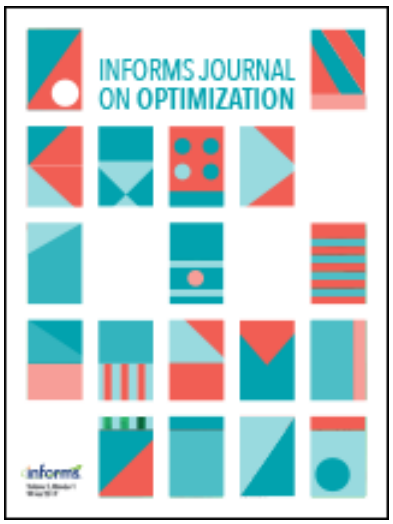


INFORMS Journal on Optimization

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Improved Linear Programs for Discrete Barycenters

Steffen Borgwardt, Stephan Patterson

To cite this article:
Steffen Borgwardt, Stephan Patterson (2020) Improved Linear Programs for Discrete Barycenters. INFORMS Journal on Optimization 2(1):14-33. https://doi.org/10.1287/ijoo.2019.0020

http://pubsonline.informs.org/journal/ijoo

INFORMS JOURNAL ON OPTIMIZATION
Vol. 2, No. 1, Winter 2020, pp. 14–33
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

Improved Linear Programs for Discrete Barycenters

Steffen Borgwardt,[a] **Stephan Patterson**[a]

[a] University of Colorado Denver, Denver, Colorado 80204

Contact: steffen.borgwardt@ucdenver.edu, http://orcid.org/0000-0002-8069-5046 (SB); stephan.patterson@ucdenver.edu (SP)

Received: April 19, 2018
Revised: September 24, 2018; February 8, 2019; April 22, 2019
Accepted: May 6, 2019
Published Online in Articles in Advance: January 27, 2020

https://doi.org/10.1287/ijoo.2019.0020

Abstract. Discrete barycenters are the optimal solutions to mass transport problems for a set of discrete measures. Such transport problems arise in many applications of operations research and statistics. The best-known algorithms for exact barycenters are based on linear programming, but these programs scale exponentially in the number of measures, making them prohibitive for practical purposes. In this paper, we improve on these algorithms. First, by using the optimality conditions to restrict the search space, we provide a reduced linear program that contains dramatically fewer variables compared with previous formulations. Second, we recall a proof from the literature that lends itself to a linear program that has not been considered for computations. We show that this second formulation is the best model for data in general position. Third, we combine the two programs into a single hybrid model that retains the best properties of both formulations for partially structured data. We study these models by analyzing their scaling in size, by showing the hardness of the required preprocessing, and through computational experiments. In doing so, we show that each of the improved linear programs becomes the best model for different types of data.

Funding: The authors gratefully acknowledge support from the Collaboration Grant for Mathematicians "Polyhedral Theory in Data Analytics" from the Simons Foundation [Grant 524210].

Keywords: discrete barycenter • optimal transport • linear programming

1. Introduction

Applications for optimization of mass transport for multiple marginals arise in a variety of fields: probabilistic Fréchet means in statistics (Munch et al. 2015, Zemel and Panaretos 2018), team matching in game theory (Carlier and Ekeland 2010, Pass 2014, Carlier et al. 2015), option prices and price equilibria in economics (Chiaporri et al. 2010, Beiglböck et al. 2013), and electron correlations in matter physics (Cotar et al. 2013), to name a few. The variety of applications has led to considerable activity on the topic; a search for "optimal mass transport problems for several marginals" on GoogleScholar returns about 19,600 hits, including a seminal book by Villani (2009) that has over 2,800 citations at the time of this writing. A search for "weighted Wasserstein barycenters" returns about 940 results.

A particularly noteworthy application is the design of deformable templates in manufacturing and image processing (Jain et al. 1998, Trouvé and Younes 2005, Cuturi and Doucet 2014). An intuitive variant is in metal shaping: a number of sheets of metal have to be pressed (deformed) into multiple, different shapes (deformations). Each shape is modeled as a continuous probability distribution by comparing it to a flat sheet of metal. Physically, a *mean deformation* is a best shape for the initial sheet of metal, that is, a shape that requires minimal energy to mold into all possible deformations. The mean deformation itself is represented as another continuous probability distribution, and the search for it can be formulated as an optimization problem to minimize the total energy cost required to obtain all desired deformations. Formal definitions of the deformations, their means, and this energy cost can be found in Boissard et al. (2015).

1.1. Wasserstein Barycenters and the Discrete Barycenter Problem

The *weighted Wasserstein barycenters* are optimal solutions to these problems. The distance between two probability measures μ and ν supported on $\mathbb{R}^d$ is calculated using the square of the quadratic Wasserstein distance

$$W_2(\mu,\nu)^2 = \inf\left\{\int_{\mathbb{R}^d\times\mathbb{R}^d} \|x-y\|^2 d\gamma(x,y), \gamma \in \Pi(\mu,\nu)\right\},$$

where $\Pi(\mu,\nu)$ denotes the set of all measures on $\mathbb{R}^d \times \mathbb{R}^d$ with μ,ν as marginals. The set $\Pi(\mu,\nu)$ represents the set of transport plans between μ and ν (see Villani 2009 and Agueh and Carlier 2011 for more details). Then,

given probability measures $P_1, \ldots, P_n$ on $\mathbb{R}^d$ and a strictly positive weight vector $\lambda \in \mathbb{R}^n_+$ with $\sum_{i=1}^n \lambda_i = 1$, a Wasserstein barycenter is a probability measure $\bar{P}$ on $\mathbb{R}^d$ satisfying

$$\varphi(\bar{P}) := \sum_{i=1}^{n} \lambda_i W_2\left(\bar{P}, P_i\right)^2 = \inf_{P \in \mathcal{P}^2(\mathbb{R}^d)} \sum_{i=1}^{n} \lambda_i W_2(P, P_i)^2, \tag{1}$$

where $\mathcal{P}^2(\mathbb{R}^d)$ is the set of all probability measures on $\mathbb{R}^d$ with finite second moments. Informally, a barycenter $\bar{P}$ is a measure such that the total transport from $\bar{P}$ to all P_i with respect to the quadratic Wasserstein distance is minimal.

Existence and uniqueness of barycenters for continuous probability measures is established in the study of the problem by Agueh and Carlier (2011), whose work is a foundation for much of the recent literature on barycenters. However, direct computation of barycenters for continuous probability measures has proven intractable outside of special cases (Cuturi and Doucet 2014, Cuturi and Peyré 2016), in part because an evaluation of the Wasserstein distance is computationally challenging in its own right. The computational effort of optimization on this distance creates a need for good approximate solutions.

This is one reason for the interest in Wasserstein barycenters where the measures P_i are supported on a *finite* set of points. We call such measures *discrete*. This setting arises through a discretization of the space of interest, as in Carlier et al. (2015), which is one of the most powerful tools for heuristics to find an approximation of a barycenter. Additionally, many applications in operations research have naturally discrete support sets. In facility location problems, the support corresponds to transport destination locations; see the firehouse location example presented in Anderes et al. (2016). The computation of a barycenter for measures with finite support is also used in distribution clustering (Ye et al. 2017).

We formally state the problem of computing a *discrete barycenter*, that is, a barycenter for a given set of *discrete probability measures* $P_1, \ldots, P_n$. We denote the finite support set of P_i as $\text{supp}(P_i) = \{\mathbf{x}_{ij} | j = 1, \ldots, |P_i|\}$, where the size $|P_i|$ is the number of support points of measure P_i. Each $\mathbf{x}_{ij} \in \text{supp}(P_i)$ has a corresponding mass $d_{ij} > 0$, and $\sum_{j=1}^{|P_i|} d_{ij} = 1$ for each P_i. In this paper, we are looking to improve exact algorithms to solve the following:

Discrete Barycenter Problem
Input: Discrete probability measures $P_1, \ldots, P_n$, weight vector $\lambda \in \mathbb{R}^n_+$
Output: Discrete barycenter $\bar{P}$ for $P_1, \ldots, P_n$ and λ.

A proof of the existence of optimal solutions to the discrete barycenter problem, as well as the linear programming formulation for producing these barycenters that is the starting point for our study, is established in Anderes et al. (2016).

1.2. Set S of Possible Barycenter Support Points

Barycenters satisfy a property that is crucial for many applications: there exists an optimal transport to the measures that is *non-mass-splitting* (Agueh and Carlier 2011, Anderes et al. 2016); that is, the mass of each barycenter support point is transported only to a single support point in each measure. This observation can be used to prove that a discrete barycenter is a discrete measure itself, supported on a subset of the set

$$S = \left\{ \sum_{i=1}^{n} \lambda_i \mathbf{x}_{ij} \;:\; \mathbf{x}_{ij} \in \text{supp}(P_i) \text{ for any } j = 1, \ldots, |P_i| \right\} := \left\{ \mathbf{x}_1, \ldots, \mathbf{x}_{|S|} \right\}.$$

This is the set of all convex combinations of support points, one from each measure P_i, given by the fixed λ_i (Anderes et al. 2016). We call its elements $\mathbf{x}_k$ the *weighted means*. (A single-indexed $\mathbf{x}_k$ always refers to an element in S, in contrast to the double-indexed $\mathbf{x}_{ij}$ for support points of the original measures.) The size of S is bounded by the product of the sizes of the input measures and thus may grow exponentially in the number of input measures. The linear program in Anderes et al. (2016) for the computation of discrete barycenters is based on finding the optimal mass on each "possible support point" in S. Thus, the exponential scaling of S can be an extreme challenge for practical computations.

The actual size of S depends greatly on the underlying data. In this paper, we consider different types of data. We say the measures $P_1, \ldots, P_n$ are *in general position* if different combinations of $\mathbf{x}_{ij} \in \text{supp}(P_i)$ always induce different weighted means $\mathbf{x}_k$; this is the worst-case scenario producing an S of largest possible size: $|S| = \prod_{i=1}^{n} |P_i|$. Additionally, we consider *structured data*: the support sets of the P_i have known properties.

Structured data has two beneficial computational properties: significant repetition in the weighted means creates a smaller set S, and the a priori knowledge allows for easier construction of the set. In particular, we consider a *highly structured* case in which all measures are supported on the same regular d-dimensional grid. Finally, the *partially structured* case refers to measures with support sets that can be partitioned into a part in general position and a part that is structured. Specifically, we study a situation in which the support of each measure is split into a part contained in a regular grid and a part outside of the grid.

1.3. Linear Programming for Discrete Barycenters

In contrast to the worst-case, exponentially sized possible support set, there always exists a barycenter with provably *sparse support*. More precisely, there is a barycenter $\bar{P}$ with

$$|\mathrm{supp}(\bar{P})| \leq \left(\sum_{i=1}^{n} |P_i|\right) - n + 1. \tag{2}$$

This is a tiny fraction of the number of possible support points in S, whose size is bounded by the product, rather than the sum, of the sizes of the original measures (Anderes et al. 2016). Such extreme sparsity is computationally promising, although no strategy to select the optimal support points from the large set S is known. The sparsity can form the basis of an approximation scheme in two primary ways: by selecting a relatively large, representative set of possible support points or by choosing a small initial set and updating the possibilities after a fixed number of iterations (Ye et al. 2017). Another approach is a strongly polynomial two-approximation algorithm based on the restriction of the set of support points for an approximate barycenter to the union of supports of the measures $P_1, \ldots, P_n$ (Borgwardt 2017). In a different, heuristic style of approximation, algorithms based on projection and utilizing an entropic regularization term produce solutions with qualitative smoothing and full support (Cuturi and Doucet 2014, Benamou et al. 2015). In this paper, we focus on exact barycenters, thus maintaining sparsely supported solutions. Our goal is to devise linear programming formulations with fewer variables and constraints. The formulations presented are also useful in LP-based approximation schemes.

When representing a barycenter $\bar{P}$, we use values z_k, $k = 1, \ldots, |S|$, to denote the mass on support point $\mathbf{x}_k \in S$. Further, the values y_{ijk} denote mass transported from $\mathbf{x}_k \in S$ to $\mathbf{x}_{ij} \in \mathrm{supp}(P_i)$ (for all $i = 1, \ldots, n$ and $j = 1, \ldots, |P_i|$). With this notation, the transportation cost (1) in the discrete setting can be written as:

$$\varphi(\bar{P}) := \inf_{P \in \mathcal{P}^2(\mathbb{R}^d)} \sum_{i=1}^{n} \lambda_i W_2(P, P_i)^2 = \min \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{|P_i|} \sum_{k=1}^{|S|} \|\mathbf{x}_k - \mathbf{x}_{ij}\|^2 y_{ijk}. \tag{3}$$

Note that (3) is a linear objective function, as the $\mathbf{x}_k$ and $\mathbf{x}_{ij}$ are part of the input.

It is open whether the discrete barycenter problem can be solved in polynomial time. The best-known algorithms for exact barycenters are based on linear programming (Carlier et al. 2015, Anderes et al. 2016, Ye et al. 2017, Srivastava et al. 2018). The program is established as follows: use variables z_k to measure the (unknown) masses of a barycenter supported on the elements $\mathbf{x}_k \in S$; thus the variables z_k themselves define the barycenter. The mass z_k is transported to each measure P_i, the amount of which is indicated by the variables y_{ijk}. This yields capacity constraints $\sum_{j=1}^{|P_i|} y_{ijk} = z_k$ (for all $i = 1, \ldots, n$ and $k = 1, \ldots, |S|$), as the mass transport to each P_i cannot exceed the available barycenter mass. Further, each support point $\mathbf{x}_{ij}$ in each measure P_i receives exactly its mass d_{ij} from the barycenter support points, which can be stated as $\sum_{k=1}^{|S|} y_{ijk} = d_{ij}$ (for all $i = 1, \ldots, n$ and $j = 1, \ldots, |P_i|$).

Thus, a linear program for the computation of a barycenter may be formulated as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{|P_i|} \sum_{k=1}^{|S|} \|\mathbf{x}_k - \mathbf{x}_{ij}\|^2 y_{ijk} \\
\text{s.t.} \quad & \sum_{j=1}^{|P_i|} y_{ijk} = z_k, \qquad \forall i = 1, \ldots, n, \ \forall k = 1, \ldots, |S|, \\
& \sum_{k=1}^{|S|} y_{ijk} = d_{ij}, \qquad \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, |P_i|, \\
& y_{ijk} \geq 0, \qquad \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, |P_i|, \ \forall k = 1, \ldots, |S|.
\end{aligned} \tag{original}$$

Because the z_k variables represent a barycenter, which itself is a probability measure, we do also require $z_k \geq 0$ and $\sum_{k=1}^{|S|} z_k = 1$; however, these properties are naturally enforced by the capacity constraints and the fact that $\sum_{j=1}^{|P_i|} d_{ij} = 1$. Any optimal vertex of the LP has sparse support, that is, it satisfies (2). See Carlier et al. (2015) and Anderes et al. (2016) for more details. We summarize this information in the following proposition.

Proposition 1. *The discrete barycenter problem can be solved using LP (original). Any optimal vertex corresponds to a sparse barycenter.*

One of the important observations about LP (original) is that its size may scale exponentially in the number n of measures (Borgwardt 2017). For simplicity in denoting the worst-case scenario, assume $|P_i| = p_{\max}$ for all $i = 1, \ldots, n$. For measures $P_1, \ldots, P_n$ with support points in general position, $|S| = \prod_{i=1}^{n} |P_i| = (p_{\max})^n$. Then LP (original) has $n(p_{\max})^{n+1} + (p_{\max})^n$ variables and $n(p_{\max})^n + np_{\max}$ equality constraints.

Most applications arise in dimension two or three, but the dimension does not actually appear as a factor in the exponential scaling of LP (original). Even for rather small input sizes, computations based on LP (original) already are challenging: the computation of a barycenter for the firehouse location problem in Anderes et al. (2016), that is, for $n = 8$ measures consisting of the same $p_{\max} = 9$ support points (a structured data set), already took several minutes on a standard laptop (2.3 GHz Intel Core i7 MacBook Pro).

1.4. Contributions and Outline

The poor scaling of LP (original) is the motivation for our research. We devise new, smaller LP formulations that allow exact computations for larger instances of the discrete barycenter problem. In this paper, we improve on LP (original), while retaining all optimal solutions. By using the optimality conditions of discrete barycenters in a better way, we present several ways to reduce the model size.

In Section 2, we formally devise these improvements. First, we explain why only a strict subset of the variables y_{ijk} is required for a better formulation LP (reduced). We show that LP (reduced) always is an improvement on LP (original): it has strictly fewer variables and the nonzero entries in the constraint matrix of LP (reduced) are a strict subset of the nonzero entries in the corresponding columns of the constraint matrix of LP (original).

Second, we turn to an alternative LP (general) that finds a discrete barycenter. It was used in Anderes et al. (2016) and Miller (2016) to show the existence of a sparse barycenter for all discrete barycenter problems. Because of an inherent, unavoidably exponential scaling that is independent of the underlying data, it has not been previously considered for computational purposes. We use this linear program to reveal that for two input measures, the discrete barycenter problem is a classical transportation problem of the type introduced by Ford and Fulkerson (1956).

Third, we combine the two above LPs in a hybrid model, LP (hybrid). The key idea is that the choice of model can be split into independent decisions for each combination of input support points. For partially structured data, a mix of the strategies of LP (reduced) and LP (general) through LP (hybrid) retains the beneficial properties of both formulations.

Section 3 is dedicated to an analysis of the sizes of the new models. In Section 3.1, we discuss data in general position. We show that both LP (reduced) and LP (general) are dramatic improvements over LP (original), and that LP (general) is the smallest model. Further, we show that a best implementation of LP (hybrid) becomes identical to LP (general).

In Section 3.2, we discuss data supported in a d-dimensional regular grid. In this highly structured setting, the size of S becomes polynomial in n for fixed dimension d. This implies that LP (original) scales polynomially (and recall that LP (reduced) and LP (hybrid) provide further improvements), in strong contrast to the exponential scaling of LP (general) for all data. We provide a formal proof that the discrete barycenter problem (for evenly weighted measures) can be solved in strongly polynomial time in this setting.

Section 4 is a study of the preprocessing required to set up the various LP formulations. In Section 4.1, we prove that the construction of all the LPs is hard (exponential unless $P = NP$), even if the resultant LPs are of polynomial size. More precisely, we show that a decision version of one small step of the necessary preprocessing is already NP-hard. This step may even be repeated exponentially many times for LPs (reduced) and (hybrid).

In Section 4.2, we discuss how expert knowledge that data are supported on a regular grid can help. Under the additional assumption that the measures are evenly weighted, we devise a simple and efficient preprocessing routine to achieve a significant improvement in setup time for LP (original). This routine also

avoids the inefficient preprocessing required for an exact setup of LPs (reduced) and LP (general), at the cost of a minor increase in linear program size.

Section 5 contains our computational experiments. We use three representative types of data: a geospatial data set in general position, the well-known MNIST digits data set (cf. LeCu et al. 1998), and a tailored data set that combines the properties of these two sets.

In Section 5.1, we briefly exhibit the computational advantages of LP (general) over LP (original) for data in general position. We use crime data for Denver County, publicly available as part of the Denver Open Data Catalog (www.denvergov.org/opendata), to construct such a data set. Specifically, we use the geographical locations of incidents in different months and years to represent a set of crime patterns. A barycenter for these crime patterns is readily interpreted as a set of locations for which police presence could lead to a fast crime response.

Section 5.2 shows our computational experiments for grid-structured data. We use the MNIST data set of handwritten digits (http://yann.lecun.com/exdb/mnist/), which is widely used for benchmarking in machine learning (see LeCu et al. 1998 for more information). Each digit is supported on a subset of a 16×16 grid. A set of barycenters, one computed for the measures representing each digit $0, 1, \dots, 9$, allows the classification of a new measure as one of the digits. It suffices to evaluate the cost of transport from each barycenter to the new measure and pick the smallest value.

In our experiments, we consider various settings: computations with and without expert knowledge (that the data are supported on a grid) and the impact of preprocessing on model sizes and computation times in these settings. The fastest results are achieved when using the preprocessing routines from Section 4.2. A combination of fast preprocessing and LP sizes that are just slightly larger than exact formulations of LP (reduced) or LP (hybrid) leads to a best practical performance.

In Section 5.3, we highlight the advantages of LP (hybrid). We construct a data set similar to the MNIST digits data: the letter "i" is recorded in a combination of two grids of different coarseness; a finer grid is used to record the position and detail of the i-dot, because most of the differences between images lie in the dot. Here, LP (hybrid) greatly outperforms the others because of the ability to adapt the representation of S to the different parts of the data. Treating the i-dot as data in general position and the rest of the letter as grid-structured gives the smallest model and fastest computation times.

We finish with some concluding remarks and open questions in Section 6.

2. Improved Linear Programs

In the following, we describe three ways to improve on the formulation of LP (original). The first one is a strict improvement; the second one is the smallest model for data in general position; and the third one is a hybrid of the former two.

2.1. LP (reduced): Optimality Conditions for y-Variables

For an initial reduction in size, we note that variables can be dropped from LP (original) while keeping all optimal solutions in the feasible set. Because of the non-mass-splitting property, we know that any $\mathbf{x}_k = \sum_{i=1}^{n} \lambda_i \mathbf{x}_{ij} \in S$ is the optimal support point for mass which is to be transported to all of the $\mathbf{x}_{ij}$ from which it is constructed. This implies that, in an optimal solution, $\mathbf{x}_k$ never transports to any $\mathbf{x}_{ij}$ not in a weighted mean calculation producing $\mathbf{x}_k$. Equivalently, in an optimal solution, $y_{ijk} = 0$ for all such pairs $(\mathbf{x}_k, \mathbf{x}_{ij})$, and we can therefore eliminate those y_{ijk} from the formulation.

Additionally, we note that the capacity constraints in LP (original),

$$\sum_{j=1}^{|P_i|} y_{ijk} = z_k, \quad \forall i = 1, \dots, n, \ \ \forall k = 1, \dots, |S|,$$

could be rewritten in a way that eliminates the variables z_k. Specifically, for each support point $\mathbf{x}_k$, the transport to each P_i has to be equal:

$$\sum_{j=1}^{|P_1|} y_{1jk} = \sum_{j=1}^{|P_2|} y_{2jk} = \cdots = \sum_{j=1}^{|P_n|} y_{njk}.$$

However, we do not implement this change. It would remove $|S|$ variables and constraints, only a minor benefit to the program size. Meanwhile, the dramatically increased number of nonzero entries in the constraint matrix has an overall negative effect on computations.

Instead, we develop a reduced formulation based solely on the removal of the extraneous y-variables. We first introduce some notation. Let S_{ij} be the set of indices k for which $\mathbf{x}_k \in S$ can be computed as a weighted mean of a set of support points that includes $\mathbf{x}_{ij}$. Formally,

$$S_{ij} = \left\{ k \ : \ \mathbf{x}_k = \lambda_i \mathbf{x}_{ij} + \sum_{l=1, l\neq i}^{n} \lambda_l \mathbf{x}_{lj'} \text{ for some } \mathbf{x}_{lj'} \in \operatorname{supp}(P_l) \right\}.$$

Conversely, let S_k be the set of index tuples (i,j) of support points $\mathbf{x}_{ij}$, which contribute to a computation of $\mathbf{x}_k$, that is,

$$S_k = \left\{ (i,j) \ : \ \mathbf{x}_k = \lambda_i \mathbf{x}_{ij} + \sum_{l=1, l\neq i}^{n} \lambda_l \mathbf{x}_{lj'} \text{ for some } \mathbf{x}_{lj'} \in \operatorname{supp}(P_l) \right\}.$$

With this notation, LP (original) can be improved to a smaller formulation as follows.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{|P_i|} \sum_{k:k\in S_{ij}} \|\mathbf{x}_k - \mathbf{x}_{ij}\|^2 y_{ijk} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in S_k} y_{ijk} = z_k, \quad \forall i = 1,\dots,n, \ \forall k = 1,\dots,|S|, \\ & \sum_{k:k\in S_{ij}} y_{ijk} = d_{ij}, \quad \forall i = 1,\dots,n, \ \forall j = 1,\dots,|P_i|, \\ & y_{ijk} \geq 0, \quad \forall i = 1,\dots,n, \ \forall j = 1,\dots,|P_i|, \ \forall k \in S_{ij}. \end{aligned} \tag{reduced}$$

Because the optimal vertices of LPs (original) and (reduced) are in one-to-one correspondence, we can solve either of them to the same result. We obtain the following theorem.

Theorem 1. *The discrete barycenter problem can be solved using LP (reduced). Any optimal vertex corresponds to a sparse barycenter.*

Note that at least one pair $(\mathbf{x}_k, \mathbf{x}_{ij})$, such that $\mathbf{x}_{ij}$ is not part of any weighted means construction of $\mathbf{x}_k$, exists for any dimension $\mathbb{R}^d$, as long as $|P_i| \geq 2$ for at least one $i = 1,\dots,n$. In other words, for nontrivial input there is an $(i,j) \notin S_k$ for some k. Thus LP (reduced) always provides a strict reduction in the number of variables and the number of nonzero entries in the constraint matrix.

Lemma 1. *For a set of measures $P_1,\dots,P_n$, at least one with $|P_i| \geq 2$, LP (reduced) has strictly fewer variables than LP (original). Further, the nonzero entries of the constraint matrix for LP (reduced) are a strict subset of the nonzero entries in the corresponding columns of the constraint matrix of LP (original).*

In Section 3, we turn to the dramatic reduction in size from LP (original) to LP (reduced)—often several orders of magnitude—in more detail.

2.2. LP (general): Fixed Transport

An alternative linear program for the discrete barycenter problem was used in Anderes et al. (2016) and Miller (2016) to prove the existence of a sparse barycenter, but it was not considered for computational purposes. However, as we will see in Sections 3.1 and 5.1, this is the best approach for data in general position.

The key idea is to treat each combination of original support points that gives a weighted mean separately, even if some combinations produce the same weighted mean $\mathbf{x}_k$. Applying this idea to LP (reduced), a variable z_k is used for each *combination* of original support points. Then $|S_k| = n$, as S_k contains precisely one pair (i,j) for each $i \leq n$. When mass is associated with variable z_k, it is also fully associated to all of the corresponding y_{ijk} with $(i,j) \in S_k$. This implies that the variables y_{ijk} can be eliminated through $y_{ijk} = z_k$ for all $(i,j) \in S_k$. Informally, assigning mass to z_k gives rise to a *fixed transport* to the measures.

We now develop an LP formulation, in our own notation, based on this idea. First, we define the set of all combinations of original support points, one from each measure, as

$$S^* = \left\{ (\mathbf{x}_{1j},\dots,\mathbf{x}_{nj}) \ : \ \mathbf{x}_{ij} \in \operatorname{supp}(P_i) \text{ for any } j = 1,\dots,|P_i| \right\} := \left\{ s^*_1,\dots,s^*_{|S^*|} \right\}.$$

There is an intimate relation of S^* and S: Each tuple $s^*_h = (\mathbf{x}_{1j},\dots,\mathbf{x}_{nj})$, $h = 1,\dots,|S^*|$, corresponds to a set of original support points $\mathbf{x}_{ij}$, $i = 1,\dots,n$, and their weighted mean $\mathbf{x}_k = \sum_{i=1}^{n} \lambda_i \mathbf{x}_{ij} \in S$. Generally, it is possible that

multiple $s_h^* \in S^*$ are associated with the same $\mathbf{x}_k \in S$. In turn, each $\mathbf{x}_k \in S$ is associated with at least one $s_h^* \in S^*$. We immediately obtain $|S^*| \geq |S|$. For data in general position, $|S^*| = |S|$ and there is a bijection between the tuples s_h^* and weighted means $\mathbf{x}_k$.

Next, we introduce a variable w_h for each s_h^*, $h = 1, \ldots, |S^*|$, representing mass associated with that combination. The corresponding cost c_h of transporting a unit of mass is

$$c_h = \sum_{i=1}^{n} \lambda_i \|\mathbf{x}_k - \mathbf{x}_{ij}\|^2.$$

Finally, we define sets S_{ij}^* similarly to the sets S_{ij}. S_{ij}^* is the set of indices h in $1, \ldots, S^*$ such that the i^{th} component of s_h^* is $\mathbf{x}_{ij}$. Formally,

$$S_{ij}^* = \{h \ : \ s_h^* = (\ldots, \mathbf{x}_{ij}, \ldots)\}.$$

We now have all the ingredients for the LP. Because of the fixed transport, the barycenter capacity constraints in LPs (original) and (reduced) are naturally enforced and can be eliminated. All variables y_{ijk} are also eliminated, and so we obtain the LP:

$$\begin{aligned} \min \quad & \sum_{h=1}^{|S^*|} c_h w_h \\ \text{s.t.} \quad & \sum_{h:h\in S_{ij}^*} w_h = d_{ij}, \quad \forall i = 1, \ldots, n, \forall j = 1, \ldots, |P_i|, \\ & w_h \geq 0, \quad \forall h = 1, \ldots, |S^*|. \end{aligned} \tag{general}$$

Barycenter masses z_k and an optimal transport y_{ijk} can readily be reconstructed from the w_h, and the optimal vertices of LPs (original) and (general) again are in one-to-one correspondence.

Theorem 2. *The discrete barycenter problem can be solved using LP (general). Any optimal vertex corresponds to a sparse barycenter.*

For a set of two discrete measures, the computation of a discrete barycenter can be modeled as a classical transportation problem; this is explained in Miller (2016) through a transformation of LP (original). We demonstrate an easier way to prove this claim by reindexing LP (general) as follows:

Denote the support elements as $\mathbf{x}_k \in \text{supp}(P_1)$ with mass d_k and $\mathbf{x}_l \in \text{supp}(P_2)$ with mass d_l. Further, represent the weights as $\lambda_1 = \lambda$ and $\lambda_2 = 1 - \lambda$. For a given $\mathbf{x}_k$ and $\mathbf{x}_l$, an optimal barycenter support point is $\mathbf{x} = \lambda\mathbf{x}_k + (1-\lambda)\mathbf{x}_l$ and the corresponding cost of transport is

$$c_{kl} = \lambda\|\mathbf{x} - \mathbf{x}_k\|^2 + (1-\lambda)\|\mathbf{x} - \mathbf{x}_l\|^2 = \big(\lambda(1-\lambda)^2 + \lambda^2(1-\lambda)\big)\|\mathbf{x}_k - \mathbf{x}_l\|^2 = \lambda(1-\lambda)\|\mathbf{x}_k - \mathbf{x}_l\|^2.$$

With w_{kl} denoting the mass associated to the combination of $\mathbf{x}_k$ and $\mathbf{x}_l$, the discrete barycenter problem for $n = 2$ can be written as the following transportation problem.

$$\begin{aligned} \min \quad & \sum_{k=1}^{|P_1|} \sum_{l=1}^{|P_2|} c_{kl} w_{kl} \\ \text{s.t.} \quad & \sum_{l=1}^{|P_2|} w_{kl} = d_k, \quad \forall k = 1, \ldots, |P_1|, \\ & \sum_{k=1}^{|P_1|} w_{kl} = d_l, \quad \forall l = 1, \ldots, |P_2|, \\ & w_{kl} \geq 0, \quad \forall k = 1, \ldots, |P_1|, \ \forall l = 1, \ldots, |P_2|. \end{aligned} \tag{transportation}$$

It is well known that transportation problems can be solved in strongly polynomial time: transportation problems have totally unimodular constraint matrices and are a special case of minimum-cost flow problems (Tardos 1986, Orlin 1993). Various specialized algorithms are available for addressing transportation problems (Kleinschmidt and Schannath 1995, Tokuyama and Nakano 1995, Brenner 2008) (see Matsui and Scheifele 2016 for a survey). The best running time for our setting is due to Kleinschmidt and Schannath (1995), with the corresponding bound stated in Theorem 3 in our notation.

Theorem 3. *Consider the discrete barycenter problem for $n = 2$ with $p_1 = |P_1|$, $p_2 = |P_2|$, and w.l.o.g. $p_1 \geq p_2$. Then the problem can be solved in strongly polynomial time*

$$O(p_1 \log p_1 \ (p_1 p_2 + p_2 \log p_2)).$$

2.3. LP (hybrid): A Hybrid Approach to Variable Introduction

The strategies of variable introduction presented in LP (reduced) and LP (general) are not mutually exclusive. For each tuple s_h^* individually, we can decide whether to use a representation with y,z-variables as in LP (reduced) or a representation with w-variables as in LP (general). To this end, we partition the set $\{1,\dots,|S^*|\}$ into two index sets $(S^*)^y$ and $(S^*)^w$ to indicate for which s_h^* we use which representation.

The original support points $\mathbf{x}_{ij}$ can then receive mass in two ways: through a transport denoted by some y_{ijk} and through a fixed transport of a mass w_h. First, we define $(S^*)_{ij}^w$ to be the set of indices h such that the i^{th} component of s_h^* is $\mathbf{x}_{ij}$. Then $(S^*)_{ij}^w \subset (S^*)^w$ precisely corresponds to those combinations s_h^* which imply a fixed transport to $\mathbf{x}_{ij}$. For a formal definition, we only have to restrict S_{ij}^* to indices $h \in (S^*)^w$, that is,

$$(S^*)_{ij}^w = \{h \ : \ s_h^* = (\dots,\mathbf{x}_{ij},\dots), h \in (S^*)^w\}.$$

For a proper indexing of the transport corresponding to $(S^*)^y$, we need index sets that mirror S_{ij} and S_k as defined in Section 2.1, but restricted to $(S^*)^y$. S_{ij}^y contains the indices k of all $\mathbf{x}_k \in S$ produced by the weighted means of tuples in $(S^*)^y$ that contain $\mathbf{x}_{ij}$. Formally,

$$S_{ij}^y = \left\{k \in S \ : \ \mathbf{x}_k = \lambda_i \mathbf{x}_{ij} + \sum_{l=1,l\neq i}^{n} \lambda_l \mathbf{x}_{lj'} \text{ for } s_h^* = (\mathbf{x}_{1j'},\dots,\mathbf{x}_{ij},\dots,\mathbf{x}_{nj'}) \in (S^*)^y\right\}.$$

Note that the index pair (i,j) is fixed in the definition of S_{ij}^y. (The use of j' in the other $\mathbf{x}_{lj'}$ indicates that the corresponding j' do not have to match j.) Further, S_k^y contains those index pairs (i,j) for which $\mathbf{x}_k \in S$ is the weighted mean of a tuple in $(S^*)^y$ that contains $\mathbf{x}_{ij}$. This gives

$$S_k^y = \left\{(i,j) \ : \ \mathbf{x}_k = \lambda_i \mathbf{x}_{ij} + \sum_{l=1,l\neq i}^{n} \lambda_l \mathbf{x}_{lj'} \text{ for some } s_h^* = (\mathbf{x}_{1j'},\dots,\mathbf{x}_{ij},\dots,\mathbf{x}_{nj'}) \in (S^*)^y\right\}.$$

Now, we are ready to state an LP that allows for the split of $\{1,\dots,|S^*|\}$ into index sets $(S^*)^y$ and $(S^*)^w$:

$$\begin{aligned}
\min \quad & \sum_{h:h\in(S^*)^w} c_h w_h + \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{|P_i|} \sum_{k:k\in S_{ij}^y} \|\mathbf{x}_k - \mathbf{x}_{ij}\|^2 y_{ijk} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in S_k^y} y_{ijk} = z_k, && \forall i = 1,\dots,n, \ \forall k \in S_{ij}^y, \\
& \sum_{h:h\in(S^*)_{ij}^w} w_h + \sum_{k:k\in S_{ij}^y} y_{ijk} = d_{ij}, && \forall i = 1,\dots,n, \ \forall j = 1,\dots,|P_i|, \\
& w_h \geq 0, && \forall h \in (S^*)^w, \\
& y_{ijk} \geq 0, && \forall i = 1,\dots,n, \ \forall j = 1,\dots,|P_i|, \ \forall k \in S_{ij}^y.
\end{aligned} \tag{hybrid}$$

Correctness of this model is a direct consequence of Theorems 1 and 2.

Theorem 4. *The discrete barycenter problem can be solved using LP (hybrid). Any optimal vertex corresponds to a sparse barycenter.*

In Section 5.3, we show computations for an example where LP (hybrid) is much smaller than the other "pure" LP formulations.

3. Model Sizes

Next, we study the advantages of LPs (reduced), (general), and (hybrid) over LP (original) in model size, that is, the number of variables and constraints, in two representative settings. We begin with data in general position and then turn to highly structured data, in which support is on a regular d-dimensional grid.

3.1. Data in General Position

For a simple notation, we assume that all P_i have the same number of support points $|P_i| = p_{\max}$. Recall that data in general position occurs when all $(p_{\max})^n$ combinations of support points in the original measures produce $(p_{\max})^n$ different weighted means $\mathbf{x}_k$. Thus $|S| = |S^*| = (p_{\max})^n$.

LP (original) is set up with a variable z_k for each of support point in $\mathbf{x}_k \in S$, as well as variables y_{ijk} that indicate transport from $\mathbf{x}_k$ to each $\mathbf{x}_{ij}$. The result is an LP with $(p_{\max})^n + n(p_{\max})^{n+1}$ variables, $(p_{\max})^n$ of type z_k and $n(p_{\max})^{n+1}$ of type y_{ijk}, and $n(p_{\max})^n + np_{\max}$ constraints; recall the discussion at the end of Section 1.3. All the other formulations significantly improve on these numbers; the results are summarized in Table 1.

First, we turn to LP (reduced). Because of the general position of the data, each $\mathbf{x}_k$ may transport only to exactly those n points $\mathbf{x}_{ij}$ from which it is constructed. Thus, we reduce the number of variables y_{ijk} for each $\mathbf{x}_k$ from $np_{\max}$ to n. The number of variables z_k and the number of constraints remain unaffected. This gives a total of $(1+n)(p_{\max})^n$ variables. A representation of the reduction of the number of variables as a percentage highlights the improvement over LP (original). It is a function dominated by $p_{\max}$:

$$1 - \frac{(1+n)(p_{\max})^n}{(p_{\max})^n + n(p_{\max})^{n+1}} = 1 - \frac{1+n}{1+np_{\max}} = \frac{n(p_{\max}-1)}{1+np_{\max}} \approx \frac{(p_{\max}-1)}{p_{\max}}.$$

For example, for $p_{\max} = 256$ the fraction indicates about a 99.61% reduction.

Moving on to LP (general), LP (general) is a strict improvement over LP (reduced). Informally, all the y-variables from LP (original) and LP (reduced) are eliminated. There is just a single variable w_h for each $\mathbf{x}_k \in S$ with the index h indicating the corresponding tuple in S^*. This further lowers the total number of variables from $(1+n)(p_{\max})^n$ in LP (reduced) to $(p_{\max})^n$ in LP (general).

A similar analysis shows that the reduction in the number of variables from LP (original) to LP (general) is about 99.9% for $n = 4$ and $p_{\max} = 256$. The percentage reduction between the number of variables in LP (reduced) and LP (general) only depends on n and moves arbitrary close to 100% for increasing n; for $n = 4$, it is already 80%.

It is worth noting that, unlike LP (reduced), LP (general) also reduces the number of constraints from $n(p_{\max})^n + np_{\max}$ in LPs (original) and (reduced) to $np_{\max}$: from exponential in n to linear in n. This can be significant for the solution time, especially in cases in which the number of required variables is comparable between different formulations. For data in general position, LP (general) always is the best model, by a significant margin.

Finally, consider LP (hybrid). For each of the $(p_{\max})^n$ different combinations of original support points, which here correspond to $(p_{\max})^n$ different weighted means $\mathbf{x}_k$, we decide whether to introduce w-variables as in LP (general) or y,z-variables as in LP (reduced). Thus, the number of variables and constraints ranges between the values for LP (general) and LP (reduced). As seen in the above, a best decision is to always introduce w-variables, which produces exactly LP (general).

Theorem 5. *For data in general position, choosing $(S^*)^w = S^*$ and $(S^*)^y = \emptyset$ gives a minimal number of variables and constraints in LP (hybrid). Then LP (hybrid) and LP (general) are identical.*

3.2. Grid-Structured Data

LPs (original), (reduced), and (hybrid) have a significant advantage over LP (general) in many practical applications: they are able to take advantage of repetition in the $\mathbf{x}_k$ constructed from different combinations s_h^* of original support points. Recall that LPs (original) and (reduced) have variables y and z both indexed on $k = 1, \dots, |S|$, where $|S|$ is the number of *distinct* weighted means. By contrast, LP (general) introduces variables w indexed on $h = 1, \dots, |S^*|$. When there is heavy repetition of the $\mathbf{x}_k$, $|S^*|$ is much larger than $|S|$.

Here, we discuss the model sizes for (evenly weighted) measures supported on a d-dimensional regular grid G_{org}. This is the most common setting in optimal mass transport problems. The MNIST data set of handwritten

Table 1. Number of Variables and Constraints of the Different LP Models for Data in General Position

LP formulation	Variables	Constraints
(original)	$n(p_{\max})^{n+1} + (p_{\max})^n$	$n(p_{\max})^n + np_{\max}$
(reduced)	$(1+n)(p_{\max})^n$	$n(p_{\max})^n + np_{\max}$
(general)	$(p_{\max})^n$	$np_{\max}$
(hybrid)	$(p_{\max})^n$ to $(1+n)(p_{\max})^n$	$np_{\max}$ to $n(p_{\max})^n + np_{\max}$

digits is a prime example for such data; see LeCu et al. (1998) for more information. The handwritten digits are recorded as a measure supported on a subset of a 16×16 grid. Scaling the masses to sum to one makes this a probability measure; see Figure 1 (left) for an example. The shades of grey indicate the amount of mass at the support points; darker points hold more mass.

We focus our comparison of model sizes on LPs (original) and (general). While both LPs (reduced) and (hybrid) provide further improvements on LP (original), they come at significant cost: expert knowledge on either the input or computationally expensive preprocessing are required. We move this discussion to Section 4.2.

So assume that all measures P_i are supported on a d-dimensional regular grid G_{org} of integer step sizes in each direction, each coordinate going from 1 to K. Because the measures are evenly weighted, that is, $\lambda_i = \frac{1}{n}$ for all $i = 1, \dots, n$, S is contained in a d-dimensional, regular grid G_{full} that is n-times-finer than the original grid in each of the d directions. We get $|S| \leq |G_{\text{full}}| = (nK - n + 1)^d$, as the finer grid only runs between the boundary points. In Figure 1 (right), we depict a barycenter computed for four evenly weighted digits. It is supported sparsely on a regular, 4-times-finer grid; the dimension is 61×61 instead of 16×16. (Note $61 = 4 \cdot 16 - 3$.)

Recall that LP (original) uses a variable z_k for each possible support point $\mathbf{x}_k \in S$, and variables y_{ijk} that allow transport from $\mathbf{x}_k$ to each $\mathbf{x}_{ij}$. The number of variables is

$$(nK - n + 1)^d \left(1 + \sum_{i=1}^{n} |P_i|\right) \leq (nK - n + 1)^d \left(1 + nK^d\right) \in O\left(n^{d+1} K^{2d}\right)$$

and the number of constraints is

$$nK^d + n(nK - n + 1)^d \in O\left(n^{d+1} K^d\right).$$

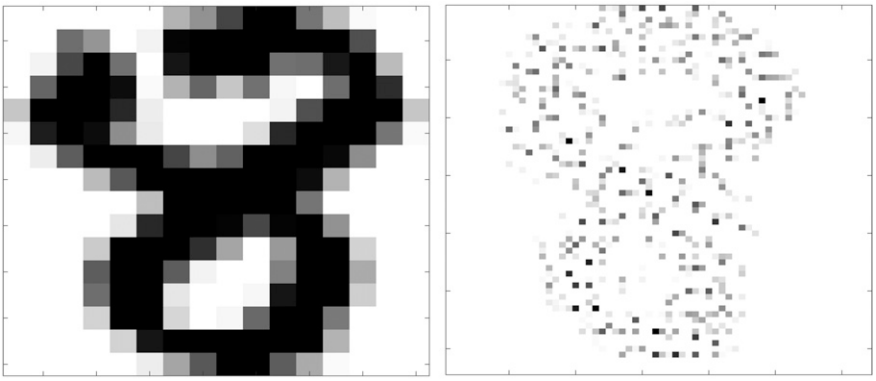
Both the number of variables and the number of constraints are polynomial in n with degree $d + 1$. In fact, it now is not hard to see that the discrete barycenter problem with grid-structured data, even weights λ_i, and input information (coordinates of support points, K) on the grid, can be solved in strongly polynomial number by using G_{full} in place of S in the formulation of LP (original). The proof is similar to the proof of existence of a strongly polynomial 2-approximation for the discrete barycenter problem in Borgwardt (2017).

Theorem 6. *For all rational data and fixed dimension d, the discrete barycenter problem for data supported on a grid G_{org}, $\lambda_i = \frac{1}{n}$ for all $i = 1, \dots, n$, and input on G_{org} itself, can be solved in strongly polynomial time in n and K using LP (original), by using G_{full} in place of S.*

Proof. We have to show that setup and solution of LP (original) using G_{full} is strongly polynomial, that is, it is possible in a polynomial number of arithmetic operations in the number of integers in the input and by performing only computations on numbers that are polynomial in the input.

In the above, we have seen that the problem has a strongly polynomial number of variables and constraints for fixed d. The only numbers in the model that might not be explicitly stated in the input are $\|\mathbf{x}_k - \mathbf{x}_{ij}\|^2$. The $\mathbf{x}_{ij}$ lie in G_{org}, the $\mathbf{x}_k$ in G_{full}; both can be efficiently computed from input on G_{org}. Then each of the strongly polynomially many terms $\|\mathbf{x}_k - \mathbf{x}_{ij}\|^2 = (\mathbf{x}_k - \mathbf{x}_{ij})^T (\mathbf{x}_k - \mathbf{x}_{ij})$ only requires the computation of the vector difference

Figure 1. Barycenter of Grid-Structured Data



Notes. Left, an example of digit 8 from the MNIST digits data set. Right, a barycenter computed for four 8's from the data set. It is supported on a 4-times-finer grid.

Table 2. Number of Variables and Constraints for n Measures Densely Supported on a Grid of Length K

LP formulation	Variables	Constraints
(original)	$(nK-n+1)^d(1+nK^d)$	$nK^d+n(nK-n+1)^d$
(general)	$(K^d)^n$	nK^d

$\mathbf{x}_k - \mathbf{x}_{ij}$ and the scalar product of the resultant vector with itself. Thus, the setup and the size of the LP indeed are strongly polynomial.

Now, recall that linear programs are generally solvable in weakly polynomial time. This indicates that the absolutes of numbers in the input are connected to the number of required arithmetic operations. However, a famous result by Tardos (1986) states that it suffices to only consider numbers in the constraint matrix, not in the objective function or right-hand sides. The constraint matrix for LP (original) consists only of 0's and 1's. This implies that the LP can be solved in strongly polynomial time. □

By contrast, recall LP (general) has up to $(p_{\max})^n$ variables (but only $np_{\max}$ constraints). If the actual support of the measures is a linear fraction of the number K^d of support points in the whole grid (we observe factors between $\tau = \frac{1}{5}$ and $\tau = \frac{1}{3}$ for the MNIST digits data set) then one obtains an exponential growth in $\Theta((\tau K^d)^n)$. In summary, LP (original) is of polynomial size in n for fixed dimension d, while LP (general) is of exponential size in n, even if d is fixed. See Table 2 for the numbers for a worst-case scenario, in which each grid point has nonzero mass.

Both LP (reduced) and (hybrid) scale polynomially in n too: by Lemma 1, LP (reduced) always is a strict improvement over LP (original) in model size. An even smaller model size can be achieved using LP (hybrid) through an optimal choice of $(S^*)^y$ and $(S^*)^w$. However, computationally expensive preprocessing is required to set up both models. In the next section, we discuss the hardness of this preprocessing in general, as well as some efficient, but less powerful preprocessing routines tailored to grid data.

4. Preprocessing

We now analyze the preprocessing that is necessary for each LP from Section 2. First, we prove that the setup of all LPs is hard (and thus computationally expensive), even if the resultant LPs are of polynomial size. In fact, a small step of the setup for LPs (reduced) and (hybrid), which may even have to be repeated exponentially many times, already is hard.

Second, we turn to evenly weighted measures supported on a regular grid (as in Section 3.2). Here, the possible support set S is of polynomial size, in contrast to the exponential size of S^*. Our goal is to reduce the model sizes *without* processing S^*. LP (original) scales polynomially for fixed dimension d in this setting (cf. Table 2). Thus, the preprocessing has to be fast, while still providing a significant improvement. To achieve this, we present preprocessing routines that use *only the grid structure* and simple properties derived from measures $P_1, \dots, P_n$ to improve computations. The positive impact on computation times is highlighted in Section 5.

4.1. Hardness of General Preprocessing

If there is no expert knowledge on structure underlying a set of measures $P_1, \dots, P_n$ (or they are in general position), we first have to process the set S^* in order to set up any of the linear programming formulations. In particular, this is necessary for determining the set S, as well as the sets S_k and S_{ij} required for LPs (reduced) and (hybrid). However, the exponential size of S^* is not the only reason why this preprocessing is computationally expensive. We begin by proving that it is already NP-hard to decide whether a given $\mathbf{x}$ lies in S or not.

Lemma 2. *Let $P_1, \dots, P_n \subset \mathbb{R}^d$ be discrete measures, let $\lambda \in \mathbb{R}^n_+$ be a weight vector with $\sum_{i=1}^n \lambda_i = 1$, and let $\boldsymbol{x} \in \mathbb{R}^d$. Then it is NP-hard to decide whether $\boldsymbol{x} \in S$, even for $d = 1$.*

Proof. For convenience, we call the decision whether $\mathbf{x} \in S$ for $d = 1$ the *possible support problem*. We prove the claim by showing that an efficient decision of the possible support problem would lead to an efficient decision of the subset sum problem, which is known to be NP-complete. A variant of the subset sum problem can be stated as follows: given a set of nonzero integers $p_1, \dots, p_n \in \mathbb{Z}$, is there a subset whose sum is a given $s \in \mathbb{Z}$? (We allow for nonproper subsets, the empty set or the whole set, in this formulation to make the following arguments a bit less technical.)

Consider an instance $\mathcal{I}_s$ of the subset sum problem. We construct an instance $\mathcal{I}_p$ of the possible support problem from it as follows: Let $P_1, \dots, P_n \subset \mathbb{Z}$ be discrete measures, where each P_i consists of the two support points p_i and 0 (both of mass $\frac{1}{2}$, but this is not relevant). Further, let $\lambda_1, \dots, \lambda_n = \frac{1}{n}$ and $\mathbf{x} = \frac{1}{n} \cdot s$. Clearly, this construction is polynomial. It remains to prove that $\mathcal{I}_s$ is a yes-instance if and only if $\mathcal{I}_p$ is a yes-instance.

If $\mathcal{I}_p$ is a yes-instance, then $\mathbf{x} = \frac{1}{n} \cdot s = \sum_{i=1}^{n} \frac{1}{n} \mathbf{x}_{ij}$, where $\mathbf{x}_{ij}$ is either p_i or 0 for all $i \leq n$. Equivalently $s = n \cdot \mathbf{x} = \sum_{i=1}^{n} \mathbf{x}_{ij}$, where $\mathbf{x}_{ij}$ is either p_i or 0 for all $i \leq n$. Thus there exists a subset of the $p_1, \dots, p_n$ adding up to s, so $\mathcal{I}_s$ is a yes-instance, too.

Conversely, let $\mathcal{I}_s$ be a yes-instance, that is, there exists a set of indices $I \subset \{1, \dots, n\}$ such that $s = \sum_{i \in I} p_i$. Then $\mathcal{I}_p$ is a yes-instance, too, as $\mathbf{x} = \frac{1}{n} \cdot s = \sum_{i=1}^{n} \frac{1}{n} \mathbf{x}_{ij}$ for $\mathbf{x}_{ij} = p_i$ for all $i \in I$, and 0 else. This proves the claim. □

Lemma 2 has several implications. First, it transfers to the set S as a whole.

Corollary 1. *Let $P_1, \dots, P_n \subset \mathbb{R}^d$ be discrete measures and let $\lambda \in \mathbb{R}^n_+$ be a weight vector with $\sum_{i=1}^{n} \lambda_i = 1$. Then it is NP-hard to decide whether a given set $S \subset \mathbb{R}^d$ is the set of possible support points.*

This implies that it is (computationally) hard to construct S from the input, *even if it is of polynomial size*. Further, Lemma 2 transfers to the hardness of constructing S_k for a fixed $\mathbf{x}_k \in S$.

Corollary 2. *Let $P_1, \dots, P_n \subset \mathbb{R}^d$ be discrete measures, let $\lambda \in \mathbb{R}^n_+$ be a weight vector with $\sum_{i=1}^{n} \lambda_i = 1$, and let $\mathbf{x}_k \in S$. Then it is NP-hard to decide whether a given set S_k is correct.*

Recall that S_k contains the pairs (i, j) of the $\mathbf{x}_{ij}$ from which a particular $\mathbf{x}_k$ can be constructed. The ability to construct S_k for a given $\mathbf{x}_k$ in polynomial time would be sufficient to decide whether a given S_k is correct in polynomial time, too, a contradiction to Corollary 2, unless $P = NP$.

4.2. Preprocessing for Grid-Structured Data

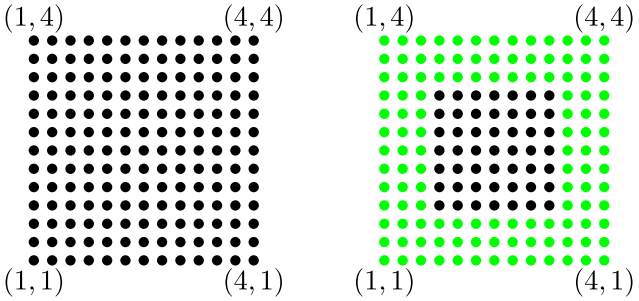
The main benefit of evenly weighted grid-structured data, that is, n measures supported in a regular grid G_{org} and $\lambda_i = \frac{1}{n'}$ is that the set S is contained in an n-times-finer grid G_{full}. This allows the efficient setup and solution of the discrete barycenter problem, by using G_{full} instead of S in LP (original); recall Theorem 6 and see Table 2 for exact model sizes. However, this approach can still be improved significantly. To this end, an exact construction of LPs (reduced) and (hybrid) may be impractical: it requires the processing of the exponential-sized set S^*, and each step of this process is hard by itself (Lemma 2).

In this section, we have a different goal. We devise some efficient preprocessing routines, only using the grid structure and simple information on $P_1, \dots, P_n$, to reduce the model size. But to devise these routines, we first have to understand what happens in an exact preprocessing. For simplicity, we assume that each of the P_i has mass on the whole grid G_{org}. In this case, $S = G_{\text{full}}$.

4.2.1. Preprocessing for LP (reduced). To set up LP (reduced), the sets S_{ij} and S_k are constructed during the processing of S^*. They are used to reduce the number of y-variables for each $\mathbf{x}_k \in S$.

The maximum size of S_k, respectively, the maximum number of y-variables, is nK^d for each $\mathbf{x}_k$. The actual size of S_k depends on the position of $\mathbf{x}_k$ in the grid. An example is depicted in Figure 2, which shows the effect for four measures in $\mathbb{R}^2$ in a regular grid with $K = 4$, resulting in a 13×13 grid. For any $\mathbf{x}_k \in G_{\text{full}}$ with a coordinate in the outer $K - 1$ rows or columns (here, three rows and columns, highlighted), there exist $\mathbf{x}_{ij} \in G_{\text{org}}$ such that no weighted mean involving that $\mathbf{x}_{ij}$ gives the $\mathbf{x}_k$. Therefore, the corresponding variables y_{ijk} are not introduced in the preprocessing. All other points—those in the "center" of the grid G—require all nK^d y-variables.

Figure 2. (Color online) Barycenter of Four Measures Densely Supported in 4×4 Grids Lies in a 13×13 Grid



Note. The outer three rows and columns have points in each original 4×4 grid to which they will not transport.

4.2.2. Preprocessing for LP (hybrid)

Next, we discuss the choice between a representation of a weighted mean through w-variables versus y,z-variables. We choose y,z-variables when there are more than nK^d+1 combinations producing an $\mathbf{x}_k$.

We can calculate the number of combinations for each support point $\mathbf{x}_k$ by counting the number of combinations of integers with a specific sum as follows. Beginning in dimension one, denote $\mathbf{x}_{ij}$ as integers between 1 and K and $s=n\cdot\mathbf{x}_k=\sum_{i=1}^{n}\mathbf{x}_{ij}$. The number of combinations of $\mathbf{x}_{ij}$ that give $\mathbf{x}_k$ is equivalent to the number $F(s,K,n)$ of combinations of n integers between 1 and K that sum up to s. This is a standard counting problem, in which n K-sided dice are tossed and the number of combinations that give sum s are calculated. Thus, $F(s,K,n)$ is given by

$$F(s,K,n)=\sum_{m=0}^{n}(-1)^m\binom{n}{m}\binom{s-mK-1}{n-1}.$$

We extend this formula to $\mathbf{x}_k\in\mathbb{R}^d$ for higher dimension d by considering each coordinate of $\mathbf{x}_k$ individually. Let s_l be the corresponding sum of the lth coordinate of $\mathbf{x}_k$; then the total number of combinations whose weighted mean is $\mathbf{x}_k$ is

$$N_k=\prod_{l=1}^{d}F(s_l,K,n).$$

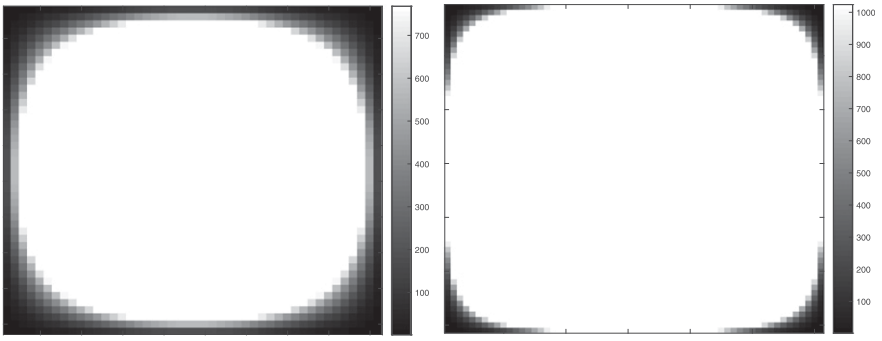
So, for any k such that N_k exceeds nK^d+1, using y,z-variables (rather than introducing individual $w_h, h=1,\dots,N_k$ for all combinations) produces fewer total variables in LP (hybrid). The $\mathbf{x}_k$ for which w-variables are preferable correspond to the "rounded" corners of G_{full} as depicted in Figure 3. The darker a support point, the fewer combinations in S^* correspond to it.

Except for small n, the majority of the grid prefers y,z-variables. Further, the corners where w-variables are preferable are contained in the areas of the grid where the number of y,z-variables are already reduced from nK^d+1 for setup of LP (reduced); see Figure 2. In the computational experiments in Section 5.2.1, we see that LP (reduced) provides a significant improvement over LP (original). On the other hand, the difference between LP (hybrid) and LP (reduced) is small in view of the computationally more expensive preprocessing needed to set up LP (hybrid). In Section 5.3, we provide a different type of data for an example in which the use of LP (hybrid) greatly outperforms the others.

4.2.3. Efficient Preprocessing to Improve on LP (original). The first step toward a good model for grid-structured data are the implementation of LP (original) on G_{full}; recall Theorem 6. We now devise another improvement in the form of a smaller subgrid $G\subseteq G_{\text{full}}$ that still contains S.

For grid-structured data, G_{full} may still contain points that are not in S. This occurs in the digits data set, because mass only exists on part of the 16×16 grid. For efficiency, G_{full} must be generated without checking if the $\mathbf{x}_{ij}$ which produce $\mathbf{x}_k$ have positive mass. Instead, to obtain a smaller subgrid G, we track the largest and smallest values of each coordinate of the dimension for each measure. The weighted means of these extreme coordinate values give sufficient minimum and maximum values for a subgrid G of G_{full} that contains S. We use this set G in LPs (original) and LP (reduced) in the computations for Section 5.2.2.

Figure 3. Set S of Support Points for 3 (Left) and 4 (Right) Measures Densely Supported on a 16×16 Grid



Note. The dark support points indicate where w-variables are preferable over y,z-variables in LP (hybrid).

LP (original) can be implemented directly on G. We have already seen that a significant reduction in model size when working over S is possible through the setup of LP (reduced). This effect also holds on G, but there is an additional challenge: for efficient preprocessing, we have to avoid processing S^*, and, thus, we do not construct the sets S_{ij} and S_k required to set up LP (reduced) exactly. We need a different approach to eliminate extraneous y-variables for G.

Recall the "boundary" effect depicted in Figure 2. The grid points $\mathbf{x}_k$ that have at least one coordinate within the minimum or maximum among all points in G have y-variables that can be eliminated – there exist points $\mathbf{x}_{ij}$ in the original grid G_{org} such that $y_{ijk} = 0$ in all optimal solutions. Because of the grid structure, it is easy to identify some of these $\mathbf{x}_{ij}$: for example, let $\mathbf{x}_k$ have the largest first coordinate. Then it can only be constructed from the $\mathbf{x}_{ij}$ with a largest first coordinate in each measure. For a second-largest first coordinate, only the second-largest or largest coordinates of $\mathbf{x}_{ij}$ are possible, and so on.

A reduction in y-variables based on this principle requires only a few comparisons for each $\mathbf{x}_k$ and guarantees the elimination of many variables: the outer $K - 1$ rows and columns have at least nK possible $\mathbf{x}_{ij}$ to which they do not transport. Thus, the reduction in model size scales with both the width K of the original grid and the number n of measures. In Section 5.2.2, we see that this reduction is significant in practice and produces the fastest total running times in our computations.

5. Computational Experiments

In this section, we compare the performance of the different LPs for different types of data through some computational experiments. For each of the LP (general), LP (reduced), and LP (hybrid) models, there exists data such that they become the smallest model with fastest total running times.

Section 5.1 is dedicated to data in general position. In Section 5.2, we study computations for grid-based data, and use these to highlight the advantages of the preprocessing routines described in Section 4.2. Finally, Section 5.3 is dedicated to an example in which using LP (hybrid) significantly outperforms the other formulations.

All computations were run on a standard laptop (MacBook Pro, 2.9 GHz Intel Core i7, 16 GB of RAM, SSD). Data processing and the setup of the LPs, in particular the preprocessing for grid-structured data from Section 4.2, were implemented in C++ and the LPs were solved using Gurobi 8.0. In all computations, the Gurobi solver parameters are fixed and the presolvers switched off. We report on solution times for a primal Simplex algorithm: all single-thread LP solvers behaved similarly. The presolving done by LP solvers for general linear programs and our tailored preprocessing routines (Section 4.2) are conceptually different and do not compete with each other. In practice, we would use both: first, our tailored preprocessing to obtain an improved model that is passed to the solver, then the internal presolving, and, finally, the actual run of an LP algorithm. However, the impact of presolvers on computation times may vary dramatically between different instances of the same model (sometimes even multiple runs of the same instance). Because of the difficulty of measuring this impact, we follow the standard practice to switch off presolvers for a comparison of computation times.

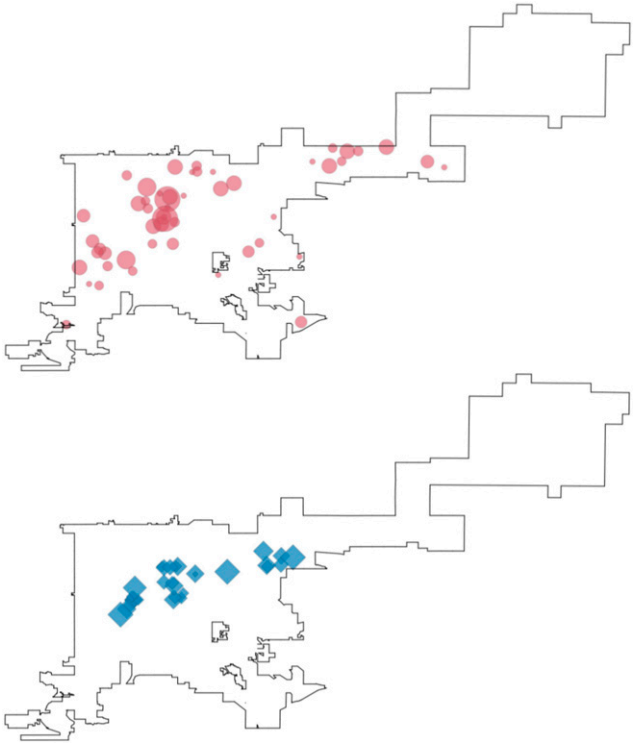
5.1. Data in General Position: Denver Crime

We demonstrate the practical advantage of LP (general) over LP (original) for data in general position through some proof-of-concept computations on crime data for Denver County, which is available as part of the Denver Open Data Catalog (www.denvergov.org/opendata). The data set is constructed from the dates and locations of murders with a year. Each month forms a measure $P_i, i = 1, \ldots, 12$, by weighting each murder evenly during the month. Thus, for months with fewer total incidents, the mass on each location is larger. In doing so, each month represents a *crime pattern*, a set of geographical locations where incidents happen "at the same time." There is no (obvious) underlying structure to the crime patterns, so we obtain a data set in general position.

We are interested in computing locations for police presence such that a fastest average response time to all incidents occurring in all crime patterns is achieved. The police presence itself should be fixed, *not* vary depending on the month. A discrete barycenter for this data set can be readily interpreted (and justified) to indicate locations for police presence: The locations are chosen such that a fast response to incidents in each crime pattern is possible. Conversely, for each incident, there is police presence at a location from which a fast response is possible. Recall that a barycenter computation uses squared Euclidean distances; this leads to a fair treatment of outlier incidents. An aggregate image of all murder locations in 2016 and a corresponding barycenter are displayed in Figure 4. The radii of the shapes in both parts of the figure are relative to their masses.

Based on the theoretical analysis in Section 3.1, we only highlight the advantage of LP (general) over LP (original): while LP (reduced) also would be an improvement over LP (original), it is dominated by the even

Figure 4. (Color online) Murder Locations in Denver County in 2016 (Top) and a Barycenter Indicating Suggested Police Presence Locations (Bottom)



better performance of LP (general). Further, a best implementation of LP (hybrid) produces the same linear program LP (general).

In Table 3, we compare the sizes of and running times for LPs (original) and (general) for five years, 2012 to 2016. The reduction in number of variables and number of constraints is substantial; the linear growth of the number of constraints is readily visible in the small number of rows, and the number of variables is typically about 2% of the number for LP (original). For two years, 2013 and 2015, LP (original) is unable to solve because of memory capacity, so, in particular, the use of LP (general) allows the computation of new solutions. The total running times using LP (general) are typically only a few seconds.

This dramatic improvement is just as expected in view of the discussion in Section 3.1. LP (general) is the smallest model for data in general position, and no sophisticated preprocessing is necessary to set it up. The other models are not competitive. The situation becomes more interesting for grid-based data, which we discuss next, and in more detail.

5.2. Grid-Structured Data: MNIST Digits

For computations of barycenters on the digits data set, we implemented LPs (original), (reduced), (general), and (hybrid) using the set S, then also implemented LPs (original) and (reduced) using the set G. Recall the discussion in Section 4.2; the set S requires the expensive processing of S^*, whereas G does not. While using G generates larger linear programs, it proves to be the much better approach when total times, including

Table 3. Total Running Times Comparing LP (original) and LP (general) for Data in General Position

	LP (original)			LP (general)		
Year	Constraints	Variables	Time in seconds	Constraints	Variables	Time in seconds
2012	1,520,675	4,976,640	47,865.53	35	138,240	1.06
2013	23,887,914	85,598,208	*	42	1,990,656	31.14
2014	663,585	1,880,064	5,824.63	33	55,296	0.43
2015	303,464,505	1,466,744,832	*	57	25,288,704	430.18
2016	26,127,412	115,395,840	223,886.9	52	2,177,280	19.15

Note. We see dramatic improvement using LP (general), including computations that were not possible before (*).

Table 4. Average Number of Constraints and Variables for Four Measures of Each Digit Using Set S

	LP (original)		LP (reduced)		LP (general)		LP (hybrid)		
Digit	Constraints	Variables	Constraints	Variables	Constraints	Variables	Constraints	Variables	$\lvert S\rvert$
0	10,598	1,317,437	10,598	740,116	522	281,656,116	10,130	739,615	2,519
1	3,083	144,200	3,083	82,834	199	6,058,800	2,867	82,624	721
2	10,589	1,072,724	10,589	596,785	421	116,865,990	10,001	596,191	2,542
3	10,999	1,212,100	10,999	679,939	459	167,401,080	10,503	679,442	2,635
4	8,964	801,204	8,964	443,410	372	74,412,360	8,260	442,725	2,148
5	10,999	1,282,464	10,999	716,753	487	206,928,720	10,587	716,328	2,628
6	7,443	617,352	7,443	343,541	347	53,316,900	7,047	343,135	1,774
7	7,944	640,974	7,944	353,378	336	48,014,460	7,432	352,843	1,902
8	10,104	1,193,794	10,104	661,078	496	211,403,136	11,964	1,424,899	2,867
9	8,787	918,280	8,787	500,554	439	131,637,120	8,207	499,940	2,087

Notes. Using LP (original) as a baseline, LP (reduced) is a significant improvement in size. LP (hybrid) is slightly smaller than LP (reduced). LP (general) is not viable.

preprocessing, are considered. Representative sizes and times were formed by running computations for several sets of images for each digit and averaging the results.

5.2.1. Computations on S: Slow Preprocessing, Small LPs. We first examine computations in which we generate S by processing each element of S^* and check for duplication in those elements already produced. This requires exponential effort, but allows us to determine precisely which $\mathbf{x}_k$ can be generated by the support set of the digits under consideration. Overall, this slow preprocessing produces the smallest LPs.

Table 4 shows the average number of constraints and variables for each formulation using this preprocessing strategy. The number of variables in LP (general) is larger, by several orders of magnitude, than for the other models. Its only advantage is the lower number of constraints, but this effect is dominated by the dramatic increase in number of variables. Moving from LP (original) to LP (reduced), there is about a 44% reduction in variables. As expected in light of the discussions in Section 3.2 and 4.2, LP (hybrid) does show a consistent further improvement in model size, but the reduction in variables using LP (hybrid) is relatively minor, typically less than 1%.

Table 5 provides the *total* running times, with subcategories of the *setup* (preprocessing) and *solution* times, for LP (original), LP (reduced) and LP (hybrid), as well as the LP solution times for LP (general) when available. (Recall that that the internal presolving of Gurobi is turned off. The setup refers to our own routines to set up the model and load it to the solver.) The LP solution times are precisely as predicted based on the sizes of the programs: LP (general) is several orders of magnitude slower than the other formulations; the (*) represent scenarios in which the LP solver was unable to complete due to memory constraints. LP (reduced)

Table 5. Average Times for Four Measures of Each Digit Using Set S

	Time in seconds									
	LP (original)			LP (reduced)			LP (hybrid)			LP (general)
Digit	Setup	Solve	Total	Setup	Solve	Total	Setup	Solve	Total	Solve
0	604.25	141.44	**746.23**	688.97	102.94	792.09	745.62	**92.95**	838.77	*
1	4.57	2.49	**7.09**	5.95	1.39	7.36	7.09	**1.20**	8.32	311.27
2	304.10	125.26	429.68	337.77	58.44	396.37	349.81	**41.95**	**391.91**	18,464.70
3	402.63	152.85	555.83	448.20	71.81	**520.18**	480.72	**68.29**	549.23	*
4	141.49	83.20	224.96	159.98	33.88	193.97	184.60	**33.52**	**182.23**	12,224.80
5	467.24	163.34	630.99	535.25	86.74	**622.24**	582.32	**61.11**	643.59	*
6	92.21	45.09	137.50	104.93	24.79	**129.80**	116.76	**17.48**	134.35	2,337.17
7	69.82	53.71	123.71	81.64	**21.41**	**103.13**	91.18	21.82	113.07	6,060.96
8	375.41	114.70	**490.47**	438.58	91.61	530.36	480.66	**51.69**	532.52	*
9	226.72	92.44	319.45	263.57	41.83	**305.52**	289.74	**34.56**	324.42	*

Notes. For all digits, the size reduction in LPs (reduced) and (hybrid) offers significant savings in solution times. Fastest solution and total times are displayed in bold. LP (general) is not competitive; solution times are several orders of magnitude larger. Because of the processing of S^*, the times for LP (original), LP (reduced), and LP (general) are dominated by the setup time. For most digits, LP (reduced) offers the best total time.

Table 6. Average Number of Constraints and Variables for Four Measures of Each Digit Using the Easily Generated, but Larger, Set G Versus Using the Difficult to Generate, but Smaller, Set S

	G					S				
	LP (original)		LP (reduced)			LP (original)		LP (reduced)		
Digit	Constraints	Variables	Constraints	Variables	$\lvert G\rvert$	Constraints	Variables	Constraints	Variables	$\lvert S\rvert$
0	12,478	1,563,247	12,478	1,139,427	2,989	10,598	1,317,437	10,598	740,116	2,519
1	3,371	158,600	3,371	117,684	793	3,083	144,200	3,083	82,834	721
2	13,417	1,371,078	13,417	999,274	3,249	10,589	1,072,724	10,589	596,785	2,542
3	12,903	1,431,060	12,903	1,026,691	3,111	10,999	1,212,100	10,999	679,939	2,635
4	11,840	1,069,391	11,840	776,727	2,867	8,964	801,204	8,964	443,410	2,148
5	11,955	1,399,096	11,955	1,016,077	2,867	10,999	1,282,464	10,99	715,753	2,628
6	8,887	742,980	8,887	541,330	2,135	7,443	617,352	7,443	343,541	1,774
7	10,340	842,837	10,340	605,340	2,501	7,944	640,974	7,944	353,378	1,902
8	11,964	1,424,899	11,964	1,029,485	2,867	10,104	1,193,794	10,104	661,078	2,402
9	10,931	1,154,120	10,931	818,342	2,623	8,787	918,280	8,787	500,554	2,087

Note. The problem sizes on G are approximately 35% larger.

and LP (hybrid) both show improved solution times over LP (original). LP (hybrid) consistently gives the fastest solution times, but the advantage over LP (reduced) is not significant.

Comparing the total running times from Table 5, as expected, the setup times dominate the solution times in contribution to the total. This setup cost increases from LP (original) to LP (reduced), and then further to LP (hybrid). LP (reduced) is typically the best choice for overall time, due to a best tradeoff of decreased solution time to increased setup time. For LP (hybrid), the minor reduction in size and solution times is insufficient to justify the increased setup time.

5.2.2. Computations on G: Fast Preprocessing, Slightly Larger LPs. Next, we show results for the same collections of digits for LP (original) and LP (reduced), using the easy-to-preprocess subgrid G of G_{full} instead of S. Recall the discussion in Section 4.2 regarding the underlying structure which reveals that an implementation of LP (hybrid) for G is not promising.

Because $S \subseteq G$, the linear programs using G are larger; however, as G is significantly easier to generate, the preprocessing effort is greatly reduced. In Table 6, we provide $|G|$ and $|S|$ for our representative digits, along with the corresponding program sizes. There is about a 35% increase in variables for LP (reduced) on G compared with the corresponding formulation on S. We also note that the number of variables for LP (reduced) on G is smaller than for LP (original) on S for all our sample runs.

The advantage of using set G is immediately apparent in Table 7. The setup of both LPs takes only a few seconds and is almost negligible in view of the total running time. Unlike for set S, the setup time for LP

Table 7. Average Setup, Solution, and Total Running Times for Four Measures of Each Digit Using the Easily Generated, but Larger, Set G

	Time in seconds					
	LP (original)			LP (reduced)		
Digit	Setup	Solve	Total	Setup	Solve	Total
0	9.38	233.97	243.64	**7.92**	**192.63**	**200.89**
1	0.42	2.94	3.18	**0.39**	**2.58**	**3.00**
2	5.00	256.18	261.35	**4.08**	**208.43**	**212.79**
3	6.73	282.34	289.25	**5.28**	**141.66**	**147.25**
4	3.77	146.12	149.99	**2.97**	**119.97**	**123.19**
5	7.31	201.88	209.39	**6.10**	**146.32**	**152.73**
6	2.65	55.87	58.51	**2.10**	**45.32**	**47.58**
7	2.71	107.41	110.13	**2.03**	**52.14**	**54.35**
8	7.53	197.01	204.75	**6.41**	**184.54**	**191.28**
9	5.06	172.58	177.79	**4.47**	**127.23**	**131.96**

Notes. Because of the simplicity of constructing the set G, the setup times are consistently low and contribute far less to the total running times than the solution times. LP (reduced) outperforms LP (original) in both setup and solution times. Fastest times are displayed in bold.

Table 8. Average Total Running Times Including Setup for Four Measures of Each Digit Using the Easily Generated Set G versus the Difficult to Generate Set S

	Total running time in seconds			
	LP (original)		LP (reduced)	
Digit	S	G	S	G
0	746.23	243.62	792.09	**200.89**
1	7.09	2.94	7.36	**2.58**
2	429.68	261.35	396.37	**212.79**
3	555.83	289.25	520.18	**147.25**
4	224.97	149.99	193.97	**123.19**
5	630.99	209.39	622.24	**152.73**
6	137.50	58.51	129.80	**47.58**
7	123.71	110.13	103.13	**54.35**
8	490.47	204.75	530.36	**191.28**
9	319.45	177.79	305.52	**131.96**

Notes. LP (reduced) on G performs best by a significant margin in all runs. Fastest total times are displayed in bold.

(reduced) on G decreases slightly from the already low setup time for LP (original) on G: The fact that there are fewer y-variables in LP (reduced) now is a direct advantage and becomes visible in the setup time. As expected, the solution times for LP (reduced) on G also are improvements over the solution times for LP (original).

Finally, in Table 8 we see the significant and consistent improvement in total running times when moving from LPs on S to LPs on G. The slower LP solution times on G are greatly outweighed by the reduction in setup times. We see up to a 75% reduction in total running time from LP (original) on S to LP (reduced) on G.

5.3. Hybrid Data: An MNIST-Style Example

For our final computational experiments, we consider an example in which LP (hybrid) performs significantly better than the other three formulations. For these experiments, we compute a barycenter (using the exact S) for four measures representing the letter "i." The main difference of these (handwritten) letters lies in the location of the dot, rather than the line at the base of the letter. Thus, we desire more accuracy for the upper portion of the image. To this end, we record the letters in a combination of two grids: the base is recorded in a 16×16 grid, in the style of the MNIST digits. The dot is recorded in what we call the *upper grid*. We compare the LP formulations as we refine this upper grid, starting from the original 16×16 grid; refinement 2 instead introduces support points from a 32×32 grid with half the step size. Two sample letters are shown in Figure 5.

Throughout the experiments, we hold the number of support points in the P_i constant; that is, the dot is always represented through the same number of support points. This allows a comparison that isolates the effects of the refinement of the upper grid itself on the resultant formulation sizes. However, the sizes of the constructed LPs vary significantly, as seen in Table 9, because of the changing amount of repetition of

Figure 5. Sample Letters "i" with Dots Mapped to a 2-Times-Finer Upper Grid

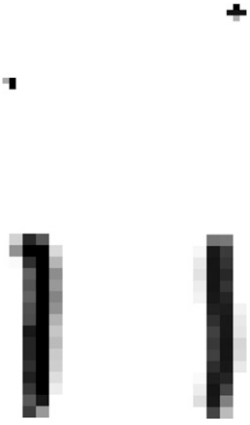


Table 9. LP Sizes for Each Formulation as the Upper Grid Is Refined

	LP (original)		LP (reduced)		LP (general)		LP (hybrid)	
Grid refinement	Constraints	Variables	Constraints	Variables	Constraints	Variables	Constraints	Variables
1	7,067	370,008	7,067	194,632	215	8,258,112	5,823	192,901
2	20,243	1,081,512	20,243	476,316	215	8,258,112	14,883	469,330
3	55,971	3,010,824	55,971	795,175	215	8,258,112	17,427	665,781
4	48,623	2,614,032	48,623	794,117	215	8,258,112	17,027	686,133

Notes. LP (general) is always the same, very large size regardless of refinement. As we refine the upper grid, the reduction in size using LP (hybrid) over LP (reduced) becomes more significant.

weighted means when producing S. For LP (original) and LP (reduced), the largest programs arise for a 3-times-finer grid. This is not surprising, as refinements by 2 or 4 naturally lead to more repetition. LP (general) does not account for this repetition, so its size remains constant.

As expected, LP (hybrid) always provides an improvement over LP (original) and LP (reduced) in model size. Essentially, it treats the few support points of the dot like data in general position through the introduction of w-variables. As we refine the grid, the size reduction for LP (hybrid) over LP (reduced) becomes more and more significant. Note that much of this reduction actually lies in the number of constraints.

Setup, solution, and total running times are shown in Table 10. In this scenario, the setup times are dominated by the solution times. As the upper grid is refined, LP (original) scales quite poorly in solution times, and thus also in total time. LP (reduced) performs better, but is still vastly outperformed by LP (hybrid): the combination of slightly fewer variables and significantly fewer constraints leads to much better solution times, especially for the finest upper grids.

Because LP (general) does not account for repetition, its size is constant through the experiments and its solution times remain constant. This essentially provides an upper bound on the running time for this number of measures regardless of refinement level; LP (hybrid) always performs better, as it can take advantage of the still-present repetition, in particular in the base of the letters. Compared with LP (original), we ultimately see as much as a 99% improvement in total running time using LP (hybrid). Further, the total running time for LP (hybrid) is approximately a 66% improvement over LP (reduced) and LP (general).

6. Concluding Remarks

In this paper, we devised new, smaller linear programs for exact solutions to the discrete barycenter problem. Each formulation is an improvement over the previously known linear program. Each is the best choice for different types of data, as seen in theoretical analysis and computational experiments.

With an abundance of applications for the barycenter problem, these better, exact models may play an important role in many current research questions. In addition to the inherent benefit of finding an exact solution for larger problem instances, they also can be used to improve LP-based approximation methods and are highly beneficial for evaluating the quality of state-of-the-art heuristic approaches.

There are a couple of natural questions arising from this work. Our computational experiments are designed to highlight the relative performance of each linear program. For practical computations, we still see significant potential for improvement in a somewhat different type of study. For example, we found that Gurobi's barrier method makes particularly good use of multiple cores when solving these problems. A more technical analysis of the behavior of the programs with respect to different solvers may be of interest to

Table 10. Setup, Solution, and Total Running Times for Each Formulation as the Upper Grid Is Refined

	Time in seconds											
	LP (original)			LP (reduced)			LP (general)			LP (hybrid)		
Dot grid	Setup	Solve	Total	Setup	Solve	Total	Setup	Solve	Total	Setup	Solve	Total
1	9.06	29.11	38.19	10.65	6.90	**17.59**	19.28	419.87	443.85	13.15	**6.35**	19.55
2	18.50	1,056.37	1,075.22	20.82	69.24	90.18	19.31	415.63	438.98	26.81	**57.87**	**84.59**
3	33.74	14,111.30	14,146.00	31.63	461.32	493.15	19.30	424.76	448.82	46.59	**92.65**	**139.32**
4	34.45	10,643.30	10,687.60	33.94	462.90	495.22	19.07	437.11	461.10	46.24	**118.35**	**164.78**

Notes. LP (hybrid) significantly outperforms the others, scaling far better with the refinement. Fastest solution and total times are displayed in bold.

optimize total running times. The similarity to classical transportation problems also suggests that the problem might be approachable through parallelized algorithms. Further, the large number of variables and relatively low number of constraints in the models, especially for LP (general), makes a column generation approach promising. Finally, a formal proof of hardness of the discrete barycenter problem is notably missing in the literature. In this paper, we only proved that the setup of the LP formulations is hard.

Acknowledgments

The authors thank Natalia Villegas Franco for the implementation of a visualization basis for the computations on Denver crime data in Figure 4.

References

Agueh M, Carlier G (2011) Barycenters in the Wasserstein space. *SIAM J. Math. Anal.* 43(2):904–924.
Anderes E, Borgwardt S, Miller J (2016) Discrete Wasserstein barycenters: optimal transport for discrete data. *Math. Methods Oper. Res.* 84(2): 389–409.
Beiglböck M, Henry-Labordere P, Penkner F (2013) Model-independent bounds for option prices — a mass transport approach. *Finance Stochastics* 17(3):477–501.
Benamou J-D, Carlier G, Cuturi M, Nenna L, Peyré G (2015) Iterative Bregman projections for regularized transportation problems. *SIAM J. Sci. Comput.* 37(2):A1111–A1138.
Boissard E, Le Gouic T, Loubes J-M (2015) Distribution's template estimate with Wasserstein metrics. *Bernoulli* 21(2):740–759.
Borgwardt S (2017) Strongly polynomial 2-approximations for discrete Wasserstein barycenters. Preprint, submitted April 18, https://arxiv.org/abs/1704.05491.
Brenner U (2008) A faster polynomial algorithm for the unbalanced Hitchcock transportation problem. *Oper. Res. Lett.* 36(4):408–413.
Carlier G, Ekeland I (2010) Matching for teams. *Econom. Theory* 42(2):397–418.
Carlier G, Oberman A, Oudet E (2015) Numerical methods for matching for teams and Wasserstein barycenters. *ESAIM Math. Model. Numer. Anal.* 49(6):1621–1642.
Chiaporri P-A, McCann R, Nesheim L (2010) Hedonic price equilibria, stable matching and optimal transport; equivalence, topology and uniqueness. *Econom. Theory* 42(2):317–354.
Cotar C, Friesecke G, Klüppelberg C (2013) Density functional theory and optimal transportation with coulomb cost. *Comm. Pure Appl. Math.* 66(4):548–599.
Cuturi M, Doucet A (2014) Fast computation of Wasserstein barycenters. Jebara T, Xing EP, eds. *Proc. 31st Internat. Conf. Machine Learn.* (ACM, New York), 685–693.
Cuturi M, Peyré G (2016) A smoothed dual approach for variational Wasserstein problems. *SIAM J. Imaging Sci.* 9(1):320–343.
Ford LR, Fulkerson DR (1956) Solving the transportation problem. *Management Sci.* 3(1):24–32.
Jain A, Zhong Y, Dubuisson-Jolly M-P (1998) Deformable template models: a review. *Signal Processing* 71(2):109–129.
Kleinschmidt P, Schannath H (1995) A strongly polynomial algorithm for the transportation problem. *Math. Programming* 68(1–3):1–13.
LeCu Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc. IEEE* 86(11):2278–2324.
Matsui T, Scheifele R (2016) A linear time algorithm for the unbalanced Hitchcock transportation problem. *Networks* 67(2):170–182.
Miller J (2016) Transportation networks and matroids: algorithms through circuits and polyhedrality. PhD thesis, University of California, Davis, Davis.
Munch E, Turner K, Bendich P, Mukherjee S, Mattingly J, Harer J (2015) Probabilistic Fréchet means for time varying persistence diagrams. *Electronic J. Statist.* 9(2015):1173–1204.
Orlin JB (1993) A faster strongly polynomial minimum cost flow algorithm. *Oper. Res.* 41(2):338–350.
Pass B (2014) Multi-marginal optimal transport and multi-agent matching problems: Uniqueness and structure of solutions. *Discrete Continuous Dynamical Systems A* 34(4):1623–1639.
Srivastava S, Li C, Dunson DB (2018) Scalable Bayes via barycenter in Wasserstein space. *J. Machine Learn. Res.* 19(8):1–35.
Tardos E (1986) A strongly polynomial algorithm to solve combinatorial linear programs. *Oper. Res.* 34(2):250–256.
Tokuyama T, Nakano J (1995) Efficient algorithms for the Hitchcock transportation problem. *SIAM J. Comput.* 24(3):563–578.
Trouvé A, Younes L (2005) Local geometry of deformable templates. *SIAM J. Math. Anal.* 37(1):17–59.
Villani C (2009) *Optimal Transport: Old and New*, Grundlehren der mathematischen Wissenschaften, vol. 338 (Springer, New York).
Ye J, Wu P, Wang JZ, Li J (2017) Fast discrete distribution clustering using Wasserstein barycenter with sparse support. *IEEE Trans. Signal Processing* 65(9):2317–2332.
Zemel Y, Panaretos V (2018) Fréchet means and procrustes analysis in Wasserstein space. Preprint, submitted January 24, https://arxiv.org/abs/1701.06876.