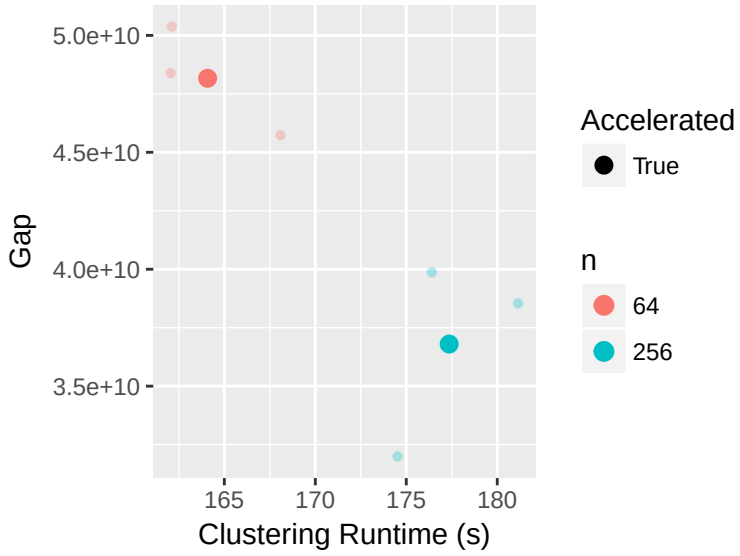


Figure 5.5: A check of the accelerated agglomerative clustering applied to the image classes *GRFmoderate*, *LogitGRF* and *White Noise* shows results which are more in line with the expectation that a higher number of clusters leads to lower objective values.

between 150 and 200 seconds, where a larger number of clusters n has a small increasing effect on the time, but Algorithm 3 needs about 4000 seconds for $n = 16$ and up to almost 7500 seconds for $n = 256$. This does not only show the large impact of having to recalculate the matrix in each step, but also the increased dependence on the number of clusters, since the calculation time of F increases with n.

Despite this disparity in runtime, choosing Algorithm 3 does have its benefits, especially when considering the resulting deviation bounds, since those decrease as expected with a higher number of clusters. We observe an increasing factor of about three, between $n = 256$ and $n = 64$ and another factor of two between $n = 64$ and $n = 16$. This does not apply to the accelerated Algorithm 2. While the results for $n = 16$ are in line with the slower algorithm, the gap values do not decrease as strongly and consistently as for Algorithm 3. For the computed instances we even see values for $n = 64$ that are lower than the average for $n = 256$.

This is a very surprising result, since fusing clusters can only increase the

values in the gap matrix and therefore in the objective functions. Since the first clustering is the same in both algorithms, it seems that the inaccuracy introduced by not recalculating the fusion matrix for the second clustering is more severe for larger n. Since the second clustering depends on the first one and thus on n, a better result with fewer clusters is mathematically – and, as the numerical experiments show, also practically – possible. The instances we chose play another role, since the same algorithms applied to different instances show the expected behaviour, that is, lower gap values for a higher number of clusters (compare Figure 5.5).

To summarize, for small n both algorithms show consistent performances and the runtime is in favor of Algorithm 2. For larger n this runtime benefit increases, but comes at the cost of a less consistent and predictable performance in terms of gap value quality.

Parameter Choices for the Random Clustering Next, we analyze the effect of the different parameters of the clustering with random representatives, Algorithm 4. In Figure 5.6 we can see the runtimes (top) and gap values (bottom) of this method for various choices of q and K, while $n = 256$ and `accel` = `true`. As expected, the runtime depends linearly on K. For q the lowest runtimes appear for $q = 2$, with $q = 1$ being only slightly higher, whereas the increase for $q = 3$ is much higher. This is due to the effort necessary to compute the power of q and depends on how this is handled in detail in R. Nevertheless, it is still remarkable that computing the third power increases the runtime by a factor of more than two.

The quality of the clusterings does not seem to depend on the power q, therefore it is safe to recommend choosing $q = 1$ or $q = 2$ for runtime reasons. A higher number of repetitions K does yield a small improvement in general, but because of the random nature of the algorithm this is not guaranteed. Because of the linearly increasing runtime, it is mostly preferable to choose $K = 1$, although if consistency is a concern the number can also be increased in order to avoid bad outliers. The results for the non-accelerated version are not shown, but they are qualitatively very similar.

If we compare the accelerated with the non-accelerated variant (Figure 5.7)

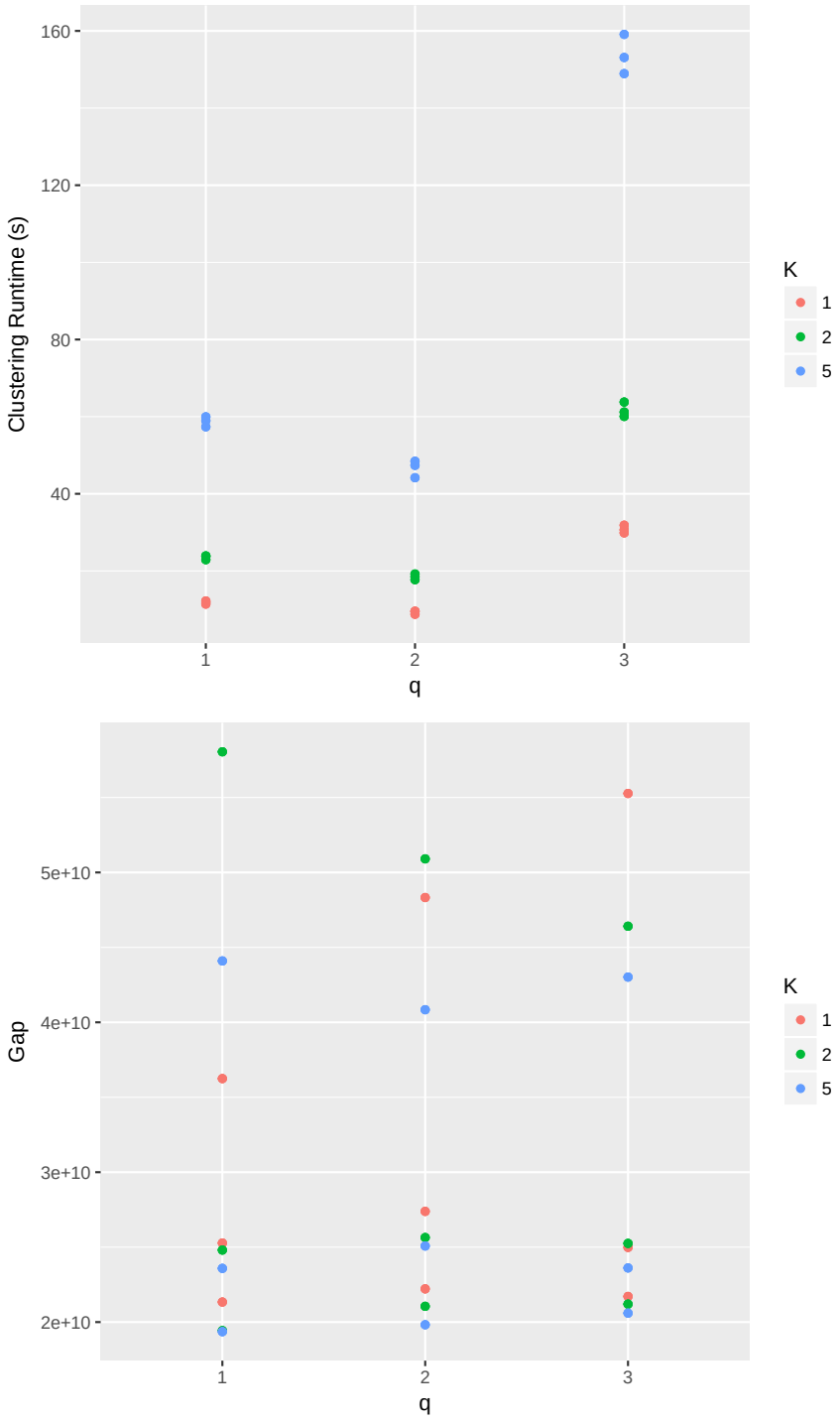


Figure 5.6: Runtime and gap value results for the clustering with random representatives depending on the parameters q and K for $n = 256$ and `accel` = `true`. *Gap* denotes the deviation bound $\min\{G_{row}, G_{col}, G_{min}\}$. The results of the non-accelerated version look qualitatively very similar.

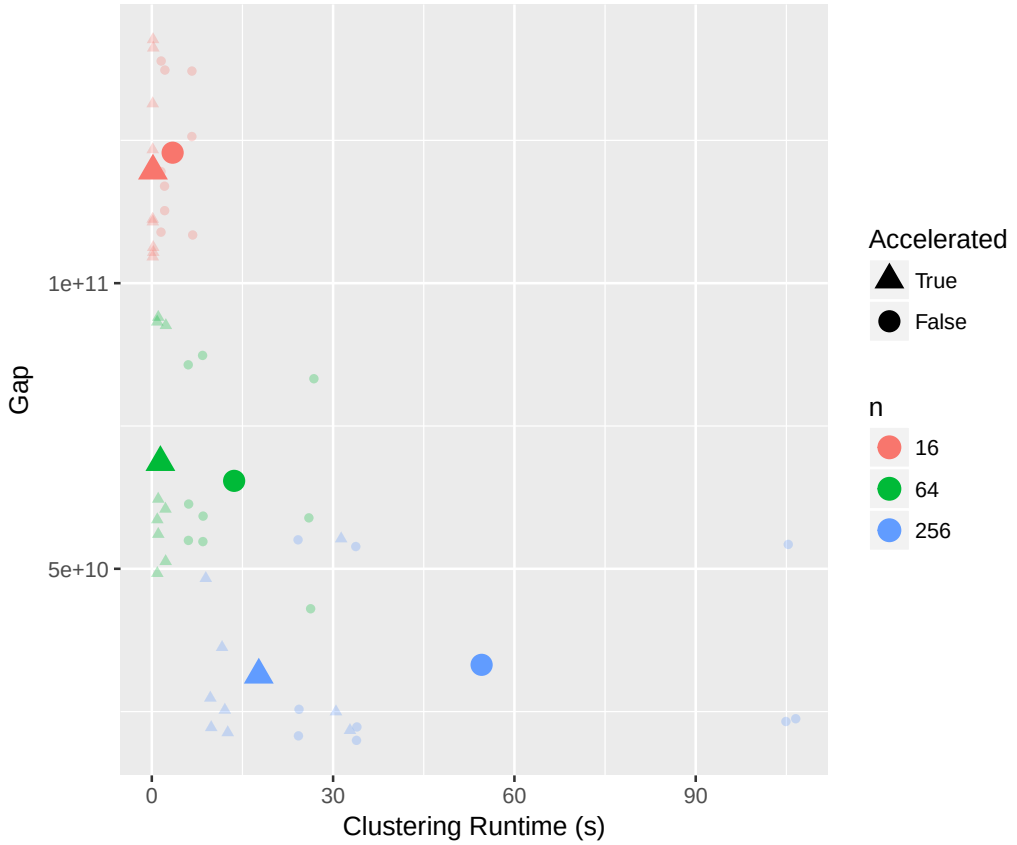


Figure 5.7: Runtime and gap value results for the clustering with random representatives comparing the accelerated with the non-accelerated Algorithm 4 with various choices for n. *Gap* denotes the deviation bound $\min\{G_{row}, G_{col}, G_{min}\}$. Results are shown for $K = 1$, but include all values of q. The outliers in terms of runtime for both options of `accel` can be explained by the choice $q = 3$ (compare Figure 5.6).

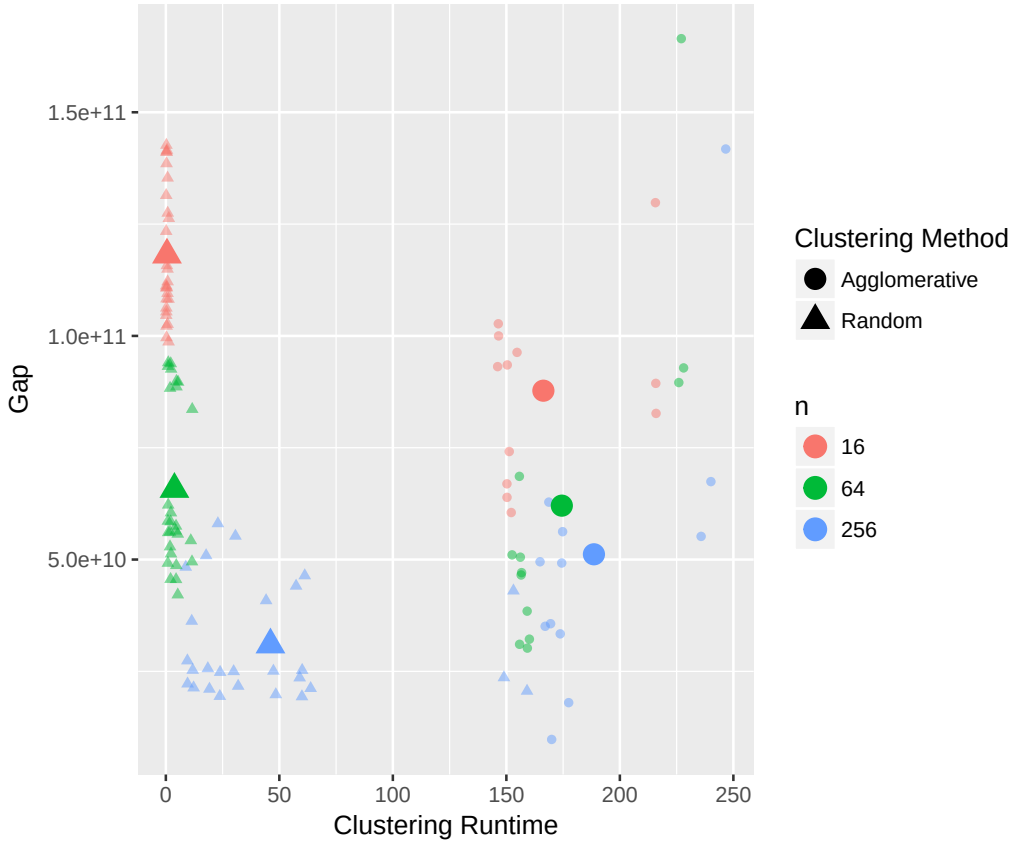


Figure 5.8: Runtime and gap value results compairing the accelerated versions of both clustering methods. For the agglomerative clustering the results of Algorithm 2 are reported for all linkage criteria.

we see a large influence in runtime. It is not as severe as for the agglomerative clustering, but on average the slower version takes about three times as long. What is more, the quality of the resulting clusterings is apparently not dependent on the version of the algorithm, as the mean gap values of the two options are comparable. This means assigning points to clusters only with respect to the costs of the representatives instead of all cost matrix entries comes with virtually no disadvantage and choosing `accel = true` in Algorithm 4 is universally the better option.

Comparing Both Clustering Methods Finally, we have a look at how the two clustering methods compare to each other. To that end, we look at

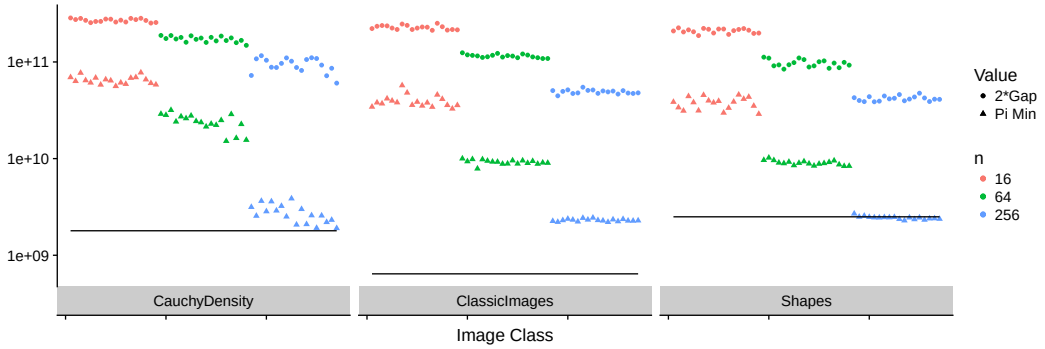


Figure 5.9: Comparing the interval lengths $2\gamma = 2 \cdot \min\{G_{row}, G_{col}, G_{min}\}$ given by the gap values, denoted 2·Gap, with the interval lengths $\langle G, \pi^*_{min} \rangle$ after computing π^*_{min}, denoted Pi Min, for the three instances. Data for different replicates of clusterings with random representatives is scattered across the x-axis. The original optimal cost values c^* are depicted by the horizontal lines for comparison. The values are shown on a log-scale.

the results for both accelerated versions in Figure 5.8. Of course, the runtimes of the clustering with random representatives are much shorter across the board even if we include the results for higher values of K and q.

For the deviation bounds we observe that the random clustering shows a higher dependence of the quality of clusterings on the number of clusters, as the decrease in the bounds with an increase in n is much more noticable compared to the effect on the agglomerative clustering. More importantly, the random clustering does not result in worse clusterings in general. While the error bounds are larger for few clusters ($n = 16$) and very similar for $n = 64$, the random clustering outperforms the agglomerative clustering on average in both runtime and bound quality for large n. However, we should note that results for all linkage criteria are shown for the agglomerative clustering and this includes the criterion G_{min}, which has been shown to have a particularly poor performance with respect to $\min\{G_{row}, G_{col}, G_{min}\}$ in Figure 5.3 and hence is responsible for some outliers here. On the other hand, the agglomerative clustering is designed to minimize gap values directly, the random clustering only indirectly.

Bound Quality and Multiscale Potential In Section 5.2.1 we discuss the interval of possible cost values of length 2γ, where $\gamma = \min\{G_{row}, G_{col}, G_{min}\}$ and the option to decrease the length of this interval by computing π^*_{min}, which decreases the length from 2γ to $\langle G,\pi^*_{min}\rangle$. In Figure 5.9 we look at how these values compare in practice. While we know that $\langle G,\pi^*_{min}\rangle \leq \gamma$ it is noteworthy that in the experiments the former is smaller by far. Especially for $n = 256$ they differ by more than one order of magnitude. When comparing to the true value c^* the deviation bounds γ are too large to give a meaningful approximation interval, even for large n. This is much better for $\langle G,\pi^*_{min}\rangle$, but even then, the best results show an interval length, which is about as large as the true value itself. This limits the usefulness of the solution on the clusters as an approximation and π^*_{min} needs to be computed in order for it to be viable. However, this is not surprising considering that the deviation bounds hold for any choice $\bar{C}$ between C_{min} and C_{max} and we can therefore not expect very tight bounds.

Despite these bad new there are also positive results. In particular, the feasible solutions obtained by the propagation method (Algorithm 5) from optimal solutions on the clusters with respect to the median cost matrix $\bar{C}$ have cost values which are much closer to the original optimal cost values than the deviation bounds might indicate. Figure 5.10 gives an in-depth look at the various bounds and values. We compare the accellerated clutering with random representatives with the grid-based coarsening, that is, 4, 16 or 64 pixels that form a square are aggregated, which results in 256, 64 and 16 clusters, respectively. Figure 5.11 shows a subset of this data.

We can infer from the plot that both the optimal cost values of the clustered instances and the cost values of the propagated solutions are significantly closer to the original optimal cost than the lower and upper bounds for all of the observed clusterings. Recall that the values of $\langle G,\pi^*_{min}\rangle$ from Figure 5.9 are exactly the difference between the lower and upper bounds reported here. Although this gives further indication that these bounds are not tight, they are rigorous and can thus still be useful. Furthermore, the clustered and refined cost values are a reliable source for approximations (see Figure 5.11).

The comparison between the random clustering with the grid-based coars-

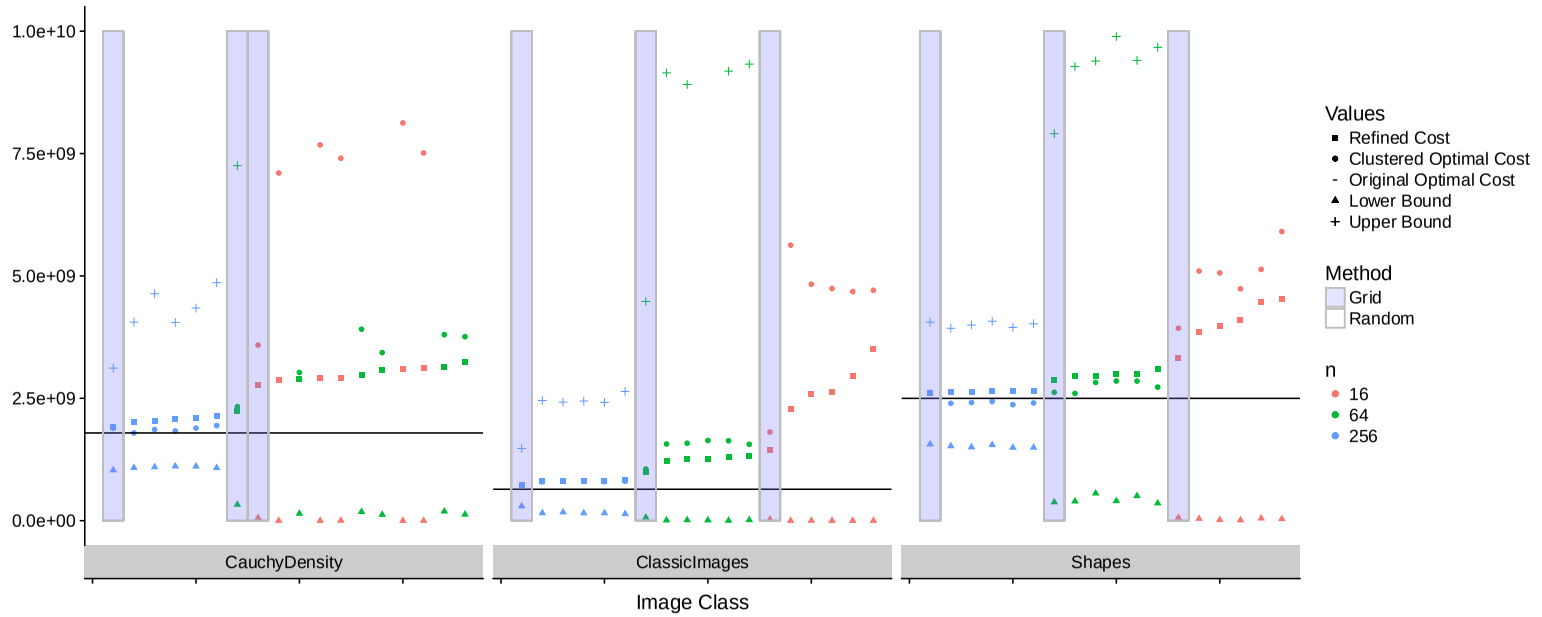


Figure 5.10: Different bounds for 5 replicates of clusterings with random representatives (accelerated, $K = 1$, $q = 2$) are compared to bounds obtained from the grid-based coarsening of images (highlighted with a blue background). The x-axis shows the three different instances. Clusterings are sorted in ascending order of the refined cost value within each instance. On the y-axis we have the values of the following quantities, which are vertically aligned for each clustering: *Lower Bound* and *Upper Bound* are the bounds c^*_{min} and $\langle C^{max}, \pi^*_{min} \rangle$ from Lemma 5.2.5, *Clustered Optimal Cost* is the optimal objective value $\bar{c}^*$ on the clusters with respect to the cost matrix $\bar{C}$, whose entries are the medians of the values in the respective submatrices of the original cost matrix C (compare Section 5.4.1). *Refined Cost* is the cost value of the feasible solution after the solution on the clusters has been propagated via Algorithm 5 and the *Original Cost Value* for each instance is plotted as a horizontal line.

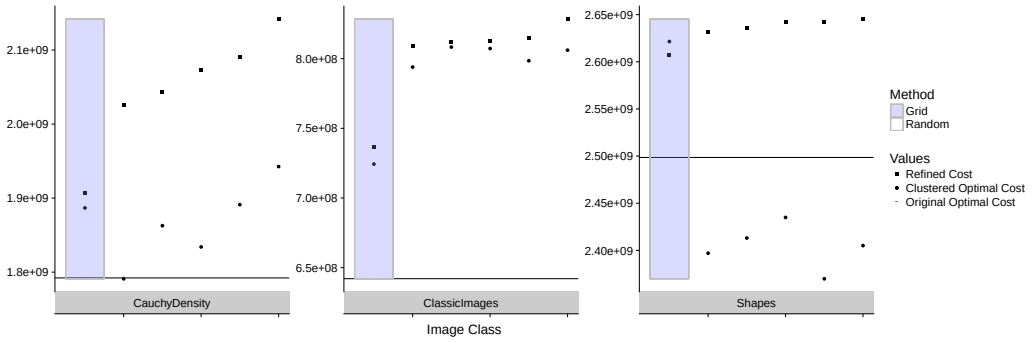


Figure 5.11: A subset of the same data as in Figure 5.10 limited to $n = 256$ for better visibility.

ening is, unsurprisingly, in favor of the coarsening, as aggregating nearby pixels seems to also work well in the context of cost-based clustering and the resulting cost values of the propagated solution are closest to the original optimal costs. In the Cauchy Density instance, the results for $n = 16$ are even better than the results of the clustering with random representatives for $n = 64$. This is further evidence that the standard aggregation of pixels works well in a multiscale scheme.

However, the results for the random clustering are not far off. In Figure 5.11 we see that the approximation quality of the grid-based coarsening is slightly better in the Cauchy Density and Classic Images instances, but this difference is negligible in the Shapes class. It is not surprising that the grid-based coarsening is the superior clustering method in grid-based instances, but the random clustering, a vastly more general method, replicates the performance reasonably well without relying on geometric information and shows promise to deliver a similar performance on instances without a metric. This opens up multiscale approaches to general optimal transport problems.

5.5 A General View on Cost-Based Clustering

As mentioned before, clustering methods usually consider only one set X with some distance function d, whereas we look at a cost function $c\colon X \times Y \to \mathbb{R}_+$

between two sets X and Y. This is a generalization, since one can choose $X = Y$ and $c = d$, and a very flexible framework that can cater to different objectives. The gap objective functions we define in Section 5.2 are examples of different weightings of the entries of the gap matrix, but any other weighting can be used as an objective function for the cost-based clustering problem. Moreover, the gap matrix can be replaced by a different notion of quality of clusters. Instead of the difference between the maximum and minimum entry of a submatrix one could use, for example, the mean absolute deviation of the values in the submatrix or other options.

Generalizations of the Clustering Algorithms The agglomerative clustering method is a general principle and can be applied to a wide range of clustering objectives. Algorithms 2 and 3 can be used as stated for other functions, provided they translate well into linkage criteria. In order to calculate the fusion matrix F we need to be able to compute increases in the objective functions for the fusion of any two clusters. It depends on the function, whether or not all of the entries in the fusion matrix change after agglomerating two clusters and to what extent. If this is the case, it might be necessary or helpful to use the slower Algorithm 3 instead of Algorithm 2.

The clustering method with random representatives can even more easily be generalized to other weightings of the gap matrix. Since Algorithm 4 generally constructs clusterings with small gap matrix entries instead of a concrete objective function, this algorithm can be directly applied. It can also be used for different cluster quality criteria, as long as small differences of costs in the same clusters are preferred, since this difference is the most important factor in the assignment decision. Otherwise, one has to adjust the decision step with respect to the quality criterion.

Examples of Geometric Clusterings Although cost-based clustering methods are not necessarily designed for Euclidean spaces, they can still be applied and yield interesting results other clustering methods would not find. The following examples are based on the Euclidean distance in $\mathbb{R}^D$ for various D and show that clusterings of X are not limited to points that are

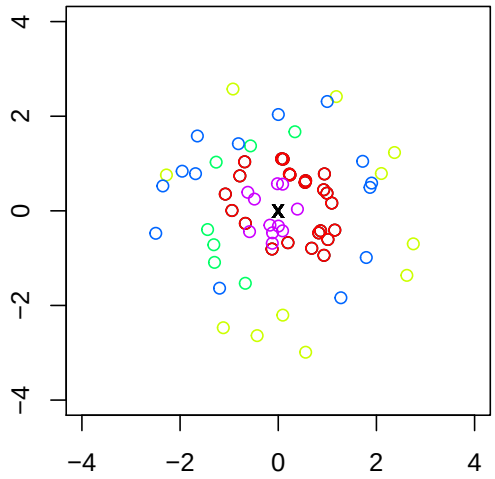


Figure 5.12: A geometric example for cost-based clustering in three dimensions. The set Y consists of twenty points evenly spread on the line segment $[-2, 2] \times \{0\} \times \{0\} \subseteq \mathbb{R}^3$, while X consists of 200 samples from a standard multivariate normal distribution in $\mathbb{R}^3$. X is clustered into twenty subsets by the clustering algorithm with random representatives using $c(x_i, y_j) = \|x_i - y_j\|_2^2$. The figure shows a projection on the (x_2, x_3)-plane. The projection of Y is $\{(0, 0)\}$, shown by a cross at the origin, and for X we only show clusters that contain points with a positive and a negative x_1-component as colored circles.

spatially close, but include features like spherical shapes or separated cluster parts. Consider the extreme case, where $Y = \{y\}$ is a singleton. Then, X is clustered based on the distances of its points to y, which means that the clusters roughly form concentric spheres around y. Similarly, if Y is (roughly) supported on an affine subspace, X is divided into slices that are orthogonal to this subspace, and each slice has the same spherical features as in the singleton case.

This can be observed in Figure 5.12. Although it seems like the mentioned example in $\mathbb{R}^2$ where Y is a singleton, this example is actually three-dimensional, and the figure shows the projection onto the second and third

component. The set X contains samples from a standard multivariate normal distribution. In the clustering it is divided into multiple layers of clusters in x_1-direction. Figure 5.12 shows one layer around the (x_2, x_3)-plane, that is, all clusters containing both points with a positive x_1-value and points with a negative x_1-value. The points in this layer are roughly divided with respect to their distance to the origin, since the entries in the cost matrix of the points in X only depend on their x_1-coordinate as well as the distance to the origin in the (x_2, x_3)-plane.

In the example shown in Figure 5.13, Y is supported on a line in $\mathbb{R}^2$ and the set X consists of samples of a bivariate normal distribution, roughly distributed on a second line orthogonal to the first (top left panel). After a rotation of the set X around the origin we observe different cluster shapes, as relation between X and Y changes (other panels). In the orthogonal case we see a division of X into slices near the origin, whereas clusters further away are divided into two parts each: one above and one below Y in a somewhat symmetrical fashion. For these points, similarity of rows in the cost matrix depends mainly on the distrance to the line. This effect decreases at different angles. At 60 degrees (top right) there still are non-connected clusters, whereas for 30 degrees and parallel lines (bottom panels) the clusters are mostly slice-structured.

5.6 Summary and Discussion

In this chapter, we introduced the cost-based clustering problem as a way to generalize multiscale approaches, which are used to great effect in reducing runtimes for grid-based or other Euclidean optimal transport problems, to more general and abstract instances of optimal transport including matching problems in economics. Our clusterings come with rigorous deviation bounds for the solutions on the clusters and while finding the optimal clustering is an NP-hard problem, it is possible to generate clusterings of good quality in reasonable time with the algorithms presented above. Moreover, we present a well-working propagation algorithm, which can be applied to any multiscale scheme and is compatible with refinement techniques, such as the

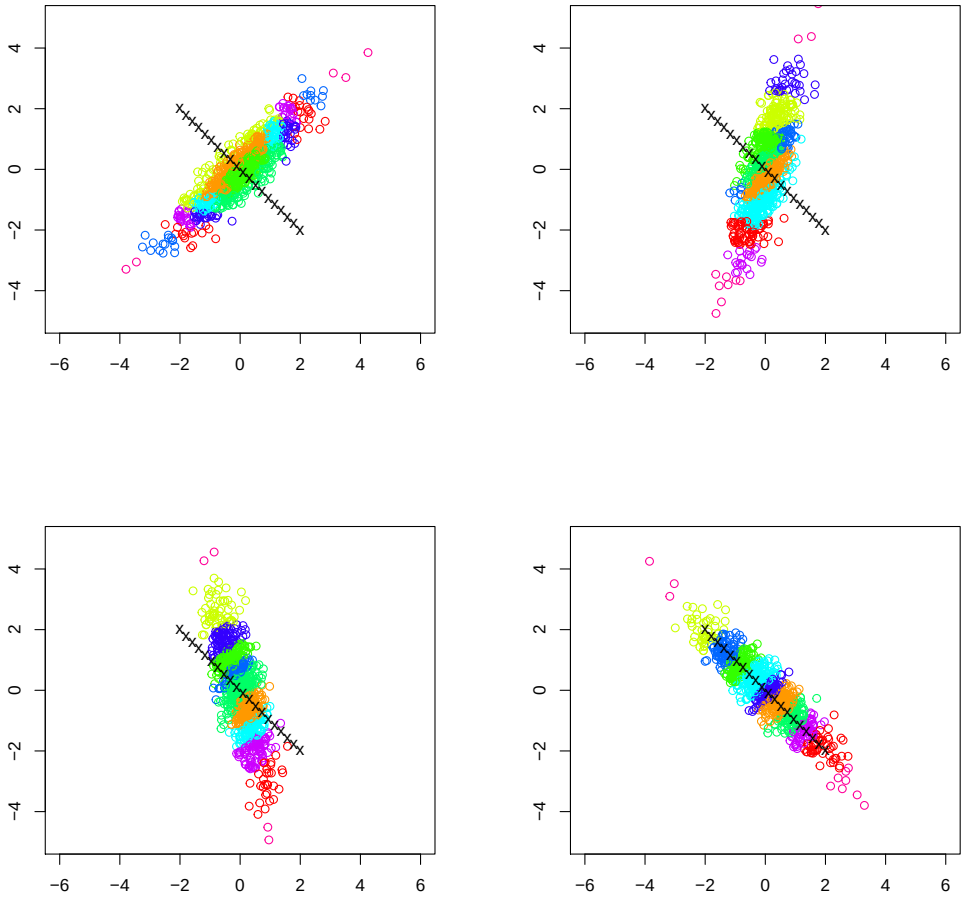


Figure 5.13: A simple example for cost-based clustering on $\mathbb{R}^2$. The set Y is fixed in all panels and consists of twenty points evenly spread across the line segment $\{(x, -x) \in \mathbb{R}^2 \colon -2 \leq x \leq 2\}$ (black crosses). The set X (colored circles) consists of 1000 samples from a bivariate normal distribution with mean vector zero and a covariance matrix with ones on the diagonal and off-diagonal entries 0.9 in the upper left panel. The same set is turned by $\pi/6$ (upper right panel), $\pi/3$ (lower left panel) and $\pi/2$ (lower right panel) around the origin. The agglomerative clustering was used in all cases with $c(x_i, y_j) = \|x_i - y_j\|_2^2$, $n = 10$ and linkage criterion G_{prod}. The ten clusters are indicated by the ten different colors.

transportation simplex.

In experiments on the DOTmark we explored the performances of the proposed clustering algorithms and analyzed parameter choices. To summarize these results, it seems that in most circumstances trying to receive higher quality clusters is not worth the additional time investment. For example, the clustering with random representatives can lead to better or worse clusterings (depending on the number of clusters) compared to the agglomerative clustering, but is much faster. Similarly, running multiple tries ($K > 1$) does not seem to make much of a difference and the same applies to the choice of the non-accelerated over the accelerated version.

This suggests the random representatives clustering algorithm as a prime candidate for applications in multiscale settings, although the agglomerative clustering has its merits in that it can easily be tuned towards constructing a hierarchical clustering, which is not as straight forward for the random clustering. Unfortunately, the theoretical and practical runtime analyses of the agglomerative clustering show that in the context of multiscale methods for optimal transport it is prohibitively slow.

The comparison between the rigorous deviation bounds of the clusterings to the optimal original cost shows that the the intervals of potential cost values is generally rather large. However, the numerical experiments also suggest that the cost values of the clustered solutions are typically much closer to the original cost values and the propagated solutions provide significantly tighter upper bounds. What is more, there seems to be a correlation between the quality of the clustering in terms of the deviation bound and the quality of the propagated solution. That means, even if the bounds given by gap objective values themselves are not highly relevant, using them as quality measures for clusterings may still be useful.

Although we do not have clear evidence of a substantial advantage of using cost-based clustering in a non-Euclidean multiscale scheme for optimal transport at this time, the results of the clustering and propagation algorithms we gathered from the simulations on the DOTmark are promising. In these grid-based instances the coarsening, which is usually used for multiscale purposes, does outperform the cost-based clustering approach as expected,

but only by small margins when comparing the cost values of the propagated solutions (compare Figure 5.11). Since there is – to the best of my knowledge – no suitable general multiscale approach for optimal transport problems without a metric context and since multiscale approaches generally lead to a considerable speedup in the grid-based case, it is a reasonable endeavor to pursue a cost-based clustering approach in cases where other multiscale approaches fail even if we do not expect quite as significant speedups.

On top of all of that, we show that the two-sided clustering approach serves as a generalization of the usual one-sided case and allows for clusterings to be constructed with respect to relations of the elements in one set to the other, instead of relations only within the set itself. The algorithms we present are adaptable beyond the context of optimal transport and we show some interesting examples where the relation between the two sets play an important role and the resulting clusterings show remarkable features that conventional clustering methods would not be able to find.

5.7 NP-Hardness Proof

This section is dedicated to the proof of Theorem 5.2.3. This proof is done via a chain of reductions from 3-satisfiability (3-SAT) over a simplified geometric version of the cost-based clustering problem and a one-sided version to the original clustering problem. This section and the overall proof are structured as follows:

First, we introduce two simplifications of the cost-based clustering problem, the one-sided cost-based clustering problem (1-CBC) and the geometric clustering problem (GC). Both have four different objective functions in correspondence to the four functions of the original clustering problem. We show that GC can be reduced to 1-CBC and this can be reduced to CBC. These reductions are done for the four different functions in parallel.

Secondly, we explain the general idea and geometric construction behind the reductions from 3-SAT to GC. The reduction scheme was introduced in [56] and we describe the features of their construction we adopt and add some aspects, which are necessary for the proof in this section.

Thirdly, we go into the details and prove the NP-hardness of GC directly. We need to perform the reduction twice as each of the two constructions covers two of the four objective functions in GC. Lastly, we prove two auxiliary lemmas, that are needed in the reduction proofs.

Overall, we prove the following chain of reductions,

$$\text{3-SAT} \leq_p \text{GC} \leq_p \text{1-CBC} \leq_p \text{CBC},$$

where $A \leq_p B$ means that problem A is polynomially reducible to problem B.

5.7.1 Geometric Clustering Simplification

We define two simplifications of the cost-based clustering problem and prove that they can be reduced to CBC.

Definition 5.7.1. Following the notation of CBC, the *one-sided cost-based clustering problem* (1-CBC) is to minimize the selected objective function $G_* \in \{G_{row}, G_{col}, G_{min}, G_{prod}\}$ over all possible clusterings $\mathcal{X}$ for X into n subsets, while the clustering $\mathcal{Y} = \{Y_1, \ldots, Y_m\}$ is fixed.

Proposition 5.7.2. *1-CBC $\leq_p$ CBC.*

Proof. We show this reduction simultaneously for the four objective functions. Technically, these are four separate reductions, but the constructions and arguments are identical. This means that everything in this proof holds for each objective function, unless stated otherwise. We handle the objective functions separately, where necessary.

From a given instance of 1-CBC we create an instance of CBC, which has the same optimal objective value and its optimal clustering contains the optimal clustering for 1-CBC. Let an instance of 1-CBC be given as X, Y, μ, ν and a cost matrix C with a fixed clustering $\mathcal{Y} = \{Y_1, \ldots, Y_m\}$, as well as the desired number of clusters $n \in \mathbb{N}$ for X. Let without loss of generality $Y_1 = \{y_1, \ldots, y_{k_1}\}, Y_2 = \{y_{k_1+1}, \ldots, y_{k_2}\}, \ldots, Y_m = \{y_{k_{m-1}+1}, \ldots, y_M\}$.

We construct an instance for CBC with sets $\bar{X}, \bar{Y}$, measures $\bar{\mu}, \bar{\nu}$, cost matrix $\bar{C}$ and the numbers of desired clusters $\bar{n}$ and $\bar{m}$ as follows:

- $\bar{X} := X \cup \{\bar{x}_1, \ldots, \bar{x}_m\}$. One element is added for each cluster of Y.
- $\bar{Y} := Y \cup \{\bar{y}\}$.
- $\bar{\mu}(x_i) := \mu_i$ for all $x_i \in X$ and $\bar{\mu}(\bar{x}_i) := \mu(X)$ for $i = 1, \ldots, m$.
- $\bar{\nu}(y_j) := \nu_j$ for all $y_j \in Y$ and $\bar{\nu}(\bar{y}) := m \cdot \mu(X)$. Note that with these definitions $\bar{\mu}(\bar{X}) = (m+1) \cdot \mu(X) = \nu(Y) + \bar{\nu}(\bar{y}) = \bar{\nu}(\bar{Y})$.
- $\bar{C}$ is defined as the cost matrix for the cost function $\bar{c}\colon \bar{X} \times \bar{Y} \to \mathbb{R}_+$ with
  - $\bar{c}(x_i, y_j) = c_{i,j}$ for all $x_i \in X, y_j \in Y$,
  - $\bar{c}(\bar{x}_k, y_j) = \begin{cases} A, & \text{if } y_j \in Y_k \\ 0, & \text{else} \end{cases}$ for all $k = 1, \ldots, m$, $y_j \in Y$, where

$$A := \frac{\mu(X)}{\min_j \nu_j} \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) + \max_{i,j} c_{i,j},$$

  - $\bar{c}(x, \bar{y}) = 0$ for all $x \in \bar{X}$.

  The structure of $\bar{C}$ is shown below. Note that $A < \infty$, since $\min_j \nu_j > 0$.
- $\bar{n} := n + m$ and $\bar{m} := m + 1$.

$$\bar{C} = \begin{array}{c|c|c|c|c|c} & Y_1 & Y_2 & \cdots & Y_m & \bar{y} \\ \hline X & & C & & & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline \bar{x}_1 & A \cdots A & & & & 0 \\ \bar{x}_2 & & A \cdots A & & 0 & 0 \\ \vdots & 0 & & \ddots & & \vdots \\ \bar{x}_m & & & & A \cdots A & 0 \end{array}$$

Now we show that this instance of CBC has the same optimal objective value $\bar{G}^*$ as the original instance for 1-CBC (G^*). Let $\mathcal{X}^* = \{X_1^*, \ldots, X_n^*\}$ be an optimal clustering of X for 1-CBC. Then an optimal solution to the CBC instance is given by the clusterings $\bar{\mathcal{X}}^* = \mathcal{X}^* \cup \{\{\bar{x}_1\}, \ldots, \{\bar{x}_m\}\}$ of $\bar{X}$ into $\bar{n}$ subsets and $\bar{\mathcal{Y}}^* = \mathcal{Y} \cup \{\{\bar{y}\}\}$ of $\bar{Y}$ into $\bar{m}$ subsets. The gap matrix for these clusterings has the form

$$\bar{g}^* = \begin{pmatrix} g^* & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{(n+m)\times(m+1)},$$

where $g^* \in \mathbb{R}^{n\times m}$ is the gap matrix of the optimal clustering for 1-CBC. Thus, the gap objective value of $\bar{g}^*$ is the same as that for g^*, which is G^*. This holds for each of the four objective functions.

It remains to be shown that this is indeed an optimal clustering for CBC. To that end, let $\bar{\mathcal{X}} = \{\bar{X}_1, \ldots, \bar{X}_{\bar{n}}\}$ and $\bar{\mathcal{Y}} = \{\bar{Y}_1, \ldots, \bar{Y}_{\bar{m}}\}$ be a clustering with gap objective value $\bar{G} < G^*$. We show that this is impossible for each of the four objective functions. Note that

$$G^* \leq \mu(X) \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right)$$

for the objective functions G_{row}, G_{col} and G_{min} and

$$G^*_{prod} \leq \mu(X)^2 \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right),$$

since these are the values of the trivial clustering for $n = m = 1$.

We show that the following two statements hold:

i) $\{\{\bar{x}_1\}, \ldots, \{\bar{x}_m\}\} \subseteq \bar{\mathcal{X}}$.

ii) $\bar{\mathcal{Y}} = \bar{\mathcal{Y}}^*$

Assume that one of these is violated. Due to the structure of the cost matrix $\bar{C}$ there are k', l' with an entry $\bar{g}_{k',l'}$ of the gap matrix of the clusterings $\bar{\mathcal{X}}, \bar{\mathcal{Y}}$, such that

$$\bar{g}_{k',l'} \geq A - \max_{i,j} c_{i,j} \geq \frac{\mu(X)}{\min_j \nu_j} \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right).$$

The corresponding cluster $\bar{X}_{k'} \in \bar{\mathcal{X}}$ has an element $\bar{x}_i \in \bar{X}_{k'} \cap (\bar{X} \setminus X)$, thus $\bar{\mu}(\bar{X}_{k'}) \geq \mu(X)$ and the corresponding cluster $\bar{Y}_{l'} \in \bar{\mathcal{Y}}$ has $\bar{\nu}(\bar{Y}_{l'}) \geq \min_{j=1,\dots,m} \nu_j$. Therefore, we have

$$\begin{aligned}
\bar{G}_{row} &= \sum_{k=1}^{\bar{n}} \bar{\mu}(\bar{X}_k) \max_{l=1,\dots,\bar{m}} \bar{g}_{k,l} \\
&\geq \bar{\mu}(\bar{X}_{k'}) \cdot \frac{\mu(X)}{\min_j \nu_j} \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) \\
&\geq \mu(X) \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) \geq G^*_{row},
\end{aligned}$$

$$\begin{aligned}
\bar{G}_{col} &= \sum_{l=1}^{\bar{m}} \bar{\nu}(\bar{Y}_l) \max_{k=1,\dots,\bar{n}} \bar{g}_{k,l} \\
&\geq \bar{\nu}(\bar{Y}_{l'}) \cdot \frac{\mu(X)}{\min_j \nu_j} \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) \\
&\geq \mu(X) \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) \geq G^*_{col},
\end{aligned}$$

$$\begin{aligned}
\bar{G}_{min} &= \sum_{k=1}^{\bar{n}} \sum_{l=1}^{\bar{m}} \min\{\bar{\mu}(\bar{X}_k), \bar{\nu}(\bar{Y}_l)\} \cdot \bar{g}_{k,l} \\
&\geq \bar{\nu}(\bar{Y}_{l'}) \cdot \frac{\mu(X)}{\min_j \nu_j} \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) \geq G^*_{min},
\end{aligned}$$

$$\begin{aligned}
\bar{G}_{prod} &= \sum_{k=1}^{\bar{n}} \sum_{l=1}^{\bar{m}} \bar{\mu}(\bar{X}_k) \cdot \bar{\nu}(\bar{Y}_l) \cdot \bar{g}_{k,l} \\
&\geq \mu(X) \cdot \nu(\bar{Y}_{l'}) \cdot \frac{\mu(X)}{\min_j \nu_j} \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) \\
&\geq \mu(X)^2 \cdot \left(\max_{i,j} c_{i,j} - \min_{i,j} c_{i,j} \right) \geq G^*_{prod},
\end{aligned}$$

which is a contradiction to $\bar{G} < G^*$ for each of the gap objective functions. Thus, the statements i) and ii) above are satisfied.

The remaining $\bar{n} - m = n$ clusters, $\bar{\mathcal{X}} \setminus \{\{\bar{x}_1\}, \dots, \{\bar{x}_m\}\}$, form a clustering for X, so a feasible clustering for the original instance of 1-CBC. The gap

objective value for this clustering is $\hat{G} \leq \bar{G}$, since it is a part of the clustering for CBC, thus $\hat{G} < G^*$. Since $\bar{\mathcal{Y}} = \bar{\mathcal{Y}}^* = \mathcal{Y} \cup \{\{\bar{y}\}\}$, this is a contradiction to the assumption that G^* is the optimal objective value of 1-CBC.

We conclude that the constructed instance of CBC has the same optimal objective value G^* as the original instance of 1-CBC and its optimal clustering contains an optimal clustering to the original problem. □

Definition 5.7.3. Let $B = \{b_1, \ldots, b_N\} \subseteq \mathbb{R}^2$ with $b_i = (b_{i,1}, b_{i,2})$ for $i = 1, \ldots, N$ and $n \in \mathbb{N}$. The *geometric clustering problem* (GC) with respect to $G_* \in \{G_{sq}, G_{r1}, G_{r2}, G_{r3}\}$ is to find a partition $\mathcal{B}$ of B into n subsets, $\mathcal{B} = \{B_1, \ldots, B_n\}$, such that $G_*(\mathcal{B})$ is minimized. The four functions are defined as follows:

$$G_{sq}(\mathcal{B}) := \sum_{k=1}^{n} \left(|B_k| \cdot \max_{b_i, b_j \in B_k} \|b_i - b_j\|_\infty \right) = \sum_{k=1}^{n} \left(|B_k| \cdot \max\{s_1(B_k), s_2(B_k)\} \right)$$

$$G_{r1}(\mathcal{B}) := \frac{N}{2} \cdot \left(\max_{k=1,\ldots,n} s_1(B_k) + \max_{k=1,\ldots,n} s_2(B_k) \right),$$

$$G_{r2}(\mathcal{B}) := \sum_{k=1}^{n} \min \left\{ |B_k|, \frac{N}{2} \right\} \cdot s(B_k),$$

$$G_{r3}(\mathcal{B}) := \frac{N}{2} \cdot \sum_{k=1}^{n} |B_k| \cdot s(B_k),$$

where we define

$$s_1(B_k) := \left(\max_{b_i \in B_k} b_{i,1} - \min_{b_i \in B_k} b_{i,1} \right),$$

$$s_2(B_k) := \left(\max_{b_i \in B_k} b_{i,2} - \min_{b_i \in B_k} b_{i,2} \right)$$

and

$$s(B_k) := s_1(B_k) + s_2(B_k)$$

as the extension of cluster B_k in x_1-direction, x_2-direction and their sum, respectively. These four objective functions correspond to the four functions in CBC and 1-CBC.

Each objective function defines one version of the problem. The first objective G_{sq} is the sum over all clusters of the amount of points in that

cluster multiplied by the side length of the smallest axis-parallel square that covers all points in the cluster. G_{r1} is $N/2$ times the sum of the largest extension of any cluster in x_1-direction and the largest extension of any cluster in x_2-direction, in other words, it is $N/4$ times the perimeter of the smallest axis-parallel rectangle that can cover any of the clusters when properly translated. G_{r2} and G_{r3} are different weightings of the rectangular extensions of the clusters.

Note that this problem is solely defined to connect CBC with 3-SAT and has no purpose beyond that.

Proposition 5.7.4. *GC $\leq_p$ 1-CBC, that is, G_{sq}, G_{r1}, G_{r2} and G_{r3} can be reduced to G_{row}, G_{col}, G_{min} and G_{prod}, respectively.*

Proof. Again, these are technically four reductions, which we show simultaneously. From an instance of GC we construct an instance of 1-CBC with a fixed clustering for Y.

Let without loss of generality $B = \{b_1, \dots, b_N\} \subseteq \mathbb{R}^2_+$. Set $X = B$ and $Y = \{y_1, y_2\}$ with $\mu_i = 1$ for all $i = 1, \dots, N$ and $\nu_1 = \nu_2 = N/2$. Fix the partition $\mathcal{Y} = \{\{y_1\}, \{y_2\}\}$. Define the cost matrix $C \in \mathbb{R}^{N\times 2}_+$ by $c_{i,j} = b_{i,j}$ for all $i = 1, \dots, N$, $j = 1, 2$.

Now any partition of X for 1-CBC is a partition of B for GC. It remains to be shown that the objective values for the two problems of a given clustering coincide. We show that $G_{row} = G_{sq}$, $G_{col} = G_{r1}$, $G_{min} = G_{r2}$ and $G_{prod} = G_{r3}$. Let $\mathcal{X} = \{X_1, \dots, X_n\}$ be a partition. Since

$$g_{k,l} = \max_{x_i \in X_k} c_{i,l} - \min_{x_i \in X_k} c_{i,l} = s_l(X_k)$$

for $k \in \{1, \dots, n\}$ and $l \in \{1, 2\}$, we have

$$\begin{aligned} G_{row} &= \sum_{k=1}^{n} \left(\mu(X_k) \cdot \max_{l=1,2} g_{k,l} \right) \\ &= \sum_{k=1}^{n} \left(|X_k| \cdot \max_{l=1,2} \left(\max_{x_i \in X_k} c_{i,l} - \min_{x_i \in X_k} c_{i,l} \right) \right) \\ &= \sum_{k=1}^{n} \left(|X_k| \cdot \max\{s_1(X_k), s_2(X_k)\} \right) = G_{sq}, \end{aligned}$$

$$\begin{aligned} G_{col} &= \sum_{l=1}^{2} \left(\nu(Y_l) \cdot \max_{k=1,\ldots,n} g_{k,l} \right) \\ &= \frac{N}{2} \cdot \max_{k=1,\ldots,n} g_{k,1} + \frac{N}{2} \cdot \max_{k=1,\ldots,n} g_{k,2} \\ &= \frac{N}{2} \cdot \left(\max_{k=1,\ldots,n} s_1(X_k) + \max_{k=1,\ldots,n} s_2(X_k) \right) = G_{r1}, \end{aligned}$$

$$\begin{aligned} G_{min} &= \sum_{k=1}^{n} \sum_{l=1}^{2} \min\{\mu(X_k), \nu(Y_l)\} \cdot g_{k,l} \\ &= \sum_{k=1}^{n} \min\left\{ |X_k|, \frac{N}{2} \right\} \cdot (g_{k,1} + g_{k,2}) \\ &= \sum_{k=1}^{n} \min\left\{ |X_k|, \frac{N}{2} \right\} \cdot s(X_k) = G_{r2} \end{aligned}$$

and

$$\begin{aligned} G_{prod} &= \sum_{k=1}^{n} \sum_{l=1}^{2} \mu(X_k) \cdot \nu(Y_l) \cdot g_{k,l} \\ &= \frac{N}{2} \cdot \sum_{k=1}^{n} |X_k| \cdot (g_{k,1} + g_{k,2}) \\ &= \frac{N}{2} \cdot \sum_{k=1}^{n} |X_k| \cdot s(X_k) = G_{r3}. \end{aligned}$$

□

5.7.2 Geometric Reduction Concept

The main part is to show that GC is NP-hard with respect to its four objective functions. This proof is done via a reduction from 3-satisfiability. In [56] Megiddo and Supowit reduce the 3-satisfiability problem to four geometric location problems proving their NP-hardness. While the reduction idea can be adapted to our geometric clustering problem, some additions and adjustments are necessary, since the details of the construction depend strongly on the type of problem and the objective function at hand.

We do this reduction proof twice. The first reduction proves the NP-hardness of GC with respect to G_{sq} and G_{r1}, while the second reduction handles G_{r2} and G_{r3}. In this section we sketch the idea behind the reduction proofs as introduced in [56], while pointing out the key similarities and differences, before we delve into the details of the proofs in the next section. We start by stating the definition of 3-SAT as given for example in [56]:

Definition 5.7.5. Let $\{u_1, \ldots, u_r\}$ be the set of (Boolean) variables and let E be a Boolean expression in 3-conjunctive normal form (3-CNF), that is, $E = E_1 \wedge E_2 \wedge \cdots \wedge E_S$ and for all $j = 1, \ldots, s$ we have a clause $E_j = x_j \vee y_j \vee z_j$ with $\{x_j, y_j, z_j\} \subseteq \{u_1, \bar{u}_1, u_2, \bar{u}_2, \ldots, u_r, \bar{u}_r\}$. The decision problem *3-satisfiability* (3-SAT) is to decide, whether or not there exists a variable assignment, such that E is satisfied.

Each reduction proof from 3-SAT to GC entails constructing an instance of the decision version of GC from a given instance of 3-SAT, that is, a given 3-CNF E. This means we need to define a suitable finite set $B \subseteq \mathbb{R}^2$, $n \in \mathbb{N}$ and a threshold value $R \in \mathbb{R}$. Then we need to show that with this definition E is satisfiable if and only if there exists a clustering $\mathcal{B}$ of B into n subsets with an objective value $G_*(\mathcal{B}) \leq R$ with respect to the objective function $G_* \in \{G_{sq}, G_{r1}, G_{r2}, G_{r3}\}$.

The constructions of B in the proofs inherit the following features from [56], which are common for all objective functions:

- For each variable u_i in E we construct a *circuit* of points, which are arranged such that there are exactly two optimal clusterings - one

representing the state *true* and the other representing the state *false* of the variable (see Figure 5.15 for an example). The circuits in [56] consist of different objects, such as circles, squares and points, whereas the circuits in our proofs only consist of points. The circuits differ based on the objective function and are specified in more detail in the proofs.

- The circuits may intersect. The *junctions* between two circuits are handled in a way that does not interfere with the variable assignments (see Figure 5.16 or Figure 5.21).

- For each clause E_j we add a point representing the clause in a location, where the three circuits for the variables appearing in the clause meet (see Figure 5.17 or Figure 5.22). This meeting point is referred to as *clause configuration*. It is constructed in a way such that if the represented clause is satisfied the additional point can be efficiently included in one of the clusters of the circuits. If not, it forces an inefficient clustering, whose objective function value inevitably exceeds the threshold value R.

In the proofs below we show that with suitable construction details we obtain the equivalence of the satisfiability of E and the existence of a clustering with objective value smaller than or equal to R. A schematic example for the whole construction is shown in Figure 5.14.

For the proof of Proposition 5.7.6 it is necessary to add the following features to the construction, both of which are related to the objective functions G_{sq} and G_{r1} and are not present in any of the proofs in [56]:

- To ensure that the optimal clustering for a circuit represents the same variable state before and after a clause configuration we have to add a *consistency check configuration* for each clause configuration the circuit belongs to by adding two points, which can be included in suitable clusters of the circuit (see Figure 5.18).

- An additional *diamond-shaped circuit* is added, which is large enough to prevent an optimal clustering formed by non-square rectangles (see

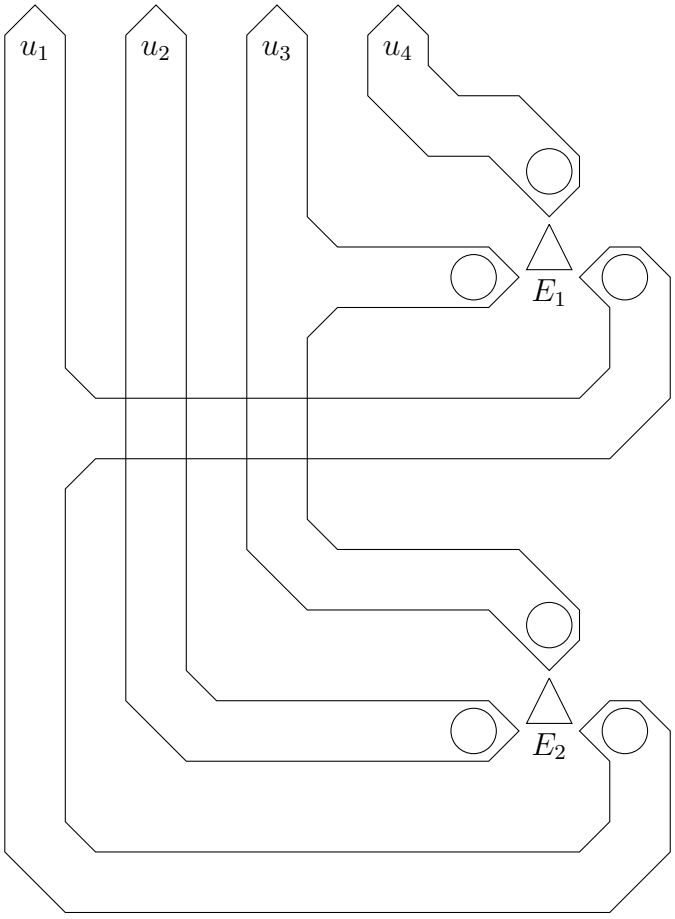


Figure 5.14: Schematic example of the whole construction with four circuits representing the variables u_1, u_2, u_3, u_4 and two clauses E_1 and E_2. The triangles represent the clause configurations and, if necessary, the consistency check configurations are located in the places indicated by the circles. The details of the construction depend on the objective function. The construction can easily be extended to an arbitrary number of variables and clauses, since additional variables can be added horizontally and additional clauses can be added vertically.

Figure 5.20). This ensures that the reduction works for both G_{sq} and G_{r1} simultaneously.

Lastly, for each of the proofs we make use of one lemma and both lemmas are proved in Section 5.7.4.

5.7.3 Geometric Reduction Proofs

Proposition 5.7.6. *GC with respect to G_{sq} and G_{r1} is NP-hard.*

Proof. First, we show this for the objective function G_{sq}. Then, with the addition of a diamond-shaped circuit, we can enforce that $G_{sq} \leq R$ and $G_{r1} \leq R$ are equivalent, which proves the NP-hardness with respect to both functions. For a given 3-CNF E the details of the construction are as follows:

For each variable u_i we have a circuit P_i containing an even number of points, that is, $P_i = \{p_1^{(i)}, \ldots, p_{2m_i}^{(i)}\} \subseteq \mathbb{R}^2$ with the following properties:

i) For all $k = 1, \ldots, 2m_i - 1$: $\|p_k^{(i)} - p_{k+1}^{(i)}\|_\infty = 1$ and $\|p_{2m_i}^{(i)} - p_1^{(i)}\|_\infty = 1$.

ii) For all k, l not affected by i): $\|p_k^{(i)} - p_l^{(i)}\|_\infty \geq 2$.

Given a circuit $P = \{p_1, \ldots, p_{2m}\}$ there are exactly two clusterings of P into m clusters, namely $P = \{p_1, p_2\} \cup \cdots \cup \{p_{2m-1}, p_{2m}\}$ and $P = \{p_2, p_3\} \cup \cdots \cup \{p_{2m-2}, p_{2m-1}\} \cup \{p_{2m}, p_1\}$, that have an objective value of $G_{sq} = |P| = 2m$. These represent the two assignments *true* and *false* for the variable u_i and we assign labels t and f to edges between subsequent points in the circuit to illustrate this. An example for a circuit is shown in Figure 5.15.

Pairs of points (p, q) from different circuits have to satisfy $\|p - q\|_\infty \geq 2$, unless they are part of the same junction or clause configuration. Figure 5.14 indicates that this is achievable, as the spaces between the circuits can be extended. Note that condition ii) may be violated in clause configurations and consistency check configurations (details below).

Since the same variable may appear in many clauses, junctions between circuits are necessary and set up as shown in Figure 5.16. The circuits have a common point at the junction and for any of the four combinations of variable

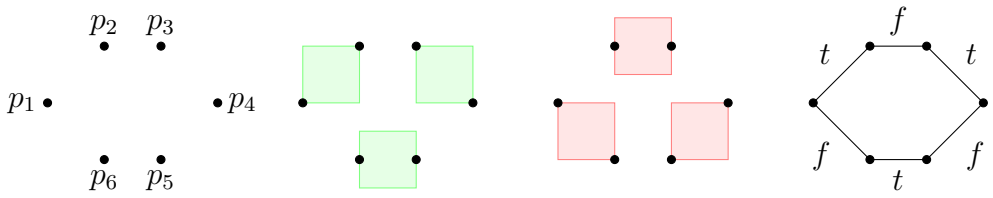


Figure 5.15: Example of a circuit (left) with the two optimal clusterings shown as square covers (middle) and the resulting labels t and f for the edges between subsequent points (right).

assignments it is possible to cover the common point together with one of each circuit saving one cluster in total. That means for r circuits with J junctions there is exactly one clustering into $n = \sum_{i=1}^{r} m_i - J$ clusters with

$$G_{sq} = |\bigcup_{i=1}^{r} P_i| = \sum_{i=1}^{r} 2m_i - J$$

for every possible variable assignment.

We demand that the indices of the common points in junctions (for example, k and l in Figure 5.16) are all odd. This ensures that there is always an odd number of points between two consecutive junctions in a circuit. This is necessary for the proof that any clustering $\mathcal{B}$ with $G_*(\mathcal{B})$ represents a consistent variable assignment. We follow the argument in [56] for this.

Now we add one point q_j for each clause E_j in E, such that q_j can be covered by one of the clusters for the circuits of variables in E_j with no further cost other than adding the point (that is, the diameter of the cluster does not increase) if and only if E_j is satisfied. This is achieved by the precise point placement in clause configurations as shown in Figure 5.17. We require the following conditions:

i) For each variable u_i that appears in E_j, there is a $k_{i,j} \in \{1, \ldots, 2m_i\}$, such that $\|q_j - p^{(i)}_{k_{i,j}}\|_\infty = \|q_j - p^{(i)}_{k_{i,j}+1}\|_\infty = 1$.

ii) The label on the edge between $p^{(i)}_{k_{i,j}}$ and $p^{(i)}_{k_{i,j}+1}$ is t if variable u_i appears as a positive literal in E_j and f if it appears as a negative literal ($\bar{u}_i$) in E_j.

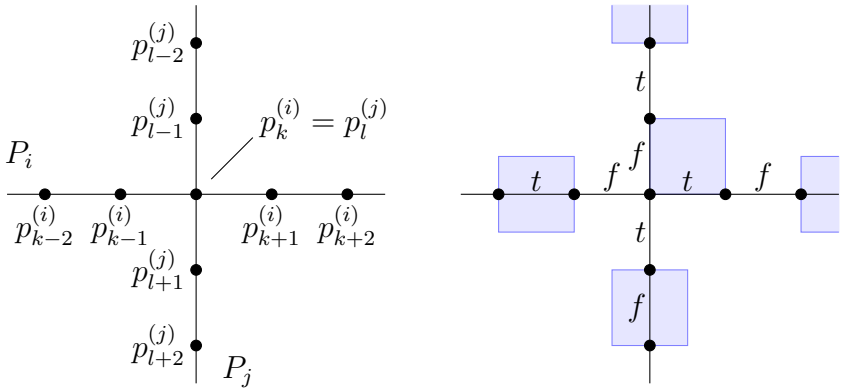


Figure 5.16: A junction between circuits P_i and P_j with a common point (left) and an optimal clustering shown as square cover (right) representing the variable assignment $u_i = true$ and $u_j = false$ with one cluster including three points.

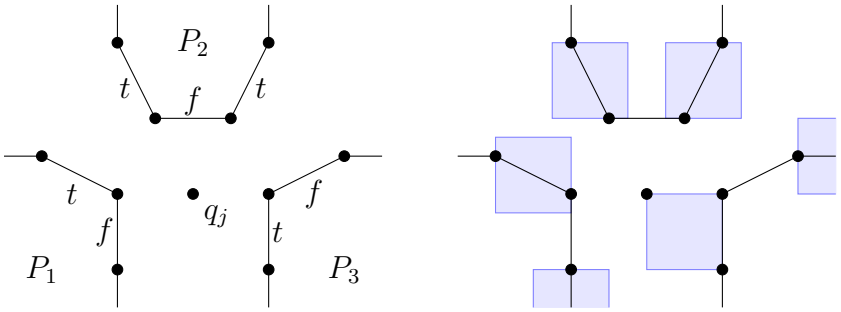


Figure 5.17: A clause configuration representing the clause $E_j = \bar{u}_1 \vee \bar{u}_2 \vee u_3$ (left). The optimal clustering shown as square cover representing the variable assignment $u_1 = true$, $u_2 = true$ and $u_3 = true$ (right). Since it satisfies E_j the point q_j can be included in one of the clusters in P_3.

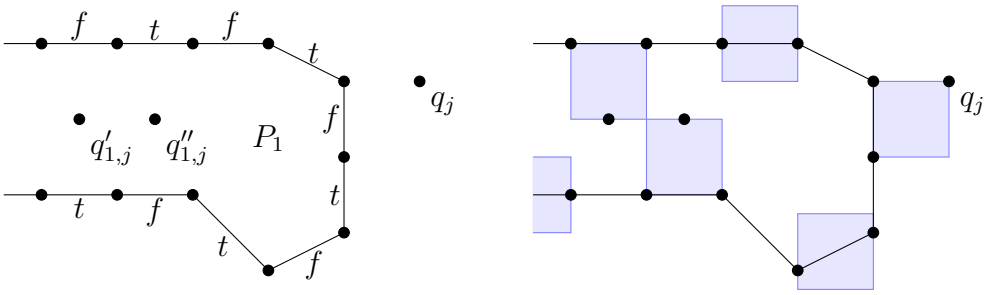


Figure 5.18: Example of a consistency check configuration with two additional points (left). This setup ensures that the two points can be included in the clusters with no further cost if and only if the variable assignment is the same before and after the clause configuration (point q_j). The clustering representing the state $false$ of the variable and satisfying clause E_j (right).

These conditions ensure that if a variable assignment satisfies a clause E_j, its point q_j can be included in the corresponding clustering in at least one of the circuits of the variables in E_j. Conversely, if a variable assignment does not satisfy a clause E_j either one of the clusters has to be enlarged, leading to a higher optimal objective value of G_{sq} in GC, or q_j is in a cluster with two points from different circuits (see Figure 5.19, left).

The latter changes the variable state represented by the affected circuits. We prevent this by adding a consistency check configuration to each circuit in each clause, which ensures that before and after each clause (and therefore in the whole circuit) the same Boolean value is represented, and a clustering that compromises the representation always has a higher objective value. This is achieved by adding two additional points, $q'_{i,j}$ and $q''_{i,j}$ for circuit P_i and clause E_j, that can be covered without enlarging any cluster if and only if the variable state represented before and after a clause in a circuit is the same (see Figure 5.18).

We now define the set B for the instance of GC as the set of all points in the construction, that is,

$$B := \bigcup_{i=1}^{r} P_i \cup \bigcup_{j=1}^{S} \left(\{q_j\} \cup \left\{ q'_{i,j}, q''_{i,j} \colon u_i \text{ appears in } E_j \right\} \right).$$

Since the circuits have one common point for each junction and we add 7

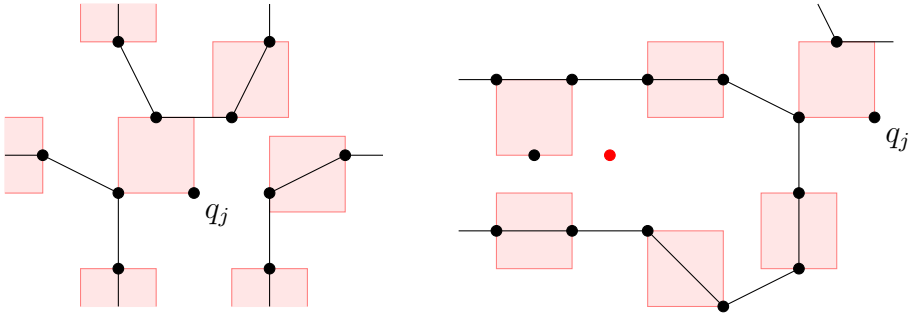


Figure 5.19: A different clustering for the clause is possible, where q_j is coupled with two points from two different circuits (left). This can be prevented through the consistency check configurations. If the clustering does not represent the same variable assignment before and after the clause one of the additional points cannot be included in any cluster without additional cost (right). This prevents an optimal clustering where one cluster contains points of different circuits.

points for each clause, the total number of points is

$$N := |B| = \sum_{i=1}^{r} 2m_i - J + 7S.$$

We define the number of clusters n as

$$n := \frac{N - 7S - J}{2} = \sum_{i=1}^{r} m_i - J$$

and the threshold value as $R := N$. We need to show that E is satisfiable if and only if there is a clustering $\mathcal{B}$ of B into n subsets with $G_{sq}(\mathcal{B}) \leq R$.

To that end, assume that E is satisfiable. This means there is a variable assignment satisfying all clauses simultaneously. Therefore, the clustering of the circuits into n subsets, which represents this variable assignment can also cover all additional points in the clause configurations and consistency check configurations, as is shown in Figures 5.17 (right) and 5.18 (right). Since all clusters $B_k \in \mathcal{B}$ have $\max\{s_1(B_k), s_2(B_k)\} = 1$, it follows that

$$G_{sq}(\mathcal{B}) = \sum_{k=1}^{n} \Big(|B_k| \cdot \max\{s_1(B_k), s_2(B_k)\} \Big) = \sum_{k=1}^{n} |B_k| = |B| = N.$$

Conversely, let $\mathcal{B} = \{B_1, \dots, B_n\}$ be a clustering of B with $G_{sq}(\mathcal{B}) \leq N$. We define a map $\sigma\colon \{1, \dots, N\} \to \mathbb{R}_+$ by

$$\sigma(k) := \begin{cases} \min\left\{\max\{s_1(\bar{B}), s_2(\bar{B})\}\colon \bar{B} \subseteq B, |\bar{B}| = k\right\}, & k = 1, k = 2, k \geq 5, \\ \frac{3}{2}, & k = 3, \\ 2, & k = 4, \end{cases}$$

which is the minimum side length of a square that can cover k points of B for all k other than three and four, while for those numbers it is defined as the second smallest possible side length. It is easy to see that $\sigma(1) = 0$, $\sigma(2) = 1$ and $\sigma(k) \geq 2 > 2 - 2/k$ for $k \geq 5$.

For $k = 3$ the smallest possible side length is 1. However, these clusters are only possible if they include either the common point of two circuits in a junction, the additional point q_j of a clause configuration or at least one of the additional points $q'_{i,j}$, $q''_{i,j}$ in a consistency check configuration. Apart from these options the smallest side length of a square covering three points is $\sigma(3) = 3/2$.

Four points can be covered by a square of side length 3/2, if they either include a clause configuration point q_j or both points $q'_{i,j}$ and $q''_{i,j}$ in one consistency check configuration. Otherwise, the minimum side length of a square covering four or more points is $\sigma(4) = 2$.

We define the following quantities:

- $a_k := \left|\left\{\bar{B} \in \mathcal{B}\colon |\bar{B}| = k\right\}\right|$ for $k = 1, \dots, N$, $k \neq 3, 4$
- $b_k := \left|\left\{\bar{B} \in \mathcal{B}\colon |\bar{B}| = k, q_j \in \bar{B} \text{ for some } j\right\}\right|$ for $k = 3, 4$
- $c_k := \left|\left\{\bar{B} \in \mathcal{B}\colon |\bar{B}| = k, \max\{s_1(\bar{B}), s_2(\bar{B})\} < \sigma(k), q_j \notin \bar{B} \text{ for all } j\right\}\right|$ for $k = 3, 4$
- $a_k := \left|\left\{\bar{B} \in \mathcal{B}\colon |\bar{B}| = k\right\}\right| - b_k - c_k$ for $k = 3, 4$

For k other than three and four, a_k indicates the number of clusters in $\mathcal{B}$ containing exactly k points. For $k = 3, 4$ we separate the clusters into three groups: clusters in clause configuratons (b_3 and b_4), small clusters apart from clause configurations (c_3 and c_4) and other clusters (a_3 and a_4).

From this definition follows directly that $b_3 + b_4 \leq S$. Since small clusters apart from clause configurations with three points contain a junction point or a point in a consistency check configuration and small clusters with four points have to contain both points in a consistency check configuration, $c_3 + 2c_4 \leq 6S + J$.

Moreover, since we have n clusters covering N points, the following equations are satisfied:

$$b_3 + c_3 + b_4 + c_4 + \sum_{k=1}^{N} a_k = n$$
$$3(b_3 + c_3) + 4(b_4 + c_4) + \sum_{k=1}^{N} k \cdot a_k = N$$

All the clusters counted by b_3 and c_3 have a side length of at least 1, the clusters counted by b_4 and c_4 have a side length of at least $3/2$ and the remaining clusters (counted by a_k) have a side length of at least $\sigma(k)$. Therefore,

$$\begin{aligned}
N \geq G_{sq} &= \sum_{l=1}^{n} |B_l| \cdot \max\{s_1(B_l), s_2(B_l)\} \\
&= \sum_{k=1}^{N} \sum_{\substack{l=1 \\ |B_l|=k}}^{n} k \cdot \max\{s_1(B_l), s_2(B_l)\} \\
&\geq 3\,(1 \cdot (b_3 + c_3) + \sigma(3) \cdot a_3) \\
&\quad + 4\left(\frac{3}{2} \cdot (b_4 + c_4) + \sigma(4) \cdot a_4\right) \\
&\quad + \sum_{\substack{k=1 \\ k \neq 3,4}}^{N} \sigma(k) \sum_{\substack{l=1 \\ |B_l|=k}}^{n} k \\
&= 3(b_3 + c_3) + 6(b_4 + c_4) + \sum_{k=1}^{N} k \cdot \sigma(k) \cdot a_k.
\end{aligned}$$

This means the system of linear inequalities in Lemma 5.7.9 as well as all

assumptions are satisfied. Thus, applying this lemma we can conclude that the only non-zero quantities are $a_2 = n - 7S - J, b_3 = S$ and $c_3 = 6S + J$. It follows that each cluster in $\mathcal{B}$ is covered by a square of at least side length 1, hence $G_{sq}(\mathcal{B}) = N$ and hence the side lengths all have to be exactly 1. Moreover, the clusters containing three points have to be located at clause and consistency check configurations ($7S$ clusters) and junctions (J clusters).

Now with the same argument as in [56] we show that this clustering represents a variable assignment. We divide the points in the circuits into junctions (we consider the common point and the four surrounding points as *belonging to the junction*) and segments between junctions (maximal subsets of consecutive points in the same circuit not belonging to junctions). Since the indices of the common points are all odd, each segment contains an odd number of points and since each segment is adjacent to two junctions and each junction is adjacent to four segments, there are $2J$ segments. The segments contain

$$\sum_{i=1}^{r} 2m_i - 6J$$

points in total, not accounting for additional points in consistency check or clause configurations. Since they each have an odd amount of points and clusters have at most a side length of 1, we need

$$\frac{1}{2} \cdot \left(\sum_{i=1}^{r} 2m_i - 6J - 2J \right) + 2J = \sum_{i=1}^{r} m_i - 2J = n - J$$

clusters in order to cover all of them, while one point in each segment takes up one cluster of its own (we call them *singletons*). Note that the clusters containing three points in consistency check and clause configurations are also used for this and thus only the J clusters for junctions remain. This means three of the five points in each junction are covered by these clusters and the two remaining points have to be assigned to the singletons in the segments. This is precisely possible, since there are exactly $2J$ of these singletons and $2J$ extra points in junctions, but it requires that each singleton is either the first or last point in its segment.

In each segment the clustering represents a variable assignment. This also

holds for segments involved in clause configurations, as all of the additional points in the consistency check configurations are contained in separate clusters with three points as shown in Figure 5.18 (right). Along one circuit, if the first point in a segment is the singleton, the first point in the next junction has to belong to the junction cluster, which means the third point in the junction does not. Thus, the first point of the next segment of the circuit again has to be the singleton. It follows that the singletons in the segments of a circuit are either always the first or always the last point, which means that the clustering represents either the value *true* or *false* of the variable represented by the circuit. Therefore, the clustering represents a consistent variable assignment for all Boolean variables u_i. Since all of the points q_j in the clause configurations are contained in clusters with side lengths 1, this variable assignment satisfies all clauses E_j and we can conclude that E is satisfiable.

This finishes the proof with respect to G_{sq}. Now we show that with the addition of a diamond-shaped circuit P_{r+1} (see Figure 5.20) to the construction with $4Q$ points, where

$$Q = \sum_{i=1}^{r} |P_i|$$

we can enforce that $G_{r1} \leq N$ is equivalent to $G_{sq} \leq N$. This circuit has no meaning for E and is disconnected from the other circuits, thus the above arguments also hold for G_{r1}.

We first show that $G_{r1} \leq N$ implies $G_{sq} \leq N$. Recall that

$$G_{r1} = \frac{N}{2} \cdot \left(\max_{k=1,\ldots,n} s_1(B_k) + \max_{k=1,\ldots,n} s_2(B_k) \right),$$

so if $G_{r1} \leq N$ then there exists an axis-parallel rectangle with side lengths a and b, such that $a + b \leq 2$ and each cluster can be covered by a translated version of this rectangle. If $0 < a < 1$ or $0 < b < 1$ then there are no two points in P_{r+1}, which share a cluster. Thus, there are at least $4Q$ clusters, but $4Q > N/2 > n$. Similarly, if $a = 0$ or $b = 0$ at least $4Q - 2$ clusters are necessary and still $4Q - 2 > n$ (compare Figure 5.20). Hence, $a = b = 1$. That means every cluster can be covered by a square with side length 1 and

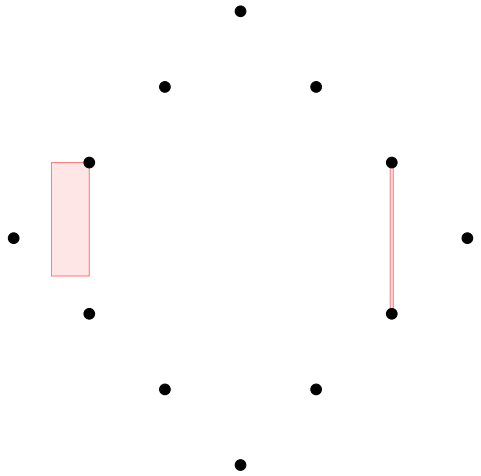


Figure 5.20: The diamond-shaped circuit P_{r+1} for $Q = 3$. A non-squared rectangle with side lengths $a + b = 2$ can never cover more than one point (left rectangle), unless $a = 0$ or $b = 0$ (right rectangle), in which case only two clusters of two points each can exist.

thus $G_{sq} \leq N$.

Conversely, if $G_{sq} \leq N$, the optimal clustering we constructed above consists of squares with side length 1, therefore all of the clusters can be covered by a translated rectangle with side length $a = b = 1$, so $G_{r1} \leq N/2 \cdot (1 + 1) = N$.

□

Remark 5.7.7. Since $G_{sq} \leq N$ and $G_{r1} \leq N$ are equivalent in the above construction, the proof also holds for the problem version, where $\min\{G_{sq}, G_{r1}\}$ is the objective function and thus the minimization of $\min\{G_{row}, G_{col}\}$ in CBC is NP-hard as well.

Proposition 5.7.8. *GC with respect to G_{r2} and G_{r3} is NP-hard.*

Proof. We reduce 3-SAT to GC in a similar way as before. Circuits represent variables and clause configurations are constructed such that there exists an optimal clustering with a certain objective value $R \in \mathbb{R}$ if and only if the given 3-CNF E is satisfiable. The objective function we consider in this proof is

$$G_{rec} := \sum_{k=1}^{n} |B_k| \cdot s(B_k).$$

Since it is a multiple of G_{r2} this shows the NP-hardness of GC with respect to G_{r2}. Later, we show that with this construction $G_{rec} \leq R$ if and only if $G_{r3} \leq R$, since there is no optimal clustering with any cluster B_k, such that $|B_k| \geq N/2$, which proves the NP-hardness of GC with respect to G_{r3}.

The construction here has a few differences compared to the one in the previous proof:

- Both the junctions and the clause configurations force clusters with three points each that also have a larger size $s(B_k)$ and therefore contribution to the objective value of the clustering. Those are the least possible increases compared to different clusterings and hence the only options for a clustering with a value below R.
- The additional consistency check configurations are not necessary, since any clustering changing the variable assignment in a clause configuration cannot be optimal due to the specific construction. The diamond-shaped circuit is not necessary, as the reduction for G_{rec} implies the NP-hardness for both G_{r2} and G_{r3}.

Each circuit is a set of an even amount of points $P_i = \{p_1^{(i)}, \ldots, p_{2m_i}^{(i)}\} \subseteq \mathbb{R}^2$ representing the Boolean variable u_i such that

i) for all $k = 1, \ldots, 2m_i - 1$: $\|p_k^{(i)} - p_{k+1}^{(i)}\|_1 = 1$ and $\|p_{2m_i}^{(i)} - p_1^{(i)}\|_1 = 1$.

ii) for all k, l not affected by i): $\|p_k^{(i)} - p_l^{(i)}\|_1 \geq 2$.

This is the same as above, but the distance is measured in the 1-norm instead of the maximum norm. We allow condition i) to be violated in junctions and condition ii) to be violated in clause configurations, since these have a specific point placement. Again, there are two optimal clusterings of the points in one circuit into m_i clusters with objective value $G_{rec} = 2m_i$, since a cluster B_k has $s(B_k) = 1$, if it contains two consecutive points and $s(B_k) \geq 2$ otherwise. We can label the edges between consecutive points alternatingly with t and f for true and false, respectively (compare Figure 5.15). Moreover, it is required that pairs of points (p, q) from different circuits have $\|p - q\|_1 \geq 2$, unless they are part of the same junction or clause configuration.

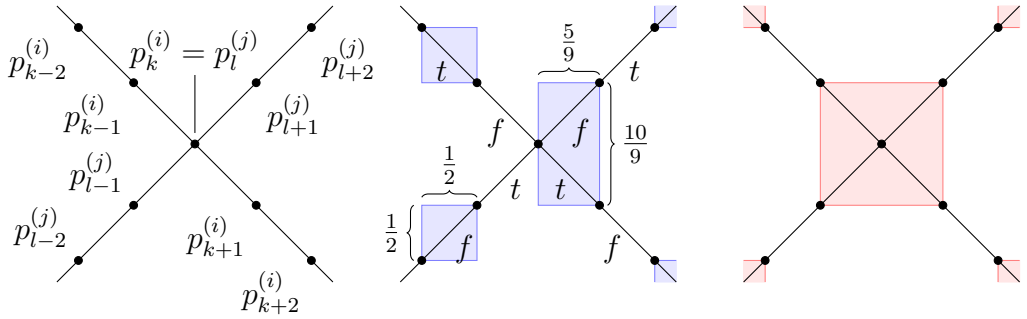


Figure 5.21: A diagonal junction between circuits P_i and P_j with a common point (left) and an optimal clustering shown as rectangle cover (middle) representing the variable assignment $u_i = true$ and $u_j = false$ with one cluster including three points with the exact distances indicated. A suboptimal cluster containing five points is shown on the right.

We require that two circuits cross diagonally in junctions such that the l_1-distance of the central point to each of the four surrounding points is 10/9, which violates condition i) above. This is shown in Figure 5.21. There are exactly four different ways a cluster B_k with $s(B_k) = 5/3$ can contain three points of the junction - one for every combination of variable assignments of the variables represented by the circuits.

As before, one point q_j is added for each clause E_j. The precise placement of the points in a clause configuration can be seen in Figure 5.22. We have to make sure that the Boolean labels on the edges of the circuits are aligned such that the edge of circuit P_i facing towards q_j has the label t if clause E_j contains u_i and f if E_j contains $\bar{u}_i$. This way, if a variable assignment satisfies E_j the point q_j can be included in one of the blue clusters in Figure 5.22. Those clusters have $s(B_k) = 5/3$. Other clusters with three points (for example the red clusters in Figure 5.22, right) have $s(B_k) > 5/3$.

Note, that there are two non-subsequent points in the left circuit (and similarly two in the right circuit) of the clause configuration, that violate circuit condition ii). However, aggregating the two points with the circuit point between them leads to another cluster B_k with $s(B_k) = 11/6 > 5/3$.

Let B be the set of all points in the construction, J be the number of

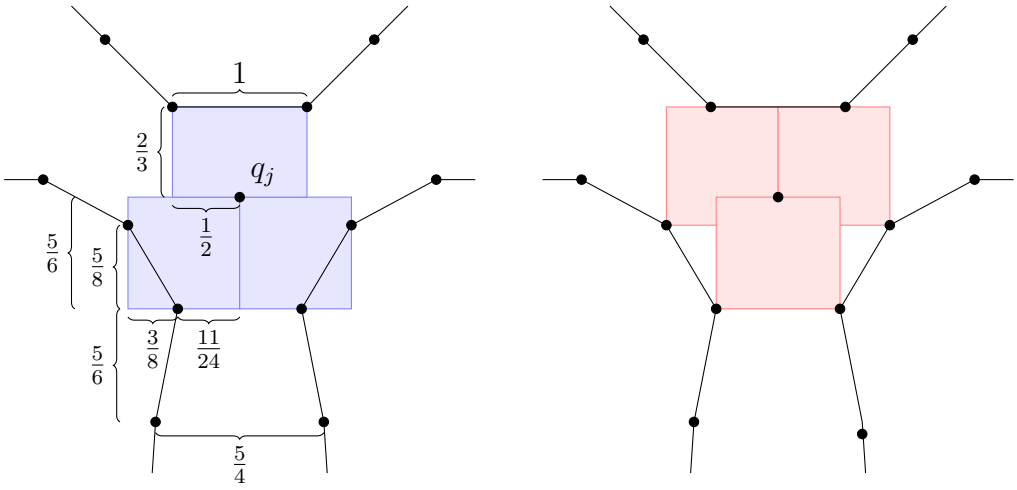


Figure 5.22: Clause configuration with the exact placements of points. One point q_j, which represents the clause E_j, is added. If E_j is satisfied by a variable assignment the point q_j can be added to one of the clusters of the circuits yielding one of the three blue clusters on the left. The right side shows three examples of clusters with three points each, two of which belong to different circuits. Since they have a larger extension s, clusterings containing one of these clusters are suboptimal.

junctions and S be the number of clauses in E. The number of points is

$$N := \sum_{i=1}^{r} 2m_i - J + S$$

and the number of clusters

$$n := \sum_{i=1}^{r} m_i - J.$$

Furthermore, we define $T := J + S = N - 2n$ and $R := N + 2T$. Then E is satisfiable if and only if there exists a clustering of the N points into n clusters, $\mathcal{B} = \{B_1, \ldots, B_n\}$, such that $G_{rec}(\mathcal{B}) \leq R$.

To prove this, we first assume E is satisfiable. Then there is a variable assignment that satisfies all of the clauses E_j in E simultaneously. Therefore, there is a clustering representing this variable assignment, such that the points q_j can be included in one of the blue clusters of Figure 5.22 for each clause

E_j. This clustering has a cluster with three points for each junction and each clause, hence T clusters with three points, each of which have an extension of $s(B_k) = 5/3$. The remaining $n - T$ clusters have two points each and an extension of $s(B_k) = 1$. It follows

$$\begin{aligned} G_{rec} &= \sum_{k=1}^{n} |B_k| \cdot s(B_k) \\ &= T \cdot 3 \cdot \frac{5}{3} + (n - T) \cdot 2 \cdot 1 \\ &= 2n + 3T = N + 2T \\ &= R. \end{aligned}$$

For the reverse direction assume we have a clustering $\mathcal{B} = \{B_1, \ldots, B_n\}$ of B with $G_{rec}(\mathcal{B}) \leq R$ and show that this implies E is satisfiable. We define a map $\sigma \colon \{1, \ldots, N\} \to \mathbb{R}_+$ by

$$\sigma(k) := \min \left\{ s(\bar{B}) \colon \bar{B} \subseteq B, |\bar{B}| = k \right\},$$

which is the smallest possible extension a cluster containing k points can have in the whole construction. The values of σ for small k can be seen in Table 5.7.3.

k	1	2	3	4	5	6	7
$\sigma(k)$	0	1	$\frac{5}{3}$	$\frac{25}{12}$	$\frac{20}{9}$	$\frac{17}{6}$	$\frac{19}{6}$

Table 5.1: Values of σ for small k.

The smallest clusters for $k = 4, 6, 7$ can be found in the clause configurations, while the smallest cluster for $k = 5$ is the center cluster in junctions (red cluster in Figure 5.21). This is assuming there are no smaller clusters aside from junctions and clause configurations, but they are easily avoidable in the construction. Since σ is non-decreasing in k, we have $\sigma(k) \geq \sigma(7) > 3$ for $k \geq 7$. Thus, σ satisfies

$$\sigma(k) > 3 - \frac{4}{k} \quad \forall\, k \geq 4.$$

Furthermore, for $k = 1, \ldots, N$ we define

$$a_k := \left|\left\{\bar{B} \in \mathcal{B} \colon |\bar{B}| = k\right\}\right|$$

as the number of clusters in the clustering $\mathcal{B}$ containing exactly k points. Since we have n clusters covering N points,

$$\sum_{k=1}^{N} a_k = n$$

and

$$\sum_{k=1}^{N} k \cdot a_k = N$$

and since $s(\bar{B}) \geq \sigma(|\bar{B}|)$ for all $\bar{B} \in \mathcal{B}$,

$$\begin{aligned}
\sum_{k=1}^{N} k \cdot \sigma(k) \cdot a_k &= \sum_{k=1}^{N} \sigma(k) \sum_{\substack{l=1 \\ |B_l|=k}}^{n} |B_l| \\
&\leq \sum_{k=1}^{N} \sum_{\substack{l=1 \\ |B_l|=k}}^{n} |B_l| \cdot s(B_l) \\
&= \sum_{l=1}^{n} |B_l| \cdot s(B_l) \\
&= G_{rec} \leq R = N + 2T.
\end{aligned}$$

With $a_k \geq 0$ for all $k = 1, \ldots, N$ and $n - T \geq 0$, this is the system of linear inequalities in Lemma 5.7.10 (see Section 5.7.4) and all of the assumptions are satisfied. Therefore, for our clustering $a_1 = 0$, $a_2 = n - T$, $a_3 = T$ and $a_k = 0$ for $k \geq 4$. Since

$$\sum_{k=1}^{N} k \cdot \sigma(k) \cdot a_k = T \cdot 3 \cdot \frac{5}{3} + (n - T) \cdot 2 \cdot 1 = N + 2T = R$$

is satisfied with equality, it follows that $s(\bar{B}) = \sigma(|\bar{B}|)$ for all $k = 1, \ldots, n$. Hence, the clustering consists of $n - T$ clusters with two points each and $s(\bar{B}) = 1$ and T clusters with three points each and $s(\bar{B}) = 5/3$.

The only places in the construction where a cluster with three points and an extension of 5/3 is possible are junctions and clause configurations with at most one cluster each. Since we have $T = J + S$ of these clusters there is one such cluster in each junction and clause configuration. Since all clusters with two points have an extension of 1, we can apply the same argument as in the last proof and conclude that the clustering represents a variable assignment and since each clause configuration has a cluster of extension 5/3, the variable assignment satisfies the clause E_j (compare Figure 5.22). Hence, it satisfies all clauses simultaneously and therefore also E.

This proves the NP-hardness of GC with respect to G_{rec}, but since all clusters $\bar{B} \in \mathcal{B}$ have $|\bar{B}| \leq 3 < N/2$, all arguments hold for G_{r3} as well. □

Proof of Theorem 5.2.3. This follows directly from the chain of reductions in the Propositions 5.7.2, 5.7.4, 5.7.6 and 5.7.8. □

5.7.4 Auxiliary Lemmas

Lemma 5.7.9. *Let $r, J, S \in \mathbb{N}$ and $m_i \in \mathbb{N}$ for $i = 1, \ldots, r$, such that*

$$\sum_{i=1}^{r} m_i - 2J - 7S \geq 0.$$

Define

$$N := \sum_{i=1}^{r} 2m_i - J + 7S \geq 0$$

and

$$n := \sum_{i=1}^{r} m_i - J.$$

Let $\sigma \colon \{1, \ldots, N\} \to \mathbb{R}_+$ be a map, such that $\sigma(1) = 0$, $\sigma(2) = 1$ and

$$\sigma(k) > 2 - \frac{2}{k} \quad \forall\ k \geq 3. \tag{5.1}$$

Then the following system of linear inequalities consisting of the variables $a_k, k = 1, \ldots, N, b_3, b_4, c_3$ *and* c_4*,*

$$b_3 + c_3 + b_4 + c_4 + \sum_{k=1}^{N} a_k = n \tag{5.2}$$

$$3(b_3 + c_3) + 4(b_4 + c_4) + \sum_{k=1}^{N} k \cdot a_k = N \tag{5.3}$$

$$3(b_3 + c_3) + 6(b_4 + c_4) + \sum_{k=1}^{N} k \cdot \sigma(k) \cdot a_k \leq N \tag{5.4}$$

$$b_3 + b_4 \leq S \tag{5.5}$$

$$c_3 + 2c_4 \leq 6S + J \tag{5.6}$$

$$b_3, b_4, c_3, c_4, a_k \geq 0 \quad \forall\ k = 1, \ldots, N \tag{5.7}$$

has the unique solution $a_1 = 0, a_2 = n - 7S - J, a_k = 0$ *for* $k = 3, \ldots, N, b_3 = S, b_4 = 0, c_3 = 6S + J$ *and* $c_4 = 0$*.*

Proof. Assume we have a solution $a_1, \ldots, a_N, b_3, b_4, c_3, c_4$ to the system (5.2) through (5.7). If we substract (5.2) from (5.3) we get

$$a_2 = N - n - 2(b_3 + c_3) - 3(b_4 + c_4) - \sum_{k=3}^{N} (k-1)a_k. \tag{5.8}$$

Inserting this into (5.4) yields

$$\begin{aligned} N &\geq 2a_2 + 3(b_3 + c_3) + 6(b_4 + c_4) + \sum_{k=3}^{N} k \cdot \sigma(k) \cdot a_k \\ &= 2N - 2n - (b_3 + c_3) + \sum_{k=3}^{N} (k \cdot \sigma(k) - 2(k-1)) \cdot a_k \\ &= N + 7S + J - (b_3 + c_3) + \sum_{k=3}^{N} (k \cdot \sigma(k) - 2(k-1)) \cdot a_k. \end{aligned} \tag{5.9}$$

Since $k \cdot \sigma(k) - 2(k-1) > 0$ for $k \geq 3$ due to (5.1), it follows from the above

inequality, (5.5), (5.6) and (5.7) that

$$\begin{aligned} 0 &\leq \sum_{k=3}^{N} (k \cdot \sigma(k) - 2(k-1)) \cdot a_k \\ &\leq b_3 + c_3 - 7S - J \\ &\leq S - b_4 + 6S + J - 2c_4 - 7S - J \\ &= -(b_4 + 2c_4) \leq 0. \end{aligned}$$

Hence, $b_4 = c_4 = 0$ and $a_k = 0$ for $k = 3, \ldots, N$. This reduces the inequality (5.9) to

$$b_3 + c_3 \geq 7S + J,$$

which, together with (5.5) and (5.6), implies $b_3 = S$ and $c_3 = 6S + J$. Solving (5.2) and (5.3) for a_1 and a_2 yields $a_1 = 0$ and $a_2 = n - 7S - J$. Since

$$a_2 = \sum_{i=1}^{r} m_i - 2J - 7S \geq 0$$

by assumption, this is indeed the unique solution to the system. □

Lemma 5.7.10. *Let $r, J, S \in \mathbb{N}$ and $m_i \in \mathbb{N}$ for $i = 1, \ldots, r$, such that*

$$\sum_{i=1}^{r} m_i - 2J - S \geq 0.$$

Define

$$N := \sum_{i=1}^{r} 2m_i - J + S,$$

$$n := \sum_{i=1}^{r} m_i - J$$

and $T := J + S = N - 2n$. Moreover, let $\sigma \colon \{1, \ldots, N\} \to \mathbb{R}_+$ be a map, such that $\sigma(1) = 0, \sigma(2) = 1, \sigma(3) = 5/3$ and

$$\sigma(k) > 3 - \frac{4}{k} \quad \forall\ k \geq 4. \tag{5.10}$$

Then the following system of linear inequalities,

$$\sum_{k=1}^{N} a_k = n \tag{5.11}$$

$$\sum_{k=1}^{N} k \cdot a_k = N \tag{5.12}$$

$$\sum_{k=1}^{N} k \cdot \sigma(k) \cdot a_k \leq N + 2T \tag{5.13}$$

$$a_k \geq 0 \quad \forall\ k = 1, \ldots, N \tag{5.14}$$

has the unique solution $a_1 = 0, a_2 = n - T, a_3 = T$ *and* $a_k = 0$ *for all* $k \geq 4$.

Proof. Assume we have a solution $a_1, \ldots, a_N$ to the system (5.11) through (5.14). If we substract (5.11) from (5.12) we get

$$a_2 = N - n - 2a_3 - \sum_{k=4}^{N} (k-1) \cdot a_k. \tag{5.15}$$

Inserting this into (5.13) yields

$$\begin{aligned} N + 2T &\geq 2a_2 + 5a_3 + \sum_{k=4}^{N} k \cdot \sigma(k) \cdot a_k \\ &= 2N - 2n + a_3 + \sum_{k=4}^{N} (k \cdot \sigma(k) - 2(k-1)) \cdot a_k \\ &= a_3 + N + T + \sum_{k=4}^{N} (k \cdot \sigma(k) - 2(k-1)) \cdot a_k \end{aligned}$$

and therefore

$$a_3 \leq T + \sum_{k=4}^{N} (2(k-1) - k \cdot \sigma(k)) \cdot a_k. \tag{5.16}$$

Now we combine (5.11), (5.14), (5.15) and (5.16) to get

$$\begin{aligned} 0 \leq a_1 &= n - a_2 - a_3 - \sum_{k=4}^{N} a_k \\ &= 2n - N + a_3 + \sum_{k=4}^{N} (k-2) \cdot a_k \\ &\leq -T + T + \sum_{k=4}^{N} (k - 2 + 2(k-1) - k \cdot \sigma(k)) \cdot a_k \\ &= \sum_{k=4}^{N} (3k - 4 - k \cdot \sigma(k)) \cdot a_k. \end{aligned} \tag{5.17}$$

We know from (5.10) that $3k - 4 - k \cdot \sigma(k) < 0$ for all $k = 4, \ldots, N$. Thus, in order for (5.17) to be satisfied, we need $a_k = 0$ for all $k = 4, \ldots, N$. It follows immediately that $a_1 = 0$. Solving (5.11) and (5.12) for a_2 and a_3 leads to $a_2 = n - T$ and $a_3 = T$. This satisfies

$$a_2 = n - T = \sum_{i=1}^{r} m_i - 2J - S \geq 0$$

and (5.13), hence it is the unique solution to the above inequation system. □

Bibliography

[1] Agueh, M., Carlier, G.: Barycenters in the Wasserstein space. SIAM J. Math. Anal. **43**(2), 904–924 (2011)

[2] Altschuler, J., Weed, J., Rigollet, P.: Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. CoRR **abs/1705.09634** (2017). URL `http://arxiv.org/abs/1705.09634`

[3] Arjovski, M., Chintala, S., Bottou, L.: Wasserstein GAN (2017). URL `https://arxiv.org/abs/1701.07875`

[4] Aurenhammer, F., Hoffmann, F., Aronov, B.: Minkowski-Type Theorems and Least-Squares Clustering. Algorithmica **20**(1), 61–76 (1998)

[5] Bazaraa, M.S., Jarvis, J.J., Sherali, H.D.: Linear Programming and Network Flows, fouth edn. Wiley (2009)

[6] Becker, G.: A Theory of Marriage: Part I. Journal of Political Economy **81**(4), 813–46 (1973). URL `https://EconPapers.repec.org/RePEc:ucp:jpolec:v:81:y:1973:i:4:p:813-46`

[7] Benamou, J.D., Carlier, G., Cuturi, M., Nenna, L., Peyré, G.: Iterative Bregman Projections for Regularized Transportation Problems. SIAM J. Sci. Comput. **37**(2), A1111–A1138 (2015)

[8] Benamou, J.D., Froese, B.D., Oberman, A.M.: Numerical solution of the Optimal Transportation problem using the Monge-Ampère equation. J. Comput. Phys. **260**, 107–126 (2014)

[9] Bertsekas, D.P.: A new algorithm for the assignment problem. Mathematical Programming **21**(1), 152–171 (1981). DOI 10.1007/BF01584237

[10] Bertsekas, D.P.: Auction Algorithms for Network Flow Problems: A Tutorial Introduction. Computational Optimization and Applications **1**(1), 7–66 (1992)

[11] Bertsekas, D.P., Castanon, D.A.: The auction algorithm for the transportation problem. Annals of Operations Research **20**(1), 67–96 (1989). DOI 10.1007/BF02216923

[12] Bickel, P.J., Freedman, D.A.: Some Asymptotic Theory for the Bootstrap. Ann. Statist. **9**(6), 1196–1217 (1981)

[13] Boissard, E., Le Gouic, T.: On the mean speed of convergence of empirical and occupation measures in Wasserstein distance. Ann. Inst. H. Poincaré Probab. Statist. **50**(2), 539–563 (2014)

[14] Bonneel, N., Rabin, J., Peyré, G., Pfister, H.: Sliced and Radon Wasserstein Barycenters of Measures. Journal of Mathematical Imaging and Vision **51**(1), 22–45 (2015)

[15] Bosc, D.: Numerical Approximation of Optimal Transport Maps. ERN: Optimization Techniques; Programming Models; Dynamic Analysis (Topic) (2010). DOI 10.2139/ssrn.1730684

[16] Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C.J., Schoelkopf, B.: From optimal transport to generative modeling: the VEGAN cookbook (2017). URL `http://arxiv.org/abs/1705.09634`

[17] Brauer, C., Lorenz, D.: Cartoon-Texture-Noise Decomposition with Transport Norms. In: Proc. SSVM, pp. 142–153. Lege-Cap Ferret, France (2015)

[18] Bravo, H.C., Theussl, S.: Rcplex: R Interface to CPLEX (2016). URL `https://CRAN.R-project.org/package=Rcplex`. R package version 0.3-3

[19] Ciscato, E., Galichon, A., Goussé, M.: Like Attract Like? A Structural Comparison of Homogamy Across Same-Sex and Different-Sex Households. SSRN Electronic Journal (2014). DOI 10.2139/ssrn.2530724

[20] Cuturi, M.: Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In: Proc. NIPS, pp. 2292–2300 (2013)

[21] Cuturi, M., Doucet, A.: Fast Computation of Wasserstein Barycenters. In: Proceedings of The 31st International Conference on Machine Learning, pp. 685–693 (2014)

[22] Defays, D.: An efficient algorithm for a complete link method. The Computer Journal **20**(4), 364–366 (1977). DOI 10.1093/comjnl/20.4.364. URL `http://dx.doi.org/10.1093/comjnl/20.4.364`

[23] Dolinsky, Y., Soner, H.M.: Martingale optimal transport and robust hedging in continuous time. Probability Theory and Related Fields **160**(1), 391–427 (2014)

[24] Dorea, C.C.Y., Ferreira, D.B.: Conditions for equivalence between Mallows distance and convergence to stable laws. Acta Math Hung **134**(1-2), 1–11 (2012)

[25] Dupuy, A., Galichon, A., Jaffe, S., Duke Kominers, S.: Taxation in Matching Markets. SSRN Electronic Journal (2017). DOI 10.2139/ssrn.3060746

[26] Flamary, R., Févotte, C., Courty, N., Emiya, V.: Optimal spectral transportation with application to music transcription. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, pp. 703–711. Curran Associates Inc., USA (2016). URL `http://dl.acm.org/citation.cfm?id=3157096.3157175`

[27] Florek, K., Lukaszewicz, J., Perkal, J., Steinhaus, H., Zubrzycki, S.: Sur la liaison et la division des points d'un ensemble fini. Colloquium Mathematicae **2**(3-4), 282–285 (1951). URL `http://eudml.org/doc/209969`

[28] Fournier, N., Guillin, A.: On the rate of convergence in Wasserstein distance of the empirical measure. Probab. Theory Relat. Fields **162**(3-4), 707–738 (2015)

[29] Freitag, G., Munk, A.: On Hadamard differentiability in k-sample semiparametric models—with applications to the assessment of structural relationships. Journal of Multivariate Analysis **94**(1), 123–158 (2005)

[30] Frogner, C., Zhang, C., Mobahi, H., Araya-Polo, M., Poggio, T.: Learning with a Wasserstein Loss. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS'15, pp. 2053–2061. MIT Press, Cambridge, MA, USA (2015). URL `http://dl.acm.org/citation.cfm?id=2969442.2969469`

[31] Fu, A., Wenyin, L., Deng, X.: Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD). IEEE Transactions on Dependable and Secure Computing **3**(4), 301–311 (2006)

[32] Galichon, A.: Optimal Transport Methods in Economics. Princeton University Press (2016)

[33] Galichon, A., Salanie, B.: Cupid's Invisible Hand: Social Surplus and Identification in Matching Models. SSRN Electronic Journal (2012)

[34] Gangbo, W., McCann, R.J.: Shape recognition via Wasserstein distance. Quarterly of Applied Mathematics **LVIII**(4), 705–737 (2000)

[35] Gerber, S., Maggioni, M.: Multiscale Strategies for Computing Optimal Transport. Journal of Machine Learning Research **18**(72), 1–32 (2017). URL `http://jmlr.org/papers/v18/16-108.html`

[36] Gneiting, T., Guttorp, P.: Continuous Parameter Spatio-Temporal Processes. In: Handbook of spatial statistics, Chapman & Hall/CRC Handb. Mod. Stat. Methods, pp. 427–436. CRC Press, Boca Raton, FL (2010)

[37] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Nets. In:

Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Associates, Inc. (2014). URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`

[38] Gottschlich, C., Huckemann, S.: Separating the Real from the Synthetic: Minutiae Histograms as Fingerprints of Fingerprints. IET Biometrics **3**(4), 291–301 (2014)

[39] Gottschlich, C., Schuhmacher, D.: The Shortlist Method for Fast Computation of the Earth Mover's Distance and Finding Optimal Solutions to Transportation Problems. PLoS ONE **9**(10), e110,214 (2014)

[40] Grauman, K., Darrell, T.: Fast Contour Matching Using Approximate Earth Mover's Distance. In: Proc. CVPR, pp. 220 –227. Washington, DC, USA (2004)

[41] Hartmann, V., Schuhmacher, D.: Semi-discrete optimal transport - the case p=1 (2017). URL `https://arxiv.org/abs/1706.07650`

[42] Hug, R., Maitre, E., Papadakis, N.: Multi-physics optimal transportation and image interpolation. ESAIM: M2AN **49**(6), 1671–1692 (2015). DOI 10.1051/m2an/2015038. URL `https://doi.org/10.1051/m2an/2015038`

[43] Ilgen, P., Stoldt, S., Conradi, L.C., Wurm, C., Rüschoff, J., Ghadimi, B., Liersch, T., Jakobs, S.: STED Super-Resolution Microscopy of Clinical Paraffin-Embedded Human Rectal Cancer Tissue. PLoS ONE **9**(7), e101,563 (2014)

[44] Jans, D., Wurm, C., Riedel, D., Wenzel, D., Stagge, F., Deckers, M., Rehling, P., Jakobs, S.: STED super-resolution microscopy reveals an array of MINOS clusters along human mitochondria. PNAS **110**(22), 8936–8941 (2013)

[45] Johnson, O., Samworth, R.: Central limit theorem and convergence to stable laws in Mallows distance. Bernoulli **11**(5), 829–845 (2005)

[46] Kantorovich, L.V.: On the translocation of masses. (Dokl.) Acad. Sci. URSS 37 **3**, 199–201 (1942)

[47] Kantorovich, L.V., Rubinstein, G.S.: On a space of completely additive functions. Vestnik Leningrad. Univ **13**(7), 52–59 (1958)

[48] Kendal, D., Hauser, C., Garrard, G., Jellinek, S., Giljohann, K., Moore, J.: Quantifying Plant Colour and Colour Difference as Perceived by Humans Using Digital Images. PLoS ONE **8**(8), e72,296 (2013)

[49] Kitagawa, J., Mérigot, Q., Thibert, B.: Convergence of a Newton algorithm for semi-discrete optimal transport (2017). URL `https://arxiv.org/abs/1603.05579`

[50] Koopmans, T.C., Beckmann, M.: Assignment Problems and the Location of Economic Activities. Econometrica **25**(1), 53–76 (1957). URL `http://www.jstor.org/stable/1907742`

[51] Kuhn, H.W., Yaw, B.: The Hungarian method for the assignment problem. Naval Res. Logist. Quart pp. 83–97 (1955)

[52] Lellmann, J., Lorenz, D., Schönlieb, C.B., Valkonen, T.: Imaging with Kantorovich–Rubinstein discrepancy. SIAM Journal on Imaging Sciences **7**(4), 2833–2859 (2014)

[53] Ling, H., Okada, K.: An Efficient Earth Mover's Distance Algorithm for Robust Histogram Comparison. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(5), 840–853 (2007)

[54] Luenberger, D.G., Ye, Y.: Linear and Nonlinear Programming, third edn. International Series in Operations Research & Management Science, 116. Springer, New York (2008)

[55] Mallows, C.L.: A Note on Asymptotic Joint Normality. Ann. Math. Statist. **43**(2), 508–515 (1972)

[56] Megiddo, N., Supowit, K.: On the complexity of some common geometric location problems. SIAM J. Comput. **13**(1), 182–196 (1984). DOI 10.1137/0213014. URL `https://doi.org/10.1137/0213014`

[57] Mérigot, Q.: A multiscale approach to optimal transport. Comput. Graph. Forum **30**(5), 1583–1592 (2011)

[58] Meyer, Y.: Oscillating Patterns in Image Processing and Nonlinear Evolution Equations. American Mathematical Society, Boston, MA, USA (2001)

[59] Monge, G.: Mémoire sur la théorie des déblais et des remblais. Histoire de l'Académie Royale des Sciences pp. 666–704 (1781)

[60] Munk, A., Czado, C.: Nonparametric Validation of Similar Distributions and Assessment of Goodness of Fit. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **60**(1), 223–241 (1998)

[61] Ni, K., Bresson, X., Chan, T., Esedoglu, S.: Local Histogram Based Segmentation Using the Wasserstein Distance. International Journal of Computer Vision **84**(1), 97–111 (2009)

[62] Oberman, A., Ruan, Y.: An efficient linear programming method for Optimal Transportation. Preprint (2015). URL `http://arxiv.org/abs/1509.03668`

[63] Papadakis, N.: Optimal Transport for Image Processing. Habilitation à diriger des recherches, Université de Bordeaux (2015). URL `https://hal.archives-ouvertes.fr/tel-01246096`

[64] Pele, O., Werman, M.: Fast and Robust Earth Mover's Distances. In: ICCV (2009)

[65] Peyré, G., Cuturi, M.: Computational Optimal Transport (2018). URL `https://arxiv.org/abs/1803.00567`

[66] R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2016). URL `https://www.R-project.org/`. Version 3.3.0

[67] Rabin, J., Ferradans, S., Papadakis, N.: Adaptive color transfer with relaxed optimal transport. In: 2014 IEEE International Conference on Image Processing (ICIP), pp. 4852–4856 (2014). DOI 10.1109/ICIP.2014.7025983

[68] Rabin, J., Papadakis, N.: Convex Color Image Segmentation with Optimal Transport Distances. CoRR **abs/1503.01986** (2015). URL `http://arxiv.org/abs/1503.01986`

[69] Rachev, S.T., Stoyanov, S.V., Fabozzi, F.J.: A Probability Metrics Approach to Financial Risk Measures. Wiley-Blackwell (2011)

[70] Rolet, A., Cuturi, M., Peyré, G.: Fast Dictionary Learning with a Smoothed Wasserstein Loss. In: A. Gretton, C.C. Robert (eds.) Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, *Proceedings of Machine Learning Research*, vol. 51, pp. 630–638. PMLR, Cadiz, Spain (2016). URL `http://proceedings.mlr.press/v51/rolet16.html`

[71] Rubner, Y., Tomasi, C., Guibas, L.J.: The Earth Mover's Distance as a Metric for Image Retrieval. International Journal of Computer Vision **40**(2), 99–121 (2000)

[72] Rudolf, D., Schweizer, N.: Perturbation theory for Markov chains via Wasserstein distance. Bernoulli **24**(4A), 2610–2639 (2018)

[73] Ruttenberg, B.E., Luna, G., Lewis, G.P., Fisher, S.K., Singh, A.K.: Quantifying spatial relationships from whole retinal images. Bioinformatics **29**(7), 940–946 (2013)

[74] Santambrogio, F.: Optimal Transport for Applied Mathematicians. Birkhäuser Basel (2015)

[75] Schlather, M., Malinowski, A., Oesting, M., Boecker, D., Strokorb, K., Engelke, S., Martini, J., Ballani, F., Moreva, O.: RandomFields: Simulation and Analysis of Random Fields (2016). URL `https://cran.r-project.org/web/packages/RandomFields/`. R package version 3.1.12

[76] Schmitzer, B.: A Sparse Multiscale Algorithm for Dense Optimal Transport. J Math Imaging Vis **56**(2), 238–259 (2016)

[77] Schmitzer, B.: Stabilized Sparse Scaling Algorithms for Entropy Regularized Transport Problems (2016). URL `https://arxiv.org/abs/1610.06519`

[78] Schrieber, J., Schuhmacher, D., Gottschlich, C.: DOTmark – A Benchmark for Discrete Optimal Transport. IEEE Access **5**, 271–282 (2016)

[79] Schuhmacher, D.: Stein's method and Poisson process approximation for a class of Wasserstein metrics. Bernoulli **15**(2), 550–568 (2009). URL `http://www.jstor.org/stable/20680164`

[80] Schuhmacher, D., Bähre, B., Gottschlich, C.: transport: Optimal Transport in Various Forms (2016). URL `https://cran.r-project.org/web/packages/transport/`. R package version 0.7-4

[81] Schwinn, J., Werner, R.: On the effectiveness of primal and dual heuristics for the transportation problem. IMA Journal of Management Mathematics (2017)

[82] Shalev-Shwartz, S., Shamir, O., Srebro, N., Sridharan, K.: Learnability, Stability and Uniform Convergence. Journal of Machine Learning Research **11**(Oct), 2635–2670 (2010)

[83] Shimer, R.: The Assignment of Workers to Jobs In an Economy with Coordination Frictions. Working Paper 8501, National Bureau of Economic Research (2001). DOI 10.3386/w8501. URL `http://www.nber.org/papers/w8501`

[84] Shirdhonkar, S., Jacobs, D.W.: Approximate earth mover's distance in linear time. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)

[85] Sibson, R.: Slink: An optimally efficient algorithm for the single-link cluster method. Comput. J. **16**(1), 30–34 (1973)

[86] Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pacific J. Math. **21**(2), 343–348 (1967)

[87] Solomon, J., de Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., Guibas, L.: Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains. ACM Trans. Graph. **34**(4), 66:1–66:11 (2015)

[88] Sommerfeld, M.: Wasserstein Distance on Finite Spaces: Statistical Inference and Algorithms (2017). URL `https://ediss.uni-goettingen.de/handle/11858/00-1735-0000-0023-3FA1-C`

[89] Sommerfeld, M., Munk, A.: Inference for empirical Wasserstein distances on finite spaces. J. R. Stat. Soc. B **80**(1), 219–238 (2018)

[90] Sommerfeld, M., Schrieber, J., Zemel, Y., Munk, A.: Optimal Transport: Fast Probabilistic Approximation with Exact Solvers (2018). URL `https://arxiv.org/abs/1802.05570`

[91] Srivastava, S., Li, C., Dunson, D.B.: Scalable Bayes via Barycenter in Wasserstein Space. arXiv:1508.05880 (2015)

[92] Tameling, C., Sommerfeld, M., Munk, A.: Empirical optimal transport on countable metric spaces: Distributional limits and statistical applications (2018). URL `https://arxiv.org/abs/1707.00973`

[93] Thai, D., Gottschlich, C.: Directional global three-part image decomposition. EURASIP Journal on Image and Video Processing **2016**(12), 1–20 (2016)

[94] Thai, D., Gottschlich, C.: Global variational method for fingerprint segmentation by three-part decomposition. IET Biometrics **5**(2), 120–130 (2016)

[95] Vasershtein, L.N.: Markov processes over denumerable products of spaces describing large system of automata. Problemy Peredači Informacii **5**(3), 64–72 (1969)

[96] Villani, C.: Optimal transport, old and new, *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, vol. 338. Springer-Verlag, Berlin (2009)

[97] Wilson, A.G.: The Use of Entropy Maximising Models, in the Theory of Trip Distribution, Mode Split and Route Split. Journal of Transport Economics and Policy **3**(1), 108–126 (1969)

[98] Wurm, C., Neumann, D., Lauterbach, M., Harke, B., Egner, A., Hell, S., Jakobs, S.: Nanoscale distribution of mitochondrial import receptor Tom20 is adjusted to cellular conditions and exhibits an inner-cellular gradient. PNAS **108**(33), 13,546–13,551 (2011)

[99] Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. Int J Comput Vision **73**(2), 213–238 (2007)

Curriculum Vitae

Jörn Schrieber

Born 5th November 1989 in Hamburg
Annastr. 5, 37075 Göttingen
joern.schrieber-1@mathematik.uni-goettingen.de

Education

since 2015	Ph.D. student and research assistant, Institute for Mathematical Stochastics, Georg-August-Universität Göttingen
2015	M.Sc. in Mathematics, Georg-August-Universität Göttingen Thesis: *Recovery to Feasibility for Integer Programs*
2013	B.Sc. in Mathematics, Georg-August-Universität Göttingen Thesis: *Das injektive Spektrum von Ringen*
2009	Abitur, Albert-Einstein-Gymnasium, Buchholz i.d.N.

Publications

- J. Schrieber, D. Schuhmacher, C. Gottschlich: DOTmark – A Benchmark for Discrete Optimal Transport. IEEE Access, Volume 5, 271–282 (2016)
- M. Sommerfeld, J. Schrieber, Y. Zemel, A. Munk: Optimal Transport: Fast Probabilistic Approximation with Exact Solvers, submitted. https://arxiv.org/abs/1802.05570 (2018)