High-precision computation of Wasserstein barycenters in low dimensions: beyond gridding

Jason Altschuler

NYU, NASC Seminar, September 2021

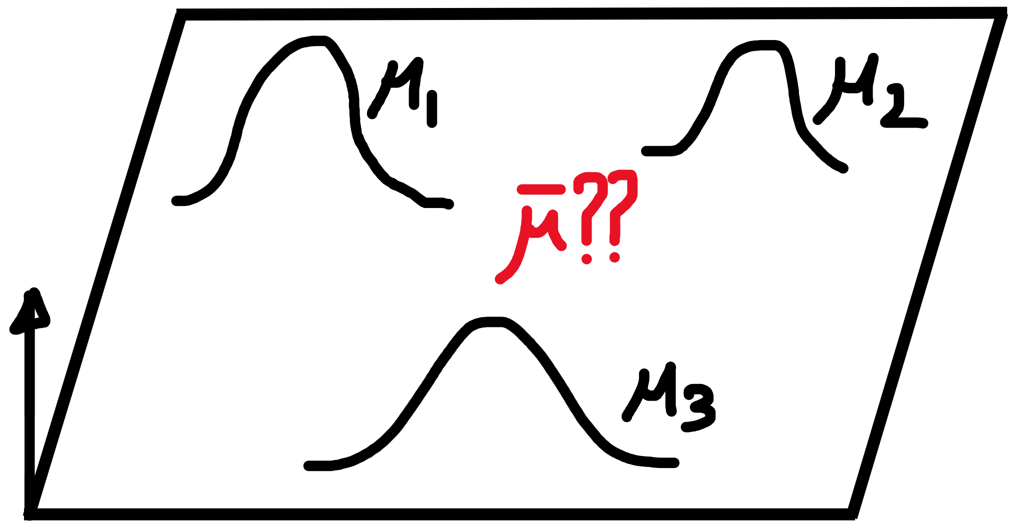
Joint work with my roommate Enric Boix during COVID quarantine

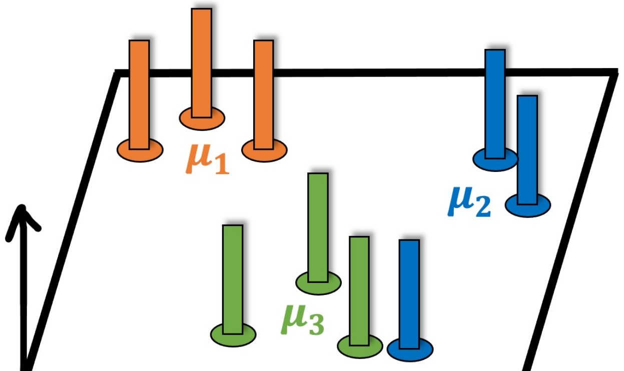
Enric with chocolate & math

Enric's quarantine bread

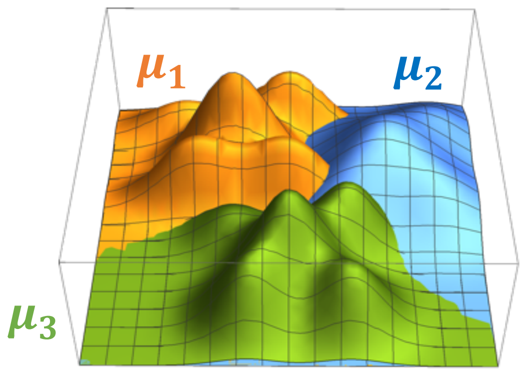
Collaborating on our home whiteboard

Modern challenge: average probability distributions





Ex: point clouds

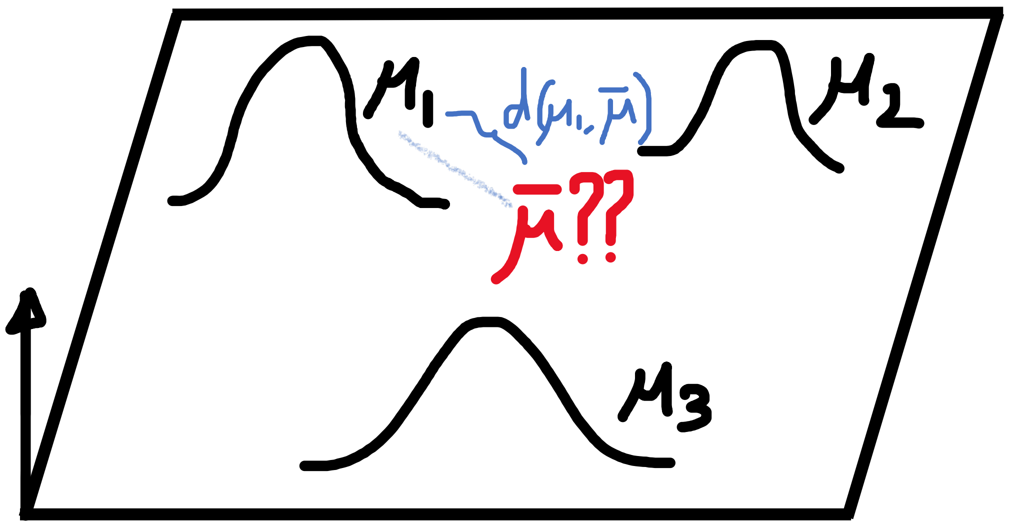


Ex: statistical models

Why average? De-noise, summarize, compute exemplar, interpolate, cluster, …

Why probability distributions? Point clouds in machine learning, posterior distributions in statistics, images in computer vision, object meshes in computer graphics, fMRI scans in neuroscience, …

Modern challenge: average probability distributions



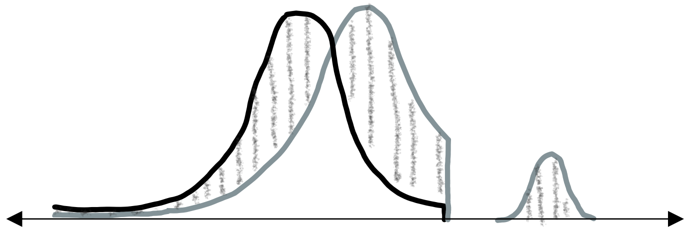
Barycenter: canonical notion of average, given distance

$$\bar{\mu} = \underset{\nu}{\operatorname{argmin}} \sum_{i=1}^{k} \boxed{d^2(\mu_i, \nu)}$$

But how to measure distance?

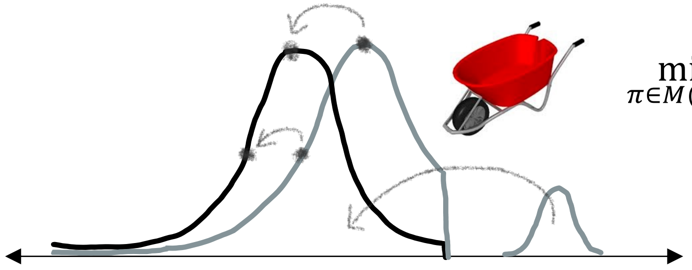
How to measure distance between distributions?

Integrate **vertical distances**



L_p norms, Kullback-Leibler, etc.

Integrate **horizontal distances**

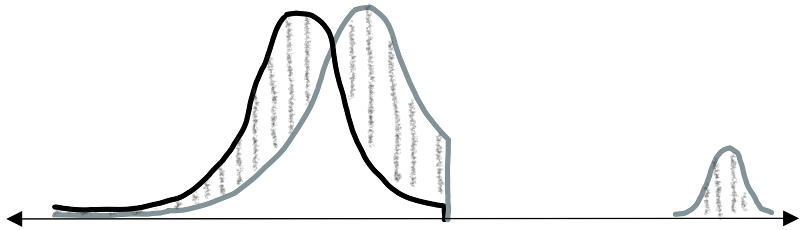


$$\min_{\pi \in M(\mu,\nu)} \int c(x,y) d\pi(x,y)$$

Wasserstein distance (aka Optimal Transport)

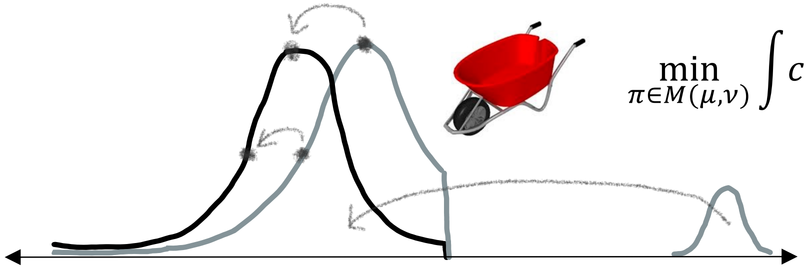
How to measure distance between distributions?

Integrate **vertical distances**



L_p norms, Kullback-Leibler, etc.

Integrate **horizontal distances**

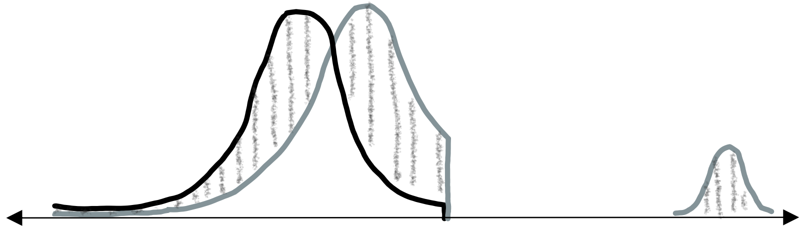


$$\min_{\pi \in M(\mu,\nu)} \int c(x,y) d\pi(x,y)$$

Wasserstein distance (aka Optimal Transport)

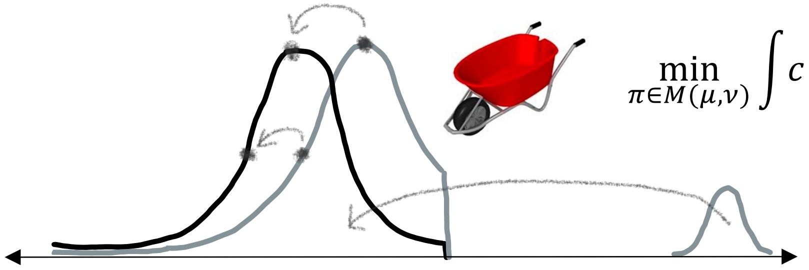
How to measure distance between distributions?

Integrate **vertical distances**



L_p norms, Kullback-Leibler, etc.

Integrate **horizontal distances**

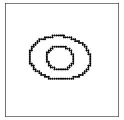
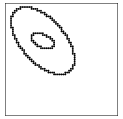
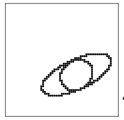


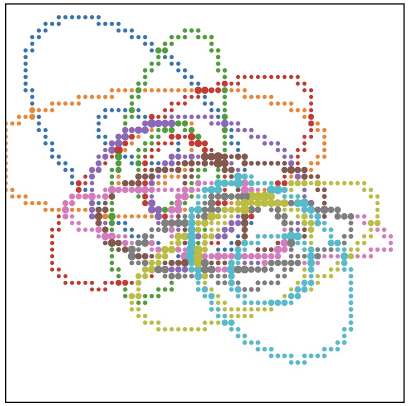
$$\min_{\pi \in M(\mu,\nu)} \int c(x,y)\, d\pi(x,y)$$

Wasserstein distance (aka Optimal Transport)

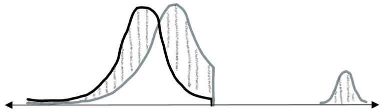
- Captures geometric properties of data!
- But, computation is more difficult…

Wasserstein distance captures the data's geometry

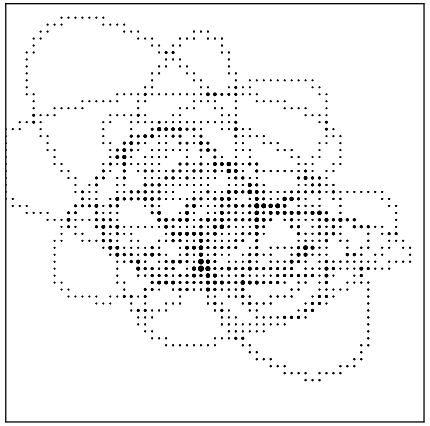
Ex: averaging images of concentric ellipses ( ,  ,  , etc.)



Integrate **vertical distances**

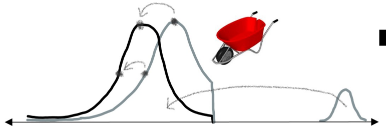


L_p norms, Kullback-Leibler, etc.

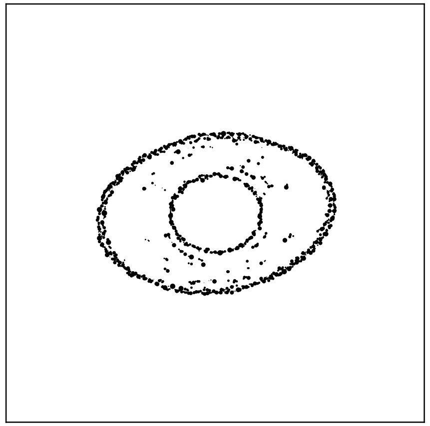


L_2 barycenter

Integrate **horizontal distances**



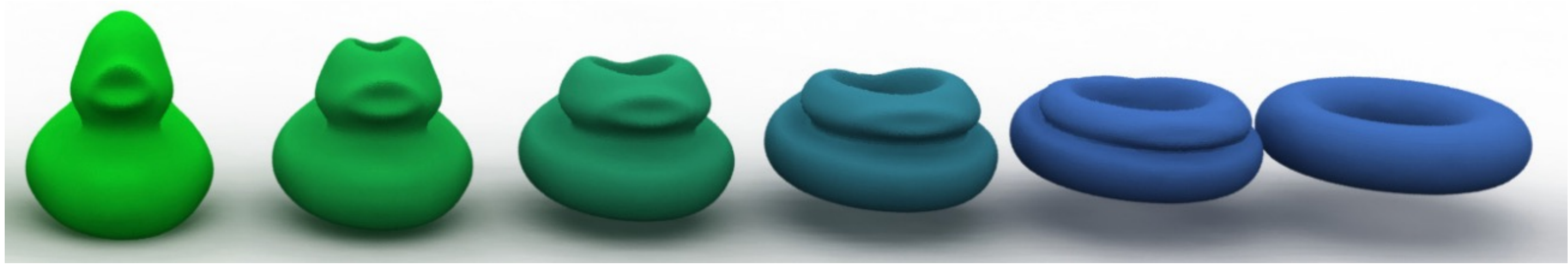
Optimal Transport (a.k.a. Wasserstein distance)



Wasserstein barycenter
(computed with our algorithm)

(points are plotted as disks with area proportional to probability mass)

Wasserstein barycenters, in the wild

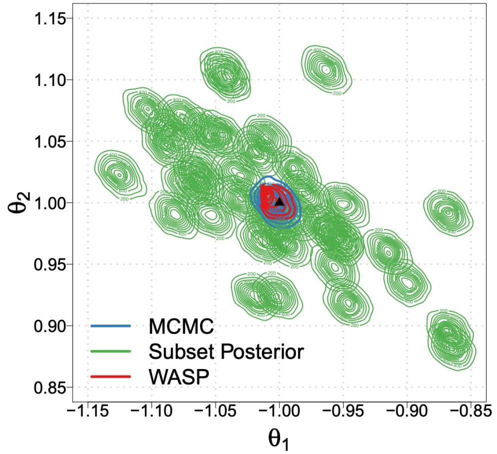


Graphics: **average 3D shapes** to interpolate [Solomon et al 2014, …]

Signal processing: **average spatial sensor measurements** to denoise [Elvander et al 2019, ...]

Probability: **couple distributions** for variance minimization [Knott-Smith 94, Rüschendorf-Uckelmann 02]

And much, much more….



ML: **average data sets** to cluster [Cuturi-Doucet 14, Ho et al 17,…]

ML: **average posterior distributions** to improve Bayesian learning [Srivastava et al 2018, …]

Outline

- Algorithmics
- Techniques
- Outlook (and our general MOT framework)

Outline

- **Algorithmics**
- Techniques
- Outlook (and our general MOT framework)

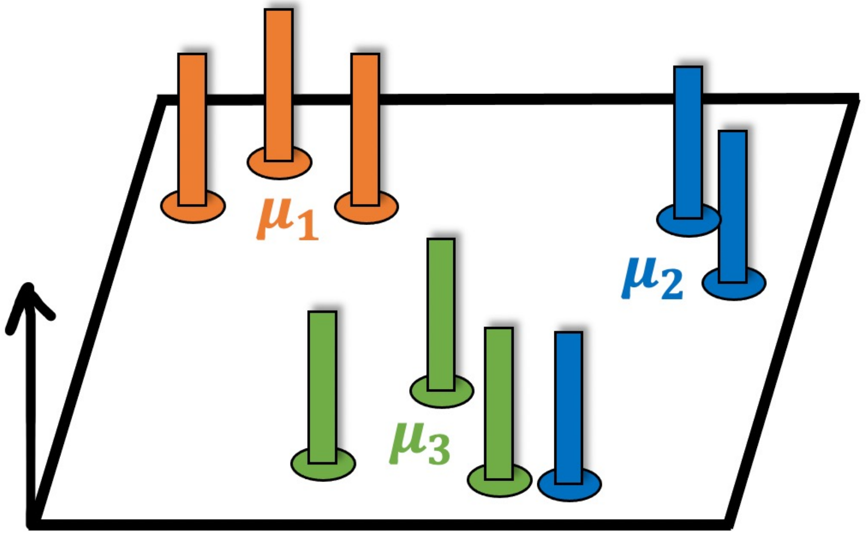
Algorithmics

Task: given ***k*** distributions, each on ***n*** points in $\boldsymbol{R^d}$, find

$$\underset{\text{distribution } \nu}{\operatorname{argmin}} \sum_{i=1}^{k} W^2(\mu_i, \nu)$$

in **poly(k,n,d) time**.

Input size is O(knd)



Ex of input with k=3 distributions, on n=3 points, in dimension d=2

Algorithmics

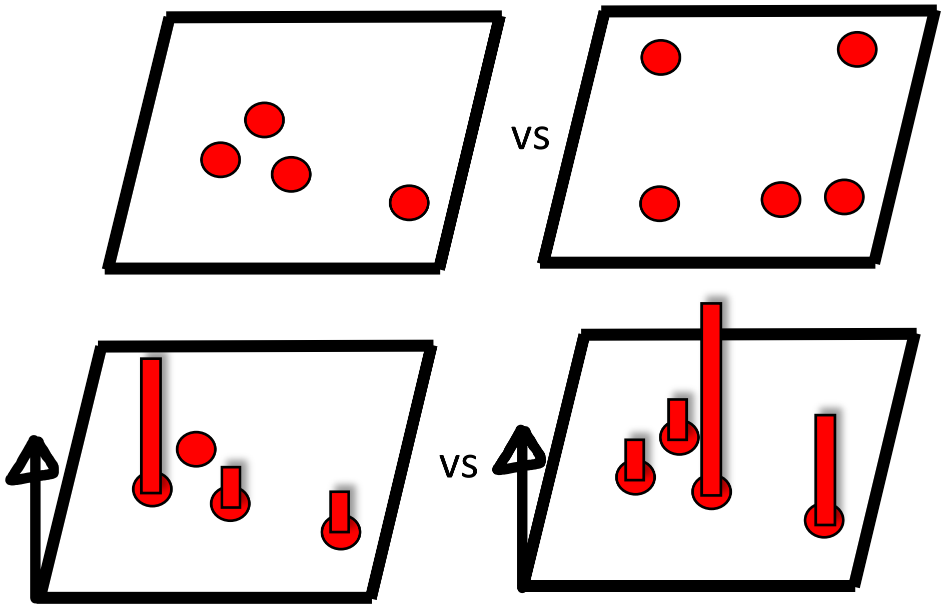
Task: given $\boldsymbol{k}$ distributions, each on $\boldsymbol{n}$ points in $\boldsymbol{R^d}$, find

$$\underset{\text{distribution } \nu}{\operatorname{argmin}} \sum_{i=1}^{k} W^2(\mu_i, \nu)$$

in **poly(k,n,d) time**.

This is joint optimization over both:

1. **Support** ("where")
2. **Mass** ("how much")



Algorithmics

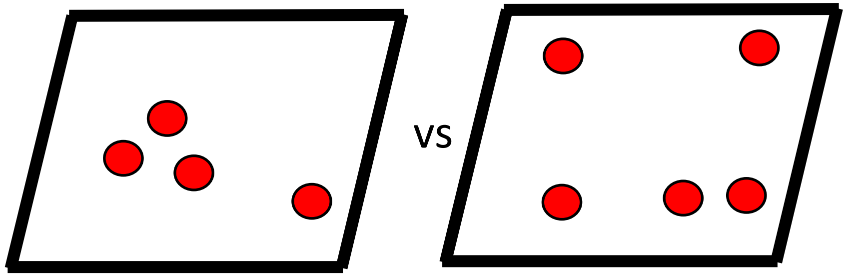
Task: given ***k*** distributions, each on ***n*** points in $\boldsymbol{R^d}$, find

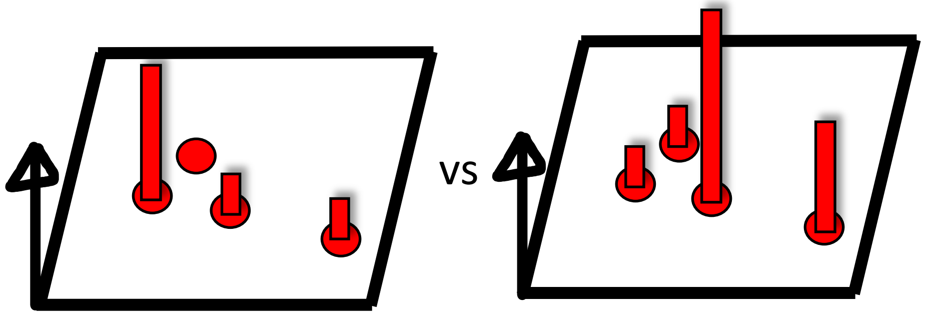
$$\underset{\text{distribution } \nu}{\operatorname{argmin}} \sum_{i=1}^{k} W^2(\mu_i, \nu)$$

in **poly(k,n,d) time**.

This is joint optimization over both:

1. **Support**





Mass: easy (linear program if fixed support)

Algorithmics

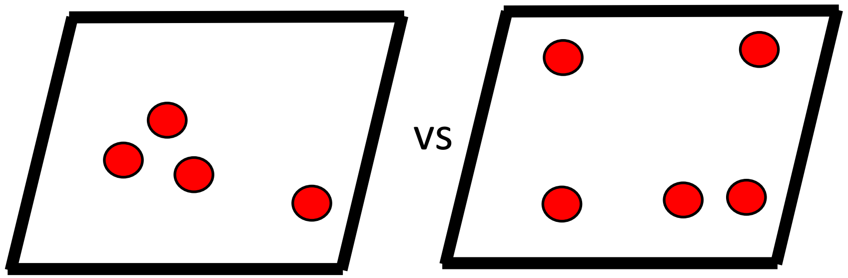
Task: given $\boldsymbol{k}$ distributions, each on $\boldsymbol{n}$ points in $\boldsymbol{R^d}$, find

$$\underset{\text{distribution } \nu}{\operatorname{argmin}} \sum_{i=1}^{k} W^2(\mu_i, \nu)$$

in **poly(k,n,d) time**.

This is joint optimization over both:

ISSUE **Support**



Mass: easy (linear program if fixed support)

Algorithmics

Task: given $\boldsymbol{k}$ distributions, each on $\boldsymbol{n}$ points in $\boldsymbol{R}^{\boldsymbol{d}}$, find

$$\underset{\text{distribution } \nu}{\operatorname{argmin}} \sum_{i=1}^{k} W^2(\mu_i, \nu)$$

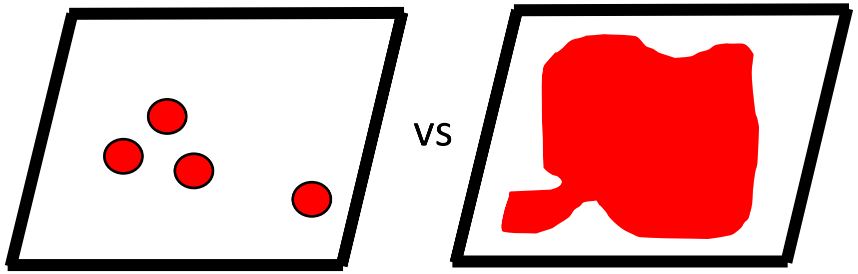
in **poly(k,n,d) time**.

This is joint optimization over both:

ISSUE

Support

➢ Optimization is infinite dimensional



Mass: easy (linear program if fixed support)

Algorithmics

Task: given $\boldsymbol{k}$ distributions, each on $\boldsymbol{n}$ points in $\boldsymbol{R}^{d}$, find

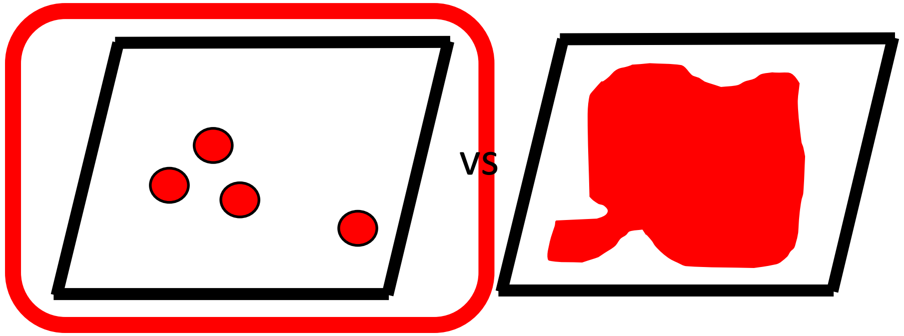
$$\underset{\text{distribution } \nu}{\operatorname{argmin}} \quad \sum_{i=1}^{k} W^2(\mu_i, \nu)$$

in **poly(k,n,d) time**.

This is joint optimization over both:

Support

- Optimization is infinite dimensional
- Exists barycenter with O(nk) support… but **how to find??**



Mass: easy (linear program if fixed support)

Previous algorithms: exponential runtime or heuristics

- **Exponential runtimes in d**

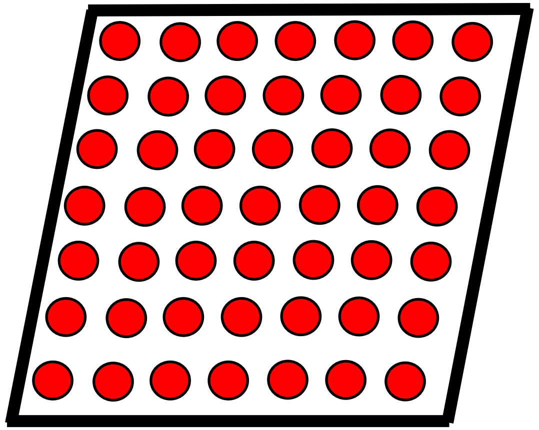
- **Exponential runtime in k**

k = # distributions, n = # points in each, d = dimension

Previous algorithms: exponential runtime or heuristics

- **Exponential runtimes in d**
  - Restrict support to ε-net → approximate answer
  - Runtime factors of $\Omega(\frac{1}{\varepsilon^d})$
  - **Intractable beyond dimension d=3**
  - **Intractable beyond few digits of accuracy**

  [Cuturi-Doucet 14, Solomon et al 15, Benamou et al 15, Carlier et al 15, Staib et al 17, Janati et al 18, Kroshnin et al 19, Shen et al 20, Lin et al 20, Ge et al 20…]



Ex: if ε=1e-5 and d=3, then $\frac{1}{\varepsilon^d} = 10^{15}$

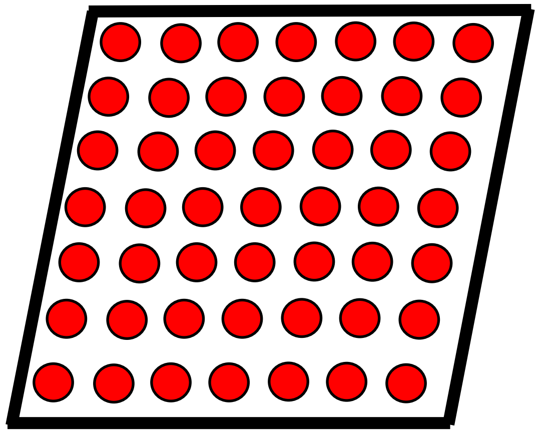
- **Exponential runtime in k**

k = # distributions, n = # points in each, d = dimension

Previous algorithms: exponential runtime or heuristics

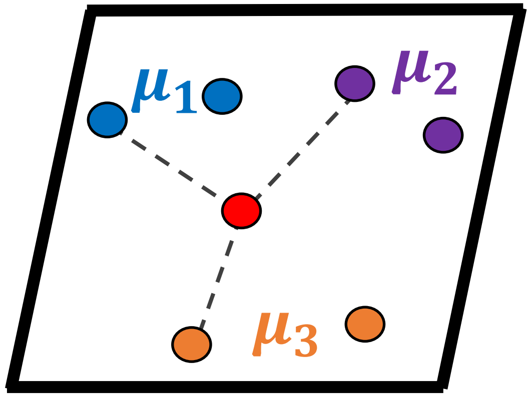
- **Exponential runtimes in d**
  - ➢ Restrict support to ε-net → approximate answer
  - ➢ Runtime factors of $\Omega(\frac{1}{\varepsilon^d})$
  - ➢ **Intractable beyond dimension d=3**
  - ➢ **Intractable beyond few digits of accuracy**

[Cuturi-Doucet 14, Solomon et al 15, Benamou et al 15, Carlier et al 15, Staib et al 17, Janati et al 18, Kroshnin et al 19, Shen et al 20, Lin et al 20, Ge et al 20…]



- **Exponential runtime in k**
  - ➢ Restrict support to special n^k points → exact answer
  - ➢ Runtime factors of $\Omega(\mathrm{n}^{\mathrm{k}})$
  - ➢ **Intractable beyond tiny inputs** (e.g. n=k=10)

[Agueh-Carlier 11, Benamou et al 15, Anderes et al 15, …]



k = # distributions, n = # points in each, d = dimension

Previous algorithms and limitations, circa 2020

Wasserstein Barycenter

- **Fixed-support (aka gridding)** — **Limitation:** LP with $\frac{1}{\varepsilon^d}$ variables
- **Multimarginal OT** — **Limitation:** LP with n^{k} variables
- **Fixed-support 2-approx** — **Limitation:** cannot get better approximation than factor of 2
- **Adaptive fixed-support**
- **Frank-Wolfe approaches**
- **Functional gradient descent**
- **Neural networks**
- **…**

Limitation: no polynomial-time guarantees

So can you compute Wasserstein barycenters or not?

Fixed dimension

Previously: only to a few digits of accuracy due to $\Omega(\frac{1}{\varepsilon^d})$ runtime factors

High dimension

Previously: only for tiny input sizes due to $\Omega(n^k)$ runtime factors

"It is open whether a discrete barycenter can be computed in polynomial time." – Borgwardt 2017

"The [Wasserstein Barycenter problem] is notoriously difficult to solve." – Ho, Lin, Cuturi, Jordan 2020

k = # distributions, n = # points in each, d = dimension

So can you compute Wasserstein barycenters or not?

Fixed dimension

Previously: only to a few digits of accuracy
due to $\Omega(\frac{1}{\varepsilon^d})$ runtime factors

<u>Theorem [AB'20]</u> For any fixed dimension d, can solve exactly in poly(n,k) time.

Explicit runtime is $(nk)^{O(d)}$

Enables computing high-precision solutions
at previously intractable scales.

High dimension

Previously: only for tiny input sizes
due to $\Omega(n^k)$ runtime factors

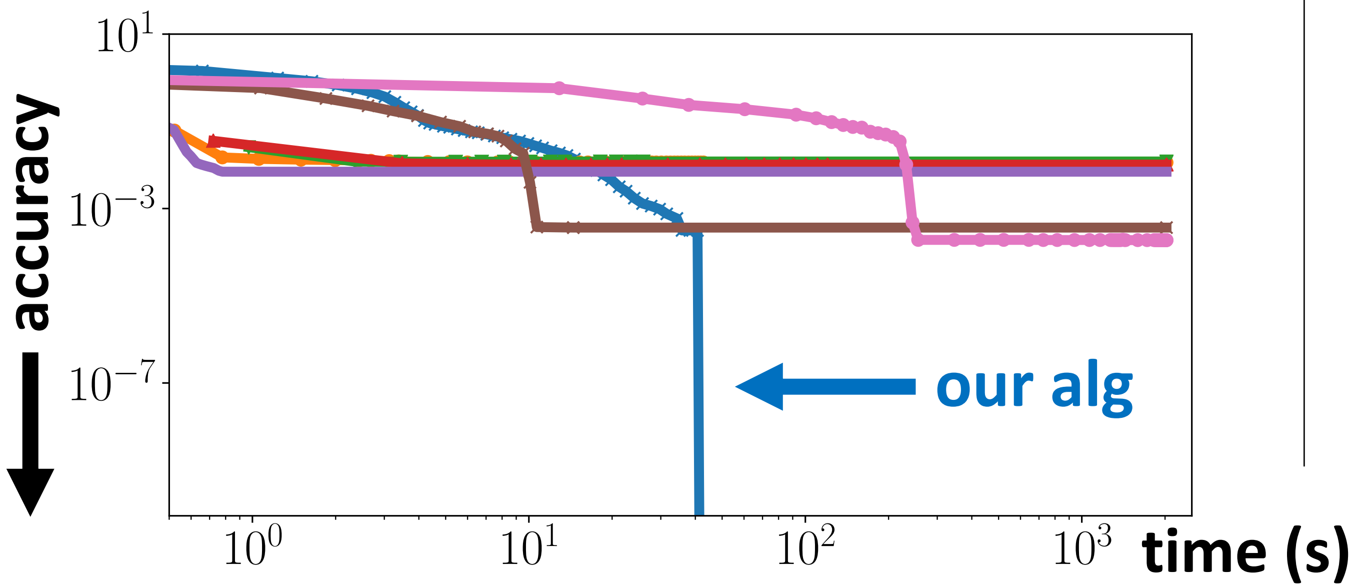
k = # distributions, n = # points in each, d = dimension

So can you compute Wasserstein barycenters or not?

Fixed dimension

Previously: only to a few digits of accuracy
due to $\Omega(\frac{1}{\varepsilon^d})$ runtime factors

Theorem [AB'20] For any fixed dimension d, can solve exactly in poly(n,k) time.



High dimension

Previously: only for tiny input sizes
due to $\Omega(n^k)$ runtime factors

k = # distributions, n = # points in each, d = dimension

So can you compute Wasserstein barycenters or not?

Fixed dimension

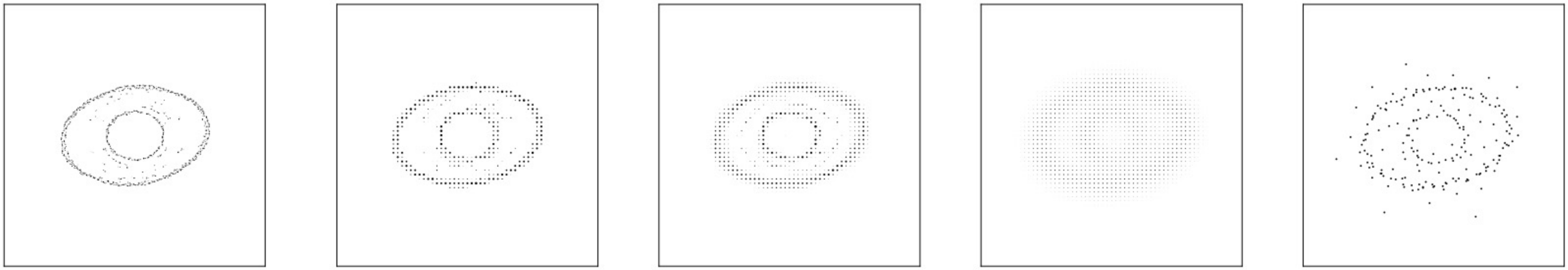
Previously: only to a few digits of accuracy due to $\Omega(\frac{1}{\varepsilon^d})$ runtime factors

<u>Theorem [AB'20]</u> For any fixed dimension d, can solve exactly in poly(n,k) time.

High dimension

Previously: only for tiny input sizes due to $\Omega(n^k)$ runtime factors

Ours | MAAIPM [19] | Debiased [23] | IBP [33] | Frank-Wolfe [28]



Cost: 0.2666 (exact) | Cost: 0.2671 | Cost: 0.2675 | Cost: 0.2723 | Cost: 0.2790

So can you compute Wasserstein barycenters or not?

Fixed dimension

Previously: only to a few digits of accuracy due to $\Omega(\frac{1}{\varepsilon^d})$ runtime factors

Theorem [AB'20] For any fixed dimension d, can solve exactly in poly(n,k) time.

Enables computing high-precision solutions at previously intractable scales.

Solution also has sparse support $(\leq nk)$. Enables fast downstream computation.

High dimension

Previously: only for tiny input sizes due to $\Omega(n^k)$ runtime factors

k = # distributions, n = # points in each, d = dimension

So can you compute Wasserstein barycenters or not?

Fixed dimension

Previously: only to a few digits of accuracy
due to $\Omega(\frac{1}{\varepsilon^d})$ runtime factors

Theorem [AB'20] For any fixed dimension d, can solve exactly in poly(n,k) time.

Enables computing high-precision solutions at previously intractable scales.

High dimension

Previously: only for tiny input sizes
due to $\Omega(n^k)$ runtime factors

Theorem [AB'21] Unless P=NP, there is no poly(n,k,d) time algorithm.

Robust phenomenon: hardness extends to approximate computation, seemingly simple cases, and other Optimal Transport metrics.

k = # distributions, n = # points in each, d = dimension

So can you compute Wasserstein barycenters or not?

Fixed dimension

Previously: only to a few digits of accuracy due to $\Omega(\frac{1}{\varepsilon^d})$ runtime factors

Theorem [AB'20] For any fixed dimension d, can solve exactly in poly(n,k) time.

Enables computing high-precision solutions at previously intractable scales.

High dimension

Previously: only for tiny input sizes due to $\Omega(n^k)$ runtime factors

Theorem [AB'21] Unless P=NP, there is no poly(n,k,d) time algorithm.

Robust phenomenon: hardness extends to approximate computation, seemingly simple cases, and other Optimal Transport metrics.

Resolves the computational complexity of Wasserstein barycenters

Uncovers "curse of dimensionality" for Wasserstein barycenters that doesn't occur for Wasserstein distance (aka, for comparing k distributions rather than just 2)

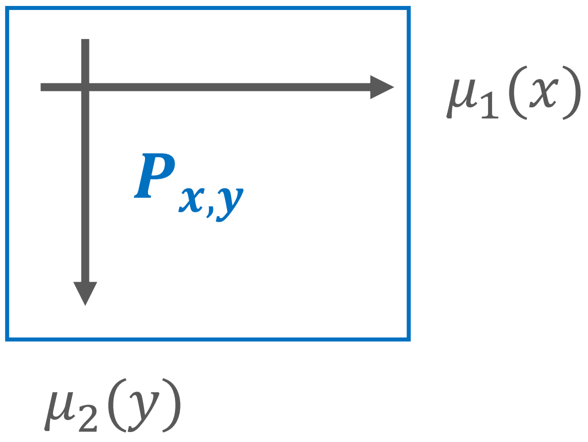
Outline

- Algorithmics
- **Techniques**
- Outlook (and our general MOT framework)

Background: LP reformulation (1/3)

- **Transportation polytope** is the set of matrices with fixed row/col sums

$$M(\mu_1, \mu_2) = \{ P \in R_{\geq 0}^{n \times n} : \sum_y P_{x,y} = \mu_1(x), \sum_x P_{x,y} = \mu_2(y) \}$$

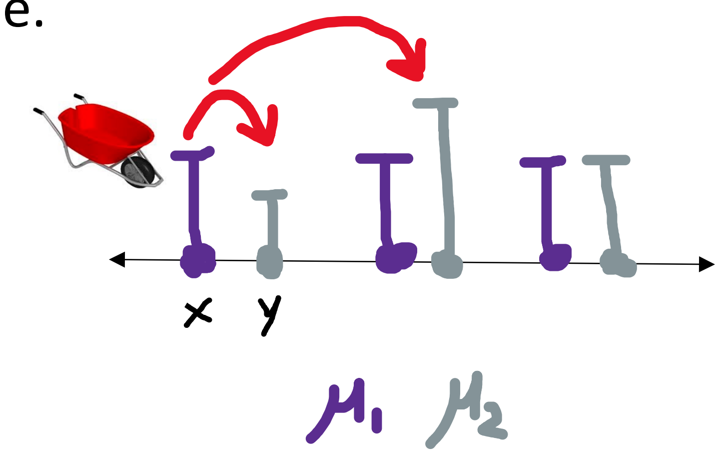


- **Note:** n^2 variables and 2n equality constraints

- **Aside:** Wasserstein distance is a linear program over this polytope, i.e.

$$\mathrm{W}^2(\mu_1, \mu_2) = \min_{\mathrm{P} \in M(\mu_1, \mu_2)} \sum_{x,y} P_{x,y} C_{x,y}$$

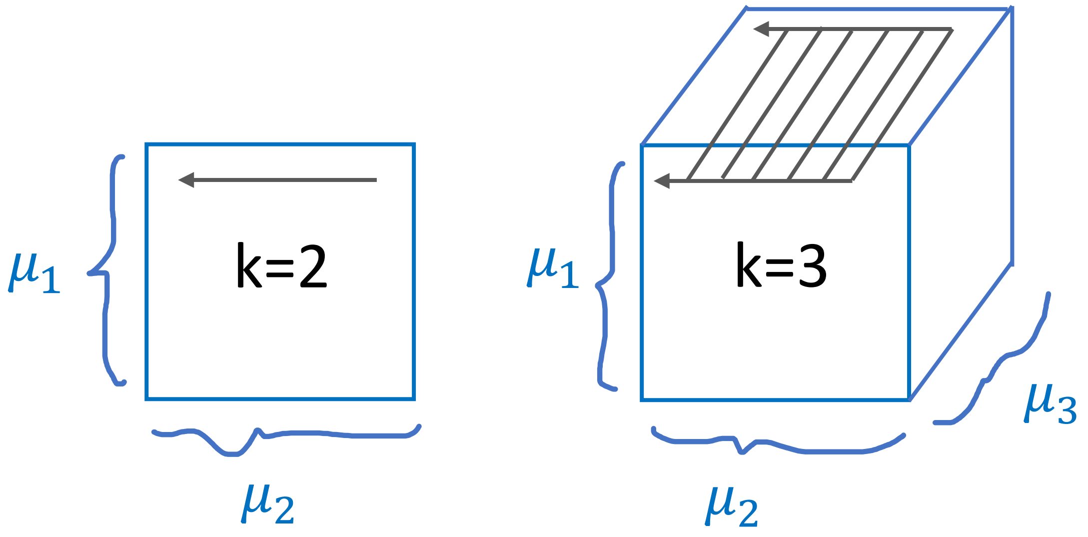
with cost $C_{x,y} = \|x - y\|^2$



Background: LP reformulation (2/3)

- **Multimarginal transportation polytope** is the set of tensors with fixed marginals

$$M(\mu_1, \dots, \mu_k) = \{\, P \in (R^n_{\geq 0})^{\otimes k} : m_i(P) = \mu_i \,\}$$



Note: n^k variables and nk equality constraints

Background: LP reformulation (3/3)

- **Multimarginal Optimal Transport** is linear programming over the transportation polytope

$$\min_{P \in M(\mu_1,\dots,\mu_k)} \sum_{x_1,\dots,x_k} P_{x_1,\dots,x_k} C_{x_1,\dots,x_k}$$

- **Lemma [AC'11].** Wasserstein Barycenter $\min_{\nu} \sum_{i=1}^{k} W^2(\mu_i, \nu)$ equals MOT with cost

$$C_{x_1,\dots,x_k} = \min_{y \in R^d} \sum_{i=1}^{k} \|x_i - y\|^2$$

- **Corollary.** Can restrict barycenter support to n^k points.

- **Key issue: this LP reformulation has n^k variables.**
  - n^k is humongous (e.g., k=100 images)
  - Can't even store cost C or solution P. And even if you could, can't solve…
  - Key obstacle for all previous attempts at polynomial-time algorithms…

Classical

1. LP re-formulation

Wasserstein barycenter ⟷ **MOT with barycenter cost**

Pro: finite size LP
Con: n^{k} variables

Steps for solution

2. Implicit LP ideas

⟷ **Separation oracle for dual MOT LP**

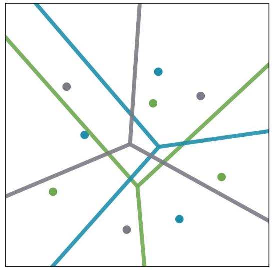
Pro: combinatorial opt
Con: n^{k} possibilities

3. Computational geometry & computational complexity ideas

Poly-time in fixed dim
[Intersect power diagrams]

NP-hard in high dim
[Reduce from Clique]

Key algorithmic insight: MOT is not a generic LP. Can solve separation oracle efficiently by exploiting the structure of low-dimensional power diagrams.



[AB1] A. & Boix, *Wasserstein barycenters can be computed in polynomial time in fixed dimension*. Journal of Machine Learning Research, 2021.
[AB2] A. & Boix, *Wasserstein barycenters are NP-hard to compute*. SIAM Journal on Mathematics of Data Science, 2021.

Classical

1. LP re-formulation

Wasserstein barycenter ⟷ **MOT with barycenter cost**

Pro: finite size LP
Con: n^k variables

Steps for solution

2. Implicit LP ideas

⟷ **Separation oracle for dual MOT LP**

Pro: combinatorial opt
Con: n^k possibilities

3. Computational geometry & computational complexity ideas

Poly-time in fixed dim
[Intersect power diagrams]

NP-hard in high dim
[Reduce from Clique]

This is an illustrative application of our **general framework for "structured" MOT problems**

MOT with cost C ⟷ **Separation oracle for dual MOT LP, with cost C**

- We show (in)tractability for general classes of costs C
- Beyond geometric structure: graphical / combinatorial / low-rank structure, etc.
- Diverse applications in the sciences, stay tuned…

Classical

1. LP re-formulation

Wasserstein barycenter ⟷ **MOT with barycenter cost**

Pro: finite size LP
Con: n^k variables

Steps for solution

2. Implicit LP ideas

next ⟷ **Separation oracle for dual MOT LP**

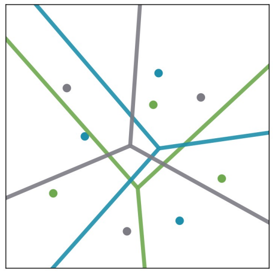
Pro: combinatorial opt
Con: n^k possibilities

3. Computational geometry & computational complexity ideas

Poly-time in fixed dim
[Intersect power diagrams]

NP-hard in high dim
[Reduce from Clique]

Key algorithmic insight: MOT is not a generic LP. Can solve separation oracle efficiently by exploiting the structure of low-dimensional power diagrams.



[AB1] A. & Boix, *Wasserstein barycenters can be computed in polynomial time in fixed dimension*. Journal of Machine Learning Research, 2021.
[AB2] A. & Boix, *Wasserstein barycenters are NP-hard to compute*. SIAM Journal on Mathematics of Data Science, 2021.

Solving MOT in poly(n,k) time seems impossible…

Obvious obstacle is n^k variables.

$$\boxed{\min_{P \in M(\mu_1,\ldots,\mu_k)} \sum_{x_1,\ldots,x_k} P_{x_1,\ldots,x_k} C_{x_1,\ldots,x_k}}$$

- Can't even write down the cost C or solution P.
- Let alone run standard LP solvers, Sinkhorn, etc. since they all have $n^{\Omega(k)}$ runtime.

Dual LP has *nk* variables…

$$\boxed{\max_{p_1,\ldots,p_k \in R^n} \sum_{i=1}^k \langle p_i, \mu_i \rangle \quad \text{s.t.} \quad \sum_{i=1}^k [p_i]_{x_i} \le C_{x_1,\ldots,x_k} \quad \text{for all } x_1, \ldots, x_k}$$

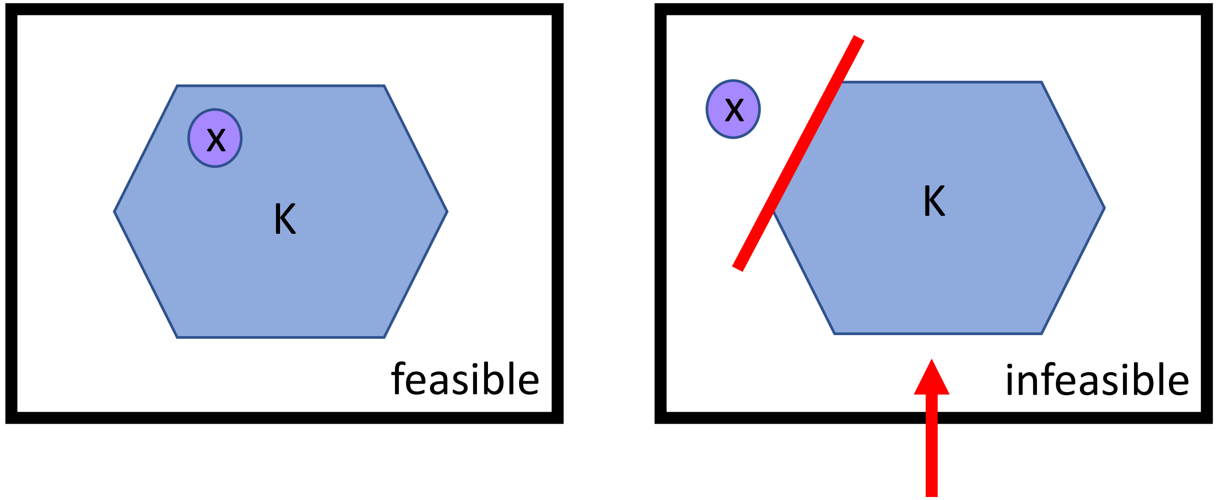
But it still has n^k constraints. [Dualizing swaps exponential primal variables -> dual constraints]

k = # distributions, n = # points in each, d = dimension

Feasible sets are sometimes simple even if many constraints

- **Theorem [K'80,GLS'81]:** Can solve convex optimization in N variables in poly(N) time if there exists poly(N) time implementation of separation oracle for its feasibility set.

- **Key point:** independent of # constraints.



For dual MOT LP, **K has exponentially many facets.**
Key Q: can you find a violated constraint efficiently?

But... does this result apply to the dual MOT LP?

- **Theorem [K'80,GLS'81]:** Can solve convex optimization in N variables in poly(N) time if there exists poly(N) time implementation of separation oracle for its feasibility set.

Key issue: efficient separation oracle?

- ➢ No for general MOT costs!
- ➢ Must exploit "structure" of the relevant MOT cost, stay tuned...

Other important technical issues:

- ➢ For algorithms: can we recover primal MOT solution? Yes [AB1]
- ➢ For hardness: can we show inapproximability? Yes [AB2]

Classical

1. LP re-formulation

Wasserstein barycenter ↔ **MOT with barycenter cost**

Pro: finite size LP
Con: n^{k} variables

Steps for solution

2. Implicit LP ideas

↔ **Separation oracle for dual MOT LP**

Pro: combinatorial opt
Con: n^{k} possibilities

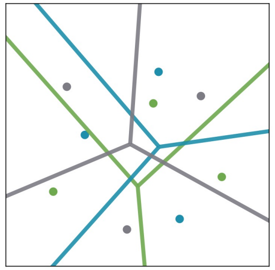
3. Computational geometry & computational complexity ideas

next

Poly-time in fixed dim
[Intersect power diagrams]

NP-hard in high dim
[Reduce from Clique]

Key algorithmic insight: MOT is not a generic LP. Can solve separation oracle efficiently by exploiting the structure of low-dimensional power diagrams.



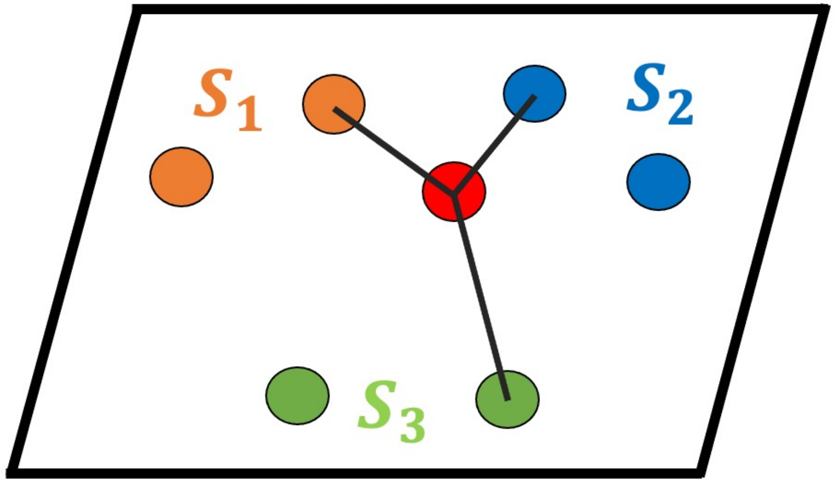
[AB1] A. & Boix, *Wasserstein barycenters can be computed in polynomial time in fixed dimension*. Journal of Machine Learning Research, 2021.
[AB2] A. & Boix, *Wasserstein barycenters are NP-hard to compute*. SIAM Journal on Mathematics of Data Science, 2021.

Separation oracle (re-interpreted & simplified)

Input: k sets of n points in R^d

Compute: $\min_{x_1 \in S_1, \ldots, x_k \in S_k} \min_{y \in R^d} \sum_{i=1}^{k} ||x_i - y||^2$

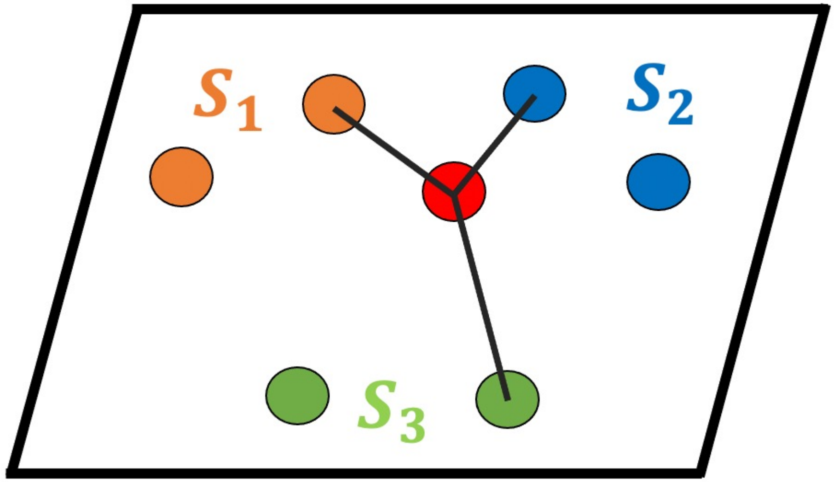
One point per set   Hub



How to solve the separation oracle?

Input: k sets of n points in R^d

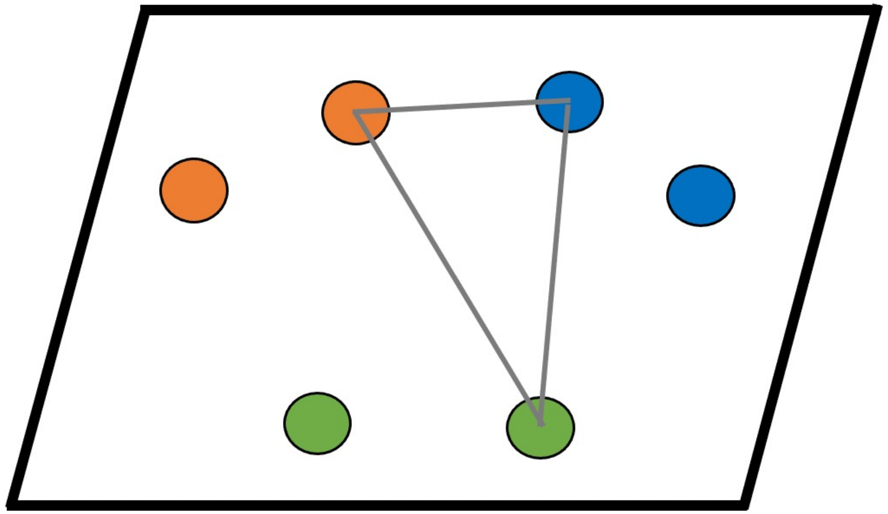
Compute: $$\min_{x_1 \in S_1, \ldots, x_k \in S_k} \min_{y \in R^d} \sum_{i=1}^{k} ||x_i - y||^2$$



Natural approach: first solve for y.
Problem becomes: find k closest points, 1 per set.

$$\min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i,j=1}^{k} \left\| x_i - x_j \right\|^2$$

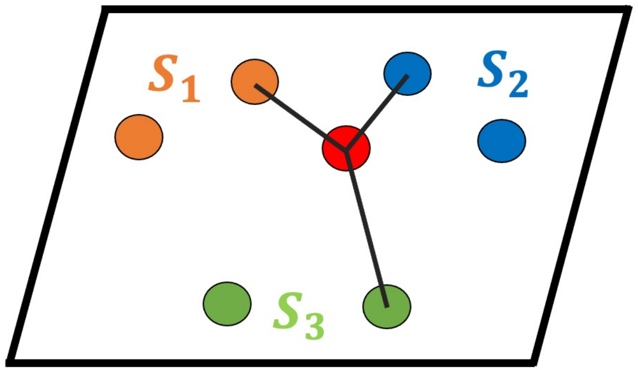
But n^k choices, unclear how to solve efficiently…



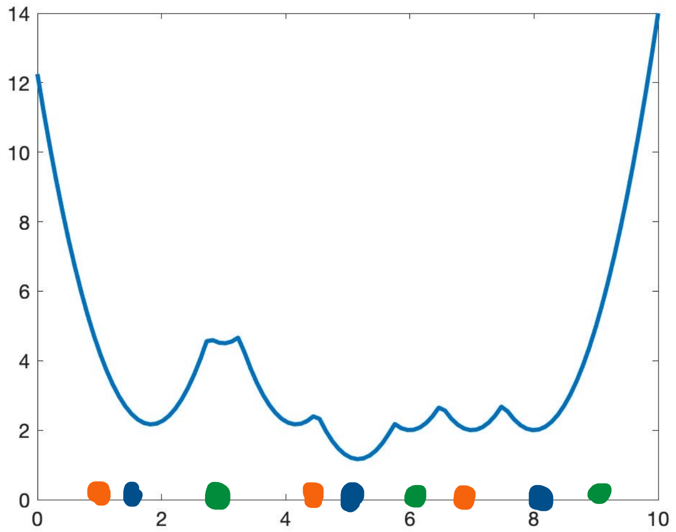
Poly(n,k) time algorithm in fixed dimension

Compute: $\min_{y \in R^d} \min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$

Easy given y: take closest point to y.



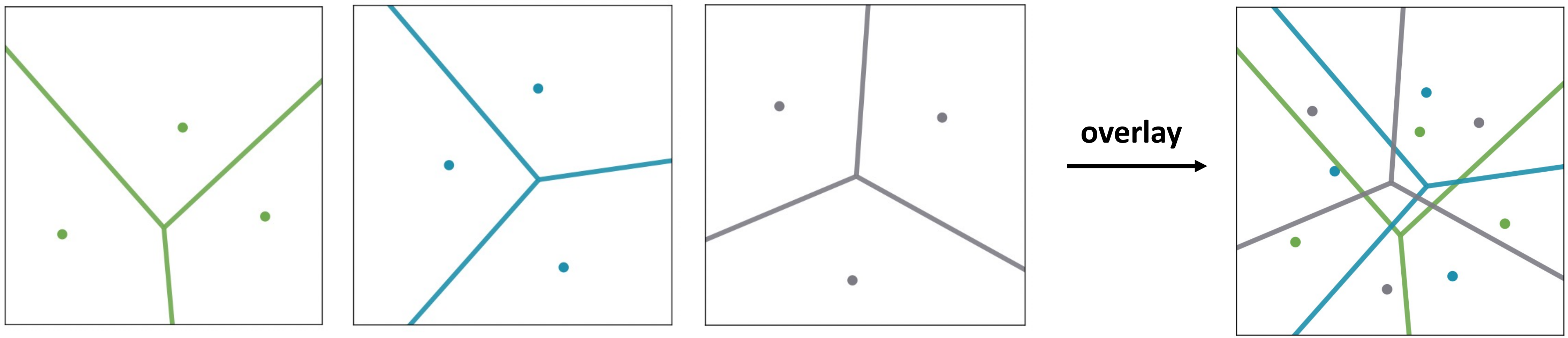
- **But, how to optimize nonconvex F(y) over $y \in R^d$?**
  - Piecewise convex on finitely many convex domains ("pieces")
  - Naive bound is n^k pieces (1 piece per tuple $x_1, \ldots, x_k$)



- **Key lemma:** For fixed d, only poly(n,k) pieces!
- **Algorithm:** Enumerate pieces. Easily optimize y on each piece. Return best.

Key lemma: $F(y) = \min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

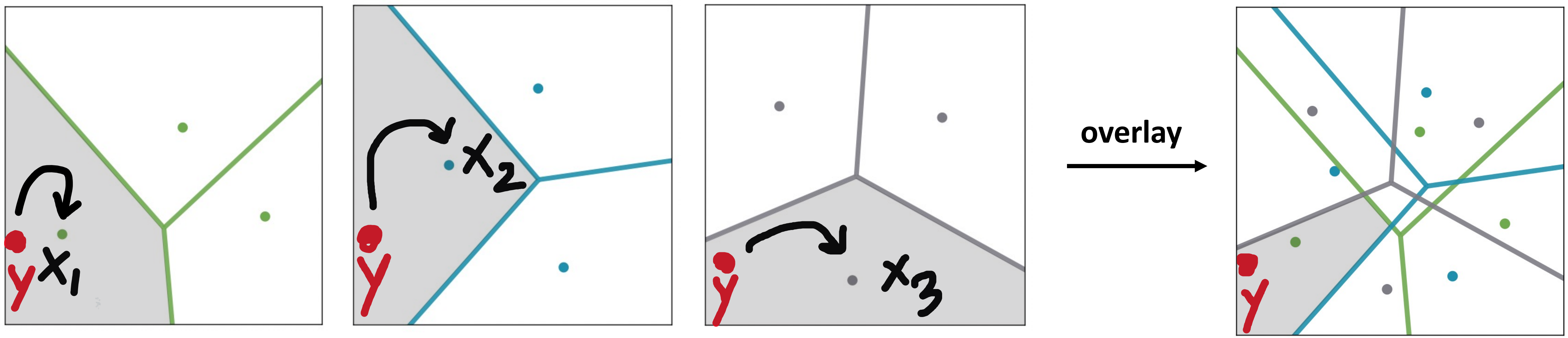
Voronoi diagram for each set S_i.



k = # sets, n = # points per set, d = dimension (fixed here)

Key lemma: $F(y) = \min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

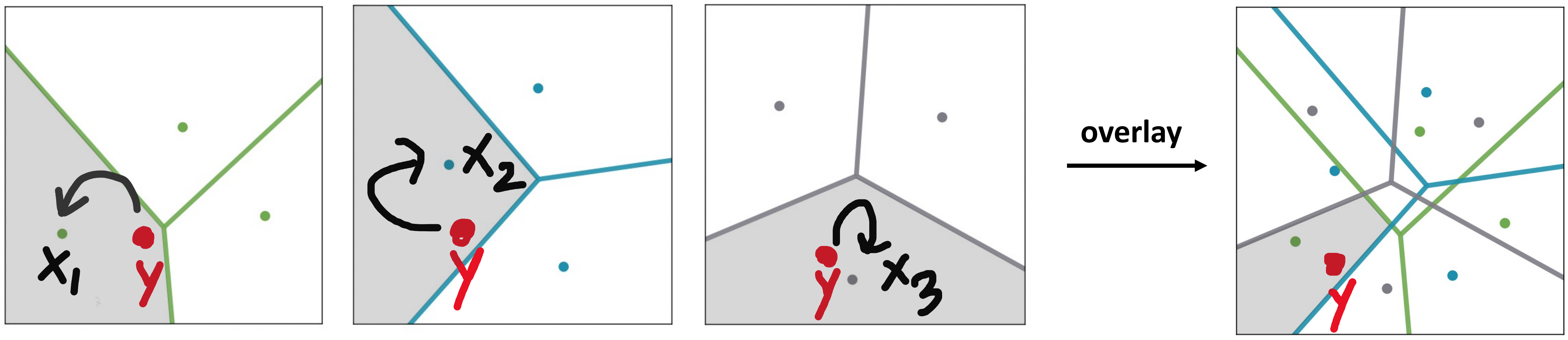
Voronoi diagram for each set S_i.



k = # sets, n = # points per set, d = dimension (fixed here)

Key lemma: $F(y) = \min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

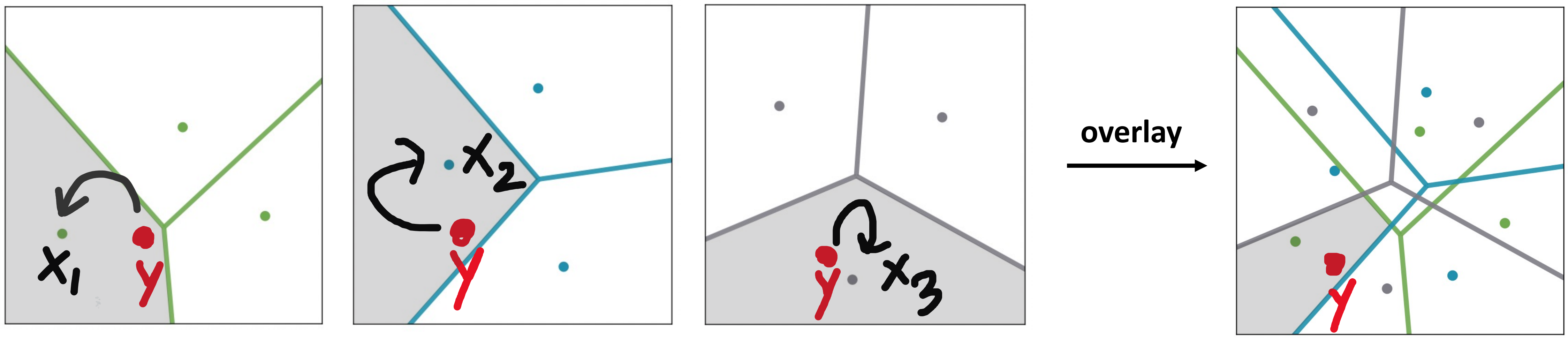
Voronoi diagram for each set S_i.



k = # sets, n = # points per set, d = dimension (fixed here)

Key lemma: $F(y) = \min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

Voronoi diagram for each set S_i.



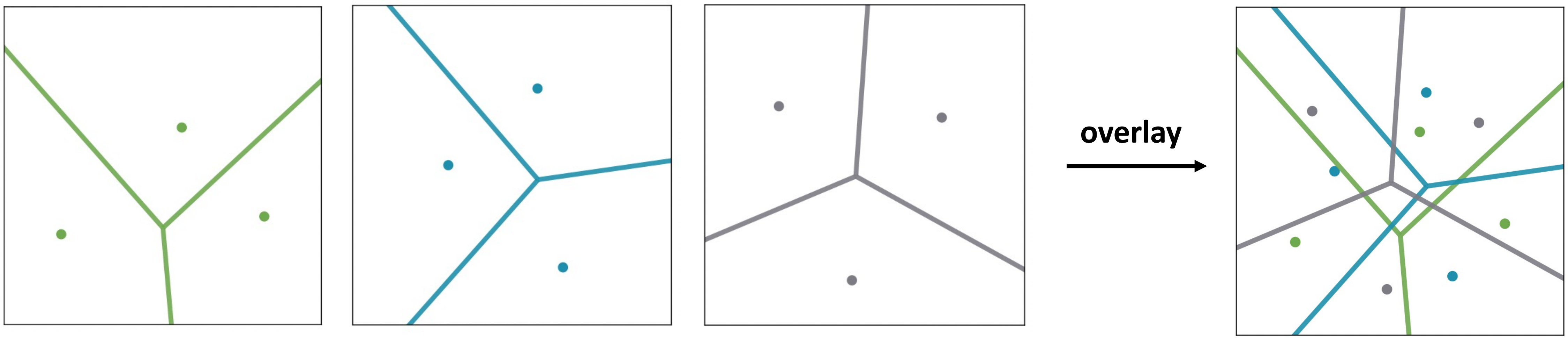
Proof. As y varies in overlay cell, the $x_1, \ldots, x_k$ are fixed.

k = # sets, n = # points per set, d = dimension (fixed here)

Key lemma: $F(y) = \min_{x_1 \in S_1, \dots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

Voronoi diagram for each set S_i.

13 cells. Less than $n^k = 3^3 = 27$!



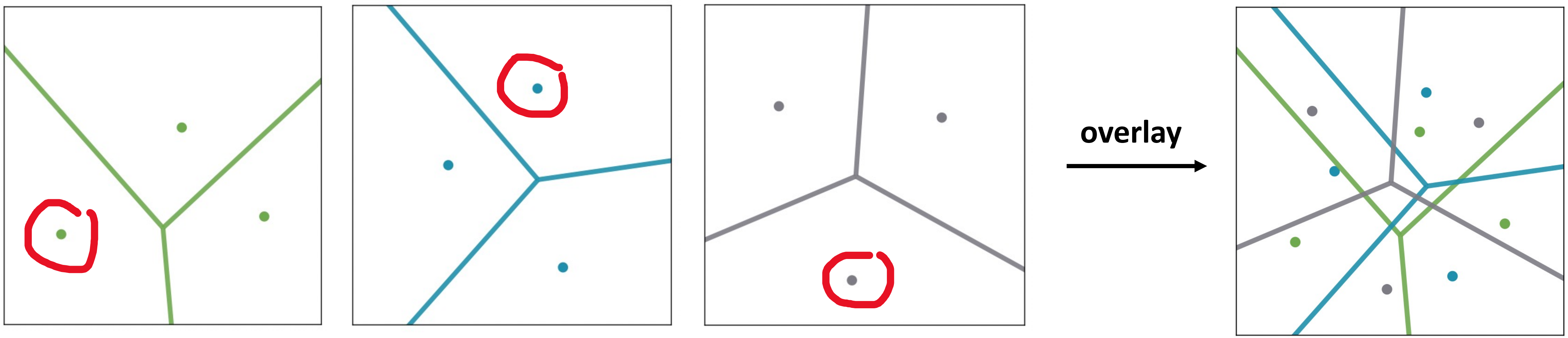
Proof. As y varies in overlay cell, the $x_1, \dots, x_k$ are fixed. So, 1 piece per **nonempty** overlay cell. How many?

k = # sets, n = # points per set, d = dimension (fixed here)

Key lemma: $F(y) = \min_{x_1 \in S_1, \dots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

Voronoi diagram for each set S_i.

13 cells. Less than $n^k = 3^3 = 27$!

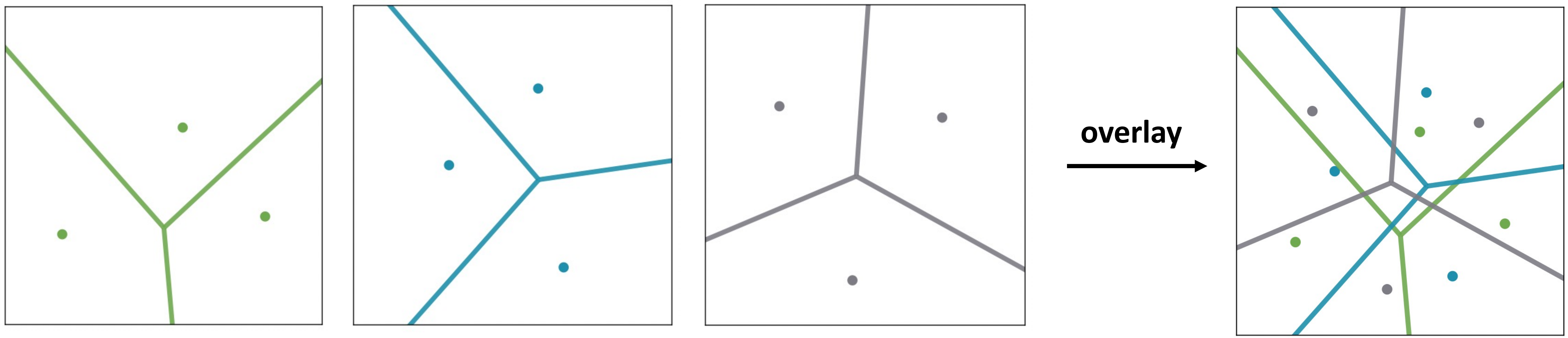


Proof. As y varies in overlay cell, the $x_1, \dots, x_k$ are fixed. So, 1 piece per **nonempty** overlay cell. How many?

k = # sets, n = # points per set, d = dimension (fixed here)

Key lemma: $F(y) = \min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

Voronoi diagram for each set S_i.



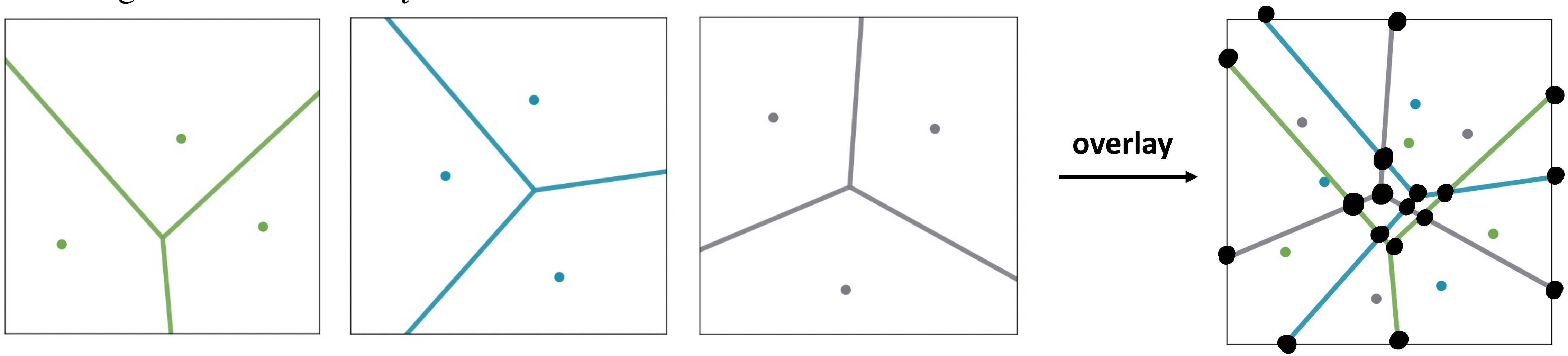
Proof. As y varies in overlay cell, the $x_1, \ldots, x_k$ are fixed. So, 1 piece per **nonempty** overlay cell. How many?

- Each Voronoi diagram is partition of R^d defined by at most $\binom{n}{2}$ hyperplanes.
- Overlaying k Voronoi diagrams unions at most $t = k\binom{n}{2}$ hyperplanes.
- **Q: How many cells are in an arrangement of hyperplanes? [Schläfli 1901]**

k = # sets, n = # points per set, d = dimension (fixed here)

Key lemma: $F(y) = \min_{x_1 \in S_1, \ldots, x_k \in S_k} \sum_{i=1}^{k} ||x_i - y||^2$ has poly(n,k) pieces.

Voronoi diagram for each set S_i.



Proof. As y varies in overlay cell, the $x_1, \ldots, x_k$ are fixed. So, 1 piece per **nonempty** overlay cell. How many?

- Each Voronoi diagram is partition of R^d defined by at most $\binom{n}{2}$ hyperplanes.
- Overlaying k Voronoi diagrams unions at most $t = k\binom{n}{2}$ hyperplanes.
- **Q: How many cells are in an arrangement of hyperplanes? [Schläfli 1901]**
- **A**: In R^2, the arrangement defines planar graph with:
    - Edges: $\mathrm{e} \leq t^2$
    - Faces: $f = e - v + 2 \leq \mathrm{t}^2 + 2 =$ **poly(n, k)**

In R^d, the bound is $\sum_{i=0}^{d} \binom{t}{i} = (\mathrm{nk})^{O(d)}$

k = # sets, n = # points per set, d = dimension (fixed here)

Classical

1. LP re-formulation

Wasserstein barycenter ⟷ **MOT with barycenter cost**

Pro: finite size LP
Con: n^{k} variables

Steps for solution

2. Implicit LP ideas

⟷ **Separation oracle for dual MOT LP**

Pro: combinatorial opt
Con: n^{k} possibilities

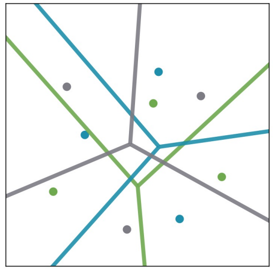
3. Computational geometry & computational complexity ideas

Poly-time in fixed dim
[Intersect power diagrams]

NP-hard in high dim
[Reduce from Clique]

Skipped today

Key algorithmic insight: MOT is not a generic LP. Can solve separation oracle efficiently by exploiting the structure of low-dimensional power diagrams.



[AB1] A. & Boix, *Wasserstein barycenters can be computed in polynomial time in fixed dimension*. Journal of Machine Learning Research, 2021.
[AB2] A. & Boix, *Wasserstein barycenters are NP-hard to compute*. SIAM Journal on Mathematics of Data Science, 2021.

Outline

- Algorithmics
- Techniques
- **Outlook (and our general MOT framework)**

Can you solve other MOT problems in poly(n,k) time?

- Remarkably, there are poly-time algorithms for several other MOT problems:
  - Generalized Euler flows [BCCNP'15], tree-structured costs [HRCK'20], …

- But, specially-tailored techniques. Unclear if extend to other applications.

- **Q: Are there general "structural" properties that make MOT solvable in poly(n,k) time?**

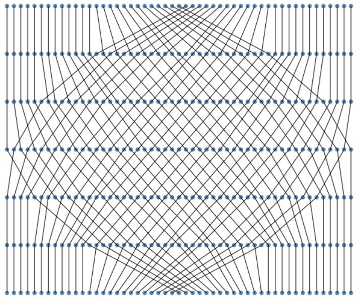
- **We identify general classes of costs C for which MOT is tractable [AB3]**
  - ➢ Leads to **first polynomial-time algorithms** for many problems thought to take exponential time.
  - ➢ Leads to **first high-precision algorithms** for all problems known to be polynomial-time solvable.

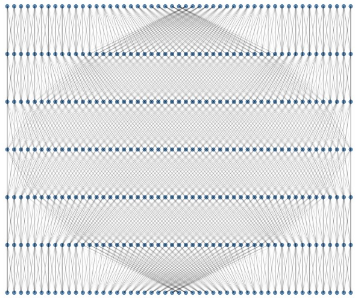
- **We show first rigorous NP-hardness for MOT problems [AB4]**
  - ➢ **Guides the algorithmic search** by showing necessity of the structures exploited by [AB3]

[AB3] A. & Boix, *Polynomial-time algorithms for Multimarginal Optimal Transport problems with structure,* arXiv:2008.03006, 2020.
[AB4] A. & Boix, *Hardness results for Multimarginal Optimal Transport problems*, Discrete Optimization, 2021.

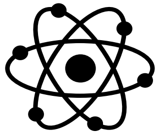
Structured MOT: a few application highlights

- **Ex in fluid dynamics:** first exact algorithm for **generalized Euler flows** [AB3]
  - This problem was the original motivation of MOT [Brenier ’89]



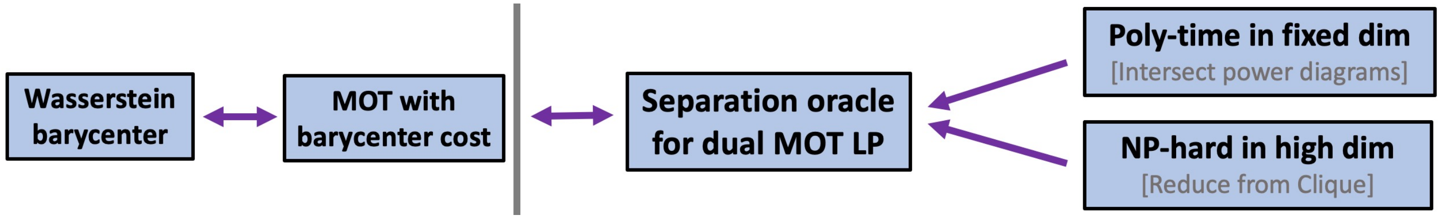


- **Ex in computational chemistry**: NP-hardness for **density functional theory** [AB4]
  - This problem captures the strong-interaction limit [Buzzano et al ‘12]



- **Ex in operations research:** first polynomial-time approximation algorithm for **quantile aggregation** [in preparation]
  - In contrast to NP-hardness for exact solution [Coffman and Yannakakis ‘84]

Takeaways



➢ What can Wasserstein Barycenters do for you?

- Average data distributions in a geometrically meaningful way.
- Applications: summarize, de-noise, cluster, …

➢ Can you compute them fast?

- Yes in fixed dimensions – now, to high precision [JMLR ’21]. Key insight: solve separation oracle can be efficiently solved by exploiting the structure of low-dimensional power diagrams.
- No in high dimensions [SIMODS ’21]. Key insight: separation oracle encodes hard combinatorial problems.

➢ General theory of when MOT is poly(n,k) solvable [arXiv ’20, Discrete Opt. ’21]

- Leads to many new applications. Leads to first exact/sparse solutions for known examples.

➢ Many important directions

- Practical heuristics: NP-hardness guides future algorithm design
- Practical scale: go beyond “poly” runtimes
- Practical paradigms: repeated solving in pipelines