Graph Sketching, Streaming and Space Efficient Optimization
Part II

Sudipto Guha and Andrew McGregor

Space Efficient Optimization for Graphs

Sampling, Connectivity, Sparsification: How do these get used?

Techniques like Dimensionality Reduction, Embeddings, $L_p \rightarrow L_q$, etc., are improving vector based computations.

Space Efficient Optimization for Graphs

Sampling, Connectivity, Sparsification: How do these get used?

Techniques like Dimensionality Reduction, Embeddings, $L_p \to L_q$, etc., are improving vector based computations.

Thesis: Graph optimization problems are natural next candidates.

Space is the final frontier. Processing Space $\neq$ Storage Space.
Streaming as a vehicle to organize accesses in an algorithm.

Space Efficient Optimization for Graphs

Sampling, Connectivity, Sparsification: How do these get used?

Techniques like Dimensionality Reduction, Embeddings, $L_p \to L_q$, etc., are improving vector based computations.

Thesis: Graph optimization problems are natural next candidates.

Space is the final frontier. Processing Space $\neq$ Storage Space.
Streaming as a vehicle to organize accesses in an algorithm.

View Sparsification as Embeddings:

- Embed the graph. Solve the optimization in the embedded space.
- Small space and faster runtimes!

Space Efficient Optimization for Graphs

Sampling, Connectivity, Sparsification: How do these get used?

Techniques like Dimensionality Reduction, Embeddings, $L_p \to L_q$, etc., are improving vector based computations.

Thesis: Graph optimization problems are natural next candidates.

Space is the final frontier. Processing Space $\neq$ Storage Space.
Streaming as a vehicle to organize accesses in an algorithm.

View Sparsification as Embeddings:

- Embed the graph. Solve the optimization in the embedded space.
- Small space and faster runtimes!

We focus on cut-sparsification. Will not always work out of the box.
We will have to change relaxations and use sparsification with care.

Plan of the Hour

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.
(a) Multiplicative Weights Method on SDPs.
Example: Correlation Clustering. Max-version.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

(a) Multiplicative Weights Method on SDPs.
    Example: Correlation Clustering. Max-version.

(b) Multiplicative Weights Method on LPs.
    Example: Correlation Clustering. Min-version.

New relaxations + oracle. Benefits in running time + space. Both cases.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

(a) Multiplicative Weights Method on SDPs.
Example: Correlation Clustering. Max-version.

(b) Multiplicative Weights Method on LPs.
Example: Correlation Clustering. Min-version.

New relaxations + oracle. Benefits in running time + space. Both cases.

Iterative (local) (Cut)-Sparsification. Multiples passes, Batch modes.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

(a) Multiplicative Weights Method on SDPs.
Example: Correlation Clustering. Max-version.

(b) Multiplicative Weights Method on LPs.
Example: Correlation Clustering. Min-version.

New relaxations + oracle. Benefits in running time + space. Both cases.

Iterative (local) (Cut)-Sparsification. Multiples passes, Batch modes.
Example 2. Non-bipartite Matching. $(1+\epsilon)$-apx.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1 + \epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

(a) Multiplicative Weights Method on SDPs.
Example: Correlation Clustering. Max-version.

(b) Multiplicative Weights Method on LPs.
Example: Correlation Clustering. Min-version.

New relaxations + oracle. Benefits in running time + space. Both cases.

Iterative (local) (Cut)-Sparsification. Multiples passes, Batch modes.
Example 2. Non-bipartite Matching. $(1 + \epsilon)$-apx.
Cornerstone of Combinatorial Optimization, Dantzig Decompositions.
Benefits in time+space+adaptivity.

Plan of the Hour

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

(a) Multiplicative Weights Method on SDPs.
Example: Correlation Clustering. Max-version.

(b) Multiplicative Weights Method on LPs.
Example: Correlation Clustering. Min-version.

New relaxations + oracle. Benefits in running time + space. Both cases.

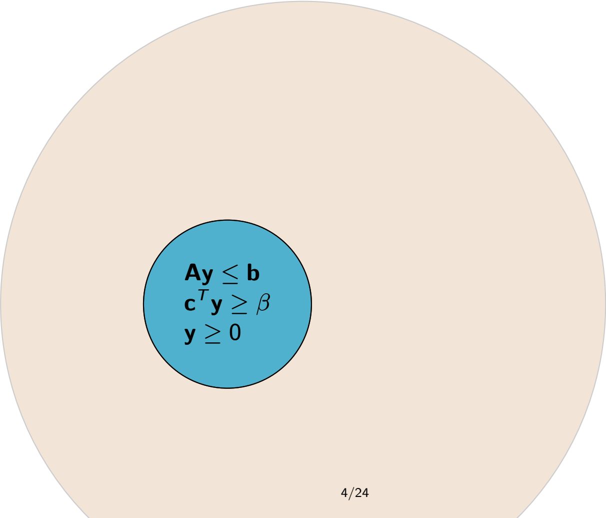
Iterative (local) (Cut)-Sparsification. Multiples passes, Batch modes.
Example 2. Non-bipartite Matching. $(1+\epsilon)$-apx.
Cornerstone of Combinatorial Optimization, Dantzig Decompositions.
Benefits in time+space+adaptivity.

Wrap-up.

Multiplicative Weights Method: A Recap

$\mathbf{Ay} \leq \mathbf{b}$
$\mathbf{c}^T\mathbf{y} \geq \beta$
$\mathbf{y} \geq 0$

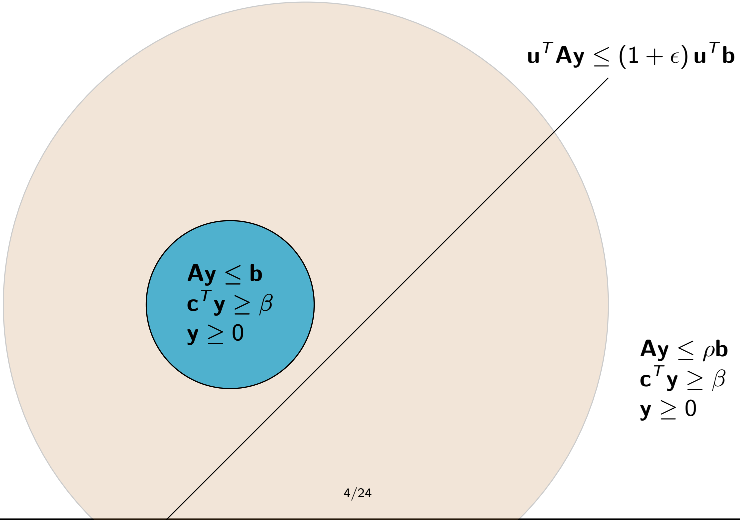
Multiplicative Weights Method: A Recap



$\mathbf{A}\mathbf{y} \leq \rho\mathbf{b}$
$\mathbf{c}^T\mathbf{y} \geq \beta$
$\mathbf{y} \geq 0$

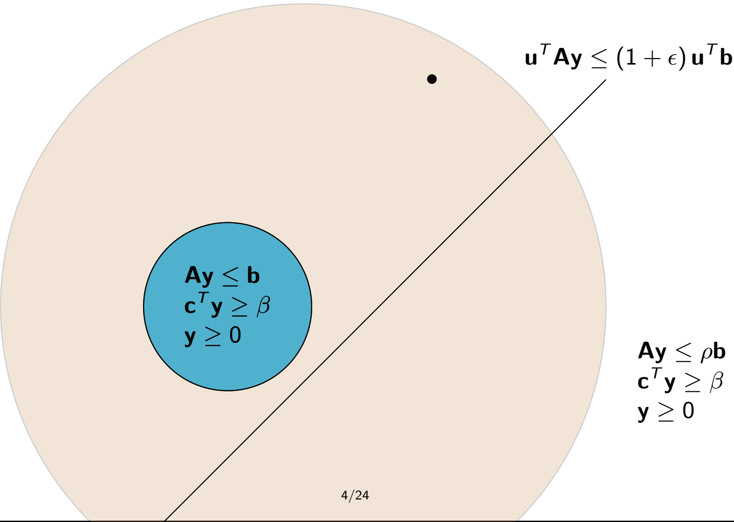
Multiplicative Weights Method: A Recap

Initially $\mathbf{u} = \mathbf{1}$.

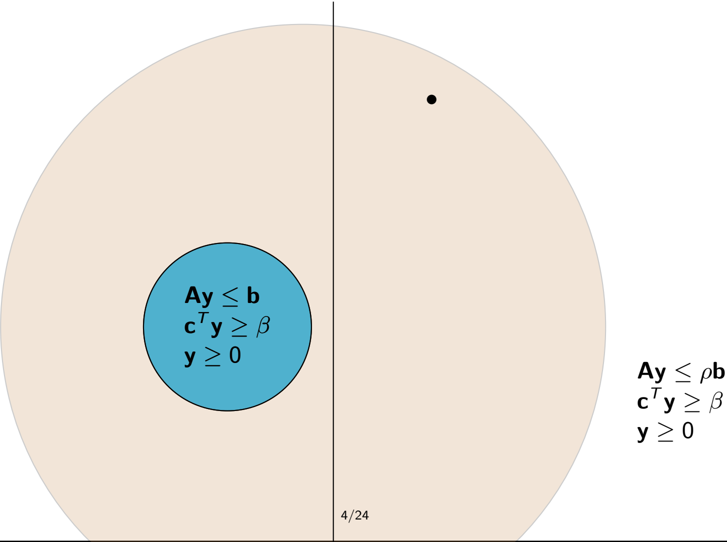


Multiplicative Weights Method: A Recap

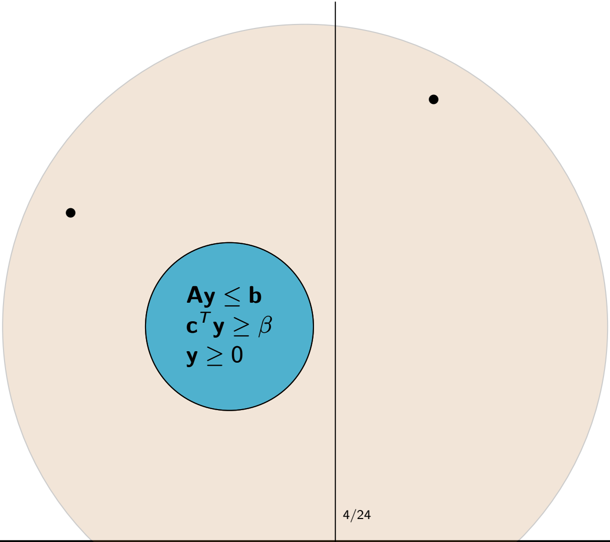
Initially $\mathbf{u} = \mathbf{1}$.
If $\mathbf{A}_i\mathbf{y} < \mathbf{b}_i$: lower $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1-\epsilon)^{(\mathbf{b}_i - \mathbf{A}_i\mathbf{y})/\mathbf{b}_i\rho}$. (Assume $\mathbf{A}, \mathbf{b} \geq \mathbf{0}$).
If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y} - \mathbf{b}_i)/\mathbf{b}_i\rho}$.



Multiplicative Weights Method: A Recap

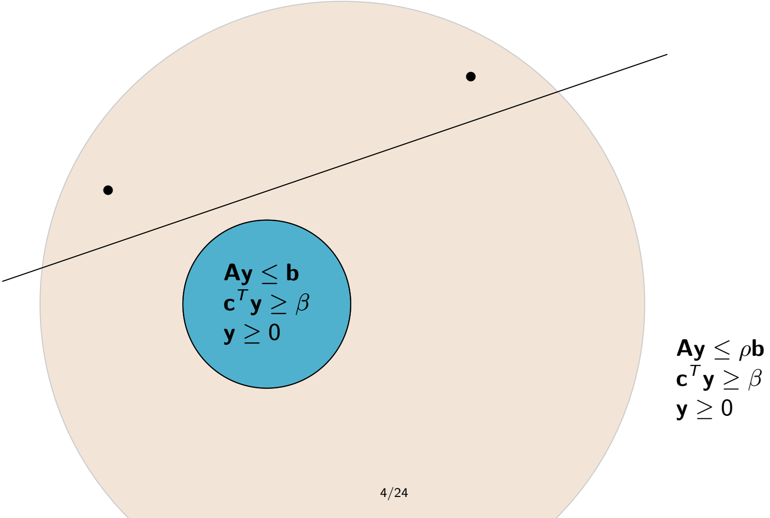


Multiplicative Weights Method: A Recap

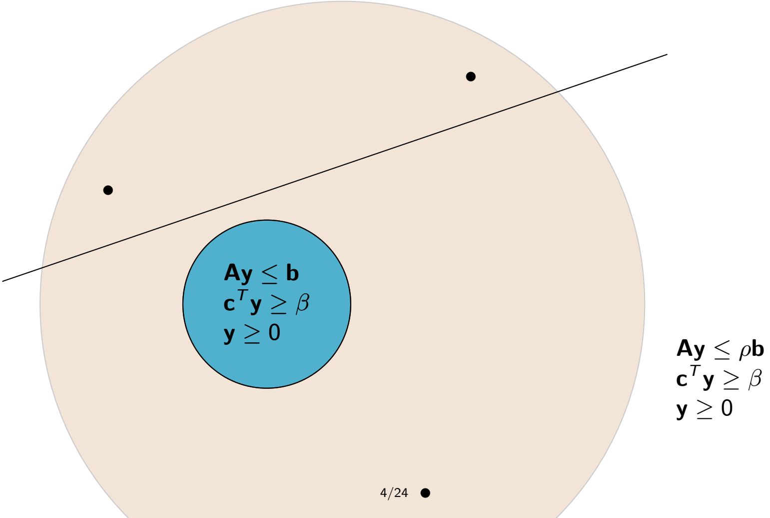


$\mathbf{A}\mathbf{y} \leq \rho\mathbf{b}$
$\mathbf{c}^T\mathbf{y} \geq \beta$
$\mathbf{y} \geq 0$

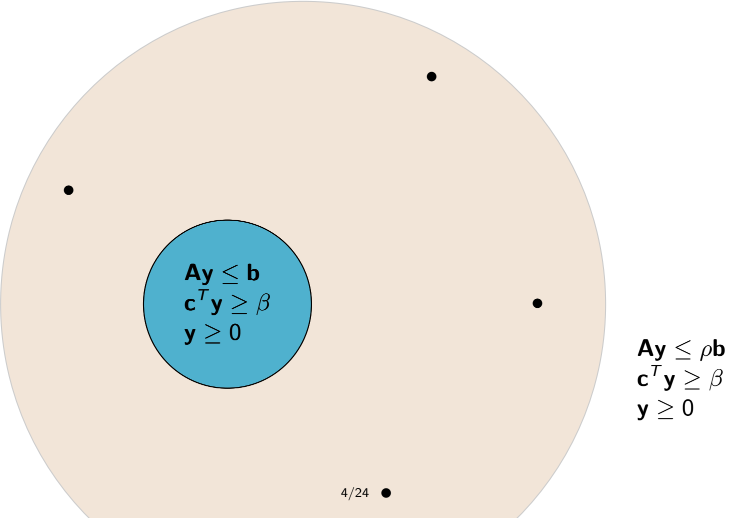
Multiplicative Weights Method: A Recap



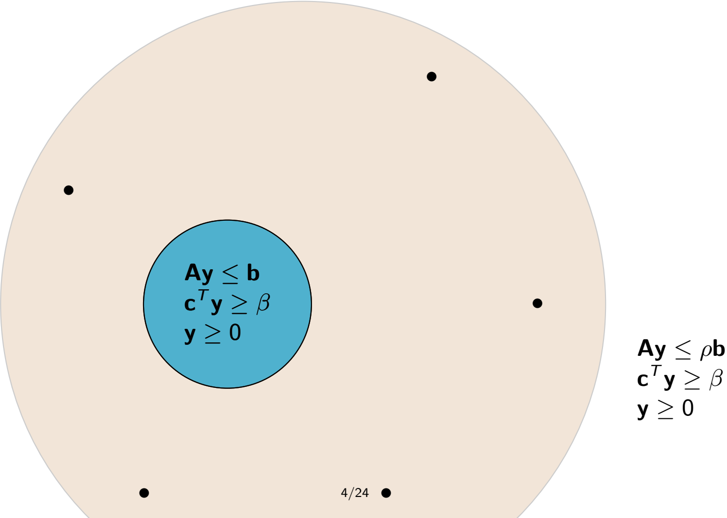
Multiplicative Weights Method: A Recap



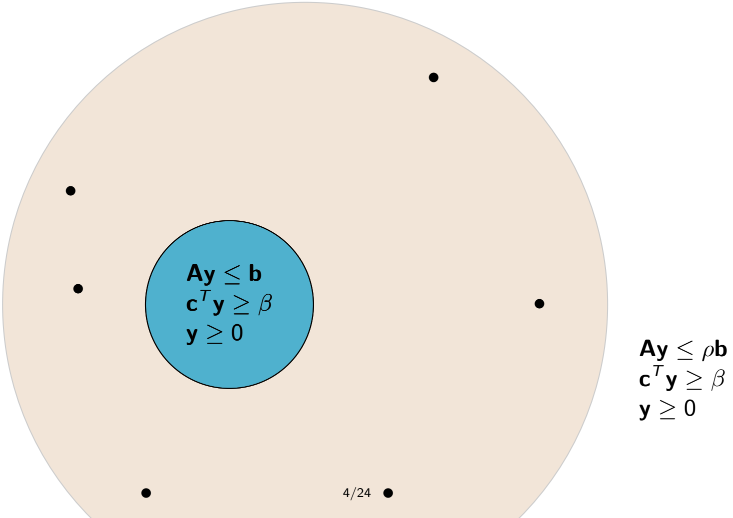
Multiplicative Weights Method: A Recap



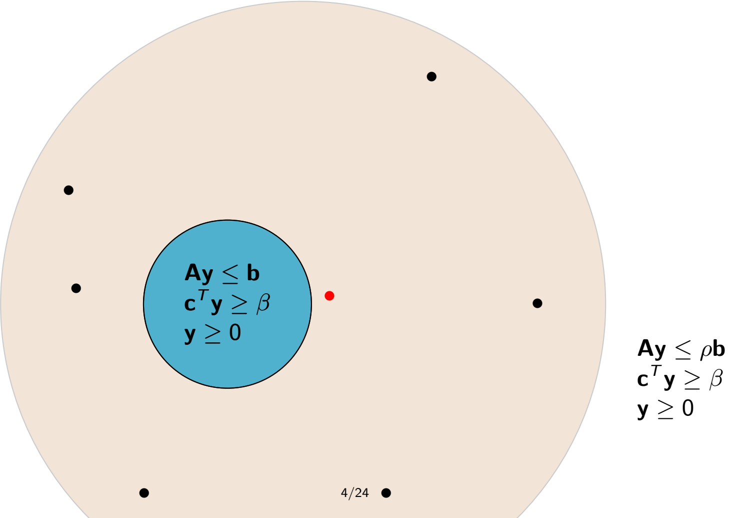
Multiplicative Weights Method: A Recap



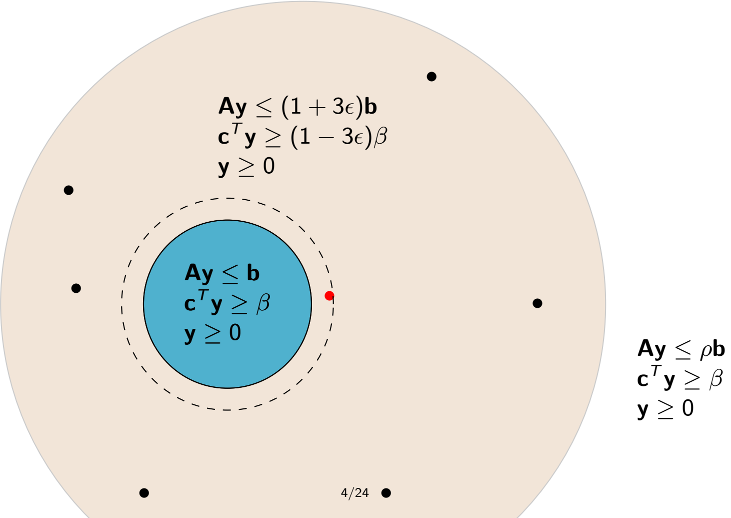
Multiplicative Weights Method: A Recap



Multiplicative Weights Method: A Recap

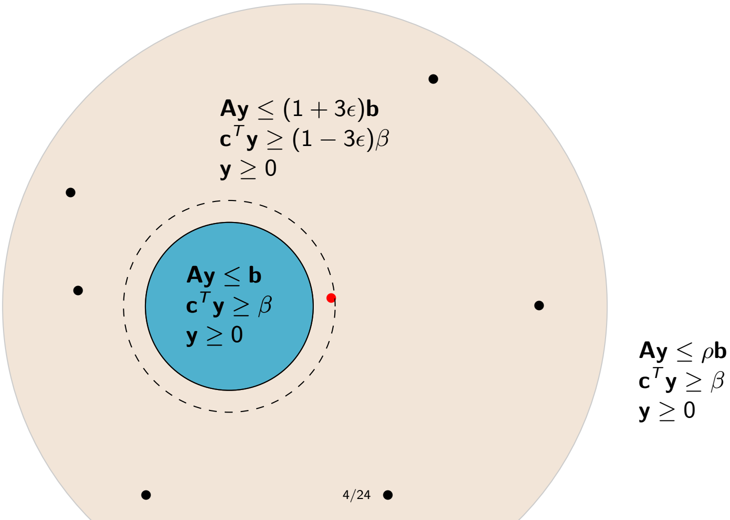


Multiplicative Weights Method: A Recap



Multiplicative Weights Method: A Recap

Number of rounds depends on ρ, ϵ and other specifics of updating **u**.
$\rho =$**width**.



MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} & & \\ & \sum_{j} y_{ij} & \leq 1 & \quad \forall i \\ & y_{ij} & \geq 0 & \quad \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.
Different from online learning. Input itself is in small pieces.

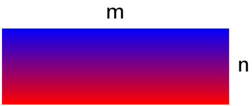
MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} & & \\ & \sum_{j} y_{ij} & \leq 1 & \quad \forall i \\ & y_{ij} & \geq 0 & \quad \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.
Different from online learning. Input itself is in small pieces.



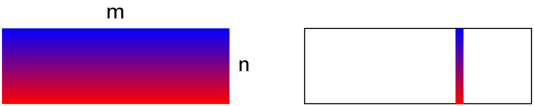
MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i, j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} \\ & \sum_{j} y_{ij} \leq 1 \quad \forall i \\ & y_{ij} \geq 0 \quad \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.
Different from online learning. Input itself is in small pieces.



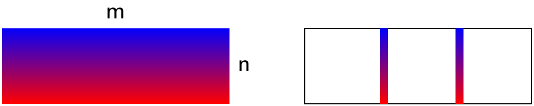
MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} & & \\ & \sum_{j} y_{ij} & \leq 1 & \quad \forall i \\ & y_{ij} & \geq 0 & \quad \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.
Different from online learning. Input itself is in small pieces.



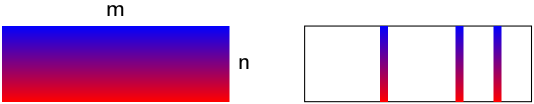
MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} & & \\ & \sum_{j} y_{ij} & \leq 1 \quad & \forall i \\ & y_{ij} & \geq 0 \quad & \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph. Different from online learning. Input itself is in small pieces.



MWM on Streams: Bipartite Matching

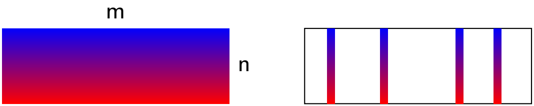
Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$
\begin{aligned}
\max \quad & \sum_{(i,j)} y_{ij} w_{ij} \\
& \sum_{j} y_{ij} \quad \leq 1 \quad \forall i \\
& y_{ij} \quad \geq 0 \quad \forall (i,j)
\end{aligned}
$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.

Different from online learning. Input itself is in small pieces.



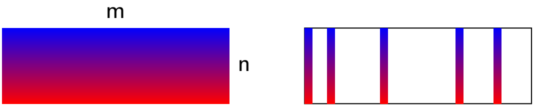
MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} & & \\ & \sum_{j} y_{ij} & \leq 1 \quad & \forall i \\ & y_{ij} & \geq 0 \quad & \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.
Different from online learning. Input itself is in small pieces.



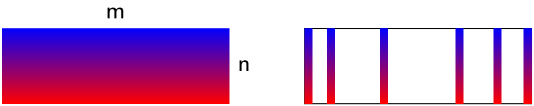
MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} & & \\ & \sum_{j} y_{ij} & \leq 1 & \quad \forall i \\ & y_{ij} & \geq 0 & \quad \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.
Different from online learning. Input itself is in small pieces.



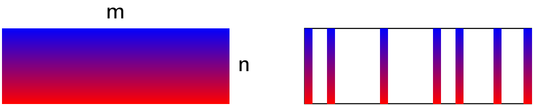
MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{aligned} \max \quad & \sum_{(i,j)} y_{ij} w_{ij} & & \\ & \sum_{j} y_{ij} & \leq 1 & \quad \forall i \\ & y_{ij} & \geq 0 & \quad \forall (i,j) \end{aligned}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.
Different from online learning. Input itself is in small pieces.



MWM on Streams: Bipartite Matching

Ahn, Guha 14.
Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i, j) implies $\in E$.)

$$
\begin{aligned}
\sum_{(i,j)} y_{ij} w_{ij} &\geq \beta \\
\sum_{j} y_{ij} &\leq 1 \quad \forall i \\
y_{ij} &\geq 0 \quad \forall (i, j)
\end{aligned}
$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.

Applying MWM: Point = candidate set of edges, in m-dim space.
Hyperplanes?

MWM on Streams: Bipartite Matching

Ahn, Guha 14.
Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$
\begin{aligned}
& & \sum_{(i,j)} y_{ij} w_{ij} &\geq \beta & \\
\mathbf{u}_i \rightarrow & & \sum_j y_{ij} &\leq 1 & \forall i \\
& & y_{ij} &\geq 0 & \forall (i,j)
\end{aligned}
$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.

Applying MWM: Point = candidate set of edges, in m-dim space.
Hyperplanes?

MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$\begin{array}{llll} & \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta & \\ \mathbf{u}_i \rightarrow & \sum_j y_{ij} & \leq 1 & \forall i \\ & y_{ij} & \geq 0 & \forall (i,j) \end{array}$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.

Applying MWM: Point = candidate set of edges, in m-dim space.
Hyperplanes? $\sum_i u_i \sum_j y_{ij} \leq \sum_i u_i \quad \Leftrightarrow \quad \sum_{(i,j)} y_{ij}(u_i + u_j) \leq \sum_i u_i$.
Store & update $\mathbf{u}$. $O(n)$ storage.

MWM on Streams: Bipartite Matching

Ahn, Guha 14.

Integer and fractional optimums coincide. ($y_{ij} = y_{ji}$, (i,j) implies $\in E$.)

$$
\begin{array}{llll}
 & \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta & \\
\mathbf{u}_i \rightarrow & \sum_{j} y_{ij} & \leq 1 & \forall i \\
 & y_{ij} & \geq 0 & \forall (i,j)
\end{array}
$$

Streams: arbitrary list of m edges, $\ldots, \langle i, j, w_{ij} \rangle, \ldots$ for an n node graph.

Applying MWM: Point = candidate set of edges, in m-dim space.

Hyperplanes? $\sum_i u_i \sum_j y_{ij} \leq \sum_i u_i \quad \Leftrightarrow \quad \sum_{(i,j)} y_{ij}(u_i + u_j) \leq \sum_i u_i$.

$$
\text{Want: } \left\{
\begin{array}{lll}
\sum_{(i,j)} y_{ij}(u_i + u_j) \sum_i u_i & \leq \sum_i u_i & \\
\sum_{(i,j)} y_{ij} w_{ij} & \geq \beta & \\
\sum_j y_{ij} & \leq \rho & \forall i \\
y_{ij} & \geq 0 & \forall (i,j)
\end{array}
\right. .
$$

MWM on Streams: Bipartite Matching

$$\text{Want: } \begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

,

MWM on Streams: Bipartite Matching

$$\text{Want:} \quad \begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_{j} y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

$$\text{Now } \exists \mathbf{y}, \;\; \forall \lambda \geq 0 \quad \begin{cases} \sum_{(i,j)} (w_{ij} - \lambda(u_i + u_j)) y_{ij} & \geq (\beta - \lambda \sum_i u_i) \\ \sum_{j} y_{ij} & \leq 1 \qquad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

MWM on Streams: Bipartite Matching

$$\text{Want: } \begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

$$\text{Now } \exists \mathbf{y}, \ \forall \lambda \geq 0 \ \begin{cases} \sum_{(i,j)} (w_{ij} - \lambda(u_i + u_j)) y_{ij} & \geq (\beta - \lambda \sum_i u_i) \\ \sum_j y_{ij} & \leq 1 \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Oracle(λ):

MWM on Streams: Bipartite Matching

Want: $\begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$

Now $\exists \mathbf{y}, \ \forall \lambda \geq 0$ $\begin{cases} \sum_{(i,j)} (w_{ij} - \lambda(u_i + u_j)) y_{ij} & \geq (\beta - \lambda \sum_i u_i) \\ \sum_j y_{ij} & \leq 1 \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$

Oracle(λ):

- Seeing (i,j) compute $(w_{ij} - \lambda(u_i + u_j))$. If -ve, discard.

MWM on Streams: Bipartite Matching

Want: $$\begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Have y, $\forall \lambda \geq 0$ $$\begin{cases} \sum_{(i,j)} (w_{ij} - \lambda(u_i + u_j))\, y_{ij} & \geq (\beta - \lambda \sum_i u_i)/c \\ \sum_j y_{ij} & \leq 1 \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Oracle(λ):

- Seeing (i,j) compute $(w_{ij} - \lambda(u_i + u_j))$. If -ve, discard.
- Find a streaming $O(n)$ space c approximation on this filtered set.

MWM on Streams: Bipartite Matching

Want:
$$\begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Have y,

$$\begin{cases} \sum_{(i,j)} (w_{ij} - \lambda(u_i + u_j))\, y_{ij} & \geq (\beta - \lambda \sum_i u_i)/c \\ \sum_j y_{ij} & \leq 1 \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Oracle(λ):

- Seeing (i,j) compute $(w_{ij} - \lambda(u_i + u_j))$. If -ve, discard.
- Find a streaming $O(n)$ space c approximation on this filtered set.

If Oracle(λ) for $\lambda = 0$ satisfies $\sum_{(i,j)} y_{ij}(u_i + u_j) \leq \sum_i u_i / c$ then we also have: $\sum_{(i,j)} w_{ij} y_{ij} \geq \beta / c$. **(easier case)**

MWM on Streams: Bipartite Matching

Want:
$$\begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Have y,
$$\begin{cases} \sum_{(i,j)} (w_{ij} - \lambda(u_i + u_j))\, y_{ij} & \geq (\beta - \lambda \sum_i u_i)/c \\ \sum_j y_{ij} & \leq 1 \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Oracle(λ):

- Seeing (i,j) compute $(w_{ij} - \lambda(u_i + u_j))$. If -ve, discard.
- Find a streaming $O(n)$ space c approximation on this filtered set.

For $\lambda = 0$ we have $\sum_{(i,j)} y_{ij}(u_i + u_j) \geq \sum_i u_i/c$.
For $\lambda = \sum_i u_i/\beta$ we have $\sum_{(i,j)} y_{ij}(u_i + u_j) \leq \sum_i u_i/c$. (Set $\mathbf{y} = 0$)

MWM on Streams: Bipartite Matching

Want: $\begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$

Have y, $\begin{cases} \sum_{(i,j)} (u_i + u_j) y_{ij} \leq \sum_i u_i / c \quad \text{and} \quad \sum_{(i,j)} w_{ij} y_{ij} \geq \beta / c \\ \sum_j y_{ij} \leq 1 \quad \forall i \\ y_{ij} \geq 0 \quad \forall (i,j) \end{cases}$

Oracle(λ):

- Seeing (i,j) compute $(w_{ij} - \lambda(u_i + u_j))$. If -ve, discard.
- Find a streaming $O(n)$ space c approximation on this filtered set.

For $\lambda = 0$ we have $\sum_{(i,j)} y_{ij}(u_i + u_j) \geq \sum_i u_i / c$.

For $\lambda = \sum_i u_i / \beta$ we have $\sum_{(i,j)} y_{ij}(u_i + u_j) \leq \sum_i u_i / c$. (Set $\mathbf{y} = 0$)

Binary search (or try values of λ in parallel).

MWM on Streams: Bipartite Matching

Want: $$\begin{cases} \sum_{(i,j)} y_{ij}(u_i + u_j) & \leq \sum_i u_i \\ \sum_{(i,j)} y_{ij} w_{ij} & \geq \beta \\ \sum_j y_{ij} & \leq \rho \quad \forall i \\ y_{ij} & \geq 0 \quad \forall (i,j) \end{cases}$$

Have y, $$\begin{cases} \sum_{(i,j)} (u_i + u_j) y_{ij} \leq \sum_i u_i / c \quad \text{and} \quad \sum_{(i,j)} w_{ij} y_{ij} \geq \beta / c \\ \sum_j y_{ij} \leq 1 \quad \forall i \\ y_{ij} \geq 0 \quad \forall (i,j) \end{cases}$$

Oracle(λ):

- Seeing (i,j) compute $(w_{ij} - \lambda(u_i + u_j))$. If -ve, discard.
- Find a streaming $O(n)$ space c approximation on this filtered set.

For $\lambda = 0$ we have $\sum_{(i,j)} y_{ij}(u_i + u_j) \geq \sum_i u_i / c$.

For $\lambda = \sum_i u_i / \beta$ we have $\sum_{(i,j)} y_{ij}(u_i + u_j) \leq \sum_i u_i / c$. (Set $\mathbf{y} = 0$)

Binary search (or try values of λ in parallel).

Multiply **y** by c. Set $\rho = c$ and we have a solution!

MWM based Bipartite Matching for Map-Reduce?

More general than streaming.

Map-Reduce based 8 approximations in $O(\log n)$ rounds exist, e.g., Lattanzi, Mosely, Suri, Vassilivitskii 11.

We can compose them. $O(\log n)$ rounds to get a c-approximation. Repeat $O(c\epsilon^{-2} \log n)$ times to get a $(1+\epsilon)$- fractional solution.

Can also round to an integral solution in small space.
A story for some other time.

Up Next ...

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. $(1+\epsilon)$-apx. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

(a) Multiplicative Weights Method on SDPs.
Example: Correlation Clustering. Max-version.

(b) Multiplicative Weights Method on LPs.
Example: Correlation Clustering. Min-version.

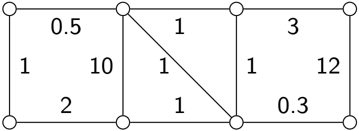
New relaxations + oracle. Benefits in running time + space. Both cases.

Iterative (local) (Cut)-Sparsification. Multiples passes, Batch modes.
Example 2. Non-bipartite Matching. $(1+\epsilon)$-apx.
Cornerstone of Combinatorial Optimization, Dantzig Decompositions.
Benefits in time+space+adaptivity.

Wrap-up.

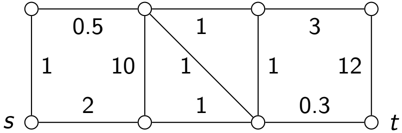
Global Sparsification: There and back again

Think of a problem on graph cuts.



Global Sparsification: There and back again

Think of a problem on graph cuts.

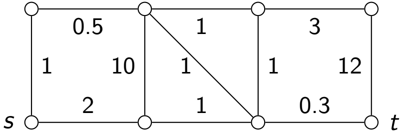


Min s-t Cut?

Sparsification preserves all cuts within $(1 \pm \epsilon)$.

Global Sparsification: There and back again

Think of a problem on graph cuts.

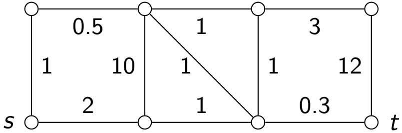


Min s-t Cut? Max s-t Cut? Max Cut?

Sparsification preserves all cuts within $(1 \pm \epsilon)$.

Global Sparsification: There and back again

Think of a problem on graph cuts.

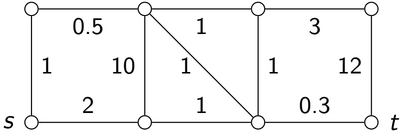


Min s-t Cut? Max s-t Cut? Max Cut? NP Hard. ≥ 0.5 apx uses SDPs.

Sparsification preserves all cuts within $(1 \pm \epsilon)$.

Global Sparsification: There and back again

Think of a problem on graph cuts.



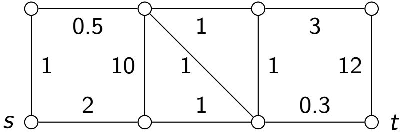
Min s-t Cut? Max s-t Cut? Max Cut? NP Hard. ≥ 0.5 apx uses SDPs.

Sparsification preserves all cuts within $(1 \pm \epsilon)$.

(a) Does not imply anything about finding specific cuts.

Global Sparsification: There and back again

Think of a problem on graph cuts.



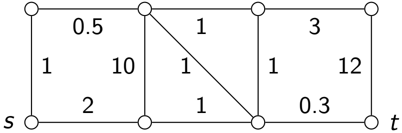
Min s-t Cut? Max s-t Cut? Max Cut? NP Hard. ≥ 0.5 apx uses SDPs.

Sparsification preserves all cuts within $(1 \pm \epsilon)$.

(a) Does not imply anything about finding specific cuts. Yet.

Global Sparsification: There and back again

Think of a problem on graph cuts.



Min s-t Cut? Max s-t Cut? Max Cut? NP Hard. ≥ 0.5 apx uses SDPs.

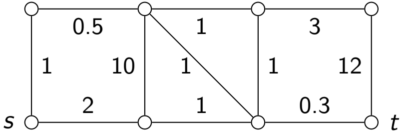
Sparsification preserves all cuts within $(1 \pm \epsilon)$.

(a) Does not imply anything about finding specific cuts. Yet.

(b) Does not obviously save space either!

Global Sparsification: There and back again

Think of a problem on graph cuts.



Min s-t Cut? Max s-t Cut? Max Cut? NP Hard. ≥ 0.5 apx uses SDPs.

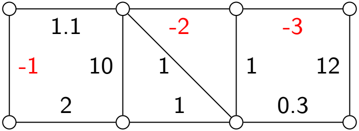
Sparsification preserves all cuts within $(1 \pm \epsilon)$.

(a) Does not imply anything about finding specific cuts. Yet.

(b) Does not obviously save space either!

We will see examples both (a)–(b) and how to overcome them.
Lets consider a variant of **clustering.** And richer graphs.

Correlation Clustering



Find a grouping that **agrees** most with the graph.

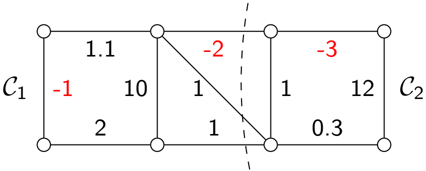
- Count +ve edges in clusters. Count -ve edges **out** of clusters.
- Use as many clusters as you like.

Alternatively we can find a grouping that **disagrees** least.

NP Hard. Bansal Blum, Chawla, 04.

Many approximation algorithms are known. For many variants.
The approximations we see here were known defore, we will not focus on the factor.

Correlation Clustering



Find a grouping that **agrees** most with the graph.

- Count +ve edges in clusters. Count -ve edges **out** of clusters.
- Use as many clusters as you like.

Alternatively we can find a grouping that **disagrees** least.

NP Hard. Bansal Blum, Chawla, 04.

Many approximation algorithms are known. For many variants.
The approximations we see here were known defore, we will not focus on the factor.

Correlation Clustering: Motivation

Tutorial in KDD 2014. Bonchi, Garcia-Soriano, Liberty.
Clustering of objects known only through relationships.
(Can have wide ranges of edge weights, +ve/-ve.)

Correlation Clustering: Motivation

Tutorial in KDD 2014. Bonchi, Garcia-Soriano, Liberty.
Clustering of objects known only through relationships.
(Can have wide ranges of edge weights, +ve/-ve.)

Consider an Entity Resolution example.

News arcticle 1: **Mr Smith** is devoted to mountain climbing. . . . **Mrs Smith** is a diver and said that she finds diving to be a sublime experience. . . . The goal is to reach new heights, said **Smith**.

Now consider a stream of such articles, with new as well as old entities.

Correlation Clustering: Motivation

Tutorial in KDD 2014. Bonchi, Garcia-Soriano, Liberty.
Clustering of objects known only through relationships.
(Can have wide ranges of edge weights, +ve/-ve.)

Consider an Entity Resolution example.

News arcticle 1: **Mr Smith** is devoted to mountain climbing. . . . **Mrs Smith** is a diver and said that she finds diving to be a sublime experience. . . . The goal is to reach new heights, said **Smith**.

Now consider a stream of such articles, with new as well as old entities.

Likely **Mr Smith** $\neq$ **Mrs Smith**. Large -ve weight.
The other references can be either. Small weights depending on context.
Weights are not a metric. Have a large range.

Max-Agreement and SDPs

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Think of vector programming over unit length vectors. $x_{ij} = v_i \cdot v_j \leq 1$.

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E(+)} w_{ij} x_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}|(1 - x_{ij}) & \\ & x_{ii} = 1 & \forall i \\ & x_{ij} \geq 0 & \forall i, j \\ & \mathbf{x} \succeq \mathbf{0} & \end{aligned}$$

Max-Agreement and SDPs

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Think of vector programming over unit length vectors. $x_{ij} = v_i \cdot v_j \leq 1$.

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E(+)} w_{ij}x_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}|(1 - x_{ij}) & \\ & x_{ii} = 1 & \forall i \\ & x_{ij} \geq 0 & \forall i,j \\ & \mathbf{x} \succeq \mathbf{0} & \end{aligned}$$

MWM (in this context): Collection of constraints. Feasible set: $\mathcal{X}$.
Given $\mathbf{x}$ provide a real symmetric $\mathbf{A}$ (satisfying some **width** bounds)

(a) $\mathbf{A} \circ \mathbf{x} \leq b - \epsilon$, note $\mathbf{A} \circ \mathbf{x} = \sum_{i,j} A_{ij}x_{ij}$.

(b) $\mathbf{A} \circ \mathbf{x}' \geq b$ for all feasible $\mathbf{x}' \in \mathcal{X}$.

Max-Agreement and SDPs

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Think of vector programming over unit length vectors. $x_{ij} = v_i \cdot v_j \leq 1$.

$$\begin{array}{lll} \max & \sum_{(i,j)\in E(+)} w_{ij}x_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}|(1 - x_{ij}) & \\ & x_{ii} = 1 & \forall i \\ & x_{ij} \geq 0 & \forall i, j \\ & \mathbf{x} \succeq \mathbf{0} & \end{array}$$

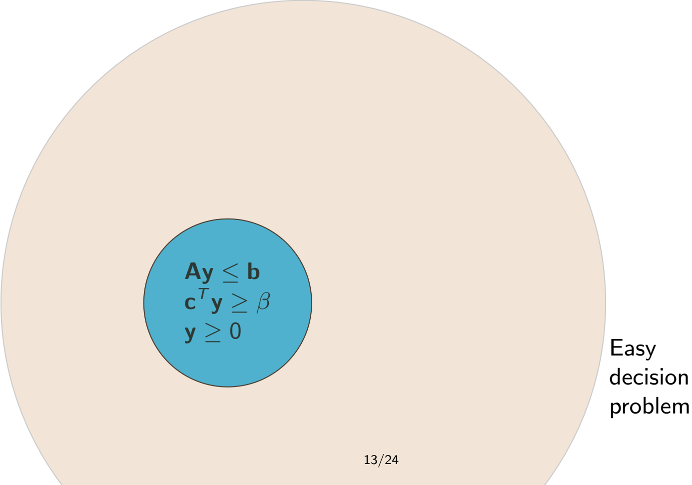
MWM (in this context): Collection of constraints. Feasible set: $\mathcal{X}$.
Given $\mathbf{x}$ provide a real symmetric $\mathbf{A}$ (satisfying some **width** bounds)

(a) $\mathbf{A} \circ \mathbf{x} \leq b - \epsilon$, note $\mathbf{A} \circ \mathbf{x} = \sum_{i,j} A_{ij} x_{ij}$.

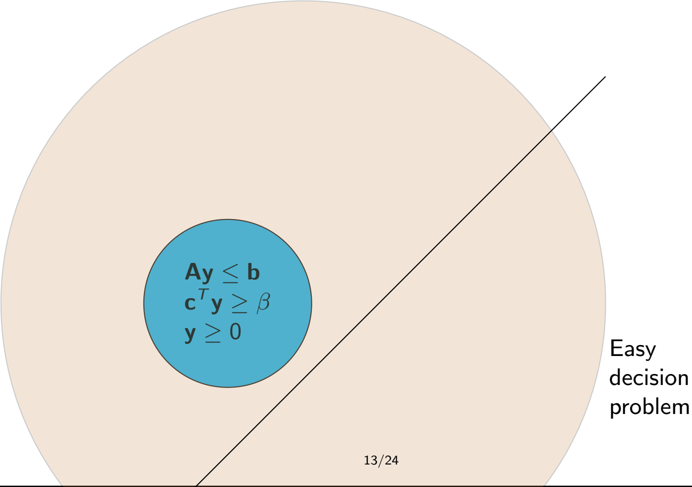
(b) $\mathbf{A} \circ \mathbf{x}' \geq b$ for all feasible $\mathbf{x}' \in \mathcal{X}$.

Why??

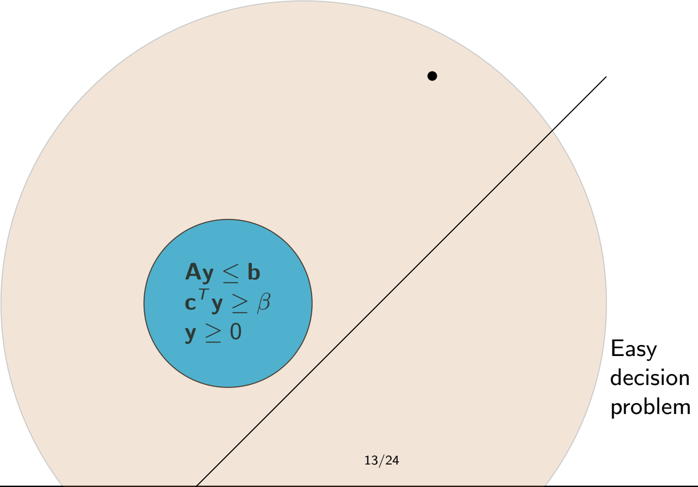
Multiplicative Weights Method: Another Recap



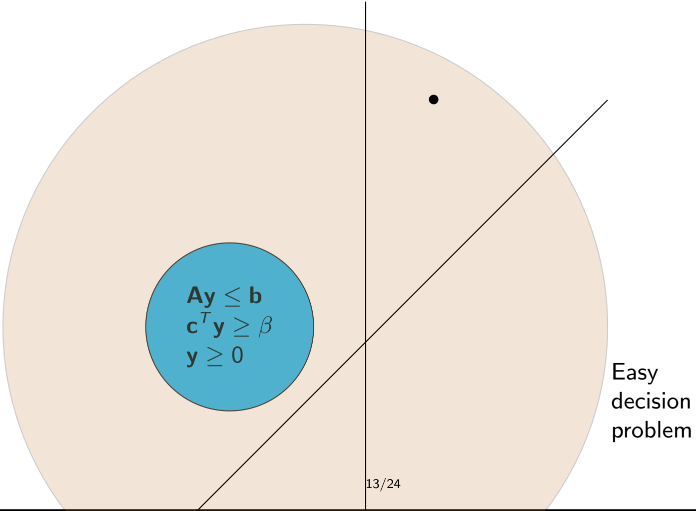
Multiplicative Weights Method: Another Recap



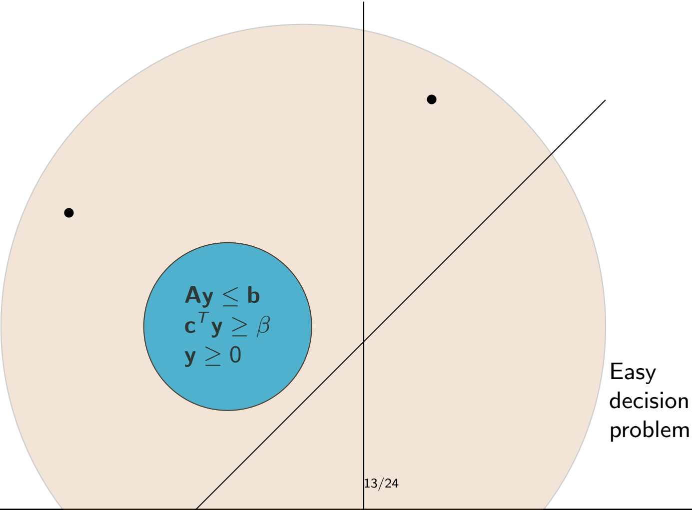
Multiplicative Weights Method: Another Recap



Multiplicative Weights Method: Another Recap

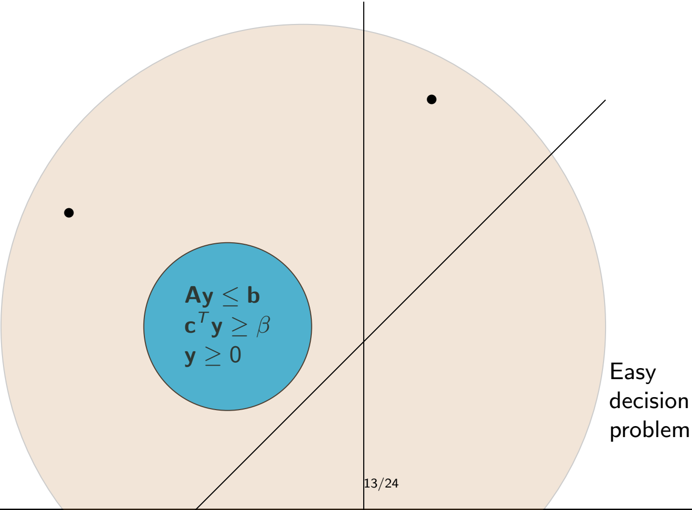


Multiplicative Weights Method: Another Recap



Multiplicative Weights Method: Another Recap

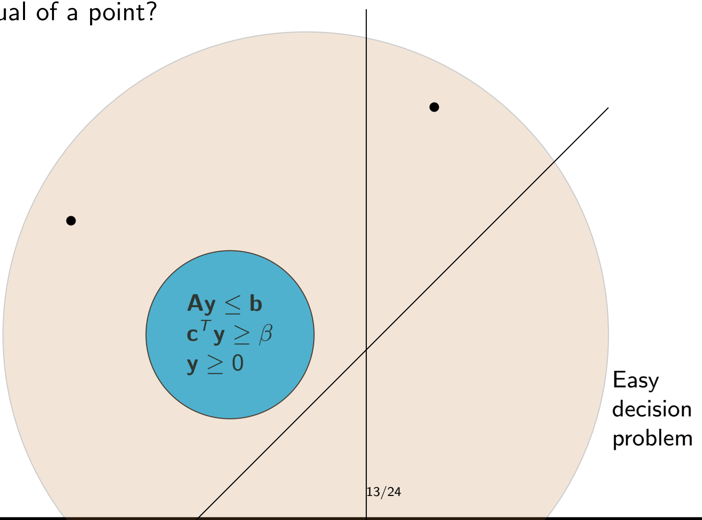
Instead of tracking violations and averaging solutions at the end,
Consider the process from the perspective of u



Multiplicative Weights Method: Another Recap

Instead of tracking violations and averaging solutions at the end,
Consider the process from the perspective of u

Dual of a hyperplane?
Dual of a point?

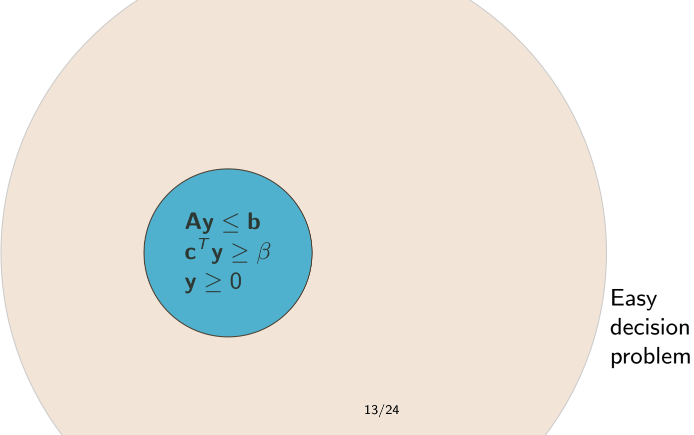


Multiplicative Weights Method: Another Recap

Instead of tracking violations and averaging solutions at the end,
Consider the process from the perspective of u

Dual of a hyperplane? Point in dual space.
Dual of a point? Constraint in dual space.

Suppose we prove [*]: $\exists \mathbf{u}$ s.t. $\mathbf{A}^T\mathbf{u} \geq \mathbf{c}$ and $\mathbf{b}^T\mathbf{u} < \beta$.

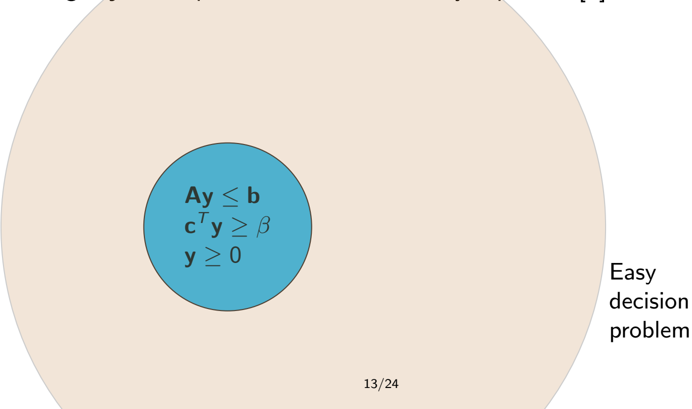


Multiplicative Weights Method: Another Recap

Instead of tracking violations and averaging solutions at the end,
Consider the process from the perspective of $\mathbf{u}$

Dual of a hyperplane? Point in dual space.
Dual of a point? Constraint in dual space.

Suppose we prove [*]: $\exists \mathbf{u}$ s.t. $\mathbf{A}^T\mathbf{u} \geq \mathbf{c}$ and $\mathbf{b}^T\mathbf{u} < \beta$.
Providing a $\mathbf{y}$ corresponds to: we have not yet proved [*].

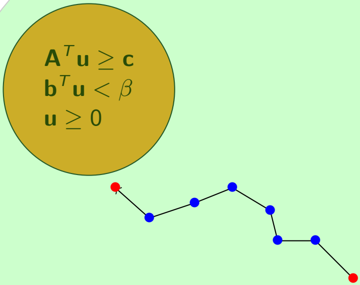


Multiplicative Weights Method: Another Recap

Instead of tracking violations and averaging solutions at the end,
Consider the process from the perspective of u

Dual of a hyperplane? Point in dual space.
Dual of a point? Constraint in dual space.

Suppose we prove [*]: $\exists \mathbf{u}$ s.t. $\mathbf{A}^T\mathbf{u} \geq \mathbf{c}$ and $\mathbf{b}^T\mathbf{u} < \beta$.
Providing a **y** corresponds to: we have not yet proved [*].
Think trajectories.
MWM on dual.
e.g., Steurer 10.



Max-Agreement and SDPs

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Think of vector programming over unit length vectors. $x_{ij} = v_i \cdot v_j \leq 1$.

$$
\begin{aligned}
\max \quad & \sum_{(i,j)\in E(+)} w_{ij}x_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}|(1 - x_{ij}) & \\
& x_{ii} = 1 & \forall i \\
& x_{ij} \geq 0 & \forall i,j \\
& \mathbf{x} \succeq \mathbf{0} &
\end{aligned}
$$

MWM (in this context): Collection of constraints. Feasible set: $\mathcal{X}$.
Given $\mathbf{x}$ provide a real symmetric $\mathbf{A}$ (satisfying some **width** bounds)

(a) $\mathbf{A} \circ \mathbf{x} \leq b - \epsilon$, note $\mathbf{A} \circ \mathbf{x} = \sum_{i,j} A_{ij}x_{ij}$.

(b) $\mathbf{A} \circ \mathbf{x}' \geq b$ for all feasible $\mathbf{x}' \in \mathcal{X}$.

Why.

Max-Agreement and SDPs

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Think of vector programming over unit length vectors. $x_{ij} = v_i \cdot v_j \leq 1$.

$$
\begin{aligned}
\beta \leq \quad & \sum_{(i,j)\in E(+)} w_{ij} x_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}|(1 - x_{ij}) & \\
& x_{ii} = 1 & \forall i \\
& x_{ij} \geq 0 & \forall i, j \\
& \mathbf{x} \succeq \mathbf{0} &
\end{aligned}
$$

MWM (in this context): Collection of constraints. Feasible set: $\mathcal{X}$.
Given $\mathbf{x}$ provide a real symmetric $\mathbf{A}$ (satisfying some **width** bounds)

(a) $\mathbf{A} \circ \mathbf{x} \leq b - \epsilon$, note $\mathbf{A} \circ \mathbf{x} = \sum_{i,j} A_{ij} x_{ij}$.

(b) $\mathbf{A} \circ \mathbf{x}' \geq b$ for all feasible $\mathbf{x}' \in \mathcal{X}$.

Why. Does not work (width is high).

Max-Agreement and SDPs

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Think of vector programming over unit length vectors. $x_{ij} = v_i \cdot v_j \leq 1$.

$$\beta \leq \sum_{(i,j)\in E(+)} w_{ij} x_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}| \sout{(1 - x_{ij})} \frac{x_{ii} + x_{jj} - 2x_{ij}}{2}$$

$$x_{ii} = 1 \quad \forall i$$

$$x_{ij} \geq 0 \quad \forall i, j$$

$$\mathbf{x} \succeq \mathbf{0}$$

MWM (in this context): Collection of constraints. Feasible set: $\mathcal{X}$.
Given $\mathbf{x}$ provide a real symmetric $\mathbf{A}$ (satisfying some **width** bounds)

(a) $\mathbf{A} \circ \mathbf{x} \leq b - \epsilon$, note $\mathbf{A} \circ \mathbf{x} = \sum_{i,j} A_{ij} x_{ij}$.

(b) $\mathbf{A} \circ \mathbf{x}' \geq b$ for all feasible $\mathbf{x}' \in \mathcal{X}$.

Why. ~~Does not work (width is high).~~ Linear Space. Linear time. 0.76-apx

Max-Agreement and SDPs

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Think of vector programming over unit length vectors. $x_{ij} = v_i \cdot v_j \leq 1$.

$$\beta \leq \sum_{(i,j)\in E(+)} w_{ij} x_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}| \cancel{(1 - x_{ij})} \frac{x_{ii} + x_{jj} - 2x_{ij}}{2}$$

$$x_{ii} = 1 \qquad \forall i$$

$$x_{ij} \geq 0 \qquad \forall i, j$$

$$\mathbf{x} \succeq \mathbf{0}$$

MWM (in this context): Collection of constraints. Feasible set: $\mathcal{X}$.
Given $\mathbf{x}$ provide a real symmetric $\mathbf{A}$ (satisfying some **width** bounds)

(a) $\mathbf{A} \circ \mathbf{x} \leq b - \epsilon$, note $\mathbf{A} \circ \mathbf{x} = \sum_{i,j} A_{ij} x_{ij}$.

(b) $\mathbf{A} \circ \mathbf{x}' \geq b$ for all feasible $\mathbf{x}' \in \mathcal{X}$.

Why. ~~Does not work (width is high).~~ Linear Space. Linear time. 0.76-apx
Relaxation needs to be compatible with trajectory.
Single pass. Sparsify $E(+)$ and $E(-)$ separately.

Ahn 13, Ahn, Cormode, Guha, McGregor, Wirth 15.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.

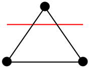
$$
\begin{aligned}
&\min \sum_{(i,j)\in E(+)} w_{ij}(1 - x_{ij}) + \sum_{(i,j)\in E(-)} |w_{ij}| x_{ij} && \\
&x_{ij} \le 1 && \forall i, j \\
&x_{ij} \ge 0 && \forall i, j \\
&(1 - x_{ij}) + (1 - x_{jk}) \ge (1 - x_{ik}) && \forall i, j, k
\end{aligned}
$$

A linear program.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.



$$\begin{aligned}
&\min \sum_{(i,j)\in E(+)} w_{ij}(1 - x_{ij}) + \sum_{(i,j)\in E(-)} |w_{ij}| x_{ij} && \\
&x_{ij} \le 1 && \forall i, j \\
&x_{ij} \ge 0 && \forall i, j \\
&(1 - x_{ij}) + (1 - x_{jk}) \ge (1 - x_{ik}) && \forall i, j, k
\end{aligned}$$

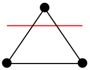
Triangle constraints

A linear program. $\Theta(n^3)$ Constraints, $\Theta(n^2)$ variables.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.



$$
\begin{aligned}
\min \sum_{(i,j)\in E(+)} w_{ij}(1-x_{ij}) + \sum_{(i,j)\in E(-)} |w_{ij}|x_{ij} & \\
x_{ij} \le 1 & \quad \forall i,j \\
x_{ij} \ge 0 & \quad \forall i,j \\
(1-x_{ij}) + (1-x_{jk}) \ge (1-x_{ik}) & \quad \forall i,j,k
\end{aligned}
$$

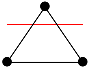
Triangle constraints

A linear program. $\Theta(n^3)$ Constraints, $\Theta(n^2)$ variables.
1 pass lower bound of $|E(-)|$ for any apx via Communication Complexity.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.



$$
\begin{aligned}
\min \sum_{(i,j)\in E(+)} w_{ij}(1 - x_{ij}) + \sum_{(i,j)\in E(-)} |w_{ij}| x_{ij} & \\
x_{ij} \le 1 & \quad \forall i,j \\
x_{ij} \ge 0 & \quad \forall i,j \\
(1 - x_{ij}) + (1 - x_{jk}) \ge (1 - x_{ik}) & \quad \forall i,j,k
\end{aligned}
$$

Triangle constraints

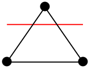
A linear program. $\Theta(n^3)$ Constraints, $\Theta(n^2)$ variables.
1 pass lower bound of $|E(-)|$ for any apx via Communication Complexity.

Sparsify $E(+)$, store $E(-)$? Will have $\tilde{O}(n) + |E(-)|$ variables.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.



$$
\begin{aligned}
&\min \sum_{(i,j)\in E(+)} w_{ij}(1 - x_{ij}) + \sum_{(i,j)\in E(-)} |w_{ij}| x_{ij} && \\
&x_{ij} \leq 1 && \forall i,j \\
&x_{ij} \geq 0 && \forall i,j \\
&(1 - x_{ij}) + (1 - x_{jk}) \geq (1 - x_{ik}) && \forall i,j,k
\end{aligned}
$$

Triangle constraints

A linear program. $\Theta(n^3)$ Constraints, $\Theta(n^2)$ variables.
1 pass lower bound of $|E(-)|$ for any apx via Communication Complexity.

Sparsify $E(+)$, store $E(-)$? Will have $\tilde{O}(n) + |E(-)|$ variables.

Does **not** work. The triangle constraints need all $\binom{n}{2}$ variables.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Set $y_{ij} = 1 - x_{ij}$ for $+$ve edges. $z_{ij} = x_{ij}$ for -ve edges.

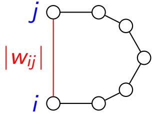
$$\begin{aligned} \min \quad & \sum_{(i,j)\in E(+)} w_{ij} y_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}| z_{ij} & \\ & y_{ij}, z_{ij} \geq 0 & \forall i,j \\ & y_{ij}, z_{ij} ? & \end{aligned}$$

Sparsify $E(+)$. Store $E(-)$. $\Theta(n^2) \rightarrow \tilde{O}(n) + |E(-)|$ variables?
$\Theta(n^3)$ Constraints

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Set $y_{ij} = 1 - x_{ij}$ for $+$ve edges. $z_{ij} = x_{ij}$ for -ve edges.



$$\min \sum_{(i,j)\in E(+)} w_{ij} y_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}| z_{ij}$$

$$y_{ij}, z_{ij} \geq 0 \qquad \forall i, j$$

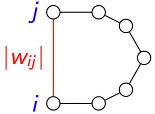
$$\sum_{(u,v)\in P(ij)} y_{uv} + z_{ij} \geq 1 \qquad \forall i, j, \text{and } i\text{-}j \text{ path } P(ij)$$

Sparsify $E(+)$. Store $E(-)$. $\Theta(n^2) \to \tilde{O}(n) + |E(-)|$ variables.
$\Theta(n^3)$ Constraints $\to$ Exponentially many constraints!

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Set $y_{ij} = 1 - x_{ij}$ for $+$ve edges. $z_{ij} = x_{ij}$ for -ve edges.



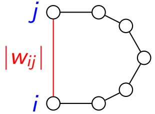
$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E(+)} w_{ij} y_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}| z_{ij} & \\
& y_{ij}, z_{ij} \geq 0 & \forall i,j \\
& \sum_{(u,v)\in P(ij)} y_{uv} + z_{ij} \geq 1 & \forall i,j, \text{and } i\text{-}j \text{ path } P(ij)
\end{aligned}$$

Sparsify $E(+)$. Store $E(-)$. $\Theta(n^2) \to \tilde{O}(n) + |E(-)|$ variables.
$\Theta(n^3)$ Constraints $\to$ Exponentially many constraints!
Solve LP (ellipsoid) & **Ball Growing**: Garg, Vazirani, Yannakakis 93.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Set $y_{ij} = 1 - x_{ij}$ for +ve edges. $z_{ij} = x_{ij}$ for -ve edges.



$$\min \sum_{(i,j)\in E(+)} w_{ij} y_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}| z_{ij}$$
$$y_{ij}, z_{ij} \geq 0 \qquad \forall i,j$$
$$\sum_{(u,v)\in P(ij)} y_{uv} + z_{ij} \geq 1 \qquad \forall i,j, \text{and } i\text{-}j \text{ path } P(ij)$$

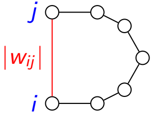
Sparsify $E(+)$. Store $E(-)$. $\Theta(n^2) \rightarrow \tilde{O}(n) + |E(-)|$ variables.
$\Theta(n^3)$ Constraints $\rightarrow$ Exponentially many constraints!
Solve LP (ellipsoid) & **Ball Growing**: Garg, Vazirani, Yannakakis 93.

MWM on the dual. $\tilde{O}(n + |E(-)|)$ space and $\tilde{O}(n^2)$ time. ACGMW 15.

Min-Disagreement

Equivalent to Max-Agreement at optimality. Not in approximation.

$x_{ij} = 1$ if in same group, and 0 otherwise. $E(+/-) = +/-$ve edge sets.
Set $y_{ij} = 1 - x_{ij}$ for $+$ve edges. $z_{ij} = x_{ij}$ for -ve edges.



$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E(+)} w_{ij} y_{ij} + \sum_{(i,j)\in E(-)} |w_{ij}| z_{ij} & \\
& y_{ij}, z_{ij} \geq 0 & \forall i, j \\
& \sum_{(u,v)\in P(ij)} y_{uv} + z_{ij} \geq 1 & \forall i, j, \text{and } i\text{-}j \text{ path } P(ij)
\end{aligned}
$$

Sparsify $E(+)$. Store $E(-)$. $\Theta(n^2) \rightarrow \tilde{O}(n) + |E(-)|$ variables.
$\Theta(n^3)$ Constraints $\rightarrow$ Exponentially many constraints!
Solve LP (ellipsoid) & **Ball Growing**: Garg, Vazirani, Yannakakis 93.

MWM on the dual. $\tilde{O}(n + |E(-)|)$ space and $\tilde{O}(n^2)$ time. ACGMW 15.
Round infeasible primal (the running average). Success $\rightarrow$ done.
Failure $\rightarrow$ violated constraint(s) $\rightarrow$ point needed for MWM on Dual.

Up Next ...

Fast and approximate recap of fast and approximate convex optimization.
Multiplicative Weights Method (MWM). LP version. Oracles.
Example: Bipartite Matching. MWM on Streams.

Global (Cut)-Sparsification. Single pass.

(a) Multiplicative Weights Method on SDPs.
Example: Correlation Clustering. Max-version.

(b) Multiplicative Weights Method on LPs.
Example: Correlation Clustering. Min-version.

New relaxations + oracle. Benefits in running time + space. Both cases.

Iterative (local) (Cut)-Sparsification. Multiples passes, Batch modes.
Example 2. Non-bipartite Matching. $(1+\epsilon)$-apx.
Cornerstone of Combinatorial Optimization, Dantzig Decompositions.
Benefits in time+space+adaptivity.

Wrap-up.

New Strategy: Putting the Horse before the Cart

A **natural** algorithm for non-bipartite matching. Ahn, Guha 15.

1. Find an initial solution.
2. We assign some prices to the edges.
3. For $O(10/\epsilon)$ steps:
   3.1 Sample $n^{1.1}$ edges using current prices.
   3.2 Find the best weighted matching in the sample.
   3.3 Maintain the best weight matching found (say β) so far.
   3.4 Update the prices.

New Strategy: Putting the Horse before the Cart

A **natural** algorithm for non-bipartite matching. Ahn, Guha 15.

1. Find an initial solution.
2. We assign some prices to the edges.
3. For $O(10/\epsilon)$ steps:
   3.1 **Sample** $n^{1.1}$ edges using current prices.
   3.2 Find the best weighted matching in the sample.
   3.3 Maintain the best weight matching found (say β) so far.
   3.4 **Update** the prices.

New Strategy: Putting the Horse before the Cart

A **natural** algorithm for non-bipartite matching. Ahn, Guha 15.

1. Find an initial solution.
2. We assign some prices to the edges.
3. For $O(10/\epsilon)$ steps:
   - 3.1 **Sample** $n^{1.1}$ edges using current prices.
   - 3.2 Find the best weighted matching in the sample.
   - 3.3 Maintain the best weight matching found (say β) so far.
   - 3.4 **Update** the prices.

Update:

1. Subdivide sampled edges in $t = O(\frac{1}{\epsilon} \log n)$ blocks
2. Simulate t steps of a primal-dual algorithm trying to prove Opt $\approx \beta$.
3. Obtain new prices on the edges.

New Strategy: Putting the Horse before the Cart

A **natural** algorithm for non-bipartite matching. Ahn, Guha 15.

1. Find an initial solution. Of the dual problem. (A trend?)
2. We assign some prices to the edges.
3. For $O(10/\epsilon)$ steps:
   - 3.1 **Sample** $n^{1.1}$ edges using current prices.
   - 3.2 Find the best weighted matching in the sample.
   - 3.3 Maintain the best weight matching found (say β) so far.
   - 3.4 **Update** the prices.

Update:

1. Subdivide sampled edges in $t = O(\frac{1}{\epsilon}\log n)$ blocks
2. Simulate t steps of a primal-dual algorithm trying to prove Opt $\approx \beta$. Feasible Dual $\leq \beta(1 + O(\epsilon))$.
3. Obtain new prices on the edges.

New Strategy: Putting the Horse before the Cart

A **natural** algorithm for non-bipartite matching. Ahn, Guha 15.

1. Find an initial solution. Of the dual problem. (A trend?)
2. We assign some prices to the edges.
3. For $O(10/\epsilon)$ steps:
   - 3.1 **Sample** $n^{1.1}$ edges using current prices.
   - 3.2 Find the best weighted matching in the sample.
   - 3.3 Maintain the best weight matching found (say β) so far.
   - 3.4 **Update** the prices.

Update:
1. Subdivide sampled edges in $t = O(\frac{1}{\epsilon}\log n)$ blocks
2. Simulate t steps of a primal-dual algorithm trying to prove Opt $\approx \beta$. Feasible Dual $\leq \beta(1 + O(\epsilon))$.
3. Obtain new prices on the edges.

$\mathbf{u}_{ij}$: the weight in MWM corresponding to constraint (i, j) of the dual. signals if the edge relevant/not.

New Strategy: Putting the Horse before the Cart

A **natural** algorithm for non-bipartite matching. Ahn, Guha 15.

1. Find an initial solution. Of the dual problem. (A trend?)
2. We assign some prices to the edges.
3. For $O(10/\epsilon)$ steps:
   - 3.1 **Sample** $n^{1.1}$ edges using current prices.
   - 3.2 Find the best weighted matching in the sample.
   - 3.3 Maintain the best weight matching found (say β) so far.
   - 3.4 **Update** the prices.

Update:

1. Subdivide sampled edges in $t = O(\frac{1}{\epsilon}\log n)$ blocks
2. Simulate t steps of a primal-dual algorithm trying to prove Opt $\approx \beta$. Feasible Dual $\leq \beta(1 + O(\epsilon))$.
3. Obtain new prices on the edges.

$\mathbf{u}_{ij}$: the weight in MWM corresponding to constraint (i, j) of the dual. signals if the edge relevant/not.

New Strategy: Putting the Horse before the Cart

A **natural** algorithm for non-bipartite matching. Ahn, Guha 15.

1. Find an initial solution. Of the dual problem. (A trend?)
2. We assign some prices to the edges.
3. For $O(10/\epsilon)$ steps:
   3.1 **Sample** $n^{1.1}$ edges using current prices.
   3.2 Find the best weighted matching in the sample.
   3.3 Maintain the best weight matching found (say β) so far.
   3.4 **Update** the prices.

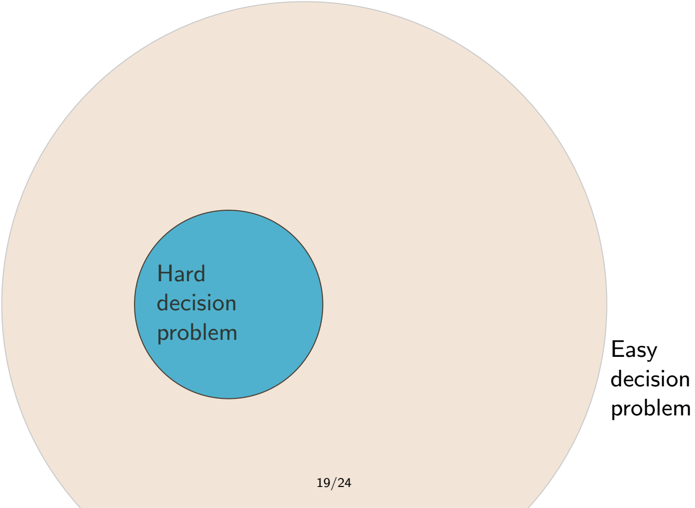
Update:
1. Subdivide sampled edges in $t = O(\frac{1}{\epsilon}\log n)$ blocks
2. Simulate t steps of a primal-dual algorithm trying to prove Opt $\approx \beta$. Feasible Dual $\leq \beta(1 + O(\epsilon))$.
3. Obtain new prices on the edges.

$\mathbf{u}_{ij}$: the weight in MWM corresponding to constraint (i, j) of the dual. signals if the edge relevant/not. **Sparsify those.**

Sparsification reveals a **subgraph** containing a near optimal solution. But only at near-optimality.

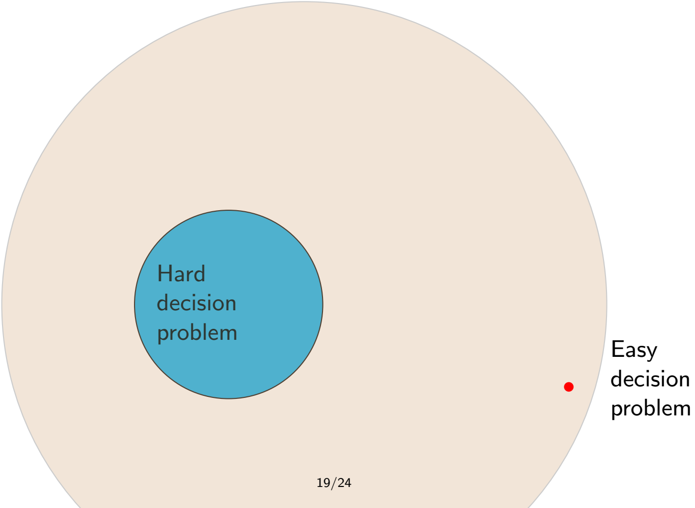
Dantzig Decompositions

A running average view (primal space).



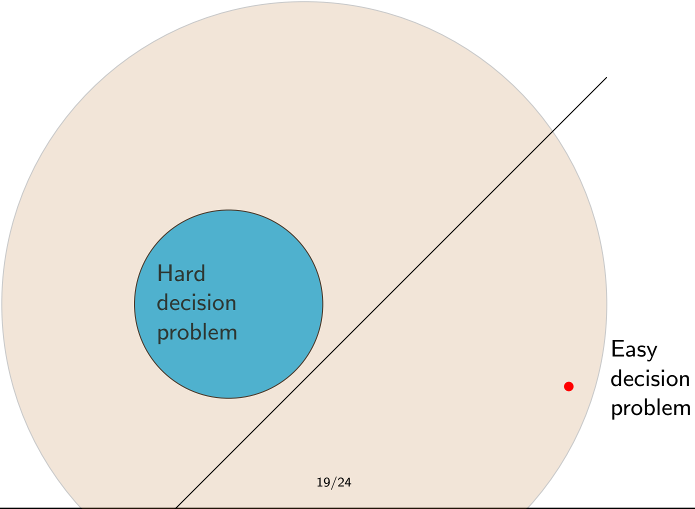
Dantzig Decompositions

A running average view (primal space).



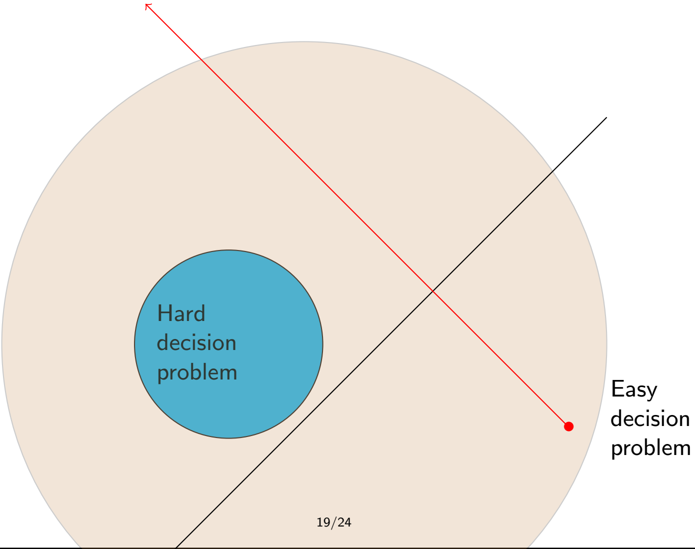
Dantzig Decompositions

A running average view (primal space).



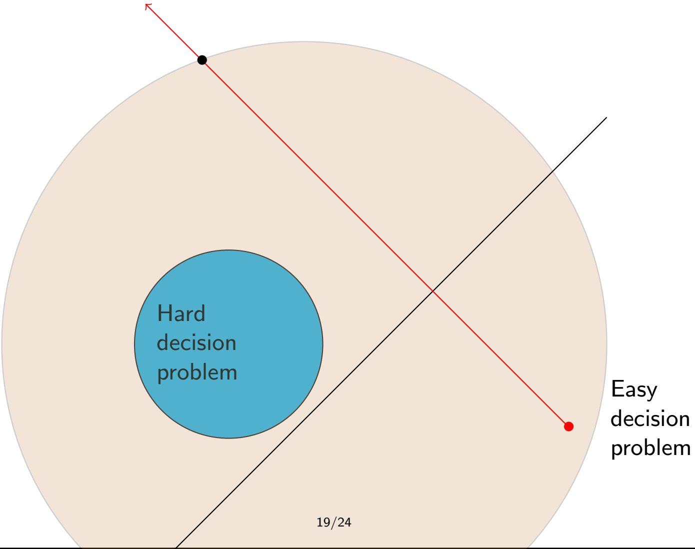
Dantzig Decompositions

A running average view (primal space).



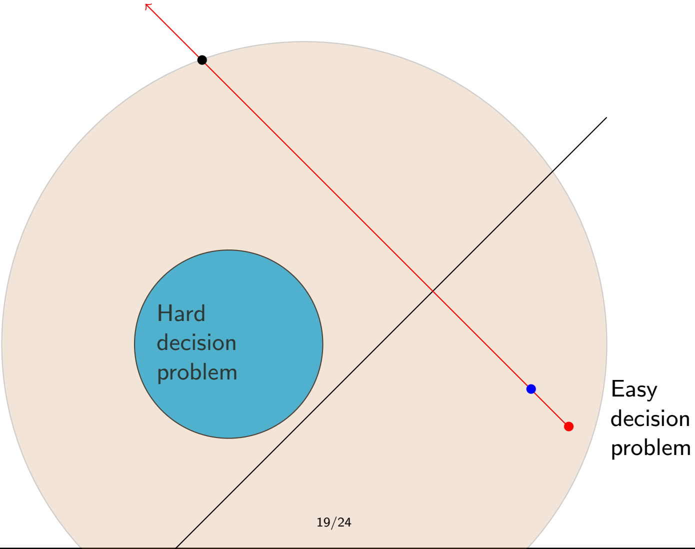
Dantzig Decompositions

A running average view (primal space).



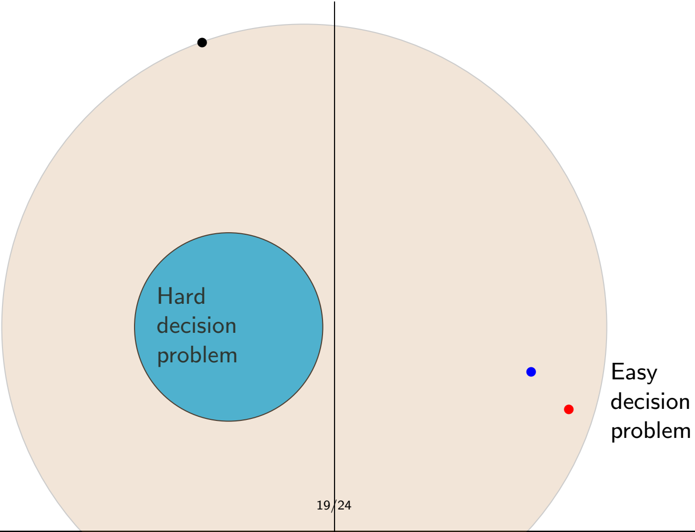
Dantzig Decompositions

A running average view (primal space).



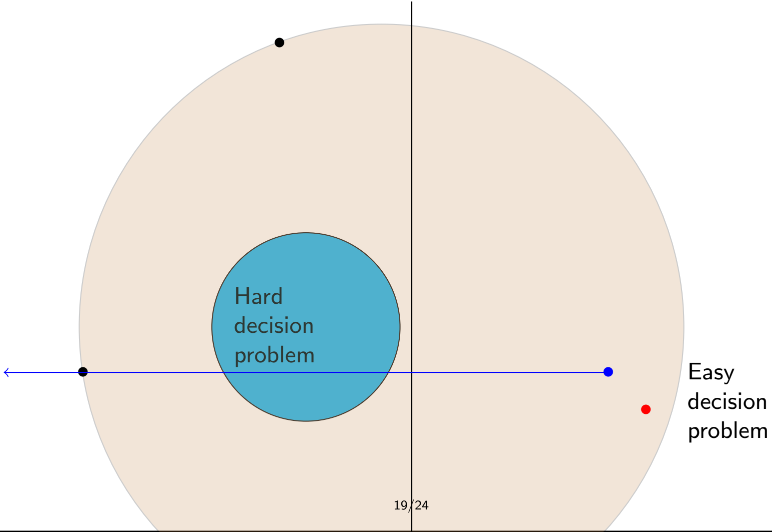
Dantzig Decompositions

A running average view (primal space).



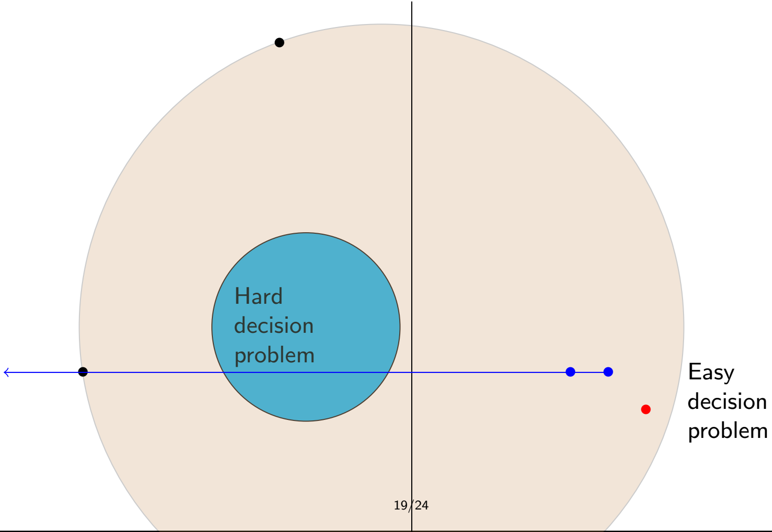
Dantzig Decompositions

A running average view (primal space).



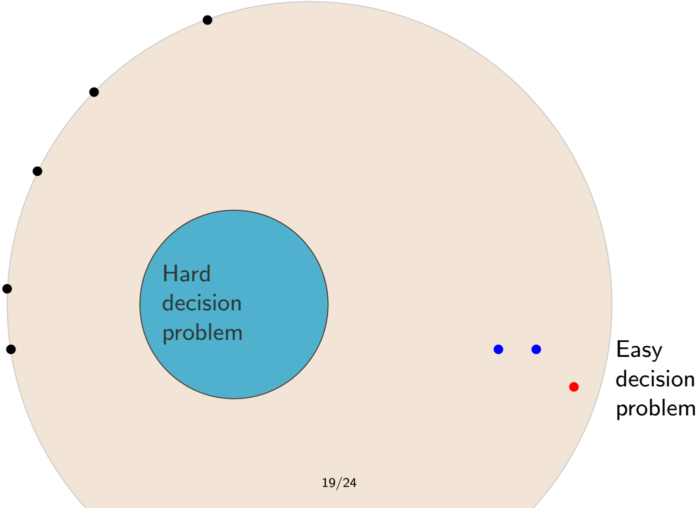
Dantzig Decompositions

A running average view (primal space).



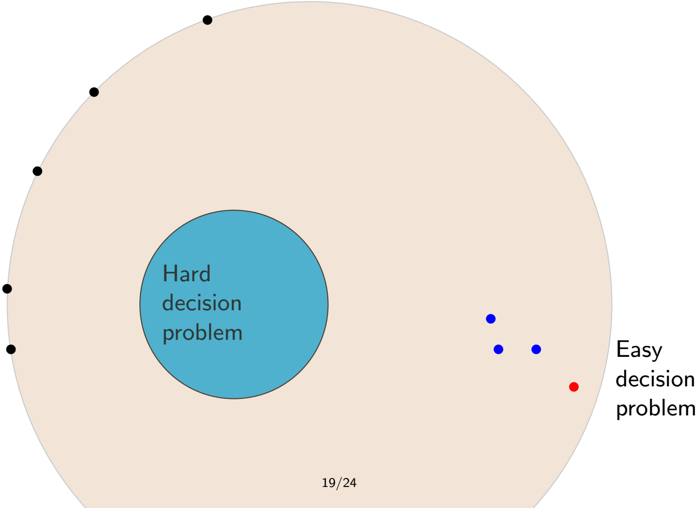
Dantzig Decompositions

A running average view (primal space).



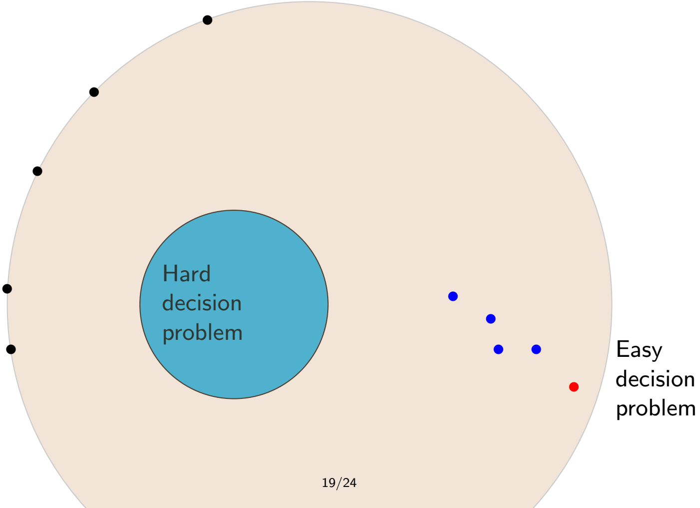
Dantzig Decompositions

A running average view (primal space).



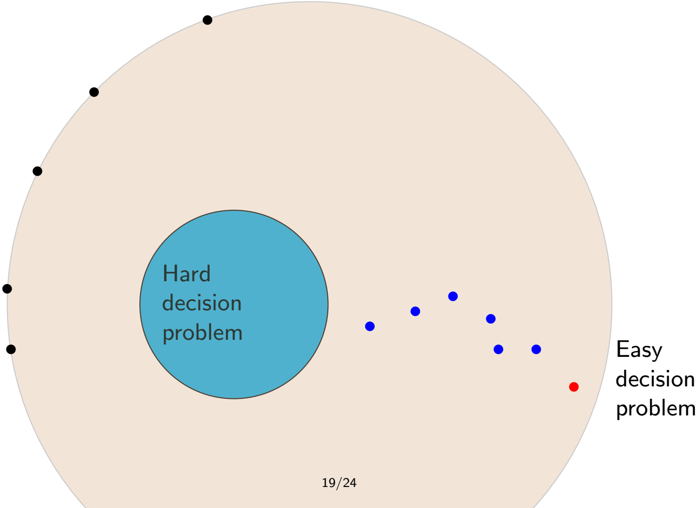
Dantzig Decompositions

A running average view (primal space).



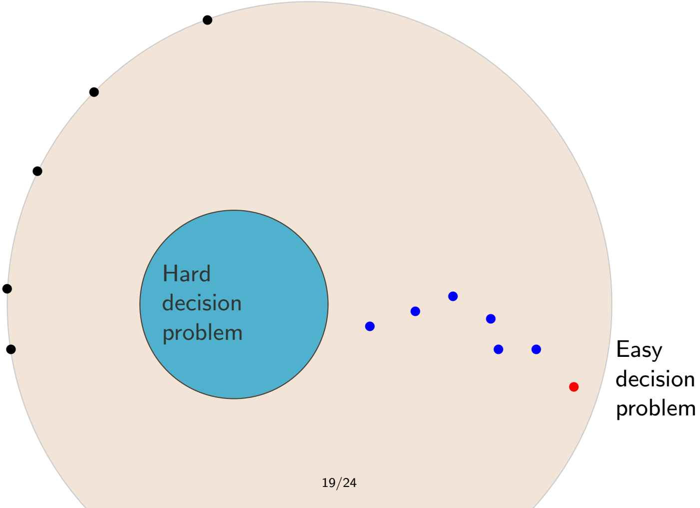
Dantzig Decompositions

A running average view (primal space).



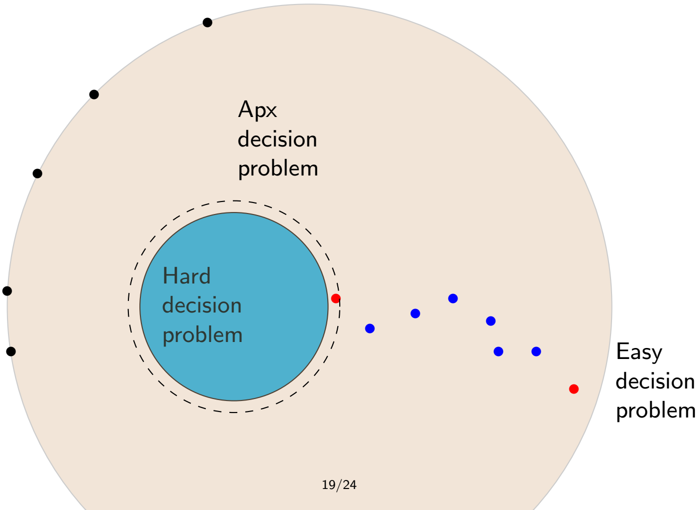
Dantzig Decompositions

A running average view (primal space).



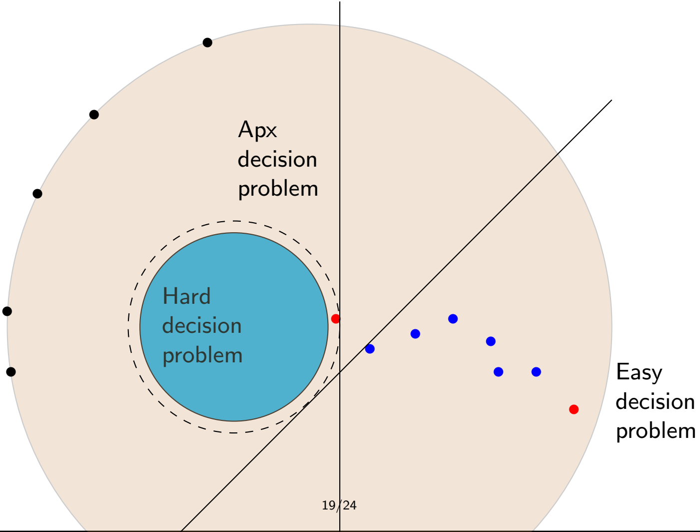
Dantzig Decompositions

A running average view (primal space).



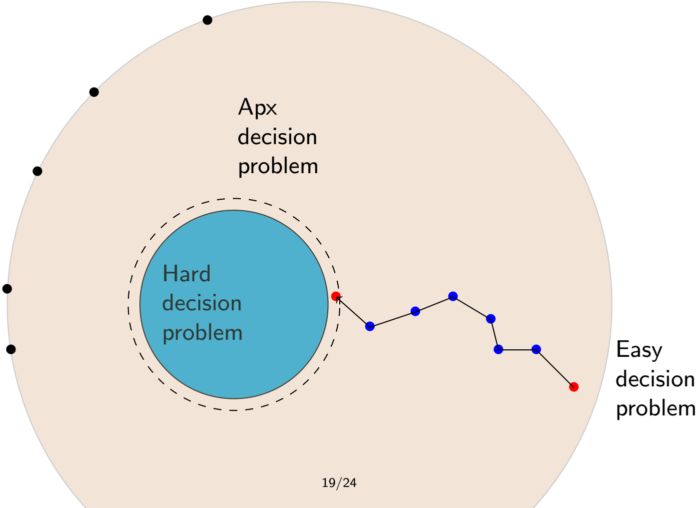
Dantzig Decompositions

A running average view (primal space).



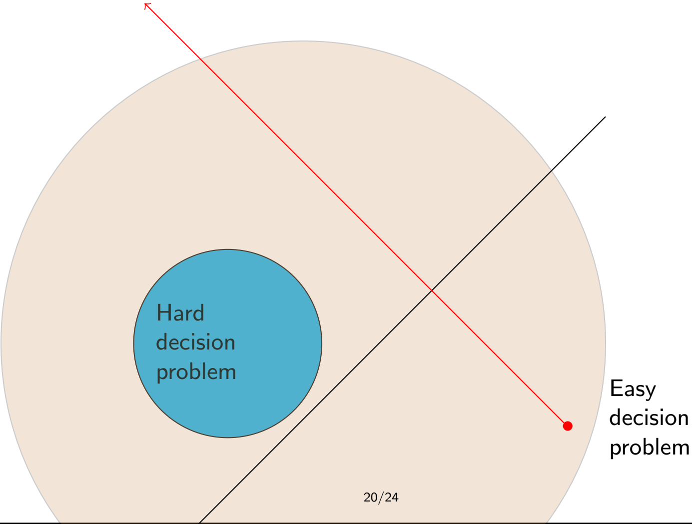
Dantzig Decompositions

A running average view (primal space).



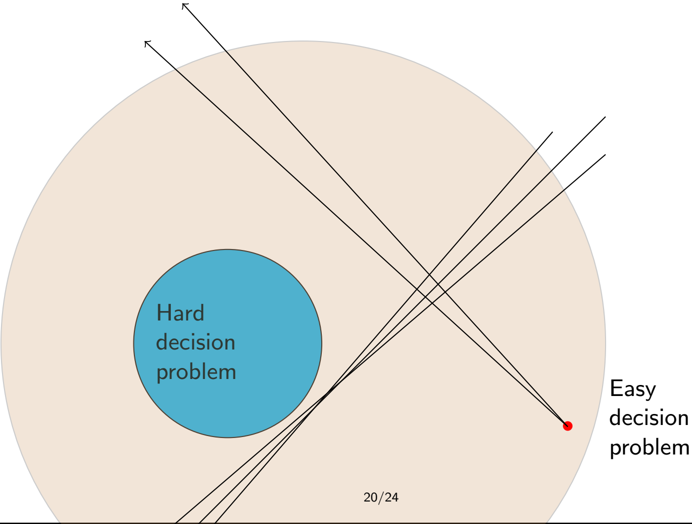
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



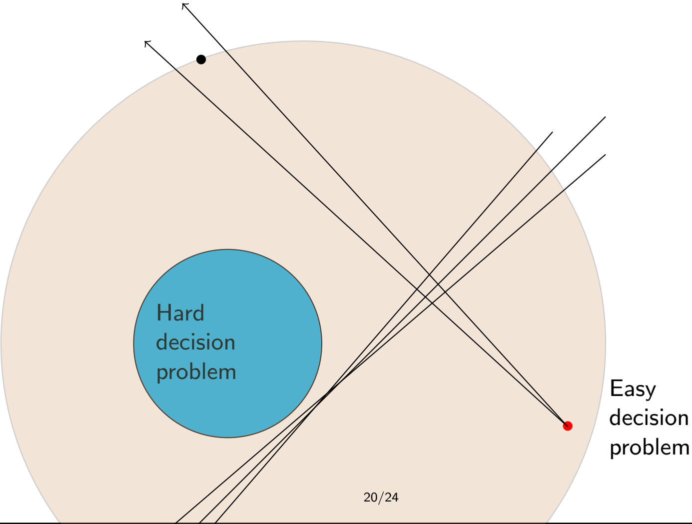
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



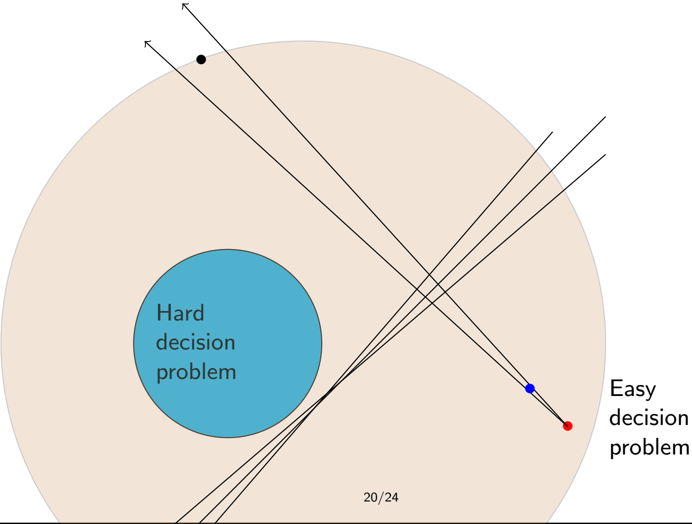
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



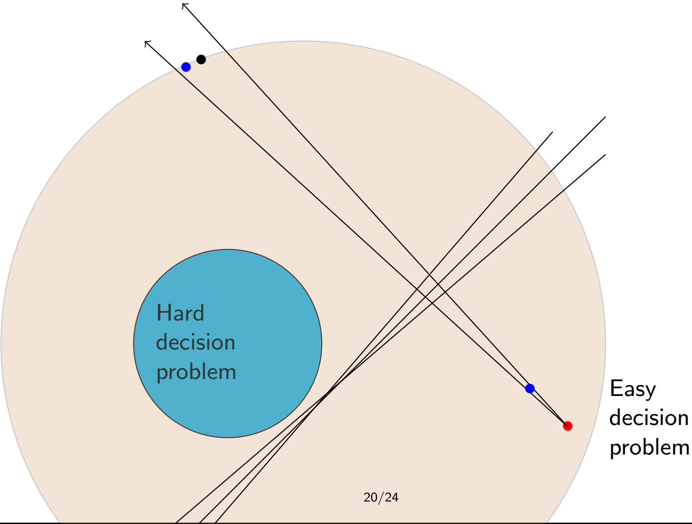
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



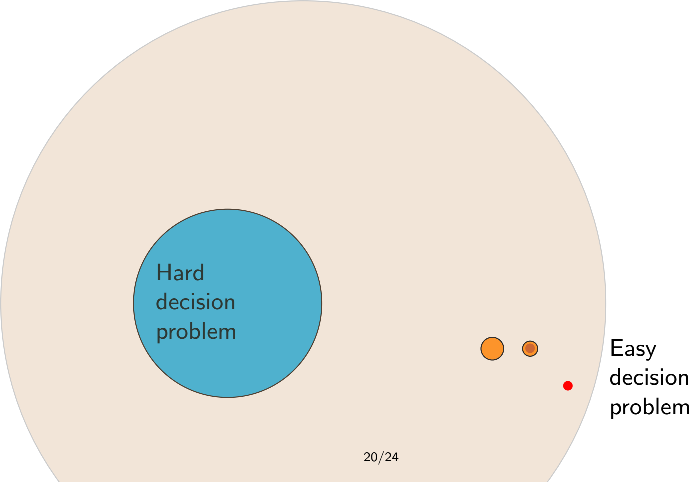
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



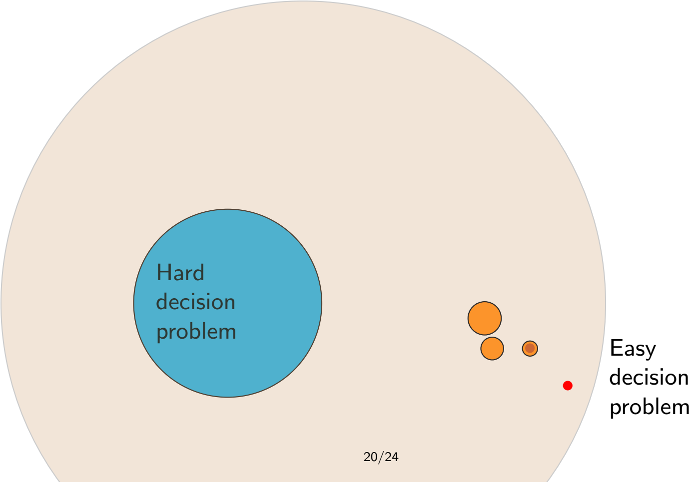
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



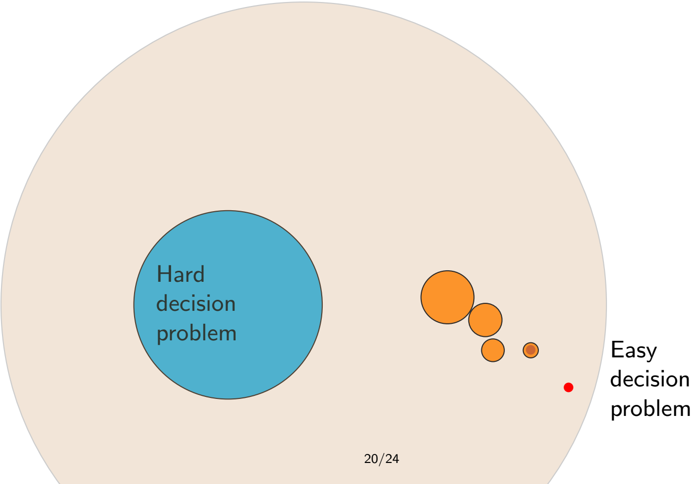
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



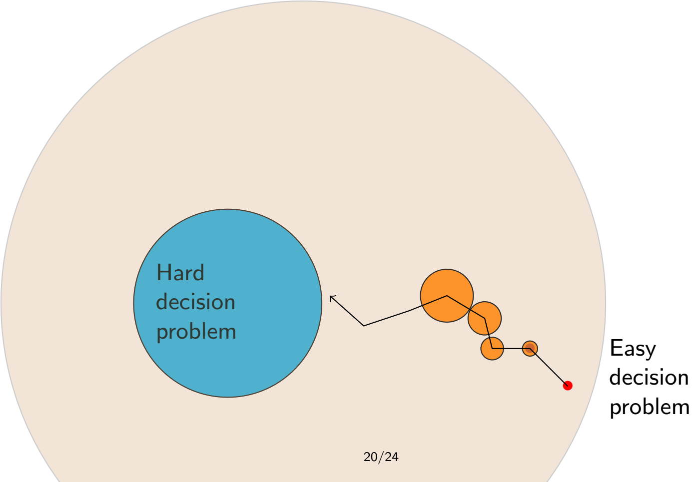
Sparsifications and Dantzig Decompositions

What if we sparsify $\mathbf{u}$?



Sparsifications and Dantzig Decompositions

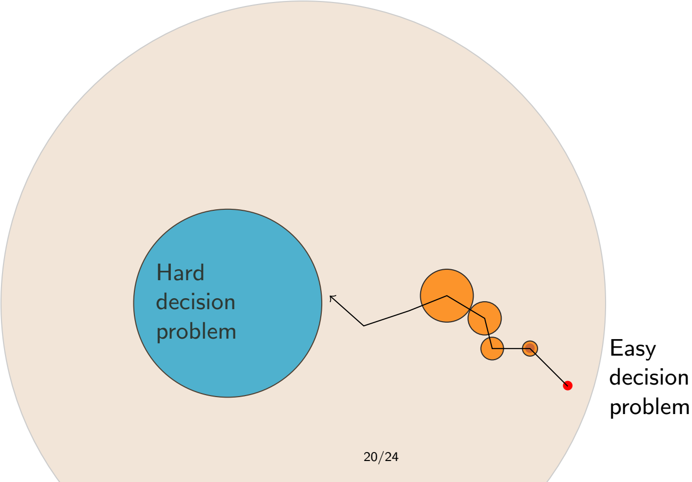
What if we sparsify $\mathbf{u}$?



Sparsifications and Dantzig Decompositions

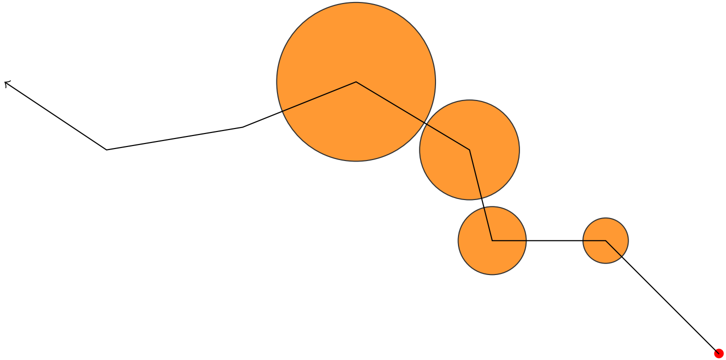
What if we sparsify $\mathbf{u}$?

Construct multiple sparsifications in parallel. Use sequentially.



Sparsify in Parallel, Use Sequentially

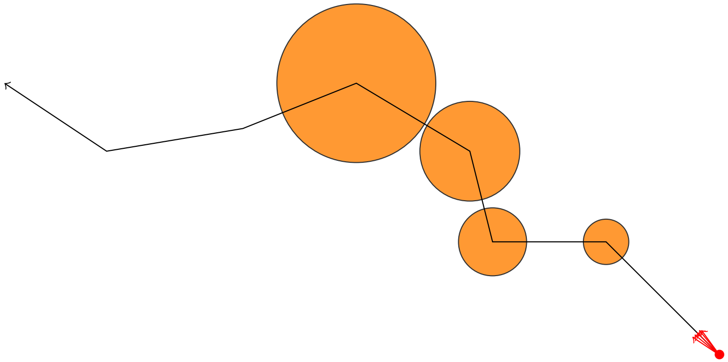
We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.



Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.

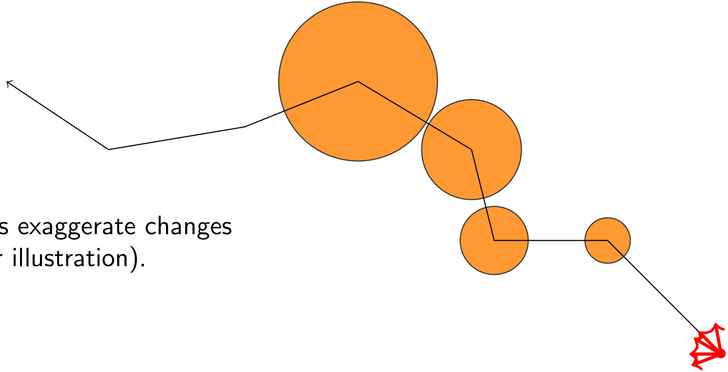
$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5 \mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.



Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.

$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5 \mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.

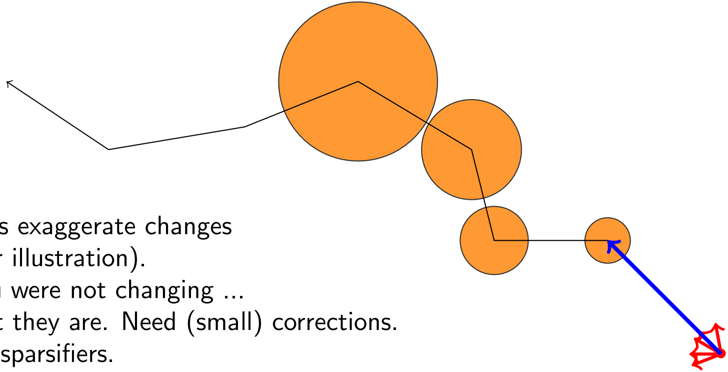


Lets exaggerate changes
(for illustration).

Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.

$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5 \mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.

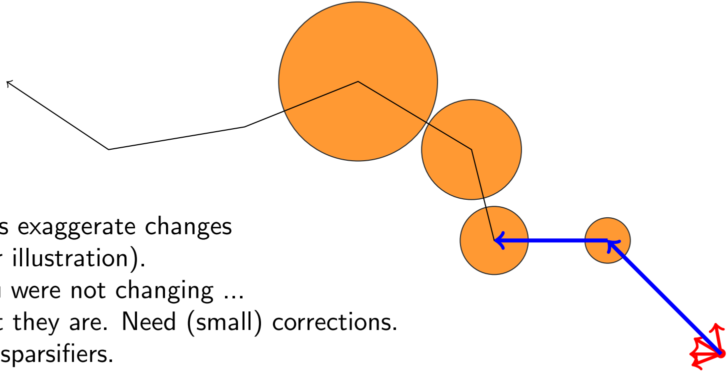


Lets exaggerate changes
(for illustration).
If $\mathbf{u}$ were not changing ...
But they are. Need (small) corrections.
Presparsifiers.

Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.

$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5 \mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.

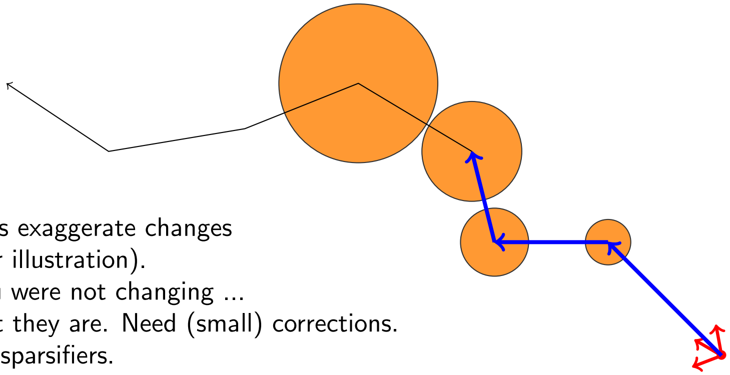


Lets exaggerate changes
(for illustration).
If $\mathbf{u}$ were not changing ...
But they are. Need (small) corrections.
Presparsifiers.

Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.

$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5 \mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.

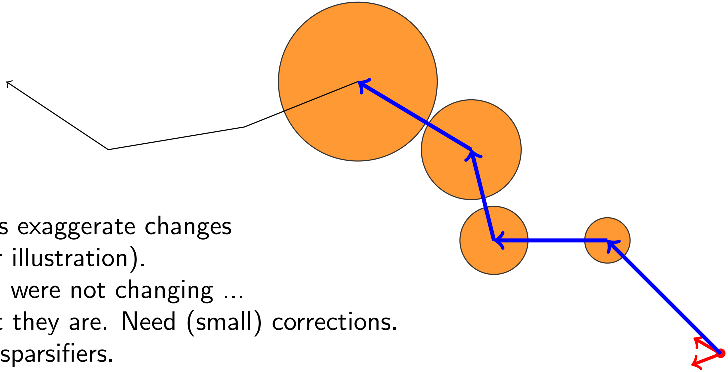


Lets exaggerate changes
(for illustration).
If $\mathbf{u}$ were not changing ...
But they are. Need (small) corrections.
Presparsifiers.

Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i \mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i (1+\epsilon)^{(\mathbf{A}_i\mathbf{y} - \mathbf{b}_i)/\mathbf{b}_i \rho}$.

$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5 \mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.

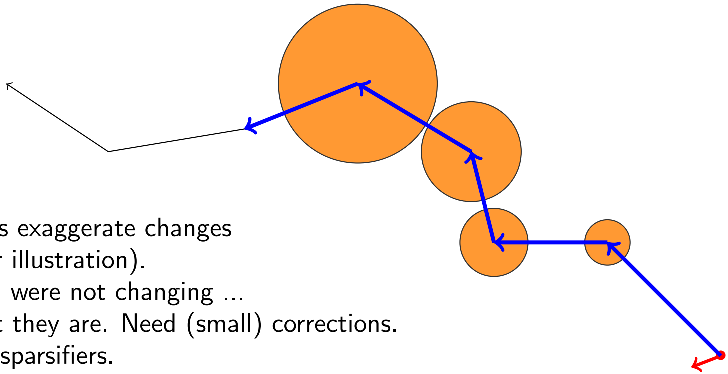


Lets exaggerate changes
(for illustration).
If $\mathbf{u}$ were not changing ...
But they are. Need (small) corrections.
Presparsifiers.

Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.

$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5\mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.

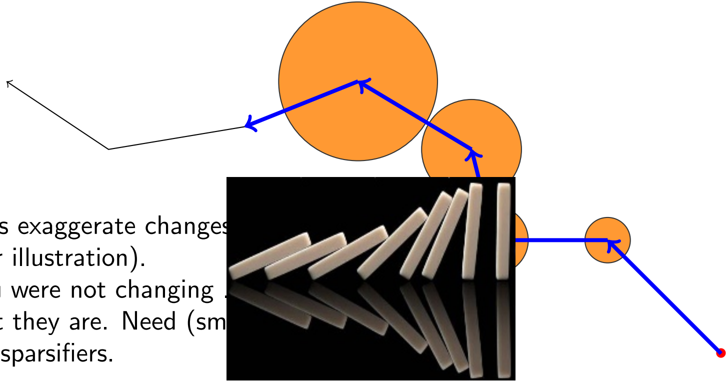


Lets exaggerate changes
(for illustration).
If $\mathbf{u}$ were not changing ...
But they are. Need (small) corrections.
Presparsifiers.

Sparsify in Parallel, Use Sequentially

We saw a version of sketch in parallel, use sequentially in connectivity.
Question: Where will we be after 5 steps of MWM?
Recall: If $\mathbf{A}_i\mathbf{y} > \mathbf{b}_i$: raise $\mathbf{u}_i$, i.e., $\mathbf{u}_i \leftarrow \mathbf{u}_i(1+\epsilon)^{(\mathbf{A}_i\mathbf{y}-\mathbf{b}_i)/\mathbf{b}_i\rho}$.

$\mathbf{u}_i(5) \in (1 \pm \epsilon)^5 \mathbf{u}_i$. Construct 5 independent sparsifications of $\mathbf{u}$.



Lets exaggerate changes
(for illustration).
If $\mathbf{u}$ were not changing .
But they are. Need (sm
Presparsifiers.

Cuts and Sparsification

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\beta^* = \max \quad \sum_{(i,j)} w_{ij} y_{ij}$$

$$\sum_j y_{ij} \quad \leq 1 \quad \forall i$$

$$y_{ij} \quad \geq 0$$

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$
\begin{aligned}
\beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\
\sum_j y_{ij} \quad & \leq 1 \quad \forall i \quad \text{(Cut constraint!)} \\
y_{ij} \quad & \geq 0
\end{aligned}
$$

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$
\begin{aligned}
\beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\
\sum_j y_{ij} \quad & \leq 1 \quad \forall i \quad \text{(Cut constraint!)} \\
\sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\
y_{ij} \quad & \geq 0
\end{aligned}
$$

Cuts and Sparsification

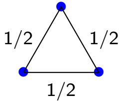
(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

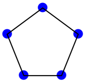
$$\beta^* = \max \quad \sum_{(i,j)} w_{ij} y_{ij}$$

$$\sum_j y_{ij} \leq 1 \quad \forall i \quad \text{(Cut constraint!)}$$

$$\sum_{i,j \in U} y_{ij} \leq \lfloor |U|/2 \rfloor \quad \forall U$$

$$y_{ij} \geq 0$$

Rules out: 



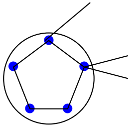
Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\begin{aligned}
\beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\
\sum_j y_{ij} \quad & \leq 1 \quad \forall i \quad \text{(Cut constraint!)} \\
\sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\
y_{ij} \quad & \geq 0
\end{aligned}$$

$$\sum_{i,j \in U} y_{ij} \leq \lfloor |U|/2 \rfloor \quad \Longleftrightarrow \sum_{i \in U} \left(\sum_j y_{ij} \right) - \left(\sum_{i \in U, j \notin U} y_{ij} \right) \leq 2 \lfloor |U|/2 \rfloor$$

$\sum_{i \in U, j \notin U} y_{ij} = Cut(U, V-U)$.



Cuts and Sparsification

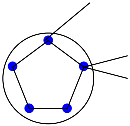
(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\begin{aligned}
\beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\
\sum_j y_{ij} \quad & \leq 1 \quad \forall i \\
\sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\
y_{ij} \quad & \geq 0
\end{aligned}$$

$$\sum_{i,j \in U} y_{ij} \leq \lfloor |U|/2 \rfloor \iff \sum_{i \in U} \left(\sum_j y_{ij} \right) - \left(\sum_{i \in U, j \notin U} y_{ij} \right) \leq 2 \lfloor |U|/2 \rfloor$$

$\sum_{i \in U, j \notin U} y_{ij} = Cut(U, V - U)$.

Find small cuts (with odd vertex sizes).



Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\begin{aligned} \beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\ \sum_j y_{ij} \quad & \leq 1 \quad \forall i \\ \sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\ y_{ij} \quad & \geq 0 \end{aligned}$$

$$\begin{aligned} & \beta^* = \min \sum_i x_i + \sum_U \left\lfloor \frac{|U|}{2} \right\rfloor z_U \\ \mathbf{u}_{ij} : \quad & x_i + x_j + \sum_{i,j \in U} z_U \geq w_{ij} \quad \forall (i,j) \in E \\ & x_i, z_U \geq 0 \end{aligned}$$

Find small cuts (with odd vertex sizes).

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$
\begin{aligned}
\beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\
\sum_j y_{ij} \quad & \leq 1 \quad \forall i \\
\sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\
y_{ij} \quad & \geq 0
\end{aligned}
$$

$$
\begin{aligned}
& \beta^* = \min \sum_i x_i + \sum_U \left\lfloor \frac{|U|}{2} \right\rfloor z_U \\
\mathbf{u}_{ij} : \quad & x_i + x_j + \sum_{i,j \in U} z_U \geq w_{ij} \quad \forall (i,j) \in E \\
& x_i, z_U \geq 0
\end{aligned}
$$

Find small cuts (with odd vertex sizes).
Standard Algorithm: Augment, contract blossoms, ... (many rounds).
Signature: feasible,. . . ,feasible (larger), . . . , feasible, (near) optimal

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\begin{aligned}
\beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\
\sum_j y_{ij} \quad & \leq 1 \quad \forall i \\
\sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\
y_{ij} \quad & \geq 0
\end{aligned}$$

$$\begin{aligned}
& \beta^* = \min \sum_i x_i + \sum_U \left\lfloor \frac{|U|}{2} \right\rfloor z_U \\
\mathbf{u}_{ij} : \quad & x_i + x_j + \sum_{i,j \in U} z_U \geq w_{ij} \quad \forall (i,j) \in E \\
& x_i, z_U \geq 0
\end{aligned}$$

Find small cuts (with odd vertex sizes).
Standard Algorithm: Augment, contract blossoms, ... (many rounds).
Signature: feasible,. . . ,feasible (larger), . . . , feasible, (near) optimal

Signature of Algorithms we just saw:
Bipartite: ($O(\epsilon^{-2} \log n)$ rounds, no sparsification)
infeasible,. . . ,infeasible (smaller), . . . , feasible, (near) optimal

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\begin{aligned} \beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\ \sum_j y_{ij} \quad & \leq 1 \quad \forall i \\ \sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\ y_{ij} \quad & \geq 0 \end{aligned}$$

$$\begin{aligned} & \beta^* = \min \sum_i x_i + \sum_U \left\lfloor \frac{|U|}{2} \right\rfloor z_U \\ \mathbf{u}_{ij} : \quad & x_i + x_j + \sum_{i,j \in U} z_U \geq w_{ij} \quad \forall (i,j) \in E \\ & x_i, z_U \geq 0 \end{aligned}$$

Find small cuts (with odd vertex sizes).
Standard Algorithm: Augment, contract blossoms, ... (many rounds).
Signature: feasible,. . . ,feasible (larger), . . . , feasible, (near) optimal

Signature of Algorithms we just saw:
Bipartite: ($O(\epsilon^{-2} \log n)$ rounds, no sparsification)
infeasible,. . . ,infeasible (smaller), . . . , feasible, (near) optimal

Non bipartite: ?

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\begin{aligned} \beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\ \sum_j y_{ij} \quad & \leq 1 \quad \forall i \\ \sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\ y_{ij} \quad & \geq 0 \end{aligned}$$

$$\begin{aligned} & \beta^* = \min \sum_i x_i + \sum_U \left\lfloor \frac{|U|}{2} \right\rfloor z_U \\ \mathbf{u}_{ij} : \quad & x_i + x_j + \sum_{i,j \in U} z_U \geq w_{ij} \quad \forall (i,j) \in E \\ & x_i, z_U \geq 0 \end{aligned}$$

Find small cuts (with odd vertex sizes).
Standard Algorithm: Augment, contract blossoms, ... (many rounds).
Signature: feasible,...,feasible (larger), ..., feasible, (near) optimal

Signature of Algorithms we just saw:
Bipartite: ($O(\epsilon^{-2} \log n)$ rounds, no sparsification)
infeasible,...,infeasible (smaller), ..., feasible, (near) optimal

Non bipartite: ($O(1/\epsilon)$ rounds, sparsification)
infeasible dual, ..., (estimate of β^* is increasing), ..., (near) optimal

Cuts and Sparsification

(Again dropping $(i,j) \in E$ in the subscripts, $y_{ij} = y_{ji}$.)

$$\begin{aligned} \beta^* = \max \quad & \sum_{(i,j)} w_{ij} y_{ij} \\ \sum_j y_{ij} \quad & \leq 1 \quad \forall i \\ \sum_{i,j \in U} y_{ij} \quad & \leq \lfloor |U|/2 \rfloor \quad \forall U \\ y_{ij} \quad & \geq 0 \end{aligned}$$

$$\begin{aligned} & \beta^* = \min \sum_i x_i + \sum_U \left\lfloor \frac{|U|}{2} \right\rfloor z_U \\ \mathbf{u}_{ij} : \quad & x_i + x_j + \sum_{i,j \in U} z_U \geq w_{ij} \quad \forall (i,j) \in E \\ & x_i, z_U \geq 0 \end{aligned}$$

Find small cuts (with odd vertex sizes).
Standard Algorithm: Augment, contract blossoms, ... (many rounds).
Signature: feasible,. . . ,feasible (larger), . . . , feasible, (near) optimal

Signature of Algorithms we just saw:
Bipartite: ($O(\epsilon^{-2} \log n)$ rounds, no sparsification)
infeasible,. . . ,infeasible (smaller), . . . , feasible, (near) optimal

Non bipartite: ($O(1/\epsilon)$ rounds, sparsification)
infeasible dual, . . . , (estimate of β^* is increasing), . . . , (near) optimal
. . . . . . keep best matching seen so far, ... (near) optimal

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.
(2) LPs (MWM) on Streams.

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.
(2) LPs (MWM) on Streams.
(3) Remember a small number of weight values.

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.
(2) LPs (MWM) on Streams.
(3) Remember a small number of weight values.
(4) Compute in sketch (sparsified) space entirely. Correlation clustering.

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.
(2) LPs (MWM) on Streams.
(3) Remember a small number of weight values.
(4) Compute in sketch (sparsified) space entirely. Correlation clustering.
(5) May need to change the natural relaxations (convergence speed).

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.
(2) LPs (MWM) on Streams.
(3) Remember a small number of weight values.
(4) Compute in sketch (sparsified) space entirely. Correlation clustering.
(5) May need to change the natural relaxations (convergence speed).
(6) May need new relaxations for correctness.

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.
(2) LPs (MWM) on Streams.
(3) Remember a small number of weight values.
(4) Compute in sketch (sparsified) space entirely. Correlation clustering.
(5) May need to change the natural relaxations (convergence speed).
(6) May need new relaxations for correctness.
(7) Round first, ~~ask questions later~~, failing, take a dual step.

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.

(2) LPs (MWM) on Streams.

(3) Remember a small number of weight values.

(4) Compute in sketch (sparsified) space entirely. Correlation clustering.

(5) May need to change the natural relaxations (convergence speed).

(6) May need new relaxations for correctness.

(7) Round first, ~~ask questions later~~, failing, take a dual step.

(8) Dual-Primal algorithms may help. SDP, Nonbipartite Matching.

Wrap up

(1) Primitives: Sampling, Sketching and Sparsification.
(2) LPs (MWM) on Streams.
(3) Remember a small number of weight values.
(4) Compute in sketch (sparsified) space entirely. Correlation clustering.
(5) May need to change the natural relaxations (convergence speed).
(6) May need new relaxations for correctness.
(7) Round first, ~~ask questions later~~, failing, take a dual step.
(8) Dual-Primal algorithms may help. SDP, Nonbipartite Matching.
(9) Think differently. The real voyage of discovery ...

Thank You