GridOT – a discrete optimal transport solver on grids

Johannes Rauch
Ulm University, Institute of Optimization and Operations Research, Germany

Leo Zanotti
Ulm University, Institute of Optimization and Operations Research, Germany

Abstract

TODO: Abstract; Leo's ORCID; Check if references are actually OT on grids

2012 ACM Subject Classification Mathematics of computing → Mathematical software

Keywords and phrases Discrete optimal transport

Supplementary Material `https://github.com/johannesrauch/GridOT/`

Acknowledgements We thank Henning Bruhn-Fujimoto for the introduction to the problem and helpful discussions.

1 Introduction

Informally, in the *optimal transport* problem we are given two probability distributions and a cost function, and we seek a minimum cost transport function that transforms one probability function into the other. In *discrete* optimal transport, the given probability distributions are discrete. Discrete optimal transport *on grids* means that the underlying cost function has geometric properties; it is for example a metric. Discrete optimal transport on grids is an important special case of the general optimal transport problem, because it has numerous applications in image processing and computer vision. For instance, Werman et al. [6] use a discrete optimal transport problem to define a distance metric for multidimensional histograms. Building on this, Peleg et al. [3] present a unified treatment of spatial and gray-level resolution in image digitization. As image pixels are conventionally arranged in two-dimensional grids, their approaches rely on solving discrete optimal transport on 2-dimensional grids. Thus, the need for a practically fast solver is evident.

1.1 Preliminaries

We introduce discrete optimal transport on grids formerly before proceeding. For a finite set S let $\mathcal{P}(S)$ denote the set of probability measures over S with the power set as the σ-algebra. Given two finite sets X and Y together with two probability measures $\mu \in \mathcal{P}(X)$ and $\nu \in \mathcal{P}(Y)$, the set of couplings between μ and ν is given by

$$\Pi(\mu,\nu) = \{\pi \in \mathcal{P}(X \times Y) : \pi(\{x\} \times Y) = \mu(x), \mu(X \times \{y\}) = \nu(y) \text{ for all } x \in X,\, y \in Y\}.$$

For a cost function $c : X \times Y \to \mathbb{R} \cup \{\infty\}$ the discrete optimal transport problem is

$$\min_{\pi \in \Pi(\mu,\nu)} C(\pi), \quad \text{where} \quad C(\pi) = \sum_{(x,y) \in X \times Y} c(x,y)\pi(x,y). \tag{P}$$

Let $[i] = \{1, 2, \dots, i\}$ for an integer i. We say that (P) is on a grid if $X = [x_1] \times \cdots \times [x_d]$ and $Y = [y_1] \times \cdots \times [y_d]$ for some positive integers d, $x_1, \dots, x_d$ and $y_1, \dots, y_d$, and c denotes the squared Euclidean distance. The discrete optimal transport problem (P) is "dense" in

the sense that an algorithm solving (P) has to consider couplings of all elements in $X \times Y$, of which there are quadratically many. If $N \subset X \times Y$ is a "small" subset, then the restriction

$$\min_{\pi \in \Pi(\mu,\nu), \operatorname{supp} \pi \subseteq N} C(\pi) \tag{P'}$$

of (P) is "sparse" in the sense that an algorithm solving (P') only has to consider couplings of all elements in N. We say that N is the *neighborhood* of (P'). Of course, $\mathrm{OPT}(P) \leq \mathrm{OPT}(P')$ holds for any neighborhood $N \subseteq X \times Y$, and ideally one would want to have a "small" neighborhood N such that $\mathrm{OPT}(P) = \mathrm{OPT}(P')$.

1.2 Related work

Schmitzer [4] devised a sparse multi-scale algorithm for dense optimal transport, which works very well in practice [5]. Ignoring the multi-scale scheme, we outline his algorithm in Algorithm 1. It repeatedly solves a restricted discrete optimal transport problem (P') in a neighborhood N and updates N to solve the discrete optimal transport problem (P) at hand. For grids he shows how to construct and update the neighborhoods N in a sparse manner explicitly. In particular, $\mathrm{OPT}(P) = \mathrm{OPT}(P')$ holds for the neighborhood N after the termination of the algorithm. We refer to his article for more details [4].

Input: An instance of discrete optimal transport (P) and a neighborhood N.
Output: An optimal coupling.
repeat
  solve the restricted discrete optimal transport problem (P') in neighborhood N
  update N
until *the cost of the coupling does not improve anymore*
output the last coupling

Algorithm 1 An outline of Schmitzer's algorithm for dense optimal transport [4] without the multi-scale scheme.

Schmitzer solves the restricted discrete optimal transport problem (P') in Algorithm 1 with the network simplex algorithm [1]. He uses the CPLEX[1] and LEMON [2] network simplex implementations for his numerical experiments[2]. Since the problems (P') are very similar in every iteration of Algorithm 1, it is natural to preserve information in the network simplex algorithm and use warmstarts. While the CPLEX provides an interface to set a basis for a warmstart, LEMON does not. To make up for this, Schmitzer uses a trick that modifies the cost function throughout the execution of Algorithm 1 with the LEMON library. Remarkably, he finds that LEMON outperforms CPLEX with this trick [4].

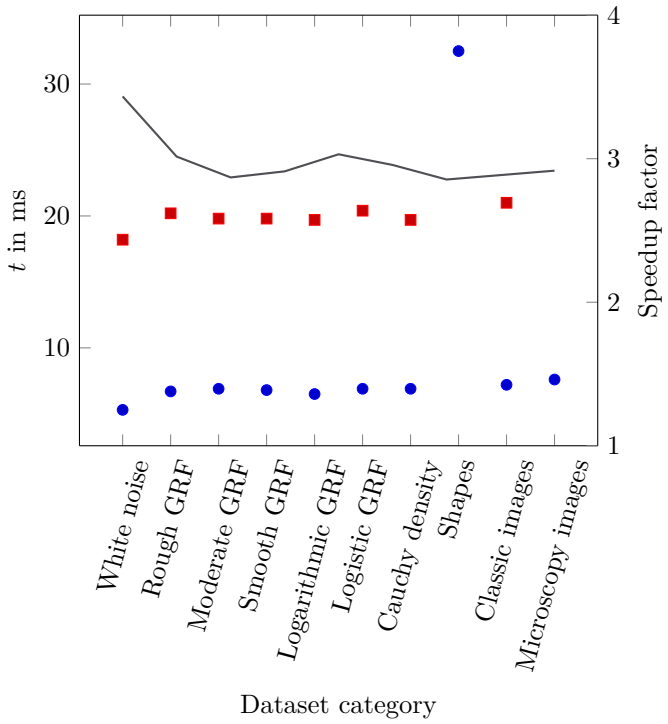
1.3 Our contribution

We provide a new C++-implementation of Schmitzer's [4] sparse multi-scale algorithm for dense optimal transport on grids and a network simplex implementation that is adapted from LEMON. Our program distinguishes itself from Schmitzer's in the following points. We adhere to LEMON's design choice and use compile-time polymorphisms (templates) instead

[1] `https://www.ibm.com/products/ilog-cplex-optimization-studio`
[2] He also uses the cost scaling implementation of LEMON [2], which is outperformed by the network simplex algorithms.

of run-time polymorphisms for efficency. Moreover, we use many of the standard library utilities to avoid dynamically allocating memory and handling raw pointers manually to avoid memory leaks and ensure memory safety. (Using Valgrind[3] we noticed memory leaks in Schmitzer's solver.) On the DOTmark benchmark [5], this results in roughly a 2–4 times faster run-time compared to Schmitzer's solver (see Section 2). Our code is open source and publicly available on GitHub[4].

Figure 1 The average runtimes in each dataset category and dimension. The runtimes of our solver are blue while the runtimes of Schmitzer's solver are red.



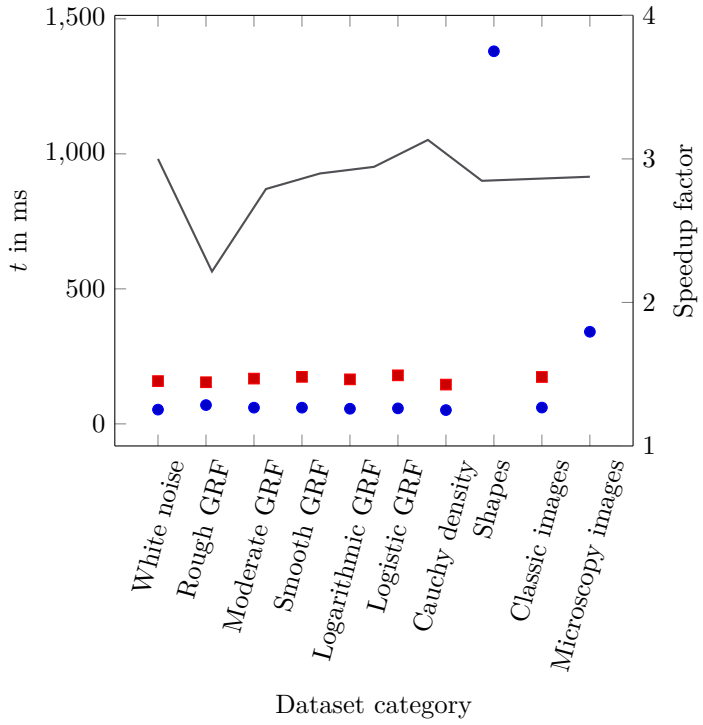
(a) Grid dimension: 32×32.

2 Results

We run the benchmark on a computer with a Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz and 2×8 GiB DDR4 DIMM @ 2133 MHz in dual channel mode. We use the DOTmark [5] dataset as a benchmark. DOTmark consists of several dataset categories: White noise, rough, moderate and smooth Gaussian random field (GRF), logarithmic and logistic GRF, Cauchy density, shapes, classic images, and microscopy images. These categories can be divided into two parts: One part is randomly generated and the other part is taken from practical applications. We refer the reader to the article of Schrieber et al. [5] for more details.

One dataset of DOTmark corresponds to a measure μ or ν, respectively, in a discrete optimal transport problem (P) on grids. In each dataset category of DOTmark, we pair every

[3] `https://valgrind.org/`
[4] `https://github.com/johannesrauch/GridOT/`



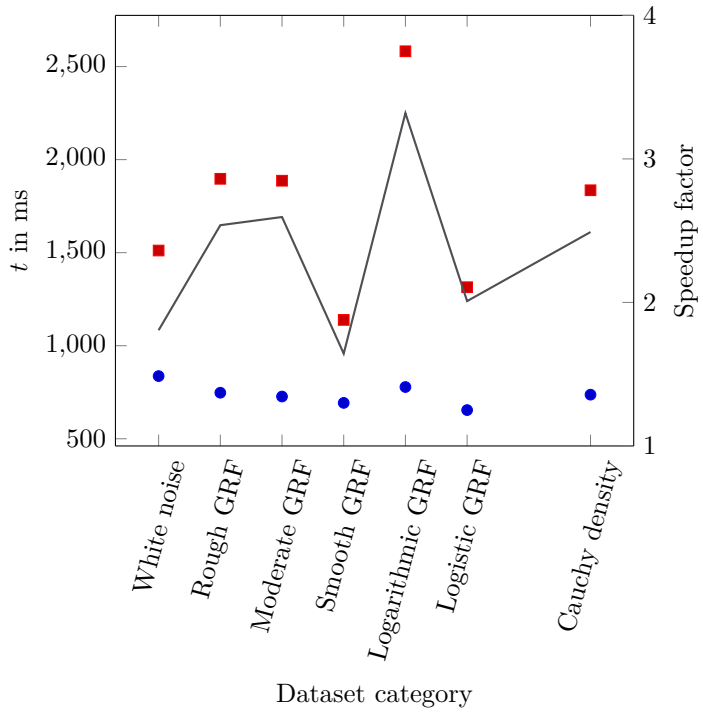
(b) Grid dimension: 64×64.

two datasets and solve the corresponding discrete optimal transport on grids instance ten times. The averages of these ten runs constitute the runtimes we consider in the subfigures of Figure 2. The average runtime in each dataset category and dimension is shown as an overview in Figure 1.

Unfortunately, Schmitzer's solver [4] can only handle strictly positive measures μ and ν, although measures with $\mu(x) = 0$ or $\nu(y) = 0$ for some $x \in X$ or $y \in Y$, respectively, are common in practical applications as DOTmark shows. Therefore, some of the datapoints in Figure 1 and Figure 2 are missing, and we are unable to compare our solver to Schmitzer's solver in every dataset categorie without modifying the DOTmark benchmark dataset.

3 Conclusion

We showed that there is room for improvement in discrete optimal transport solvers, which are still competitive when compared to continuous, numeric solvers [5]. We focused on the case where the domains X and Y of (P) are grids and the cost function c is the squared Euclidean distance and achieved roughly a 2–4 times faster runtime compared to Schmitzer's work. It would be nice to have improved solvers for other geometric constellations of X and Y and other cost functions c. For this, we think that the work of Schmitzer [4] is again a good starting point, since he not only considers grids and squared Euclidean distance, but also $X, Y \subseteq \mathbb{R}^d$ with squared Euclidean distance or strictly convex cost functions, and $X, Y \subseteq S_d = \{x \in \mathbb{R}^d : \|x\| = 1\}$ with squared geodesic distance. Another direction for future work is to implement a general interface to support warmstarts in LEMON's network simplex algorithm. As evident in Schmitzer's article [4], LEMON is a powerful, open source library and the network simplex algorithm is an important tool in other algorithms. Therefore, it would be a good idea to further develop LEMON and unfurl its full potential.



(c) Grid dimension: 128×128.

References

1 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

2 Balázs Dezso, Alpár Jüttner, and Péter Kovács. LEMON - an open source C++ graph template library. In Zoltán Porkoláb and Norbert Pataki, editors, *Proceedings of the Second Workshop on Generative Technologies, WGT@ETAPS 2010, Paphos, Cyprus, March 27, 2010*, volume 264 of *Electronic Notes in Theoretical Computer Science*, pages 23–45. Elsevier, 2010.

3 Shmuel Peleg, Michael Werman, and Hillel Rom. A unified approach to the change of resolution: Space and gray-level. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):739–742, 1989.

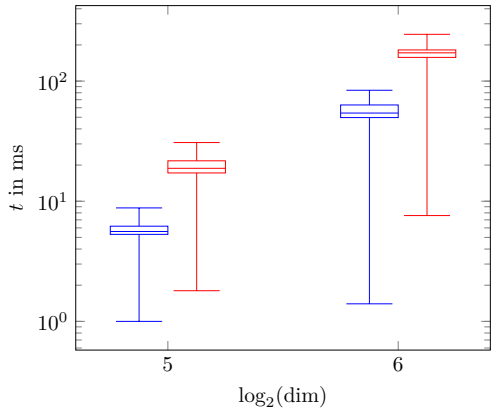
4 Bernhard Schmitzer. A sparse multiscale algorithm for dense optimal transport. *Journal of Mathematical Imaging and Vision*, 56:238–259, 2016.

5 Jörn Schrieber, Dominic Schuhmacher, and Carsten Gottschlich. DOTmark - A benchmark for discrete optimal transport. *IEEE Access*, 5:271–282, 2017.

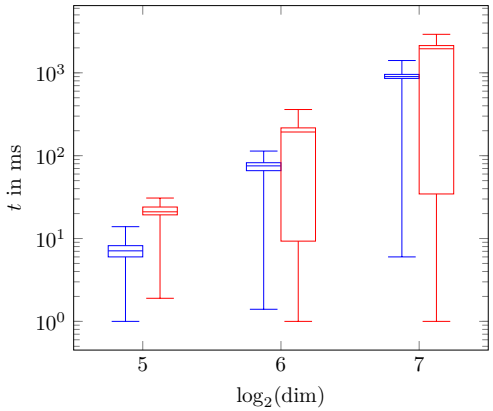
6 Michael Werman, Shmuel Peleg, and Azriel Rosenfeld. A distance metric for multidimensional histograms. *Comput. Vis. Graph. Image Process.*, 32(3):328–336, 1985.

A Boxplots

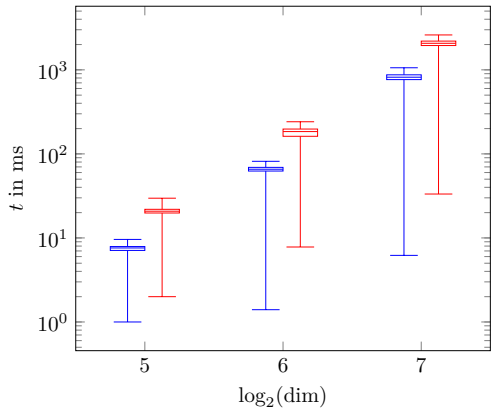
Figure 2 Boxplots of the averaged runtimes for each pair in each dataset category of DOTmark. The runtimes of our solver are blue while the runtimes of Schmitzer's solver are red.



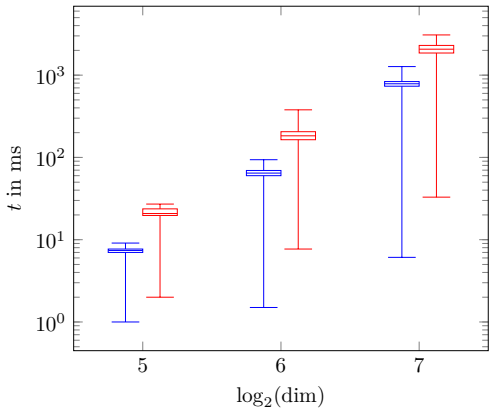
(a) White noise.



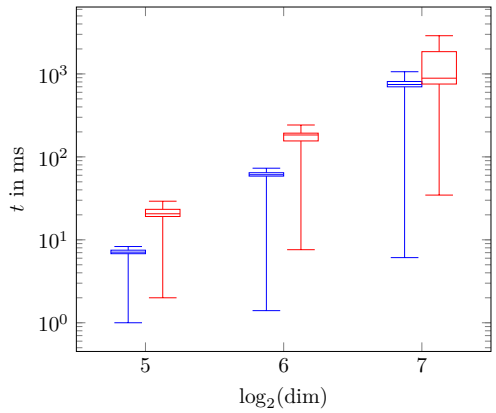
(b) Rough GRF.



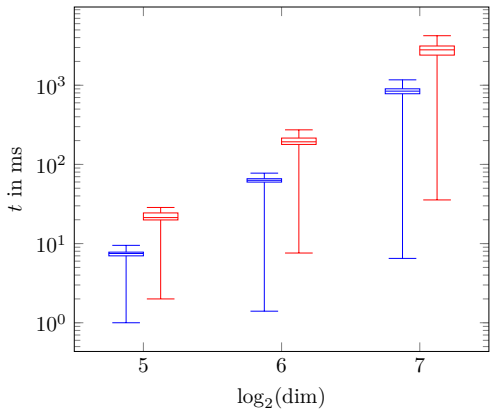
(c) Moderate GRF.



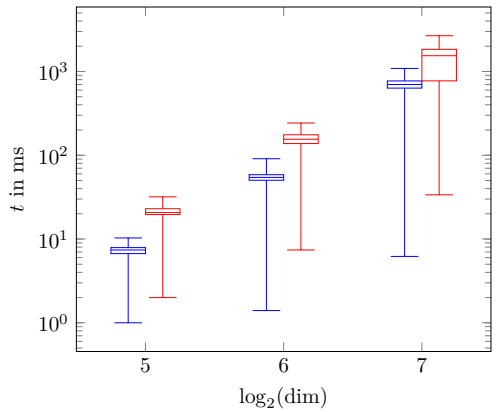
(d) Smooth GRF.



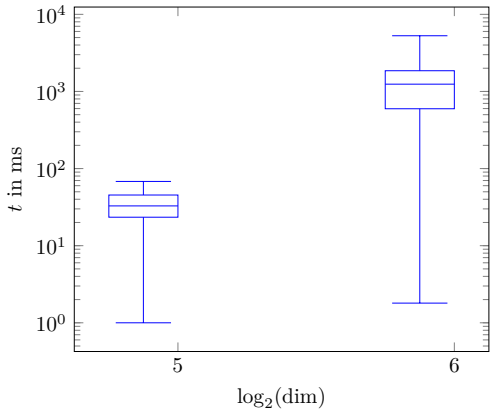
(e) Logarithmic GRF.



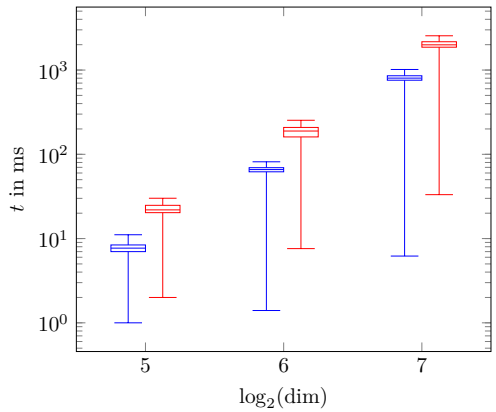
(f) Logistic GRF.



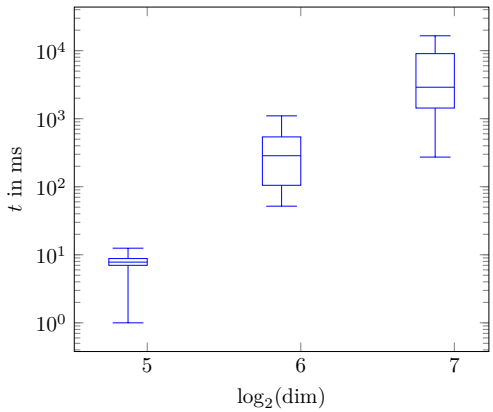
(g) Cauchy density.



(h) Shapes.



(i) Classic images.



(j) Microscopy images.