An Integer Program for Pricing Support Points of Exact Barycenters

Steffen Borgwardt[1] and Stephan Patterson[2]

[1] steffen.borgwardt@ucdenver.edu; University of Colorado Denver
[2] stephan.patterson@lsus.edu; Louisiana State University in Shreveport

Abstract. The computation of exact barycenters for a set of discrete measures is of interest in applications where sparse solutions are desired, and to assess the quality of solutions returned by approximate algorithms and heuristics. The task is known to be NP-hard for growing dimension and, even in low dimensions, extremely challenging in practice due to an exponential scaling of the linear programming formulations associated with the search for sparse solutions.
A common approach to facilitate practical computations is an approximation based on the choice of a small, fixed set of combinations of support points from the measures that may be assigned mass. Through classic integer programming techniques, we model an integer program to compute additional combinations of support points that, when added to the fixed set, allow for a better approximation of the underlying exact barycenter problem. The approach improves on the scalability of previous column generation approaches: instead of a pricing problem that has to evaluate exponentially many reduced cost values, we solve a mixed-integer program of quadratic size. The properties of the model, and practical computations, reveal a tailored branch-and-bound routine as a good solution strategy.

Keywords: discrete barycenter, integer programming, column generation
MSC 2010: 90B80, 90C08, 90C10, 90C11, 90C46

1 Introduction

Barycenter problems have seen much attention in the literature due to their relevance for applications in a wide range of fields; see Section 1.1. Data for these problems is typically represented as so-called *discrete (probability) measures* P_i, i.e., probability measures with a finite number $|P_i|$ of support points.

The input for the *discrete barycenter problem* is a set of discrete measures $P_1, \dots, P_n$ whose support points lie in $\mathbb{R}^d$ and a corresponding set of positive weights $\lambda_1, \dots, \lambda_n$ with $\sum_{i=1}^n \lambda_i = 1$. Then the discrete barycenter problem is: find a measure $\bar{P}$ such that

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

where W_2 is the quadratic Wasserstein distance and $\mathcal{P}^2(\mathbb{R}^d)$ is the set of all probability measures on $\mathbb{R}^d$ with finite second moments (Agueh and Carlier 2011). Because the measures $P_1, \dots, P_n$ are discrete, all barycenters $\bar{P}$ are also supported on a finite subset of $\mathbb{R}^d$. In addition to information on the support set and associated masses, computing the associated transport cost $\phi(\bar{P})$ requires identifying the *destination (support) points* in each $P_1, \dots, P_n$ to which mass is transported from each support point of $\bar{P}$. Then the optimal transport cost $\phi(\bar{P})$ can be computed as the summation of the weighted squared Euclidean distances between each barycenter support point and its destination points (Anderes et al. 2016, Carlier et al. 2015, Villani 2009).

1.1 Fixed-Support Approximations

The computation of barycenters arises in a vast range of applications, from statistics (Munch et al. 2015, Zemel and Panaretos 2019), over economics (Beiglböck et al. 2013, Chiaporri et al. 2010), game theory (Carlier and Ekeland 2010, Carlier et al. 2015), physics and the material sciences (Cotar et al. 2013, Jain

et al. 1998, Trouvé and Younes 2005), to image processing (Janati et al. 2020a, Simon and Aberdam 2020, Cohen et al. 2020) and machine learning (Heitz et al. 2021, Ho et al. 2019, 2017, Schmitz et al. 2018, Yan et al. 2021). We refer the reader to the recent surveys by Peyré and Cuturi (2019), Panaretos and Zemel (2019) and the textbooks by Villani (2009) and Natarajan (2021).

In recent years, there has been growing interest in approaching clustering and learning applications for high-dimensional data (Cohen et al. 2020, Ho et al. 2019, 2017, Singh et al. 2020), such as those arising in natural language processing (Xu et al. 2018, Ye et al. 2017). However, the computation of a barycenter or an approximation is challenging: the barycenter problem is known to be NP-hard for growing dimension, and this hardness extends to approximate computations, some special cases, and other optimal transport metrics (Altschuler and Boix-Adserà 2022). A variation in which one wants to find a sparse barycenter, i.e., one with a low number of support points, remains NP-hard even for dimension two and three measures (Borgwardt and Patterson 2021). By contrast, the barycenter problem is solvable in polynomial time for fixed dimension (Altschuler and Boix-Adserà 2021).

There is a wide field of research that addresses these challenges through approximate and heuristic methods. Barycenter problems are readily interpreted as multi-marginal optimal transport problems with a special geometric cost structure. Many recent algorithmic improvements for structured multi-marginal optimal transport problems, see for example (Benamou et al. 2015, 2016, 2019, Nenna 2016, Haasler et al. 2021) and references therein, have been particularly impactful for barycenters, and the complexity results in (Altschuler and Boix-Adserà 2021, 2022) build on the intimate connection.

A popular approach is to a priori specify a support set $S_0 \subset \mathbb{R}^d$ of *possible support points* and then optimize over measures supported on S_0, i.e., over the masses associated to each support point. There are many variations of such *fixed-support approximations* (Altschuler and Boix-Adserà 2022). In particular, a restriction of S_0 to the union of original support points in the measures leads to a strongly-polynomial-time 2-approximation in any dimension (Borgwardt 2022). For n measures supported on a shared grid, a barycenter is contained in an n-times finer grid (Borgwardt and Patterson 2020).

In general, the fixation of a polynomial-size S_0 leads to a polynomial-size linear program (LP) (and avoids an exponential scaling in d) which can be solved directly using blackbox LP solvers or through specially tailored approaches such as entropic regularization introduced by Cuturi (2013), Cuturi and Doucet (2014). In the latter approach, the LP is made strongly convex through the addition of a small entropy cost, so that more efficient solvers can be applied. This approach has been refined for competitive, approximate large-scale computations; see, e.g., Benamou et al. (2015), Janati et al. (2020b), Kroshnin et al. (2019), Lin et al. (2020), Solomon et al. (2015).

An alternative approach to obtain an LP is to use the *non-mass splitting property* (Anderes et al. 2016) of barycenters and the Wasserstein distance: given a set of single support points in n measures, the optimal location for joint transport of mass to them is their weighted mean; we describe this observation in some detail in Section 2. A barycenter decomposes into such *combinations of support points* (also called couplings in multi-marginal optimal transport theory), each associated some mass, of the original measures. In turn, this allows one to perform an exact barycenter computation through modeling an LP that assigns mass optimally to the set S^* of all combinations. The resulting LP is a formulation as a general multi-marginal optimal transport problem with a special cost. See, e.g., Anderes et al. (2016), Benamou et al. (2015), Borgwardt and Patterson (2022). Note that S^* generally is of exponential size $|S^*| = \prod_{i=1}^{n} |P_i|$. Again, it is of interest to approximate a barycenter through a smaller set of *possible combinations* S_0^*.

Recent algorithms that take alternative approaches, i.e., do not build on S_0 or S_0^*, such as the Frank-Wolfe algorithm of Luise et al. (2019) and the Functional Gradient Descent algorithm of Shen et al. (2020) are rare exceptions. They underscore a desire to step away from a fixation of S_0 or S_0^*; at the same time, it remains desirable to retain the ability to connect to and use tools from the long line of research on fixed-support approximations.

1.2 Contributions and Outline

For both types of fixed-support approximations, based on optimization over a given support set S_0 or a set of combinations S_0^*, the size of S_0 or S_0^* is the bottleneck. In this work, we are interested in the generation of support points or combinations that can be used for S_0 or S_0^*, respectively. We describe an integer program, and develop a branch-and-bound approach, to find a new combination to add to a given set S_0^* that could

allow an improvement of an optimal barycenter approximation over the new S_0^*. The combination corresponds to a new support point for S_0, too. As such, our method can be used to, *a posteriori*, improve an approximate barycenter computed through *any* method in the literature, or to certify optimality thereof.

When used in an iterative scheme, our method would improve any approximate barycenter to an *exact* barycenter. Thus, it can be used to compete with the previous column generation method in Borgwardt and Patterson (2022) and will outperform it as soon as the exponential-size reduced-cost vector becomes prohibitively large compared to setup and solution of a polynomial-size integer program. Depending on the previous method's implementation, the advantages lie in significantly faster speed at only slightly larger memory or similar computational speed at significantly smaller memory and setup time. One obtains the ability to compute exact barycenters for larger instances than before.

However, this favorable behavior also reveals its main challenges: by design, the method solves a pricing problem for exact barycenters, which itself is already known to be NP-hard from hardness of the separation oracle for the underlying dual LP and, if run in an iterative scheme, it solves the NP-hard exact barycenter problem (Altschuler and Boix-Adserà 2021, 2022). In turn, the approach is computationally expensive and can only be expected to be applied to problems of small-to-moderate size. Further, multiple combinations may have to be computed before a strict improvement is possible and accuracy of computations requires special attention.

The paper is structured as follows. In Section 2, we first recall some background on an (exponential-size) LP formulation for the barycenter problem based on S^* and the non-mass splitting property. Then we turn to our main contribution: we devise an integer program (IP) to search for a combination of support points to add to a given S_0^* such that an improved approximation may be possible. We then show that this IP can be transformed into a simpler mixed-integer program (MIP) through the use of some optimality conditions.

In Section 3, we discuss a practical implementation. We first turn to a theoretical study of properties of the mixed-integer program, such as the fractionality of optimal solutions in a relaxation, which inform the design of solution strategies. Then we analyze the practical results of computational experiments, and discuss the potential and limitations of the proposed approach. The experiments highlight better scaling in speed or memory and reveal viable approaches in practice. We conclude in Section 4 with some final remarks and promising directions of research for further improvements.

2 An Integer Program for Support Point Generation

2.1 A Linear Program for Exact Barycenter Computations

We begin by recalling the construction of a linear program for the computation of a barycenter based on the set S^* of all combinations of support points of measures $P_1, \dots, P_n$, i.e., as a multi-marginal optimal transport problem with special cost. It has been shown (Anderes et al. 2016) that all barycenters satisfy the *non-mass-splitting property*, that is, all mass associated with a barycenter support point is transported to a single destination point x_i in each P_i, $i = 1, \dots, n$. Furthermore, for a given combination of destination points, the mass assigned to that combination must be located at the weighted mean $\sum_{i=1}^n \lambda_i x_i$, as any other location produces a strictly increased cost. For this reason, a barycenter can be described as a set of combinations of destination points and corresponding masses, and the location of the mass assignment can be computed as the weighted mean, when needed. Specifically, S^* contains elements $s_h = (x_1^h, \dots, x_n^h)$ for $h = 1, \dots, \prod_{i=1}^n |P_i|$, and each s_h has a corresponding fixed unit-mass transport cost c_h for each potential combination of destination points. The cost c_h can be computed without computing the weighted mean itself, shown in Borgwardt and Patterson (2022) to be

$$c_h = \sum_{i=1}^{n-1} \lambda_i \sum_{j=i+1}^{n} \lambda_j ||x_i^h - x_j^h||^2.$$

Then the transport cost of the barycenter $\phi(\bar{P})$ can be computed as the sum of the mass assigned to s_h times the unit transport cost c_h.

A resulting linear program for solving the discrete barycenter problem is as follows. Let $c = (c_h)$ be the vector stating the unit-mass transport costs for each element of S^*, and let $d = (d_{ik})$ be the vector stating the mass at support point x_{ik}, the k^{th} support point of P_i. The $0,1$-matrix A gives constraints that

enforce that each support point in each measure receives the correct mass: each row has 1-entries for all combinations s_h in which x_{ik} appears, and 0 otherwise. The variable vector $\mathbf{w} = (w_h)$ denotes mass assigned to each combination s_h. This gives a standard form linear program.

$$\begin{aligned} \min \quad & c^T \mathbf{w} \\ \text{s.t.} \quad & A\mathbf{w} = d \\ & \mathbf{w} \geq 0. \end{aligned} \tag{Bary-LP}$$

The number of variables in LP (Bary-LP) is $\prod_{i=1}^{n} |P_i|$, which scales exponentially in the number of measures n. However, LP (Bary-LP) contains only $\sum_{i=1}^{n} |P_i|$ constraints, which makes it a prime candidate for column generation as explored in Borgwardt and Patterson (2022).

2.2 An Integer Program for Support Point Generation

Classic column generation begins with a (reduced) master problem containing only a small number of all possible variables; for LP (Bary-LP), this is a small number of all possible combinations. Information from the current solution is used to choose a new variable, found by solving a separate pricing problem, for introduction to the master problem.

Specifically, a reduced master problem begins with a set of possible combinations S_0^* that is a small subset of the set of all combinations S^*. Then, the classic pricing problem uses the dual vector y from the current master problem, whose elements correspond to each constraint in the master problem, and A_h, the column of A corresponding to each s_h. By minimizing $c_h - y^T A_h$ (or, equivalently, maximizing $y^T A_h - c_h$), an index h^* corresponding to combination s_{h^*} is found, and s_{h^*} is the chosen combination for introduction to the master problem.

It is possible to solve the classic pricing problem by fully processing the vector c, which has one entry for each combination $s_h \in S^* \backslash S_0^*$, and selecting the best value (Borgwardt and Patterson 2022); in an iterative scheme, the processing of c can be implemented by computing and storing it a priori or by recomputing its entries as needed. The primary goal and contribution of the work in this paper is to reformulate the pricing problem *without* searching the exponentially-scaling c. Instead, we seek to choose a combination s_{h^*} by formulating a pricing program that will individually select an element x_i of each measure P_i. Our approach will make use of properties of the costs c_h specific to barycenter problems, and thus does not readily transfer to general multi-marginal optimal transport problems. Because of these properties, we are able to arrive at a quadratic, and later linear, objective function for an integer program.

We begin the development of a new formulation by expanding the expression for c_h from Section 2.1 in the same manner as in Borgwardt and Patterson (2022):

$$\begin{aligned} c_h &= \sum_{i=1}^{n-1} \lambda_i \sum_{j=i+1}^{n} \lambda_j (x_i^h - x_j^h)^T (x_i^h - x_j^h) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \lambda_i \lambda_j ((x_i^h)^T x_i^h - 2(x_i^h)^T x_j^h + (x_j^h)^T x_j^h). \end{aligned}$$

We reformulate this cost expression, returning to using the original list of support points x_{ik} instead of the combination h. For each i and k, let $\mathbf{z}_{ik}$ be a $0,1$-variable where we interpret $\mathbf{z}_{ik} = 1$ to mean $x_i^h = x_{ik}$. Then the first term $(x_i^h)^T x_i^h$ of the expansion above can be rewritten as $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \lambda_i \lambda_j (x_i^h)^T x_i^h = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \lambda_i \lambda_j x_{ik}^T x_{ik} \mathbf{z}_{ik}$, with the other terms being reformulated similarly.

Additionally, at the optimal index h, the vector $(\mathbf{z}_{ik})$ is precisely A_h, and so $y^T A_h$ may be reformulated as well. In contrast to the classic pricing objective function $\max_h y^T A_h - c_h$, in our objective function these terms will be summed up (and through a set of constraints we guarantee that all but one summand are 0). This allows a reformulation that merges terms for different combinations. Since the constraints in LP (Bary-LP) ensure that x_{ik}, the k^{th} support point of measure P_i, receives the correct mass d_{ik}, we match the formatting on the index of the elements of the dual vector y as y_{ik}, $i = 1, \ldots, n$ and $k = 1, \ldots, |P_i|$, giving

$y = (y_{ik})$, and $y^T A_h = \sum_{i=1}^{n} \sum_{k=1}^{|P_i|} y_{ik} \mathbf{z}_{ik}$.

$$\max_{(\mathbf{z}_{ik})} \sum_{i=1}^{n} \sum_{k=1}^{|P_i|} y_{ik} \mathbf{z}_{ik} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \sum_{m=1}^{|P_j|} \lambda_i \lambda_j (x_{ik}^T x_{ik} - 2 x_{ik}^T x_{jm} + x_{jm}^T x_{jm}) \mathbf{z}_{ik} \mathbf{z}_{jm}.$$

To enforce that we choose exactly one element from each measure, we introduce constraints

$$\sum_{k=1}^{|P_i|} \mathbf{z}_{ik} = 1 \qquad \forall i = 1, \ldots, n.$$

Satisfaction of these constraints allows us to expand the quadruple-sum in the objective and simplify two of its three terms to

$$\begin{aligned} \max_{(\mathbf{z}_{ik})} & \sum_{i=1}^{n} \sum_{k=1}^{|P_i|} y_{ik} \mathbf{z}_{ik} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \lambda_i \lambda_j x_{ik}^T x_{ik} \mathbf{z}_{ik} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{m=1}^{|P_j|} \lambda_i \lambda_j x_{jm}^T x_{jm} \mathbf{z}_{jm} \\ & + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \sum_{m=1}^{|P_j|} \lambda_i \lambda_j x_{ik}^T x_{jm} \mathbf{z}_{ik} \mathbf{z}_{jm}. \end{aligned}$$

However, this remains a quadratic objective function due to the products $\mathbf{z}_{ik} \mathbf{z}_{jm}$. The resulting quadratic program is as follows.

$$\begin{aligned} \max_{(\mathbf{z}_{ik})} \; & \sum_{i=1}^{n} \sum_{k=1}^{|P_i|} y_{ik} \mathbf{z}_{ik} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \lambda_i \lambda_j x_{ik}^T x_{ik} \mathbf{z}_{ik} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{m=1}^{|P_j|} \lambda_i \lambda_j x_{jm}^T x_{jm} \mathbf{z}_{jm} \\ & + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \sum_{m=1}^{|P_j|} \lambda_i \lambda_j x_{ik}^T x_{jm} \mathbf{z}_{ik} \mathbf{z}_{jm} \\ \text{s.t.} \; & \sum_{k=1}^{|P_i|} \mathbf{z}_{ik} = 1 && \forall i = 1, \ldots, n \\ & \mathbf{z}_{ik} \in \{0, 1\} && \forall i = 1, \ldots, n, \forall k = 1, \ldots, |P_i| \end{aligned} \tag{Gen-QP}$$

To reformulate the quadratic objective, we use some classic integer programming techniques. Recall that the $\mathbf{z}_{ik}$ are $0,1$-variables. First, we introduce new variables $\mathbf{z}_{ijkm}$ to represent each product $\mathbf{z}_{ik} \mathbf{z}_{jm}$. Second, we add linear constraints that link the new variables $\mathbf{z}_{ijkm}$ to the original $\mathbf{z}_{ik}, \mathbf{z}_{jm}$:

$$\begin{aligned} \mathbf{z}_{ijkm} &\leq \mathbf{z}_{ik} \\ \mathbf{z}_{ijkm} &\leq \mathbf{z}_{jm} \\ \mathbf{z}_{ijkm} &\geq \mathbf{z}_{ik} + \mathbf{z}_{jm} - 1. \end{aligned}$$

The result is a linearization of the integer program in which $\mathbf{z}_{ik} = 0$ or $\mathbf{z}_{jm} = 0$ forces $\mathbf{z}_{ijkm} = 0$ due to the first two constraints, and $\mathbf{z}_{ikjm} = 1$ if $\mathbf{z}_{ik} = \mathbf{z}_{jm} = 1$ due to the third constraint. This implies that $\mathbf{z}_{ijkm} = \mathbf{z}_{ik} \mathbf{z}_{jm}$ for all $\mathbf{z}_{ik}, \mathbf{z}_{jm} \in \{0, 1\}$. These constraints only need to be stated for pairs i, j with $i < j$. In the following, the vector $\mathbf{z} = \binom{\mathbf{z}^1}{\mathbf{z}^2}$ lists all $\mathbf{z}^1 = (\mathbf{z}_{ik})$, followed by all $\mathbf{z}^2 = (\mathbf{z}_{ijkm})$. We use the notation $\mathbf{z} \in \{0, 1\}$ to denote that all its components lie in the specified domain.

We sum up the construction through a full formulation of the integer program for generating a combination of optimal reduced cost, and a corresponding formal statement.

$$
\begin{aligned}
\max_{\mathbf{z}} \quad & \sum_{i=1}^{n}\sum_{k=1}^{|P_i|} y_{ik}\mathbf{z}_{ik} - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{|P_i|}\lambda_i\lambda_j x_{ik}^T x_{ik}\mathbf{z}_{ik} - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{m=1}^{|P_j|}\lambda_i\lambda_j x_{jm}^T x_{jm}\mathbf{z}_{jm} \\
& +2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{|P_i|}\sum_{m=1}^{|P_j|}\lambda_i\lambda_j x_{ik}^T x_{jm}\mathbf{z}_{ijkm} \\
\text{s.t.} \quad & \sum_{k=1}^{|P_i|}\mathbf{z}_{ik} = 1 && \forall i = 1,\dots,n \\
& \mathbf{z}_{ijkm} \le \mathbf{z}_{ik}, && \forall i = 1,\dots,n, \forall j = 1,\dots,n, j > i, \\
& && \forall k = 1,\dots,|P_i|, \forall m = 1,\dots,|P_j| \\
& \mathbf{z}_{ijkm} \le \mathbf{z}_{jm}, && \forall i = 1,\dots,n, \forall j = 1,\dots,n, j > i, \\
& && \forall k = 1,\dots,|P_i|, \forall m = 1,\dots,|P_j| \\
& \mathbf{z}_{ijkm} \ge \mathbf{z}_{ik} + \mathbf{z}_{jm} - 1, && \forall i = 1,\dots,n, \forall j = 1,\dots,n, j > i, \\
& && \forall k = 1,\dots,|P_i|, \forall m = 1,\dots,|P_j| \\
& \mathbf{z} \in \{0,1\}
\end{aligned}
\tag{Gen-IP}
$$

Theorem 1. *Given a set of possible combinations S_0^*, and an optimal fixed-support approximation of the barycenter problem over S_0^*, IP (Gen-IP) finds a new combination of best reduced cost from the set of all combinations S^*.*

Note that the input of S_0^* and the best solution for that set of possible combinations are required for the specification of $y = (y_{ik})$. By contrast, the set of constraints of IP (Gen-IP) is independent from this input.

2.3 A Simpler Mixed-Integer Program

Next, we simplify IP (Gen-IP) to a smaller mixed-integer program (MIP), i.e., a program in which some of the variables are not required to be integer. The advantages over IP (Gen-IP) will be twofold: in addition to the relaxation of some integer variables, we will be able to eliminate a type of main constraints.

To this end, we assume that $x > 0$ in all measures. This assumption is not a restriction, as the region containing all x_{ik} can always be shifted to lie in the positive orthant of $\mathbb{R}^d$ without changing the optimal solution. This can be verified by recalling that $c_h = \sum_{i=1}^{n-1}\lambda_i \sum_{j=i+1}^{n}\lambda_j ||x_i^h - x_j^h||^2$ only takes into account (squared) pairwise distances of x_i^h, x_j^h. The consequence of this assumption is that $x_{ik}^T x_{jm} > 0$ for all x_{ik}, x_{jm}.

We begin by showing that, in any optimal solution, $\mathbf{z}_{ijkm}$ is chosen as large as possible, even without explicit specification of the constraints $\mathbf{z}_{ijkm} \ge \mathbf{z}_{ik} + \mathbf{z}_{jm} - 1$ or domain constraints $\mathbf{z}_{ijkm} \in \{0,1\}$.

Lemma 1. *Let $x > 0$ and let $\boldsymbol{z}^*$ be an optimal solution to an LP relaxation of IP (Gen-IP) without specification of the constraints $\boldsymbol{z}_{ikjm} \ge \boldsymbol{z}_{ik} + \boldsymbol{z}_{jm} - 1$. Let further $\boldsymbol{z}_{ik}^* \in \{0,1\}$ for all $i = 1,\dots,n$ and $k = 1,\dots,|P_i|$. Then $\boldsymbol{z}^* \in \{0,1\}$.*

Proof. We have to prove that the auxiliary variables $\mathbf{z}_{ijkm}^*$ satisfy $\mathbf{z}_{ijkm}^* \in \{0,1\}$. In the maximization objective, they only appear in the term

$$
+2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{|P_i|}\sum_{m=1}^{|P_j|}\lambda_i\lambda_j x_{ik}^T x_{jm}\mathbf{z}_{ijkm}.
$$

Recall that $x > 0$ by assumption, and that $\lambda > 0$ by definition. Thus, $\lambda_i\lambda_j x_{ik}^T x_{jm} \ge 0$. For fixed $\mathbf{z}_{ik}^*, \mathbf{z}_{jm}^* \in \{0,1\}$, a maximal objective function value can only be achieved by choosing $\mathbf{z}_{ijkm}^*$ as large as possible. As $\mathbf{z}_{ijkm}^* \le \mathbf{z}_{ik}^*, \mathbf{z}_{jm}^*$, one obtains $\mathbf{z}_{ijkm}^* = \min\{\mathbf{z}_{ik}^*, \mathbf{z}_{jm}^*\} \in \{0,1\}$. This proves the claim. □

Lemma 1 shows that, in an optimum of such a relaxation of IP (Gen-IP), if $\mathbf{z}_{ik}^* = \mathbf{z}_{jm}^* = 1$ then $\mathbf{z}_{ijkm}^* = 1$, and if $\mathbf{z}_{ik}^* = 0$ or $\mathbf{z}_{jm}^* = 0$ then $\mathbf{z}_{ijkm}^* = 0$. More generally, integrality of the $\mathbf{z}_{ik}^*$ implies integrality of the auxiliary variables $\mathbf{z}_{ijkm}^*$, and $\mathbf{z}_{ijkm}^* \ge \mathbf{z}_{ik}^* + \mathbf{z}_{jm}^* - 1$ is automatically satisfied. Thus, we can drop the constraints $\mathbf{z}_{ijkm} \ge \mathbf{z}_{ik} + \mathbf{z}_{jm} - 1$ and domain constraints $\mathbf{z}_{ijkm} \in \{0,1\}$ from IP (Gen-IP) to obtain an MIP that retains the same optimal set. Let us state this MIP; recall the notation $\mathbf{z}^1 = (\mathbf{z}_{ik})$.

$$
\begin{aligned}
\max_{\mathbf{z}} \quad & \sum_{i=1}^{n}\sum_{k=1}^{|P_i|} y_{ik}\mathbf{z}_{ik} - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{|P_i|}\lambda_i\lambda_j x_{ik}^T x_{ik}\mathbf{z}_{ik} - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{m=1}^{|P_j|}\lambda_i\lambda_j x_{jm}^T x_{jm}\mathbf{z}_{jm} \\
& +2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{|P_i|}\sum_{m=1}^{|P_j|}\lambda_i\lambda_j x_{ik}^T x_{jm}\mathbf{z}_{ijkm} \\
\text{s.t.} \quad & \sum_{k=1}^{|P_i|}\mathbf{z}_{ik} = 1 && \forall i = 1,\dots,n \qquad \text{(Gen-MIP)} \\
& \mathbf{z}_{ijkm} \le \mathbf{z}_{ik}, && \forall i = 1,\dots,n, \forall j = 1,\dots,n, j > i, \\
& && \forall k = 1,\dots,|P_i|, \forall m = 1,\dots,|P_j| \\
& \mathbf{z}_{ijkm} \le \mathbf{z}_{jm}, && \forall i = 1,\dots,n, \forall j = 1,\dots,n, j > i, \\
& && \forall k = 1,\dots,|P_i|, \forall m = 1,\dots,|P_j| \\
& \mathbf{z}^1 \in \{0,1\}
\end{aligned}
$$

In fact, the maximal choice of $\mathbf{z}^*_{ijkm} = \min\{\mathbf{z}^*_{ik}, \mathbf{z}^*_{jm}\}$ shown in the proof of Lemma 1 also holds when $\mathbf{z}^*$ is fractional, i.e., when $\mathbf{z}^*_{ik}, \mathbf{z}^*_{jm} \neq 0, 1$. This will be helpful in the further analysis and for the design of a viable solution approach in Section 3.

We sum up these observations with a formal statement on the equivalence of the optimal sets of IP (Gen-IP) and MIP (Gen-MIP).

Theorem 2. *Let $x > 0$. Given a set of possible combinations S^*_0, and an optimal fixed-support approximation of the barycenter problem over S^*_0, MIP (Gen-MIP) finds a new combination of best reduced cost from the set of all combinations S^*. The sets of optimal solutions $\boldsymbol{z}^*$ of IP (Gen-IP) and MIP (Gen-MIP) are identical.*

We conclude this section with a closer look at the size of MIP (Gen-MIP) in terms of the number of variables and constraints. For simplicity, let all n measures have p support points. Then there are $n \cdot p$ $0,1$-variables of type $\mathbf{z}_{ik}$. In the following, we refer to support points by their variables $\mathbf{z}_{ik}$.

Each $\mathbf{z}_{ik}$ is matched with $(n-1)\cdot p$ variables $\mathbf{z}_{ijkm}$, one for each point $\mathbf{z}_{jm}$ in another measure. As, $i < j$, there exist $\frac{1}{2}((n\cdot p)\cdot((n-1)\cdot p)) = \binom{n}{2}\cdot p^2$ continuous variables $\mathbf{z}_{ijkm}$. Together, there are a total of

$$
n \cdot p \;\; 0,1\text{-variables and } \binom{n}{2}\cdot p^2 \;\; \text{continuous variables.}
$$

The main constraints are comprised of two types: first, there are n equalities $\sum_{k=1}^{p}\mathbf{z}_{ik} = 1$, one for each measure P_i. Second, for each $\mathbf{z}_{ijkm}$, there are two inequalities $\mathbf{z}_{ijkm} \le \mathbf{z}_{ik}, \mathbf{z}_{jm}$. As $i < j$, this gives $\binom{n}{2}\cdot 2p^2 = n(n-1)\cdot p^2$ constraints. In total, we have

$$
n + n(n-1)\cdot p^2 \;\; \text{main constraints.}
$$

Thus, MIP (Gen-MIP) has a quadratic number of continuous variables and main constraints in the number n of measures and number p of support points in each measure, as well as a linear number of $0,1$-variables. While this contrasts sharply with the number p^n of possible combinations, and while it is a significant improvement over IP (Gen-IP) (where the $\mathbf{z}_{ijkm}$ are $0,1$-variables and each has an additional main constraint), of course, MIPs with such scaling generally quickly become hard to solve. In the next section, we discuss our strategy for and implementation of a solution approach.

3 Solution Approach and Practical Implementation

We begin with the design of a viable solution approach to MIP (Gen-MIP), informed by the observations in Section 2.3. Then, we present some computational results and discuss advantages and disadvantages of our approach.

3.1 A Branch-and-Bound Strategy

Our main goal is the design of a strategy for the solution of MIP (Gen-MIP). In general, mixed-integer programs can be difficult to solve. However, the observations in Section 2.3 lead to a quite natural and promising approach. Recall that the variables that are required to be integer are only the $\mathbf{z}_{ik}$: if and only if these form a $0,1$-vector, the result corresponds to a combination of support points from each measure.

By Theorem 2, the optimal sets of MIP (Gen-MIP) and IP (Gen-IP) are identical. This allows an immediate application of Lemma 1 to an LP relaxation of MIP (Gen-MIP): if an optimal solution $\mathbf{z}^*$ in such a relaxation exhibits $\mathbf{z}^*_{ik} \in \{0,1\}$, then all variables $\mathbf{z}^*_{ijkm}$ 'automatically' satisfy $\mathbf{z}^*_{ijkm} \in \{0,1\}$, too. This means that a branch-and-bound approach for solving MIP (Gen-MIP) requires branching only over the set of $n \cdot p$ $0,1$-variables $\mathbf{z}_{ik}$. We are going to use such a branch-and-bound approach.

First, let us take a closer look at the LP relaxation of MIP (Gen-MIP). Typically, constraints of type $\mathbf{z}^1 \in \{0,1\}$ are relaxed to $0 \leq \mathbf{z}^1 \leq 1$. However, note that $\mathbf{z}^1 \leq 1$ is implied by $\sum_{k=1}^{|P_i|} \mathbf{z}_{ik} = 1$ for all $i = 1, \ldots, n$. Thus it suffices to add the constraints $0 \leq \mathbf{z}^1$ to the system. We arrive at the following LP.

$$
\begin{aligned}
\max_{\mathbf{z}} \quad & \sum_{i=1}^{n} \sum_{k=1}^{|P_i|} y_{ik}\mathbf{z}_{ik} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \lambda_i\lambda_j x_{ik}^T x_{ik}\mathbf{z}_{ik} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{m=1}^{|P_j|} x_{jm}^T x_{jm}\mathbf{z}_{jm} \\
& +2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{|P_i|} \sum_{m=1}^{|P_j|} \lambda_i\lambda_j x_{ik}^T x_{jm}\mathbf{z}_{ijkm} \\
\text{s.t.} \quad & \sum_{k=1}^{|P_i|} \mathbf{z}_{ik} = 1 && \forall i = 1, \ldots, n \\
& \mathbf{z}_{ijkm} \leq \mathbf{z}_{ik}, && \forall i = 1, \ldots, n, \forall j = 1, \ldots, n, j > i, \\
& && \forall k = 1, \ldots, |P_i|, \forall m = 1, \ldots, |P_j| \\
& \mathbf{z}_{ijkm} \leq \mathbf{z}_{jm}, && \forall i = 1, \ldots, n, \forall j = 1, \ldots, n, j > i, \\
& && \forall k = 1, \ldots, |P_i|, \forall m = 1, \ldots, |P_j| \\
& \mathbf{z}^1 \geq 0
\end{aligned}
\tag{Gen-LP}
$$

For a simple discussion, we again assume that all n measures have p support points for the remainder of this section. LP (Gen-LP) then has the same number of $n \cdot p + \binom{n}{2} \cdot p^2$ variables as the MIP, and an additional $n \cdot p$ constraints that replace the $0,1$-domain of the $\mathbf{z}_{ik}$, for a total of $n + n(n-1) \cdot p^2 + np$ constraints. This quadratic scaling leads to relatively low memory requirements for moderate problem sizes; see Section 3.2.

It suffices to search for an *optimal, integral vertex* $\mathbf{z}^*$ of LP (Gen-LP) to match the optimum for MIP (Gen-MIP) due to containment of the optimal set of the LP in the $[0,1]$-unit hypercube in the underlying space. In the following, we use the tag (Gen-LP) to also refer to the underlying polyhedron. To measure 'how far' one is from having an integer solution, we now study the number of fractional components – we refer to this as *fractionality* – of a vertex of polyhedron (Gen-LP). Early computations immediately revealed that the LP can lead to fractional optimal vertices. In particular, while the constraint matrix has only $0,1$-entries, it is not totally-unimodular; we provide a brief, elementary argument in an appendix. This is not surprising: pricing for exact barycenters is NP-hard (Altschuler and Boix-Adserà 2022), unless the dimension is fixed (Altschuler and Boix-Adserà 2021). The dimension of the underlying data affects LP (Gen-LP) only in the size of scalar products in the objective function. The polyhedron does not depend on the dimension and right-hand sides are integral, so one would get an efficiently solvable pricing problem if the matrix was totally-unimodular; a contradiction.

The study of fractionality serves two purposes. First, we show that there are not only fully-fractional solutions, i.e., solutions where all components are fractional, but even fully-fractional *vertices*. This further justifies the use of classic integer programming techniques, such as branch-and-bound, to 'break up' a high fractionality. In other situations, specifically if the number of fractional variables is provably low, simpler approaches, such as tailored rounding routines, sometimes suffice in practice. Such an example appeared in our previous work, for example, in Borgwardt et al. (2011). Second, we are able to identify a repeated structure in the fractional components of a vertex. This helps with the selection of promising branching rules.

We begin by showing the existence of a fully-fractional vertex of polyhedron (Gen-LP). It is not hard to state a fully-fractional solution $\mathbf{z}^*$ of LP (Gen-LP). Consider an input of $n \geq 2$ measures of $p \geq 2$ support points. Then $\mathbf{z}^*$ defined by setting $\mathbf{z}^*_{ik} = \frac{1}{p}$ for all $i \leq n, k \leq p$ and $\mathbf{z}^*_{ijkm} = \min\{\mathbf{z}^*_{ik}, \mathbf{z}^*_{jm}\} = \frac{1}{p}$ clearly gives a feasible solution to LP (Gen-LP). We now show that $\mathbf{z}^*$ is, in fact, a vertex.

Lemma 2. *Let $n \geq 2, p \geq 2$. Then $\boldsymbol{z}^*$ defined by $\boldsymbol{z}^*_{ik} = \frac{1}{p}$ for all $i \leq n, k \leq p$ and $\boldsymbol{z}^*_{ijkm} = \frac{1}{p}$ for all $i \leq n, j \leq n, j > i$ and $k \leq p, m \leq p$ is a vertex of polyhedron (Gen-LP).*

Proof. Recall that, given a representation of a polyhedron, to form a vertex there has to exist a set of active constraints whose normals form a matrix of full rank equal to the dimension. In other words, one has to identify a row submatrix of active constraints with a rank equal to the dimension of the underlying space, i.e., equal to the number of variables $n \cdot p + \binom{n}{2} \cdot p^2$ of LP (Gen-LP).

To this end, note that the second and third types of constraints of LP (Gen-LP), rewritten as

$$\begin{aligned} \mathbf{z}_{ijkm} - \mathbf{z}_{ik} &\leq 0, \quad \forall i = 1, \ldots, n, \forall k = 1, \ldots, p \\ \mathbf{z}_{ijkm} - \mathbf{z}_{jm} &\leq 0, \quad \forall j = 1, \ldots, n, j > i, \forall m = 1, \ldots, p \end{aligned}$$

form a collection of rows with a single $+1$ and a single -1, and 0 everywhere else. The corresponding row submatrix is the transpose of the node-arc incidence matrix of a directed, bipartite graph $G = (V, E)$ with vertex set $V = A \dot{\cup} B$, where A is the set of $\mathbf{z}_{ijkm}$, B the set of $\mathbf{z}_{ik}$, and E the set of directed arcs from A to B that connect each $\mathbf{z}_{ijkm}$ with $\mathbf{z}_{ik}$ and $\mathbf{z}_{jm}$.

For a node-incidence matrix, a column set is linearly independent if and only if it does not form a cycle in the underlying undirected graph. Based on this idea, we identify a set T of $n \cdot p + \binom{n}{2} \cdot p^2 - 1$ arcs in this graph that do not form a cycle. These arcs correspond to a row submatrix of rank $n \cdot p + \binom{n}{2} \cdot p^2 - 1$. Afterwards, we then have to identify only one additional, independent row from LP (Gen-LP).

First, note that each $\mathbf{z}_{ijkm}$ is connected to only $\mathbf{z}_{ik}$ and $\mathbf{z}_{jm}$. To form a cycle in G that includes a $\mathbf{z}_{ijkm}$, one has to select *both* incident arcs. This allows us to begin construction of T by adding *exactly one* of these two arcs for all $\mathbf{z}_{ijkm}$, which gives $\binom{n}{2} \cdot p^2$ arcs. Informally, we start with a maximal matching in the underlying bipartite graph.

It remains to identify $n \cdot p - 1$ further arcs that can be added to T without closing a cycle. First, we add all the missing $(n-1) \cdot p$ arcs to connect $\mathbf{z}_{11}$ to all $\mathbf{z}_{jm}$ for $2 \leq j \leq n$, $m \leq p$ by a path of two edges incident to $\mathbf{z}_{1j1m}$. Second, we add all the missing $p - 1$ arcs to connect $\mathbf{z}_{21}$ to all $\mathbf{z}_{1m}$ for $2 \leq m \leq p$ by a path of two edges incident to $\mathbf{z}_{12m1}$.

We arrive at a total of $(n-1) \cdot p + (p-1) = n \cdot p - 1$ additional arcs that do not form a cycle in the graph: only $\mathbf{z}_{11}$ and $\mathbf{z}_{21}$ are connected (by paths of two edges) to other nodes of type $\mathbf{z}_{ik}$, but a cycle would have to include at least one more $\mathbf{z}_{ik}$. Informally, we have constructed a spanning tree in the underlying bipartite graph.

It remains to identify one additional independent row in the system. Of course, the equality constraints $\sum_{k=1}^{p} \mathbf{z}_{ik} = 1$ for $i \leq n$ are always satisfied. The rows corresponding to set T (augmented with their right-hand sides 0) form a system with feasible solution set $z = c \cdot (1, \ldots, 1)^T$ for all $c \in \mathbb{R}$, and adding any of the equalities $\sum_{k=1}^{p} \mathbf{z}_{ik} = 1$ gives precisely $c = \frac{1}{p}$. Thus, we add the row of one of the equalities ($\sum_{k=1}^{p} \mathbf{z}_{ik} = 1$ for any i) to T and obtain a row submatrix of full rank. This shows that $\mathbf{z}^*$ is a vertex of polyhedron (Gen-LP). □

While the above example might seem pathological, highly and fully-fractional solutions do appear in our practical computations in Section 3.2; c.f., Table 7. As we will see, classic branching rules, such as branching on lowest-index variables or those closest or furthest from integrality, all are viable to break up the high fractionality.

We conclude this section by proving a sufficient property for an optimal vertex $\mathbf{z}^*$ of LP (Gen-LP) to be integral. We show that the fractional values in $\mathbf{z}^1$, i.e., the variables $\mathbf{z}^*_{ik}, \mathbf{z}^*_{jm}$, cannot all be different. In fact, there must be a repeated value $\mathbf{z}^*_{ik} = \mathbf{z}^*_{jm}$ for some $i \neq j$ and some k, m. Otherwise $\mathbf{z}^* \in \{0, 1\}$.

Lemma 3. *Let $n \geq 2, p \geq 2$. Let further $\boldsymbol{z}^*$ be an optimal vertex for LP (Gen-LP) and let $\boldsymbol{z}^*_{ik} \neq \boldsymbol{z}^*_{jm}$ for all $0 < \boldsymbol{z}^*_{ik}, \boldsymbol{z}^*_{jm} < 1$ with $i \neq j$. Then $\boldsymbol{z}^* \in \{0, 1\}$.*

Proof. We prove the claim by contradiction. Assume $\mathbf{z}^* \notin \{0, 1\}$ and $\mathbf{z}^*_{ik} \neq \mathbf{z}^*_{jm}$ for all $0 < \mathbf{z}^*_{ik}, \mathbf{z}^*_{jm} < 1$ with $i \neq j$. We will show that the number of (independent) active constraints is strictly less than the number $n \cdot p + \binom{n}{2} \cdot p^2$ required for a vertex of polyhedron (Gen-LP). We now build this set T of active constraints.

Note that the n equality constraints are always active and independent from each other; they are added to T. Note further that variables $\mathbf{z}_{ik}$ only appear in the i-th equality constraint and that, for each i, at least one $\mathbf{z}^*_{ik}$ satisfies $\mathbf{z}^*_{ik} > 0$. Thus one can add any active domain constraints $\mathbf{z}_{ik} \geq 0$ to T while retaining linear

independence. As $\mathbf{z}^* \notin \{0,1\}$, at most $n \cdot (p-1) - 1$ domain constraints of type $\mathbf{z}_{ik} \geq 0$ are active. At this point, the size $|T|$ of T is bounded above by $|T| \leq n + n \cdot (p-1) - 1 = n \cdot p - 1$.

The remaining $\binom{n}{2} \cdot p^2 + 1$ active, independent constraints need to come from the auxiliary constraints

$$\begin{aligned} \mathbf{z}_{ijkm} &\leq \mathbf{z}_{ik} \\ \mathbf{z}_{ijkm} &\leq \mathbf{z}_{jm} \end{aligned}$$

for the different $\mathbf{z}_{ijkm}$. As $\mathbf{z}^*$ is optimal, $\mathbf{z}^*_{ijkm} = \min\{\mathbf{z}^*_{ik}, \mathbf{z}^*_{jm}\}$. Thus at least one of the auxiliary constraints is active for each $\mathbf{z}_{ijkm}$. Such a constraint can be added to T while retaining linear independence, as the variable $\mathbf{z}_{ijkm}$ is not used in any of the other constraints in T.

As $\mathbf{z}^*_{ik} \neq \mathbf{z}^*_{jm}$ for all $0 < \mathbf{z}^*_{ik}, \mathbf{z}^*_{jm} < 1$ with $i \neq j$, only one auxiliary constraint is active whenever at least one of the variables is fractional. That constraint is added to T. For $\mathbf{z}^*_{ik} = \mathbf{z}^*_{jm} = 0$ or $= 1$, however, both constraints are active. If $\mathbf{z}^*_{ik} = \mathbf{z}^*_{jm} = 0$, the corresponding domain constraints $\mathbf{z}_{ik} \geq 0$ and $\mathbf{z}_{jm} \geq 0$ already are in T. The system

$$\begin{aligned} \mathbf{z}_{ijkm} &\leq \mathbf{z}_{ik} \\ \mathbf{z}_{ijkm} &\leq \mathbf{z}_{jm} \\ \mathbf{z}_{ik} &\geq 0 \\ \mathbf{z}_{jm} &\geq 0 \end{aligned}$$

only has rank 3, as it involves 3 variables. Thus the second constraint involving $\mathbf{z}_{ijkm}$ may not be added to T without violating independence.

Finally, if $\mathbf{z}^*_{ik} = \mathbf{z}^*_{jm} = 1$, then the system

$$\begin{aligned} \sum_{l=1}^{p} \mathbf{z}_{il} &= 1 \\ \mathbf{z}_{il} &\geq 0 \qquad \forall l \leq p, l \neq k \end{aligned}$$

is of full rank p. The same holds for

$$\begin{aligned} \sum_{l=1}^{p} \mathbf{z}_{jl} &= 1 \\ \mathbf{z}_{jl} &\geq 0 \qquad \forall l \leq p, l \neq m. \end{aligned}$$

Combining these two systems with the constraints

$$\begin{aligned} \mathbf{z}_{ijkm} &\leq \mathbf{z}_{ik} \\ \mathbf{z}_{ijkm} &\leq \mathbf{z}_{jm} \end{aligned}$$

gives a system of rank $2p + 1$, as it involves $2p + 1$ variables. Again, the second constraint involving $\mathbf{z}_{ijkm}$ may not be added to T without violating independence.

Thus exactly $\binom{n}{2} \cdot p^2$ auxiliary constraints – one for each $\mathbf{z}_{ijkm}$ – can be added to T while retaining linear independence. The total number of active constraints is at most $n \cdot p + \binom{n}{2} \cdot p^2 - 1 < n \cdot p + \binom{n}{2} \cdot p^2$, strictly less than what is required for a vertex of polyhedron (Gen-LP). This proves the claim. □

Lemma 3 implies that there must be repeats of a fractional value in any fractional optimal vertex. This observation makes it promising to try a branching rule that prioritizes the elimination of such repeats and compare its performance to standard branching rules. Further, such a repeat has to appear for fractional values in *different* measures. Thus at least two measures are split fractionally; at least four variables are fractional. This means that the depth of a branch-and-bound tree is bounded by $n \cdot p - 3$, regardless of the branching rule.

3.2 Computational Experiments

The purpose of our computational experiments is twofold: one, to confirm the predicted behavior of solving LP (Gen-LP) in regards to the fractionality of the vertices, and two, to explore the practicality of using MIP (Gen-MIP). To these ends, we implemented a column generation routine using the Gurobi Mixed Integer Program (MIP) solver using the C++ API; source code is available at `https://github.com/StephanPatterson/PricingIP`. This MIP solver, as is standard in commercial solvers, is based on branch-and-bound, runs in parallel, and by default supplements with cutting plane methods and other heuristics. We also implemented a custom branch-and-bound routine fully in C++ (that only uses the Gurobi Optimizer as part of its node exploration) for comparing different branching strategies; this source code is available at `https://github.com/StephanPatterson/IP-Pricing-BB`. Throughout this section, we refer to these two implementations as the *MIP approach*. For comparisons involving computation times or memory requirements, we relate to the direct use of the Gurobi MIP solver for MIP (Gen-MIP). For information on fractionality, best branching strategies, and the size of the branch-and-bound tree, we relate to our custom implementation.

When solving MIP (Gen-MIP) through the MIP solver in Gurobi, we found the solution times to be highly influenced by solver options. We first disabled the cutting plane methods, as adding constraints (particularly those without the same structure currently present in the problem) can increase solution times. Disabling heuristics also improved solution times, while disabling the presolver dramatically increased solution times, and remains enabled in the following experiments. We also set the MIP solver to focus on proving optimality (MIPFocus = 2). With these settings, we performed a comparison of solution times of MIP (Gen-MIP) to the quadratic program (Gen-QP). Allowing Gurobi to handle the quadratic objective directly was typically slightly faster, but with slightly higher memory utilization; since the scaling with problem size was almost identical, we focus our experiments on MIP (Gen-MIP). Source code for a direct solution of (Gen-QP) is available at `https://github.com/StephanPatterson/ColGen_QuadIPPrice`.

A Comparison of Methods in Low and Higher Dimension. We compare two implementations of a previous column generation strategy, based on a direct evaluation of the reduced cost vector $y^T A - c$ as in Borgwardt and Patterson (2022), with the MIP approach in view of computational speed and memory requirement. In both of our implementations of the direct evaluation, we efficiently form the product $y^T A$ in each iteration of column generation, i.e., without storing the matrix A. In the implementation *"CG SaveC"*, the vector c of approximate size p^n is stored in memory, while in the second implementation *"CG RecompC"* we recompute each element of c as needed. Note that CG SaveC has exponential memory requirement and comes with high setup costs for growing dimension due to the evaluation of d-dimensional scalar products for each of the exponentially many elements c_h; in contrast, CG RecompC has a minimal memory requirement and setup cost (essentially just storing the original input) but recomputes the elements c_h in each iteration, which again requires the evaluation of d-dimensional scalar products and introduces a scaling with the dimension in each iteration. One of the primary benefits of the MIP approach is that its feasible set does not scale with the dimension, and memory and setup cost are small; the objective function is set up, just once, by forming the scalar product for each pair of support points.

We performed experiments on data sets in dimensions 2, 12, and 48, to highlight effects that already become visible in low dimension and effects in higher dimension. Recall that a 2-dimensional data set is especially favorable for CG SaveC and CG RecompC. Our experiments were run on a laptop (MacBook Pro, 2.4 GHz Intel Core i9, 32 GB of RAM, SSD) with 16 cores for parallel computations. These experiments consist of running column generation on n measures, with n between 3 and 36, and each measure containing $2-12$ support points for an average of $2.3-4.7$ support points per measure. We report on the averages for many repeated runs, with $10-100$ repeats depending on problem size. For higher dimensions, these problem sizes (especially for the larger values of n) are already very challenging for an exact barycenter computation (Anderes et al. 2016, Altschuler and Boix-Adserà 2022).

We begin with experiments on 2-dimensional data. The data for the experiments consists of event locations given in latitude and longitude (support points). These locations do not underlie an obvious structure, which is the hardest setting for barycenter computations (Altschuler and Boix-Adserà 2022, Borgwardt and Patterson 2020). The events are grouped by the month and year in which they occurred (measures). Tables 1 and 2 display the results. In the tables in this section, n refers to the number of measures, *support* refers to the total number of support points, and *variables* refers to the number of variables in MIP (Gen-MIP).

Table 1 exhibits the gap in memory measured after the first solve of the pricing problem and the addition of the first column to the master problem. Whereas the memory requirement for CG RecompC and the MIP approach are almost constant for the experiments, and almost negligible for CG RecompC, the memory for CG SaveC scales exponentially. As problem size grows, the memory gap widens and quickly reaches some orders of magnitude. The same effect happens for setup times before the first iteration, which remain negligible for CG RecompC and the MIP approach, but grow exponentially for CG SaveC, reaching up to 15 minutes for the largest instances.

In contrast, the high cost of recomputation for CG RecompC is evident in Table 2, which exhibits the gap in average time per solution of the pricing problem. For small problem sizes, our experiments reveal some overhead cost for the MIP approach in each iteration that is not needed for the classic approaches; see the first row in the table. For the larger problems, the MIP approach and CG SaveC perform similarly and run significantly faster than CG RecompC. As CG SaveC requires dramatically more memory and has significant initial setup cost, the MIP approach performs best among the three options already in dimension 2.

Before moving to higher-dimensional experiments – where the strengths of the MIP approach lie by design – we would like to note that, in dimension 2 and in absolute terms, neither of the above approaches comes close to the performance of the fastest exact algorithm in the literature: the algorithm in Altschuler and Boix-Adserà (2021) is reported to solve a problem with 10 measures and a total support of 200 points exactly in about 50 seconds. We generated some data sets mirroring our problem sizes (more measures, fewer support points) from Tables 1 and 2 and ran the algorithm on our equipment, validating overall running times below 45 seconds. To the best of our knowledge, an implementation of this algorithm is not available for higher dimensions.

n	Total Support	Variables	CG SaveC	CG RecompC	MIP
12	42	1,990,656	38	1	120
14	57	28,449,792	220	2	104
15	50	31,850,496	245	3	94
18	56	254,803,968	1,990	3	110
15	71	731,566,080	10,910	3	143
19	60	1,019,215,872	22,910	3	146
17	73	1,567,641,600	23,370	3	139
16	69	1,820,786,688	41,000	3	146

Table 1. Total memory used in MB for two-dimensional experiments.

n	Total Support	Variables	CG SaveC	CG RecompC	MIP
12	42	1,990,656	0.088	2.95	4.18
15	50	31,850,496	1.12	22.6	10.07
14	57	28,449,792	1.16	24.7	18.450
18	56	254,803,968	9.09	179.1	18.12
15	71	731,566,080	93.86	481.1	110.39
19	60	1,019,215,872	30.73	717.7	22.30
17	73	1,567,641,600	90.04	1092.3	115.28
16	69	1,820,786,688	126.47	1276.7	70.22

Table 2. Average time per solution of the pricing problem in seconds for two-dimensional experiments.

Next, we take a closer look at some experiments in which each support point has 12 or 48 characteristics in its support, i.e., on a 12-dimensional and 48-dimensional data set, respectively. The support points were constructed from an open data set: the Heart Disease data set available at `https://data.world/kudem/`

`heart-disease-dataset`. Our goal is to exhibit the differences in performance to the previous 2-dimensional runs. The overall problem sizes are similar to the top half of the experiments in Tables 1 and 2; as we will see, for larger problems we run into scaling issues for CG SaveC and CG RecomputeC.

Tables 3 and 4 show the memory requirements and setup times. As expected, the memory requirements are very similar to the 2-dimensional experiments; c.f., Table 1. CG RecompC requires minimal memory to encode the original input, MIP requires a bit more due to encoding MIP (Gen-MIP), and CG SaveC exhibits an exponential scaling. Major differences to the previous experiments lie in the computational cost for the setup of CG SaveC. While that cost remains negligible for CG RecompC and the MIP approach at these problem sizes, for CG SaveC, the computation of the exponentially-growing vector c requires the evaluation of scalar products that are 6 times or 24 times larger (compared to the 2-dimensional experiments) for each element of c. In the tables, instances that exceeded 30 minutes for setup are marked with a (*).

Tables 5 and 6 show the average time per solution of the pricing problem. The results for CG SaveC and the MIP approach again are similar to each other and similar to the 2-dimensional experiments; c.f., Table 1. This is as expected, as after setup the underlying dimension is not 'visible' anymore and the selection of a best reduced cost or the solution of MIP (Gen-MIP) work just as before. Significant differences arise in the scaling of CG RecompC, where the fact that much larger scalar products have to be formed in each iteration (again 6 or 24 times larger than before) lead to an even larger gap in running times. In the tables, instances that exceeded 30 minutes for their iteration time are marked with a (*).

n	Total Support	Variables	CG SaveC		CG RecompC		MIP	
			MB	sec	MB	sec	MB	sec
12	42	1,990,656	33	8.79	3	0.009	50	0.007
22	47	12,582,912	194	80.54	3	0.009	77	0.008
15	45	14,348,907	208	71.69	3	0.007	45	0.007
15	50	31,850,496	488	157.4	3	0.006	60	0.006
16	52	63,700,992	974	322.5	3	0.023	61	0.009
17	54	127,401,984	1900	682.7	3	0.011	59	0.006
20	55	191,102,976	3800	1396	3	0.011	71	0.025
18	56	254,803,968	*	*	3	0.015	71	0.007

Table 3. Total memory used in MB and setup times in seconds for 12-dimensional experiments. (*) These runs exceeded half an hour of setup time.

n	Total Support	Variables	CG SaveC		CG RecompC		MIP	
			MB	sec	MB	sec	MB	sec
12	42	1,990,656	33	28.34	3	0.014	45	0.029
22	47	12,582,912	155	276.1	3	0.017	48	0.009
15	45	14,348,907	221	237.7	3	0.63	43	0.021
15	50	31,850,496	483	558.96	3	0.035	62	0.025
16	52	63,700,992	488	1124.0	3	0.033	71	0.028
17	54	127,401,984	*	*	3	0.021	65	0.007
20	55	191,102,976	*	*	3	0.027	82	0.014
18	56	254,803,968	*	*	3	0.032	67	0.02

Table 4. Total memory used in MB and setup times in seconds for 48-dimensional experiments. (*) These runs exceeded half an hour of setup time.

Summing up, the MIP approach performs as expected. It cannot resolve the underlying challenge in the computations, but its scaling is better than previous approaches. Especially for larger dimensions, the

n	Total Support	Variables	CG SaveC	CG RecompC	MIP
12	42	1,990,656	0.039	7.84	0.38
22	47	12,582,912	0.25	76.4	8.4
15	45	14,348,907	0.24	65.7	0.32
15	50	31,850,496	0.73	144.4	0.99
16	52	63,700,992	1.19	299.3	1.19
17	54	127,401,984	2.64	650.9	1.57
20	55	191,102,976	5.77	1098.6	2.39
18	56	254,803,968	5.9	*	1.99

Table 5. Average time per solution of the pricing problem in seconds for 12-dimensional experiments. (*) These runs exceeded a half-an-hour iteration time.

n	Total Support	Variables	CG SaveC	CG RecompC	MIP
12	42	1,990,656	0.036	20.1	0.42
22	47	12,582,912	0.253	171.2	8
15	45	14,348,907	0.24	140.9	0.52
15	50	31,850,496	0.65	302.8	0.96
16	52	63,700,992	1.3	1002.1	1.25
17	54	127,401,984	**	*	1.22
20	55	191,102,976	**	*	2.37
18	56	254,803,968	**	*	1.49

Table 6. Average time per solution of the pricing problem in seconds for 48-dimensional experiments. (*) These runs exceeded a half-an-hour iteration time. (**) These runs exceeded a half-an-hour setup time.

advantage takes either the form of computational speed (if recomputing costs) or the form of lower memory requirement and setup times (if storing the costs).

Fractionality of Solutions. Next, we turn to the fractionality of solutions in LP (Gen-LP), the relaxation of MIP (Gen-MIP). In Lemma 2, we saw that there exist pathological vertex solutions that are fully fractional.

To measure the fractionality in practice, we generated random subsets of the input measures in Section 3.2 and an initial set of combinations for a feasible solution, in turn giving us the vector y for the pricing objective. Then, we solved the resulting LP (Gen-LP) and measured the percentage of the variables $\mathbf{z}_{ik}$ which were fractional, along with the number of unique (different) fractional values.

Table 7 shows the results of these computations. They align with the theoretical analysis in Section 3.1 and further justify the design and use of a branch-and-bound method. The number of fractional values is very high throughout, sometimes reaching 100%, while the number of unique values is low. A good branching strategy would be one that not only 'attacks' the high fractionality (in a sense, any branching strategy would do that), but prioritizes a reduction in the number of unique fractional values, which are already low to begin with.

Design of Branch-and-Bound for MIP (Gen-MIP). Finally, we discuss the design of a custom branch-and-bound method to solve MIP (Gen-MIP). We begin with some general information and then turn to a comparison of different branching strategies. The method begins with solving the relaxation LP (Gen-LP), whose solution provides an upper bound (or 'maximum potential') for the objective value of MIP (Gen-MIP). We will explore different strategies for choosing a variable $\mathbf{z}_{ik}$ for branching; once chosen, in the left-hand subtree, $\mathbf{z}_{ik}$ is set to 0 and in the right-hand subtree, it is set to 1. Since at most one decision per variable is required to reach integrality, the maximum depth of the produced tree is $\sum_{i=1}^{n} |P_i|$; in fact, the more refined analysis at the end of Section 3.1 exhibits that this bound can be improved to $(\sum_{i=1}^{n} |P_i|) - 3$. Further, by setting a variable to 1, one implicitly sets all other variables for the same measure to 0. This results in a theoretical maximum number of nodes in the tree of $\prod_{i=1}^{n} |P_i|$. As we will see, in our practical computations, we only visit a small fraction of these exponentially many possible nodes. The depth of the

n	Total Support	Fractional Values	Unique Fractional Values
3	21	100%	1
8	29	86.2%	4
9	41	87.8%	5
12	57	100%	5
14	63	98.4%	6
18	56	96.4%	3
23	71	70.4%	3
36	132	84.8%	5

Table 7. Percentage of variables $\mathbf{z}_{ik}$ observed to be fractional when solving LP (Gen-LP) and the number of unique fractional values.

tree where pruning occurs, and how many nodes are avoided, will depend on the branching strategy, but, regardless, many right-hand branches will be pruned during the process. Further, our branch-and-bound implementation begins with an initial lower bound that also immediately allows for pruning branches that fall below it.

Recall that the initialization of column generation requires an initial feasible solution. We generate such an initial solution using the greedy strategy described in Borgwardt and Patterson (2022), which iteratively creates combinations of support points out of the first support points in each measure available to receive mass, and assigns to each generated combination the maximum mass receivable by those support points. The generated solution has the aforementioned second use: the total transportation cost associated with it leads to an initial lower bound on the optimal value of MIP (Gen-MIP) for the purpose of pruning nodes.

During the custom branch-and-bound, processing a node creates exactly two child nodes and the resulting (relaxed) MIPs are solved immediately. Solving the two new MIPs is implemented using the new Multiple Scenario feature in Gurobi 9.0. The assigned values of the branching variable (left-hand branch 0, right-hand branch 1) is achieved using the scenario upper bound and lower bound attributes, ScenNUB and ScenNLB, by setting the left-hand branch to have an upper bound of 0, and the right-hand branch to a lower bound of 1. Gurobi's output reports that the two scenarios are infeasible only if both scenarios are infeasible, so we check each scenario individually for infeasibility and prune the tree accordingly.

We are now ready to explore three branching strategies. Our goal is to identify a strategy that produces the smallest branch-and-bound tree, i.e., that processes the fewest nodes.

First, we consider a naive direct branching strategy in which the variables are branched on in the order they are given. That is, first $\mathbf{z}_{11}$ is processed, then $\mathbf{z}_{12}$, until the last support point of measure P_1 is reached; then the branching moves on to the points in measure P_2, and so on. We refer to this simple strategy as *Index Order*.

Second, we consider a *Closest to Integer* strategy in which the variable whose value is closest to 0 or 1 (but still some small tolerance away from integer to avoid issues of numerical stability) is branched on. Essentially this is a rounding strategy, based on the general idea that, in practice, a good solution to an integer program may sometimes be found by rounding a relaxed solution to integer values.

Third, we explore a strategy in which we select a (first) variable whose current fractional value is repeated most frequently in all variables. As observed earlier, fractional solutions in which all variables are assigned the same value are both theoretically possible and appear in practice. The goal of this branching strategy is to break apart this highly repetitious fractionality the most directly. We refer to this strategy as *Most Repeated*.

In our initial experiments, we leave the measures in their naturally occurring order ('unsorted'). Results are shown in Table 8. We observed that the fewest nodes are processed for the *Most Repeated* strategy, in which we focus on breaking apart the highly fractional nature of the solutions of the relaxation. This aligns with the results of our theoretical analysis in Section 3.1, where we identified such a strategy as a particularly promising angle of attack.

The naive *Index Order* strategy performed worst. The performance of the rounding strategy *Closest to Integer* was generally between that of the *Index Order* and *Most Repeated* strategies: for larger instances it performed almost as well as the *Most Repeated* strategy; for smaller instances it performed similarly to

n	Total Support	Nodes Processed: Unsorted		
		Index Order	Closest to Integer	Most Repeated
5	32	2282	2281	1028
6	34	4567	3755	2057
7	31	2870	3007	2204
8	29	2247	2032	1727
9	26	4225	3071	3071
9	32	15423	9366	8639
9	34	7212	4055	3455
10	29	15489	6143	6143

Table 8. The number of nodes processed for three branching strategies. The measures were passed to the algorithm in their naturally occurring order.

the *Index Order* strategy. It is worth noting that in all strategies, only a small percentage of the theoretical maximum number $\prod_{i=1}^{n} |P_i|$ of nodes of a branch-and-bound tree is ever explored. Using the average number of support points in each measure to approximate the bound, the theoretical maxima for the experiments in Table 8 would lie between $1.07 \cdot 10^4$ (for $n = 5$ and 32 support points total) and $1.57 \cdot 10^5$ (for $n = 9$ and 34 support points total).

In previous work on column generation (Borgwardt and Patterson 2022), we observed that sorting the measures so that the smallest measures (lowest number of support points) are first can be beneficial for solution times. We theorized that such a sorting may also be beneficial in our branch-and-bound method, as particularly in the *Index Order* strategy the variables for entire measures would be fixed at high levels of the tree. Table 9 shows how the number of processed nodes changes with this preprocessing. The number of processed nodes dropped significantly for all strategies. As expected, the benefit is the largest for the *Index Order* strategy, but *Most Repeated*, which focuses on breaking apart the highly repetitious fractionality of solutions, still outperforms the other strategies.

n	Total Support	Nodes Processed: Sorted		
		Index Order	Closest to Integer	Most Repeated
5	32	1560	1785	1028
6	34	3123	2941	2057
7	31	2185	2748	1575
8	29	1564	1339	1080
9	26	4696	4292	3240
9	32	1597	1027	1023
9	34	4426	3768	3456
10	29	4797	3238	3071

Table 9. The number of nodes processed for three branching strategies. The measures were presorted so that those with the fewest support points were first.

4 Conclusion and Outlook

Fixed-support approximations are the arguably most popular approach to the barycenter problem, in particular for exact barycenter computations. In this paper, we devised an integer program, IP (Gen-IP), to generate an optimal support point to add to such a fixed support for a better approximation. We proved that a simpler MIP (Gen-MIP) can be solved instead of IP (Gen-IP), and we designed a branch-and-bound method that uses the symmetry of the underlying problem for a tailored branching strategy. The advantage of the new approach lies in significantly faster computational speed at only slightly larger memory requirement,

or in a significantly smaller memory requirement and setup time while retaining similar speed. It facilitates larger-scale computations than before. We see two practical uses of the methods in this paper. First, one can perform a limited number of iterations to improve on a given approximate barycenter. Second, the ability to find exact barycenters for larger high-dimensional instances than before leads to a more reliable assessment of the quality of solutions returned by algorithms in the literature.

There are a few promising directions for practical and theoretical future work. Commercial solvers such as Gurobi have branch-and-bound routines that are highly optimized, for example allowing for the parallel solution of several subproblems, but do not allow the integration of a tailored branching strategy, as we propose. Our custom implementation is a proof of concept designed to identify a best branching strategy and to identify the impact of presorting measures by size. It would not be hard to improve the custom implementation and close some of the performance gap. We are also planning to compare different options of commercial solvers, and follow their development cycle – additional options for tweaking algorithms are added regularly – for a better transfer of our strategies to a state-of-the-art solver.

We see the most exciting research direction in striving for a deeper understanding of the set of all possible combinations S^*, whose size and scaling is the bottleneck in exact barycenter computations and also for the column generation approach in this work and previous work. The size of S^* drives the exponential scaling of the reduced-cost vector in Borgwardt and Patterson (2020) and determines the size of IP (Gen-IP). In all of these settings, the ability to a priori rule out combinations from S^* that cannot appear in an optimal solution may dramatically improve practical performance. For other combinations, it may be possible to bound the approximation error incurred if left out of consideration. This would be a step to leaving the setting of exact barycenter computations and working towards the efficient computation of improving support points for approximate large-scale computations in higher dimensions.

Acknowledgments

The first author was supported by Air Force Office of Scientific Research grant FA9550-21-1-0233 and NSF grant 2006183, Algorithmic Foundations, Division of Computing and Communication Foundations, and Simons Foundation grant 524210 before.

Biographies

Steffen Borgwardt is a faculty member in the Department of Mathematical and Statistical Sciences at the University of Colorado Denver. His research lies on the intersection of combinatorial optimization, polyhedral theory, and linear programming. He holds a habilitation on *Data Analysis through Polyhedral Theory*, is a lifetime Humboldt fellow, and received a joint EURO Excellence in Practice Award for his work on optimization in land consolidation.

Stephan Patterson is a faculty member in the Department of Mathematics at Louisiana State University Shreveport. He received his Ph.D. in Applied Mathematics from the University of Colorado Denver in 2020. As an applied mathematician, his research areas reach into computer science, optimization, and statistics.

The ideas presented in this paper represent the most recent progress in a multi-year effort to create a practical computational algorithm for computing exact barycenters. We began by exploring linear programming formulations based on the underlying structure of the input measures (published in INFORMS Optimization in 2020). Since then, we explored the use of column generation methods, which led to significant computational improvements, but cannot overcome the underlying exponentiality of the problem as the dimension grows. However, through a shift of this difficulty to a separate pricing problem, now taking the form of a mixed-integer program, we arrived at the most competitive column generation approach for the dynamic generation of exact support points in higher dimensions.

Bibliography

M. Agueh and G. Carlier. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2): 904–924, 2011.

J. Altschuler and E. Boix-Adserà. Wasserstein barycenters can be computed in polynomial time in fixed dimension. *Journal of Machine Learning Research*, 22:1–19, 2021.

J. Altschuler and E. Boix-Adserà. Wasserstein barycenters are NP-hard to compute. *SIAM Journal on Mathematics of Data Science*, 4(1):179–203, 2022.

E. Anderes, S. Borgwardt, and J. Miller. Discrete Wasserstein Barycenters: Optimal Transport for Discrete Data. *Mathematical Methods of Operations Research*, 84(2):389–409, 2016.

M. Beiglböck, P. Henry-Labordere, and F. Penkner. Model-independent bounds for option prices – a mass transport approach. *Finance and Stochastics*, 17(3):477–501, 2013.

J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré. Iterative Bregman Projections for Regularized Transportation Problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

J.-D. Benamou, G. Carlier, and L. Nenna. A Numerical Method to Solve Multi-Marginal Optimal Transport Problems with Coulomb Cost. *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 577–601, 2016.

J.-D. Benamou, G. Carlier, and L. Nenna. Generalized incompressible flows, multi-marginal transport and Sinkhorn algorithm. *Numerische Mathematik 142*, 142:33—-54, 2019.

S. Borgwardt. An LP-based, Strongly Polynomial 2-Approximation Algorithm for Sparse Wasserstein Barycenters. *Operational Research*, 22:1511–1551, 2022.

S. Borgwardt and S. Patterson. Improved Linear Programs for Discrete Barycenters. *INFORMS Journal on Optimization*, 2:14–33, 2020.

S. Borgwardt and S. Patterson. On the Computational Complexity of Finding a Sparse Wasserstein Barycenter. *Journal of Combinatorial Optimization*, 41:736–761, 2021.

S. Borgwardt and S. Patterson. A Column Generation Approach to the Discrete Barycenter Problem. *Discrete Optimization*, 43:100674, 2022.

S. Borgwardt, A. Brieden, and P. Gritzmann. Constrained minimum-k-star clustering and its application to the consolidation of farmland. *Operational Research*, 11(1):1–17, 2011.

G. Carlier and I. Ekeland. Matching for teams. *Economic Theory*, 42(2):397–418, 2010.

G. Carlier, A. Oberman, and E. Oudet. Numerical methods for matching for teams and Wasserstein barycenters. *ESAIM: Mathematical Modeling and Numerical Analysis*, 49(6):1621–1642, 2015.

P-A. Chiaporri, R. McCann, and L. Nesheim. Hedonic price equilibria, stable matching and optimal transport; equivalence, topology and uniqueness. *Economic Theory*, 42(2):317–354, 2010.

S. Cohen, M. Arbel, and M. P. Deisenroth. Estimating barycenters of measures in high dimensions. *eprint arXiv:2007.07105*, 2020.

C. Cotar, G. Friesecke, and C. Klüppelberg. Density functional theory and optimal transportation with Coulomb cost. *Communications on Pure and Applied Mathematics*, 66(4):548–599, 2013.

M. Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Tansportation Distances. In *Advances in Neural Information Processing Systems*, volume 26, pages 2292–2300, 2013.

M. Cuturi and A. Doucet. Fast Computation of Wasserstein Barycenters. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 685–693, 2014.

I. Haasler, R. Singh, Q. Zhang, J. Karlsson, and Y. Chen. Multi-marginal Optimal Transport and Probabilistic Graphical Models. *IEEE Transactions on Information Theory*, 67(7):4647–4668, 2021.

M. Heitz, N. Bonneel, D. Coeurjolly, M. Cuturi, and G. Peyré. Ground Metric Learning on Graphs. *Journal of Mathematical Imaging and Vision*, 63:89–107, 2021.

N. Ho, X. L. Nguyen, M. Yurochkin, H. H. Bui, V. Huynh, and D. Phung. Multilevel clustering via Wasserstein means. In *International Conference on Machine Learning*, pages 1501–1509, 2017.

N. Ho, V. Huynh, D. Phung, and M. Jordan. Probabilistic multilevel clustering via composite transportation distance. In *International Conference on Artificial Intelligence and Statistics, PMLR*, pages 3149–3157, 2019.

A. Jain, Y. Zhong, and M.-P. Dubuisson-Jolly. Deformable template models: A review. *Signal Processing*, 71(2): 109–129, 1998.

H. Janati, T. Bazeille, B. Thirion, M. Cuturi, and A. Gramfort. Multi-subject MEG/EEG source imaging with sparse multi-task regression. *Neuroimage*, 220:116847, 2020a.

H. Janati, M. Cuturi, and A. Gramfort. Debiased Sinkhorn barycenters. In *International Conference on Machine Learning, PMLR*, page 4692–4701, 2020b.

A. Kroshnin, N. Tupitsa, D. Dvinskikh, P. Dvurechensky, A. Gasnikov, and C. Uribe. On the complexity of approximating Wasserstein barycenters. In *International Conference on Machine Learning, PMLR*, pages 3530–3540, 2019.

T. Lin, N. Ho, M. Cuturi, and M. Jordan. Fixed-support Wasserstein barycenters: Computational hardness and fast algorithm. In *Advances in Neural Information Processing Systems*, volume 33, pages 5368–5380, 2020.

G. Luise, S. Salzo, M. Pontil, and C. Ciliberto. Sinkhorn barycenters with free support via Frank-Wolfe algorithm. In *Advances in Neural Information Processing Systems*, volume 32, pages 9322–9333, 2019.

E. Munch, K. Turner, P. Bendich, S. Mukherjee, J. Mattingly, and J. Harer. Probabilistic Fréchet means for time varying persistence diagrams. *Electronic Journal of Statistics*, 9:1173–1204, 2015.

K. Natarajan. *Optimization with Marginals and Moments*. Dynamic Ideas LLC, 2021.

L. Nenna. Numerical Methods for Multi-Marginal Optimal Transportation, 2016. Ph.D. thesis, Université Paris-Dauphine.

V. Panaretos and Y. Zemel. Statistical Aspects of Wasserstein Distances. *Annual Review of Statistics and Its Application*, 6(1):405–431, 2019.

G. Peyré and M. Cuturi. Computational Optimal Transport. *Foundations and Trends in Machine Learning*, 11(5-6): 355–607, 2019.

M. Schmitz, M. Heitz, N. Bonneel, F. Ngolé, D. Coeurjolly, M. Cuturi, G. Peyré, and J.-L. Starck. Wasserstein Dictionary Learning: Optimal Transport-Based Unsupervised Nonlinear Dictionary Learning. *SIAM Journal on Imaging Sciences*, 11(1):643–678, 2018.

Z. Shen, Z. Wang, A. Riebeiro, and H. Hassani. Sinkhorn barycenters via functional gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pages 986–996, 2020.

D. Simon and A. Aberdam. Barycenters of natural images - constrained wasserstein barycenters for image morphing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7907–7916, 2020.

S. P. Singh, A. Hug, A. Dieuleveut, and M. Jaggi. Context mover's distance & barycenters: Optimal transport of contexts for building representations. In *International Conference on Artificial Intelligence and Statistics, PMLR*, pages 3437–3449, 2020.

J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34:1–11, 2015.

A. Trouvé and L. Younes. Local Geometry of Deformable Templates. *SIAM Journal on Mathematical Analysis*, 37(1):17–59, 2005.

C. Villani. *Optimal transport: old and new*, volume 338. Springer-Verlag Berlin Heidelberg, 2009.

H. Xu, W. Wang, W. Liu, and L. Carin. Distilled Wasserstein learning for word embedding and topic modeling. In *Advances in Neural Information Processing Systems 32*, pages 1723–1732, 2018.

Y. Yan, S. Duffner, P. Phutane, A. Berthelier, C. Blanc, C. Garcia, and T. Chateau. 2D Wasserstein Loss for Robust Facial Landmark Detection. *Pattern Recognition*, 116(C):107945, 2021.

J. Ye, Y. Li, Z. Wu, J. Z. Wang, W. Li, and J. Li. Determining gains acquired from word embedding quantitatively using discrete distribution clustering. In *Proceedings of the Association for Computational Linguistics*, pages 1847–1856, 2017.

Y. Zemel and V. M. Panaretos. Fréchet means and Procrustes analysis in Wasserstein space. *Bernoulli*, 25(2): 932–976, 2019.

Appendix A - Constraint matrix of MIP (Gen-MIP)

We exhibit that the constraint matrix of MIP (Gen-MIP) is not totally-unimodular. While it is a $0,1$-matrix, it is not hard to verify any of the well-known criteria that rule out total unimodularity.

For example, consider the submatrix corresponding to variables $\mathbf{z}_{11}, \mathbf{z}_{12}, \mathbf{z}_{21}, \mathbf{z}_{1211}, \mathbf{z}_{1221}$ and to constraints

$$\begin{aligned} \mathbf{z}_{11} + \mathbf{z}_{12} + \cdots &= 1 \\ -\mathbf{z}_{11} + \mathbf{z}_{1211} &\leq 0 \\ -\mathbf{z}_{21} + \mathbf{z}_{1211} &\leq 0 \\ -\mathbf{z}_{12} + \mathbf{z}_{1221} &\leq 0 \\ -\mathbf{z}_{21} + \mathbf{z}_{1221} &\leq 0. \end{aligned}$$

The corresponding coefficient matrix has the form

$$U = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix},$$

which is Eulerian, i.e., it has even row and column sums, but the total sum of elements is 2, and not a multiple of 4. This violates one of the properties of totally-unimodular matrices. Other criteria are easy to check as well. For example, the determinant of U is $\det(U) = -2$.