

# How to Use SDPs in the Design of Fast Spectral Algorithms

Finding Balanced Cuts and Large Induced Expanders

Lorenzo Orecchia – UC Berkeley

Joint work with Nisheeth K. Vishnoi, Microsoft Research Bangalore.

# Graph Partitioning Methods

- Spectral algorithms
  - Favored in applications.
  - Fast in many cases of interest.
- Flow algorithms
  - Better approximations.
  - Usually slower in practice.
- SDP algorithms
  - Mostly theoretical applications.

# Spectral and SDP

- Spielman-Teng (ST) spectral algorithm for balanced-cut problem is an important primitive in **sparsification** and solutions of **systems of linear equations**.
- The combinatorial framework for solving SDPs of **Arora and Kale** has yielded **new fast algorithms** for many graph partitioning problems.
- **THIS TALK:**  
Use SDP methods to design a spectral algorithm for balanced cut achieving **better approximation** in the **same running time** as ST.

# Spectral and SDP

- Spielman-Teng (ST) spectral algorithm for balanced-cut problem is an important primitive in **sparsification** and solutions of **systems of linear equations**.
- The combinatorial framework for solving SDPs of **Arora and Kale** has yielded **new fast algorithms** for many graph partitioning problems.
- **THIS TALK:**  
Use SDP methods to design a spectral algorithm for balanced cut achieving **better approximation** in the **same running time** as ST.

**Additional advantages: simple analysis, simple to implement.**

# Notation

- Instance Graph: Undirected Unweighted Graph  $G$

$$G = (V, E_G)$$

$$|V| = n$$

$$|E_G| = m$$

- Degree of Vertex  $i$  is  $d_i$

- Volume of Set  $S$

$$\text{vol}(S) = \sum_{i \in S} d_i$$

# Conductance and Balance

- **Conductance of cut  $S$**

$$\phi(S) = \frac{|E(S, \bar{S})|}{\min\{\text{vol}(S), \text{vol}(\bar{S})\}}$$

- **Balance of cut  $S$**

$S$  is  $b$ -balanced iff

$$\min\{\text{vol}(S), \text{vol}(\bar{S})\} \geq b \cdot \text{vol}(V) = b \cdot 2m$$

# The Balanced-Cut Problem

- Decision Problem. On Input  $(G, \gamma, b)$ , either

- Output **b-balanced** cut  $S$  in  $G$  with

$$\phi(S) \leq \gamma$$

or

- Exhibit a **certificate** that for any  $b$ -balanced cut  $T$

$$\phi(T) > \gamma$$

# The Balanced-Cut Problem

- Decision Problem. On Input  $(G, \gamma, b)$ , either

- Output **b-balanced** cut  $S$  in  $G$  with

$$\phi(S) \leq \gamma$$

or

**NP-HARD**

- Exhibit a **certificate** that for any b-balanced cut  $T$

$$\phi(T) > \gamma$$

# Approximation Version

- Decision Problem. On Input  $(G, \gamma, b)$ , either

- Output **b-balanced** cut  $S$  in  $G$  with

$$\phi(S) \leq f(\gamma)$$

or

- Exhibit a **certificate** that for any b-balanced cut  $T$

$$\phi(T) > \gamma$$

# Approximation Version

- Decision Problem. On Input  $(G, \gamma, b)$ , either
  - Output **b-balanced** cut  $S$  in  $G$  with

$$\phi(S) \leq f(\gamma)$$

Or

- Exhibit a **certificate** that for any b-balanced cut  $T$

$$\phi(T) > \gamma$$

Algorithm	Method	$f(\gamma)$	Running Time
Leighton-Rao	Flow	$\gamma \cdot \log n$	$\tilde{O}(n^2)$
Arora-Rao-Vazirani	SDP	$\gamma \cdot \sqrt{\log n}$	$\tilde{O}(m + n^{1.5})$
ST, ACL	Spectral	$\sqrt{\gamma \cdot \log n}$	$\tilde{O}(m/\gamma)$

# Approximation Version

- Decision Problem. On Input  $(G, \gamma, b)$ , either
  - Output **b-balanced** cut  $S$  in  $G$  with

$$\phi(S) \leq f(\gamma)$$

Or

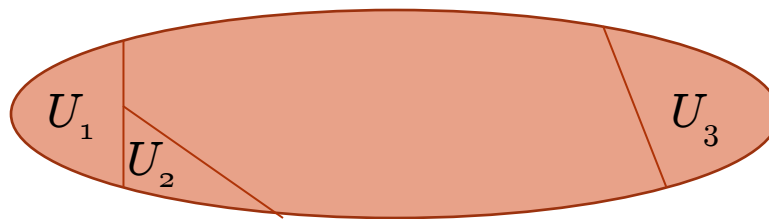
- Exhibit a **certificate** that for any b-balanced cut  $T$

$$\phi(T) > \gamma$$

Algorithm	Method	$f(\gamma)$	Running Time
Leighton-Rao	Flow	$\gamma \cdot \log n$	$\tilde{O}(n^2)$
Arora-Rao-Vazirani	SDP	$\gamma \cdot \sqrt{\log n}$	$\tilde{O}(m + n^{1.5})$
ST, ACL	Spectral	$\sqrt{\gamma \cdot \log n}$	$\tilde{O}(m/\gamma)$

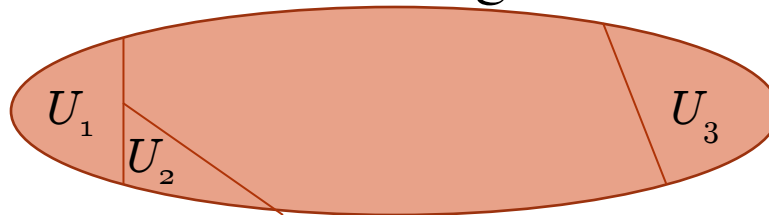
# The Spielman-Teng Algorithm

- Iterative algorithm.
- Cuts off small unbalanced cut at every iteration.
  - Compare with recursive eigenvector.



# The Spielman-Teng Algorithm

- Iterative algorithm.
- Cuts off sparse unbalanced cut at every iteration.
  - Compare with recursive eigenvector.

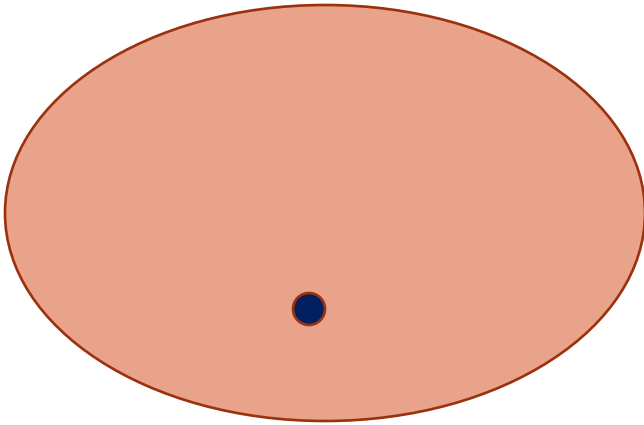


- Main idea: local random walks.
- Running time to cut  $U$  is proportional to

$$\tilde{O}(\text{vol}(U))$$

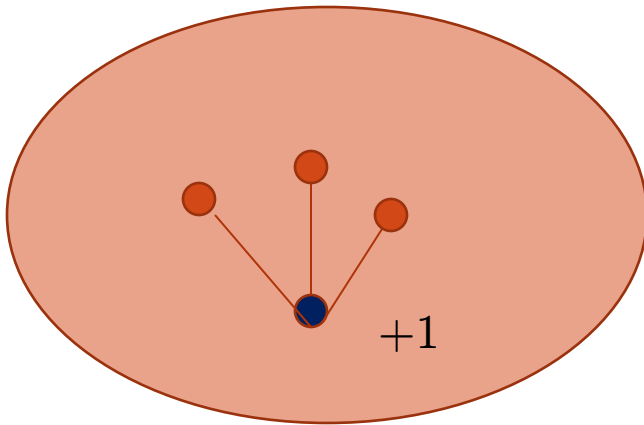
# Local Random Walks

- Pick random seed.
- Assign +1 unit of charge.
- Spread charge in graph by random walk averaging process



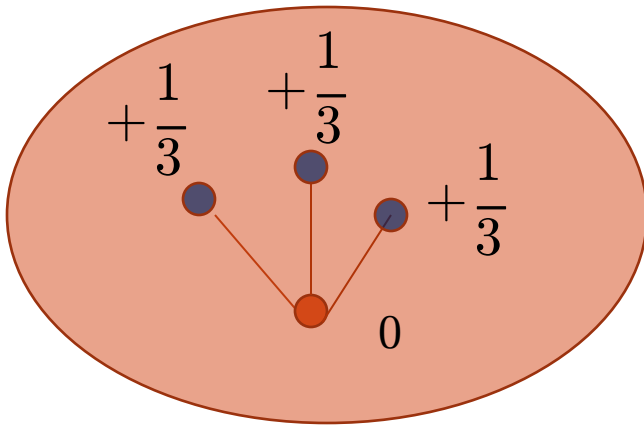
# Local Random Walks

- Pick random seed.
- Assign  $+1$  unit of charge.
- Spread charge in graph by random walk averaging process



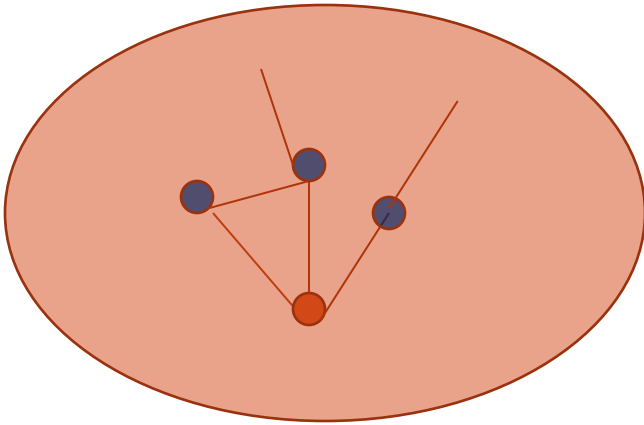
# Local Random Walks

- Pick random seed.
- Assign +1 unit of charge.
- Spread charge in graph by random walk averaging process



# Local Random Walks

- Pick random seed.
- Assign +1 unit of charge.
- Spread charge in graph by random walk averaging process
- Repeat for random number of steps  $[0, T]$ .

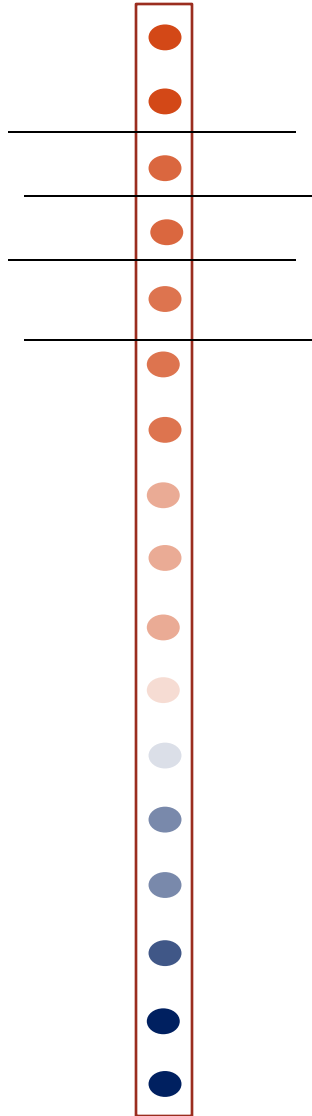


# Local Random Walks



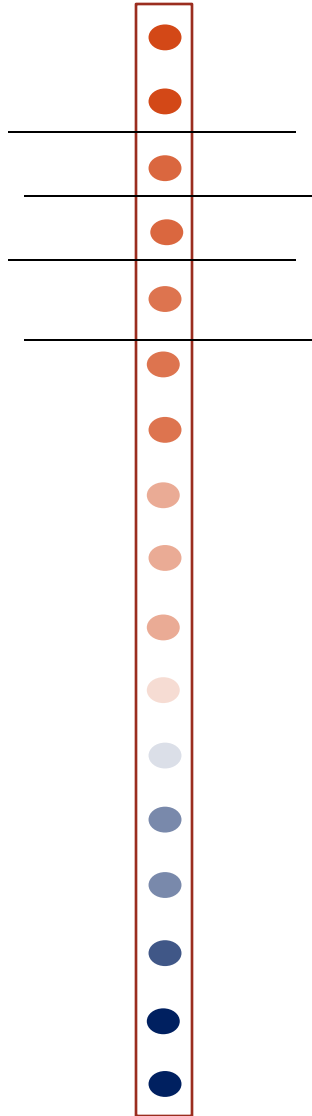
- Pick random seed vertex.
- Assign +1 unit of charge.
- Spread charge in graph by random walk averaging process
- Repeat for random number of steps  $[0, T]$ .
- Sort vertices by charge.

# Local Random Walks



- Pick random seed vertex.
- Assign +1 unit of charge.
- Spread charge in graph by random walk averaging process.
- Repeat for random number of steps  $[0, T]$ .
- Sort vertices by charge.
- Pick sweep cut of least conductance.

# Local Random Walks



- Pick random seed vertex.
- Assign +1 unit of charge.
- Spread charge in graph by random walk averaging process.
- Repeat for random number of steps  $[0, T]$ .
- Sort vertices by charge.
- Pick sweep cut of least conductance.

**NB:**

truncation of low values makes walk local.

# Different Local Random Walks

- ST: **effective resistance** walk, **uniform** distribution.

- Approximation:  $\sqrt{\gamma \log^3 n}$
- Running Time:  $\tilde{O}(m/\gamma)$

- ACL: **PageRank** walk, **geometric** distribution.

- Approximation:  $\sqrt{\gamma \cdot \log n}$
- Running Time:  $\tilde{O}(m/\gamma)$

- Chung: **heat kernel** walk, **Poisson** distribution.

- Approximation:  $\sqrt{\gamma}$
- Running Time: ---

# Our Result

On Input  $(G, \gamma, b)$ , we either

- Output **b-balanced** cut  $S$  in  $G$  with

$$\phi(S) \leq O(\sqrt{\gamma})$$

or

- Exhibit a **certificate** that for any  $b$ -balanced cut  $T$

$$\phi(T) > \gamma$$

in time  $\tilde{O}(m/\gamma)$ .

# Our Result

On Input  $(G, \gamma, b)$ , we either

- Output **b-balanced** cut  $S$  in  $G$  with

$$\phi(S) \leq O(\sqrt{\gamma})$$

or

- Exhibit a **certificate** that for any  $b$ -balanced cut  $T$

$$\phi(T) > \gamma$$

in time  $\tilde{O}(m/\gamma)$ .

**BETTER PRIMITIVE FOR SPARSIFICATION AND LINEAR SYSTEMS**

# Our Algorithm

- At iteration  $t$ , we start with graph  $G_t$  and a rate  $\eta_t$ .
- $G_0 = G$  and  $\eta_0 = 1$ .

At iteration  $t$ , simulate a small number of **heat kernel random walks** with rate  $\eta_t$  on  $G_t$ .

Use the resulting distributions to find either

- a sparse balanced cut, or
- an obstacle to finding a sparse balanced cut:
  - The local random walks are not mixing enough.
  - We find a sparse unbalanced cut.

Remove obstacle by **modifying graph** and iterate.

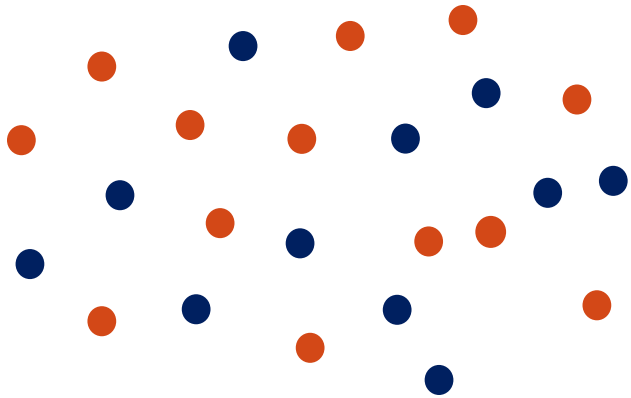
# Our Algorithm – One Iteration

● = +1 charge

● = -1 charge

At iteration  $t$ , repeat  $O(\log n)$  times:

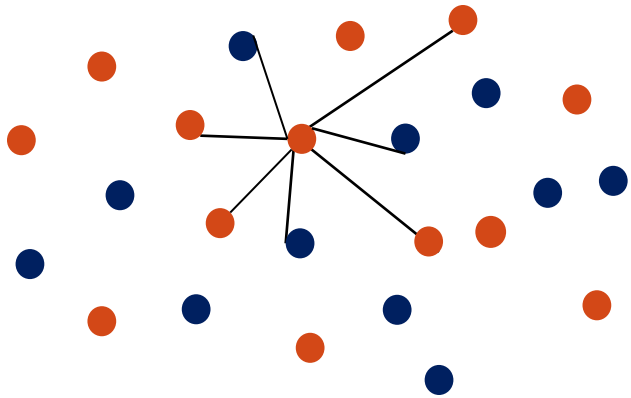
- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.



# Our Algorithm – One Iteration

● = +1 charge

● = -1 charge



At iteration  $t$ , repeat  $O(\log n)$  times:

- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.
- Mix charge along edges of  $G_t$  using heat kernel with rate  $\eta_t$

# Our Algorithm – One Iteration

● = +1 charge



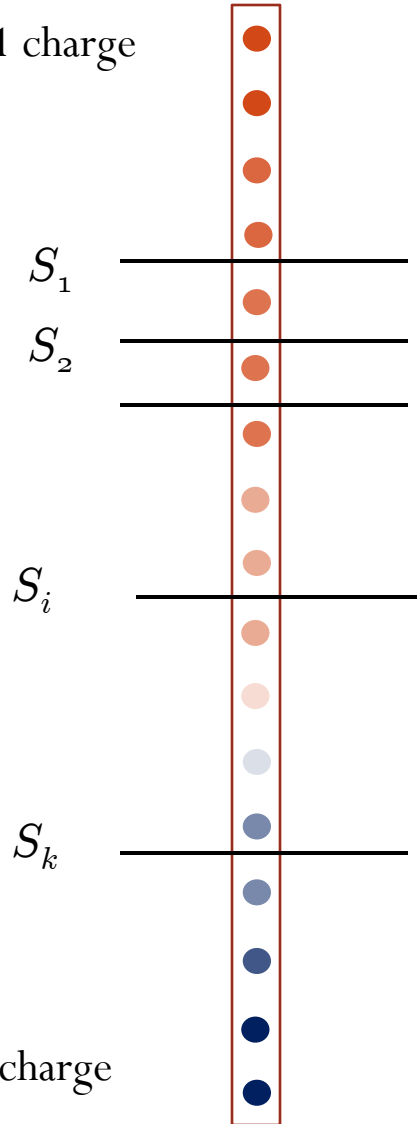
● = -1 charge

At iteration  $t$ , repeat  $O(\log n)$  times:

- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.
- Mix charge along edges with rate  $\eta_t$ .
- Consider final distribution, sorted by charge.

# Our Algorithm – One Iteration

● = +1 charge



● = -1 charge

At iteration  $t$ , repeat  $O(\log n)$  times:

- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.
- Mix along edges with rate  $\eta_t$ .
- Consider final distribution, sorted by charge.
- Check all  $\Omega(b)$ -balanced sweep cuts  $S_1, \dots, S_k$  for  $\phi(S_i) \leq O(\gamma^{1/2})$  in  $G$ .

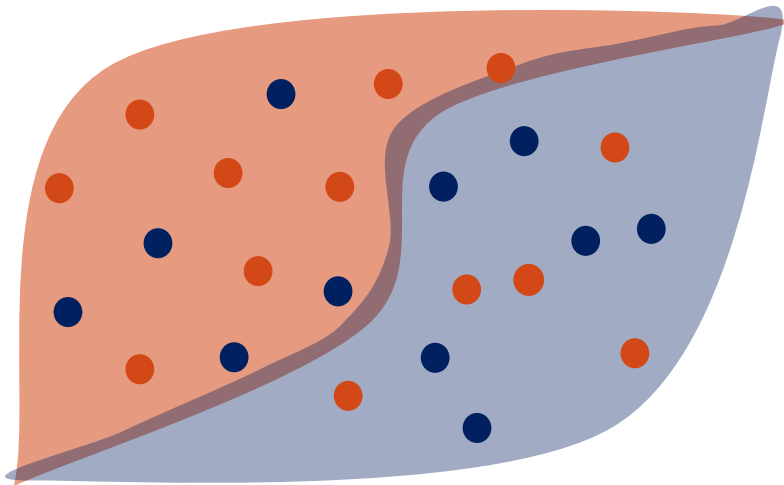
# Our Algorithm – Intuition

● = +1 charge

● = -1 charge

At iteration  $t$ , repeat  $O(\log n)$  times:

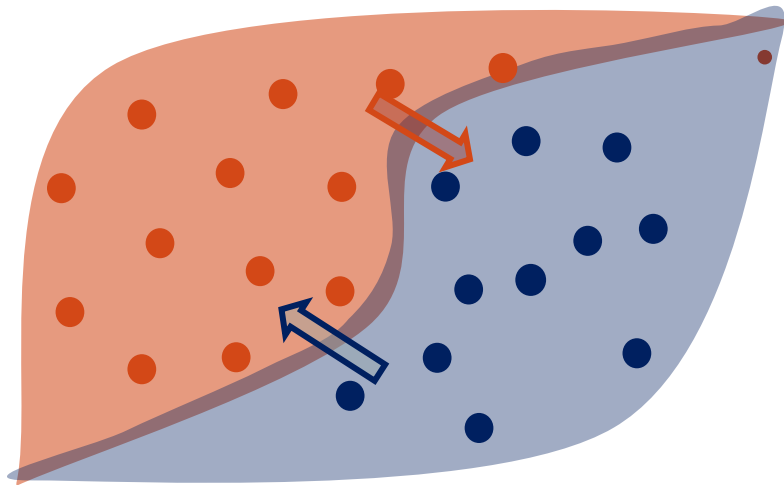
- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.



# Our Algorithm – Intuition

● = +1 charge

● = -1 charge



At iteration  $t$ , repeat  $O(\log n)$  times:

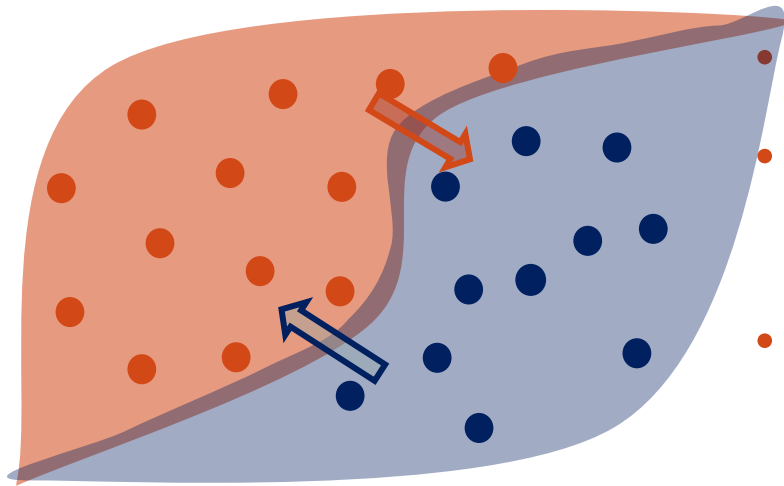
- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.
- Mix along edges with rate  $\eta_t$ .

**If cut is sparse, most mixing takes place within each side**

# Our Algorithm – Intuition

● = +1 charge

● = -1 charge



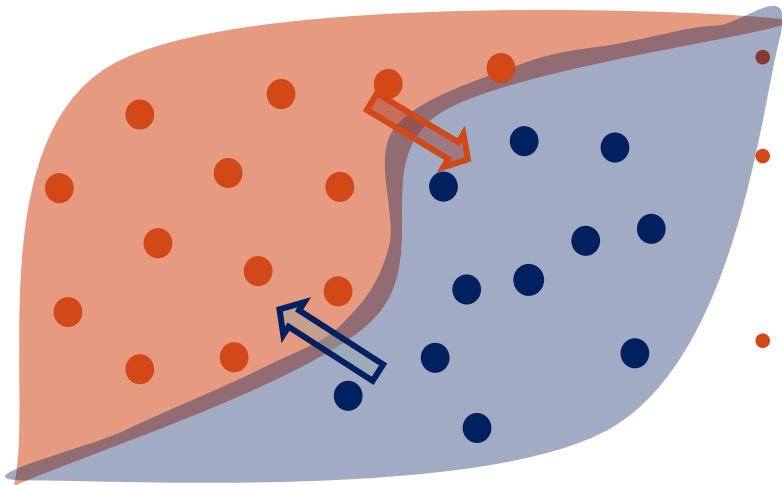
At iteration  $t$ , repeat  $O(\log n)$  times:

- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.
- Mix along edges with rate  $\eta_t$ .
- Consider final distribution, sorted by charge.
- Check all  $\Omega(b)$ -balanced sweep cuts  $S_1, \dots, S_k$  for  $\phi(S_i) \leq O(\gamma^{1/2})$ .

# Our Algorithm – Intuition

● = +1 charge

● = -1 charge



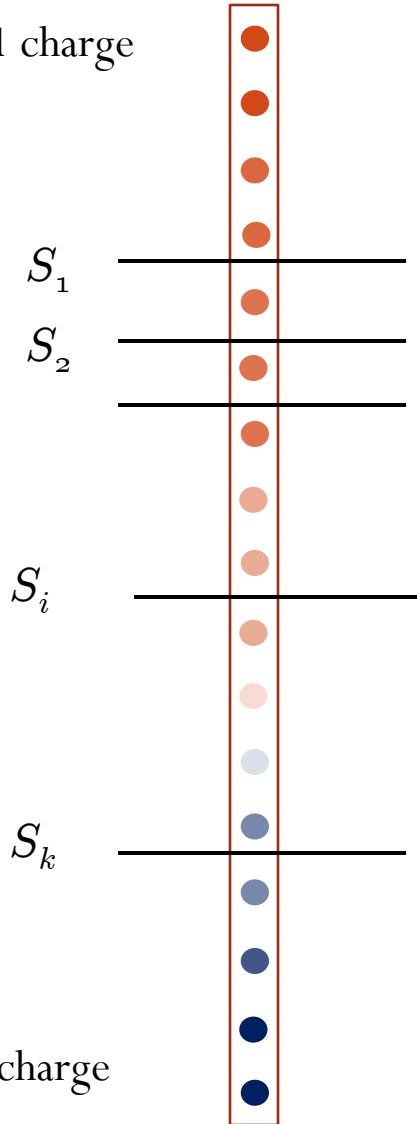
At iteration  $t$ , repeat  $O(\log n)$  times:

- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.
- Mix along edges with rate  $\eta_t$ .
- Consider final distribution, sorted by charge.
- Check all  $\Omega(b)$ -balanced sweep cuts  $S_1, \dots, S_k$  for  $\phi(S_i) \leq O(\gamma^{1/2})$ .

**Possible Problems:** 1) Less sparse cuts could also be highlighted.  
2) Bias towards small cuts.

# Our Algorithm – One Iteration

● = +1 charge



● = -1 charge

At iteration  $t$ , repeat  $O(\log n)$  times:

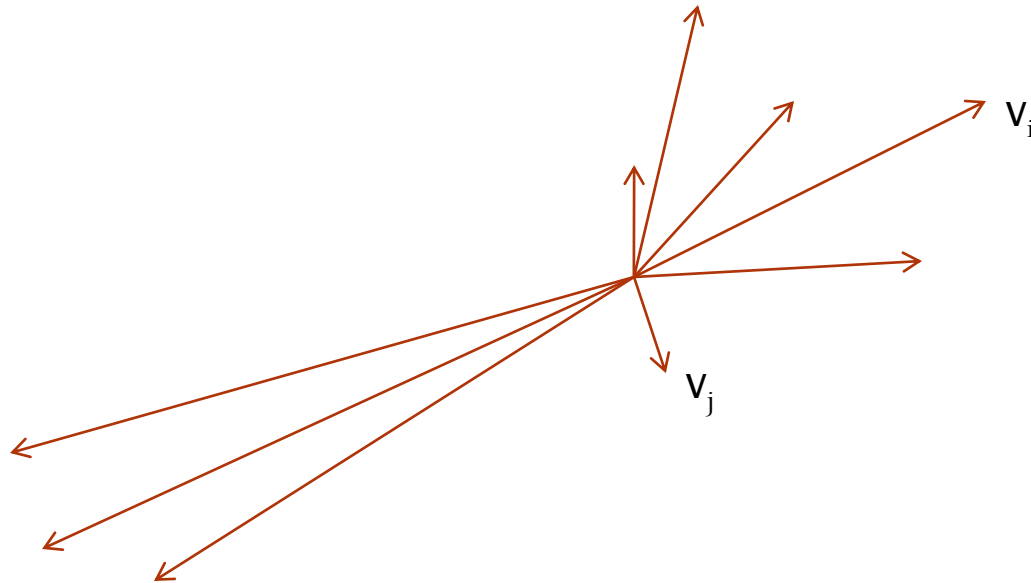
- Consider  $G_t$ .
- Pick random assignment of  $\pm 1$  charge to the vertices.
- Mix along edges with rate  $\eta_t$ .
- Consider final distribution, sorted by charge.
- Check all  $\Omega(b)$ -balanced sweep cuts  $S_1, \dots, S_k$  for  $\phi(S_i) \leq O(\gamma^{1/2})$  in  $G$ .

What if no sparse balanced cut is found?

# Our Algorithm – One Iteration

- What if no sparse balanced cut is found?

Consider the  $O(\log n)$ -dimensional vector embedding of vertices given by the final distributions.

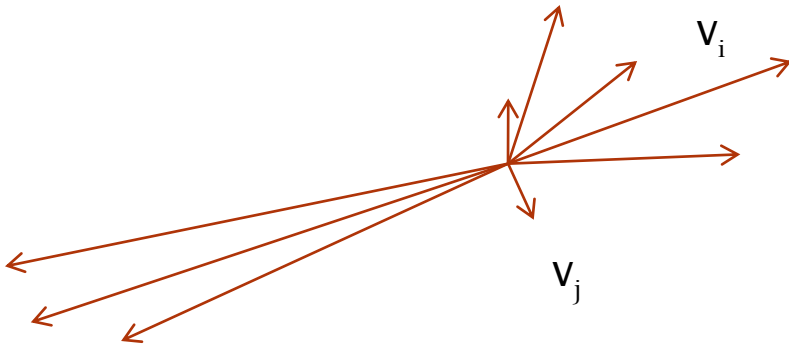


# Our Algorithm – One Iteration

- If  $\sum_{\{i,j\} \in E} \|v_i - v_j\|^2 \geq \gamma \cdot \sum_{i \in V} \|v_i\|^2$

random walks may not have mixed enough. **How to fix it?**

Increase rate.  $\eta_{t+1} = \eta_t + 1$



# Our Algorithm – One Iteration

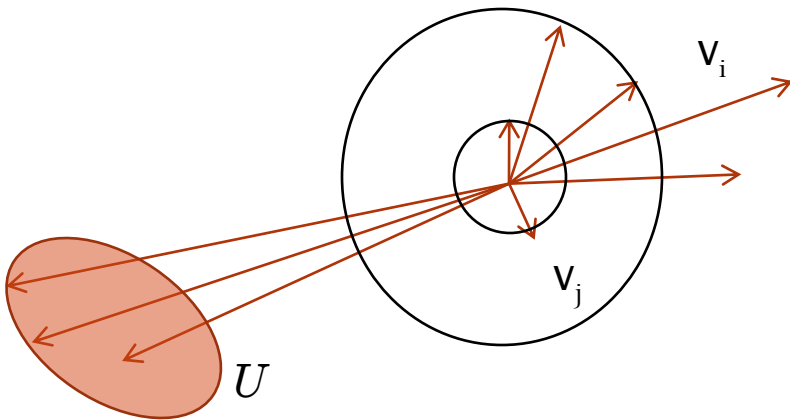
- If  $\sum_{\{i,j\} \in E} \|v_i - v_j\|^2 \geq \gamma \cdot \sum_{i \in V} \|v_i\|^2$

random walks may not have mixed enough. **How to fix it?**

Increase rate.  $\eta_{t+1} = \eta_t + 1$

- Otherwise, it is possible to find an unbalanced cut  $U$  with

$$\phi(U) \leq O(\gamma^{1/2}).$$



# Our Algorithm – One Iteration

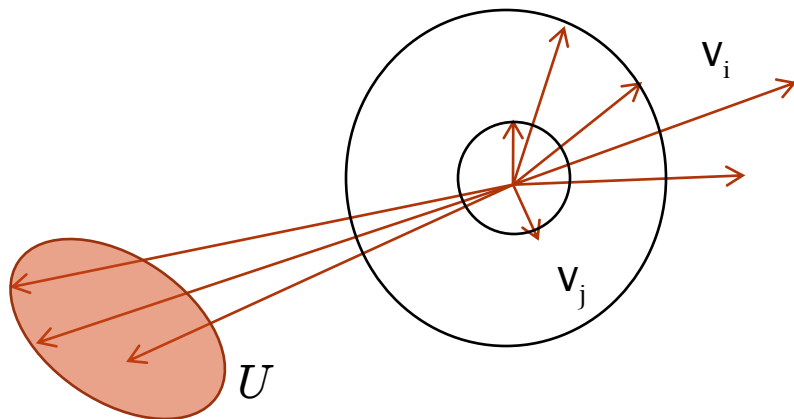
- If  $\sum_{\{i,j\} \in E} \|v_i - v_j\|^2 \geq \gamma \cdot \sum_{i \in V} \|v_i\|^2$

random walks may not have mixed enough. **How to fix it?**

Increase rate.  $\eta_{t+1} = \eta_t + 1$

- Otherwise, it is possible to find an unbalanced cut  $U$  with

$$\phi(U) \leq O(\gamma^{1/2}).$$



**Long vectors imply random walks got stuck in  $U$**

# Our Algorithm – One Iteration

- If  $\sum_{\{i,j\} \in E} \|v_i - v_j\|^2 \geq \gamma \cdot \sum_{i \in V} \|v_i\|^2$

random walks may not have mixed enough. **How to fix it?**

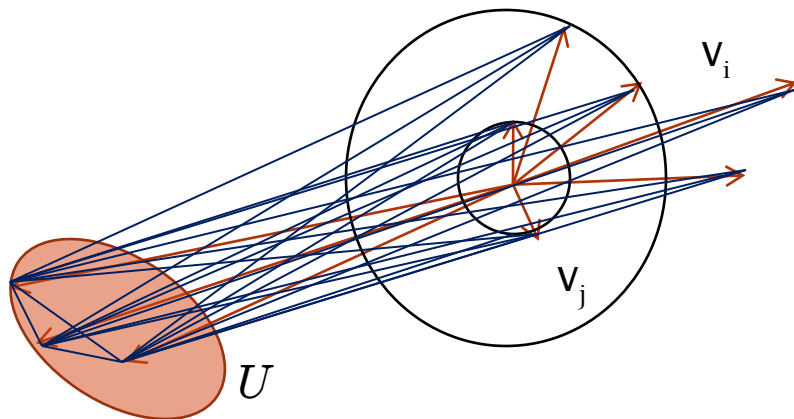
Increase rate.  $\eta_{t+1} = \eta_t + 1$

- Otherwise, it is possible to find an unbalanced cut  $U$  with

$$\phi(U) \leq O(\gamma^{1/2}).$$

**How to fix it?**

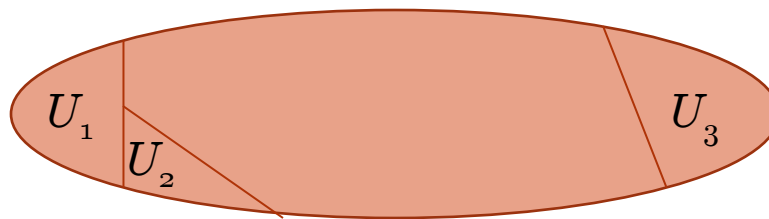
$$G_{t+1} = G_t + 2 \frac{\gamma}{n} \sum_{i \in U} \text{Star}_i$$



**Long vectors imply random walks got stuck in  $U$**

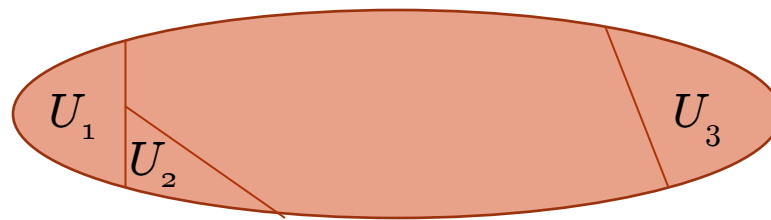
# Our Algorithm – Termination

- Terminate if:
  - Sparse balanced cut is found,
  - Union of sparse unbalanced cuts found is balanced.



# Our Algorithm – Termination

- Terminate if:
  - Sparse balanced cut is found,
  - Union of sparse unbalanced cuts found is balanced.



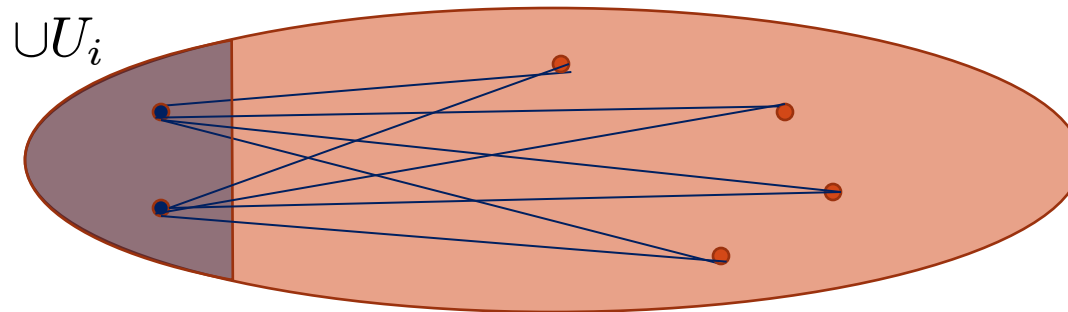
- If no sparse balanced cut found in  $T$  iterations and

$$T = O\left(\frac{\log n}{\gamma}\right),$$

we can show that  $G_T$  is a **certificate** that  $G$  has **no**  $b$ -balanced cuts of conductance less than  $\gamma$ .

# Our Algorithm - Certificate

- We can show that  $\Phi(G_T) \geq \gamma$



- Yields certificate as  $\text{vol}(U_i) \cdot O(b) \cdot 2m$ .
- Additional property:  $\phi(\cup U_i) \cdot \sqrt{\gamma}$
- Sufficient for applications to **sparsification and linear systems**.
- Allows us to find a large induced expander as in ST.

# Our Algorithm - Conclusion

- Running Time

$$\tilde{O}(m) \times \tilde{O}\left(\frac{1}{\gamma}\right) = \tilde{O}\left(\frac{m}{\gamma}\right)$$

Time  
per iteration

Number of  
iterations

Matches  
ST, ACL

# Our Algorithm - Conclusion

- **Running Time:**

$$\tilde{O}(m) \times \tilde{O}\left(\frac{1}{\gamma}\right) = \tilde{O}\left(\frac{m}{\gamma}\right)$$

Time                      Number of                      Matches  
per iteration                      iterations                      ST, ACL

- **Approximation:** outputs  $O(\sqrt{\gamma})$   $\Omega(b)$ -balanced cut or certifiat
- **Analysis:** direct consequence of Arora-Kale framework.

# Our Algorithm - Conclusion

- **Running Time:**

$$\tilde{O}(m) \times \tilde{O}\left(\frac{1}{\gamma}\right) = \tilde{O}\left(\frac{m}{\gamma}\right)$$

Time                      Number of                      Matches  
per iteration                      iterations                      ST, ACL

- **Approximation:** outputs  $O(\sqrt{\gamma})$   $\Omega(b)$ -balanced cut or certifiat
- **Analysis:** direct consequence of Arora-Kale framework.

**OUR ALGORITHM IS JUST SOLVING A SDP**

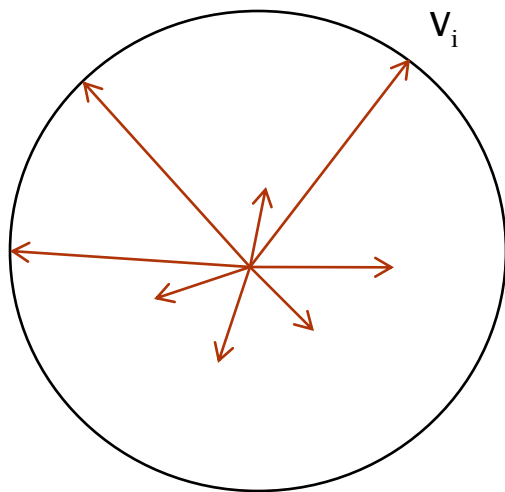
# SDP Analysis

- Our algorithm is approximately solving this SDP

$$\mathbf{SDP}(\mathbf{G}, \gamma, \mathbf{b}): \quad \mathbb{E}_{\{i,j\} \in E_G} \quad \|v_i - v_j\|^2 \cdot \gamma,$$

$$\mathbb{E}_{\{i,j\} \in V \times V} \quad \|v_i - v_j\|^2 = \frac{1}{2m},$$

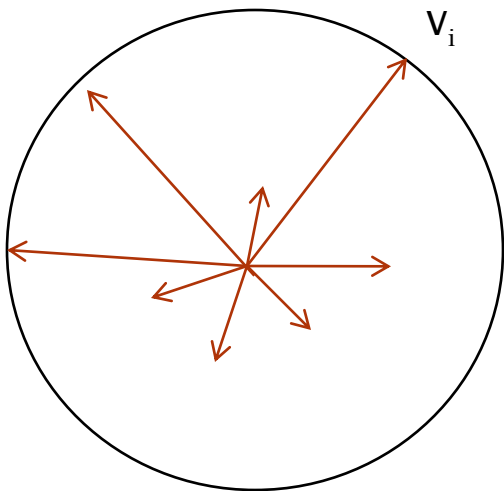
$$\forall i \in V \quad \mathbb{E}_{j \in V} \quad \|v_i - v_j\|^2 = \frac{1}{b} \cdot \frac{1}{2m}.$$



# SDP Analysis

- Our algorithm is approximately solving this SDP

$$\text{SDP}(\mathbf{G}, \gamma, \mathbf{b}): \quad \mathbb{E}_{\{i,j\} \in E_G} \quad \|v_i - v_j\|^2 \leq \gamma,$$
$$\mathbb{E}_{\{i,j\} \in V \times V} \quad \|v_i - v_j\|^2 = \frac{1}{2m},$$
$$\forall i \in V \quad \mathbb{E}_{j \in V} \quad \|v_i - v_j\|^2 = \frac{1}{b} \cdot \frac{1}{2m}.$$



- Star inequality equivalent to radius upper bound.
- Relaxation of question:  
is there a  $b$ -balanced cut of conductance less than  $\gamma$ ?
- Given approximately feasible solution, standard rounding (projection, Cheeger) yields approximation.

# Dual SDP

- If SDP is infeasible, algorithm produces solution to dual SDP.

$$\alpha - \frac{1}{b} \sum_{i \in V} \beta_i \geq \gamma \cdot 2m,$$

$$L(G) + \sum_{i \in V} \beta_i L(\text{Star}_i) \succeq \alpha L(K_n).$$

- Add few stars to  $G$  to **push its conductance over  $\gamma$** .

# Dual SDP

- If SDP is infeasible, algorithm produces solution to dual SDP.

$$\alpha - \frac{1}{b} \sum_{i \in V} \beta_i \geq \gamma \cdot 2m,$$

$$L(G) + \sum_{i \in V} \beta_i L(\text{Star}_i) \succeq \alpha L(K_n).$$

- Add few stars to  $G$  to **push its conductance over  $\gamma$** .

**How to solve SDP?**

# Arora-Kale Framework

- A roadmap to solving SDP fast.
- Yields combinatorial algorithms.

Given feasibility problem  $(G, \gamma, b)$ , visualize as a game.



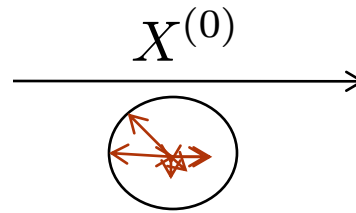
PRIMAL PLAYER  
(UPDATE)

DUAL PLAYER  
(ORACLE)

**GOAL:** Primal wants to show  $\text{SDP}(G, \gamma, b)$  is **feasible**.  
Dual wants to show it is **infeasible**.

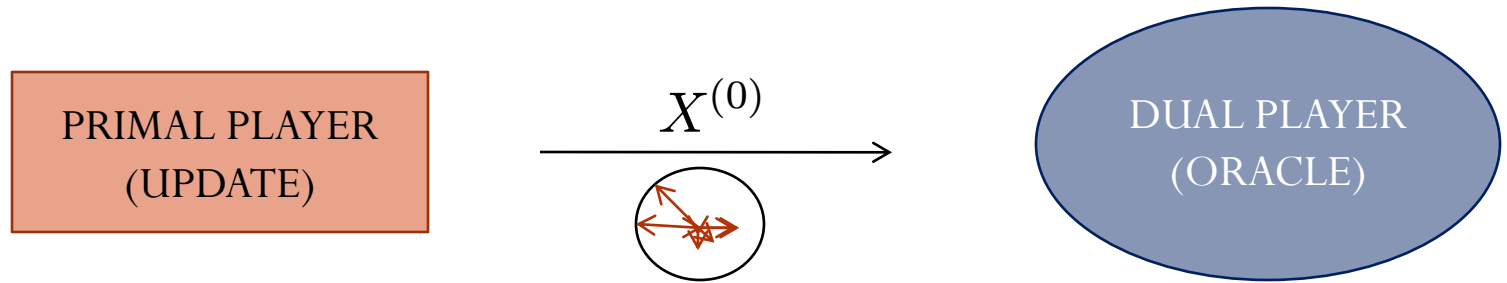
# Arora-Kale Framework

PRIMAL PLAYER  
(UPDATE)



DUAL PLAYER  
(ORACLE)

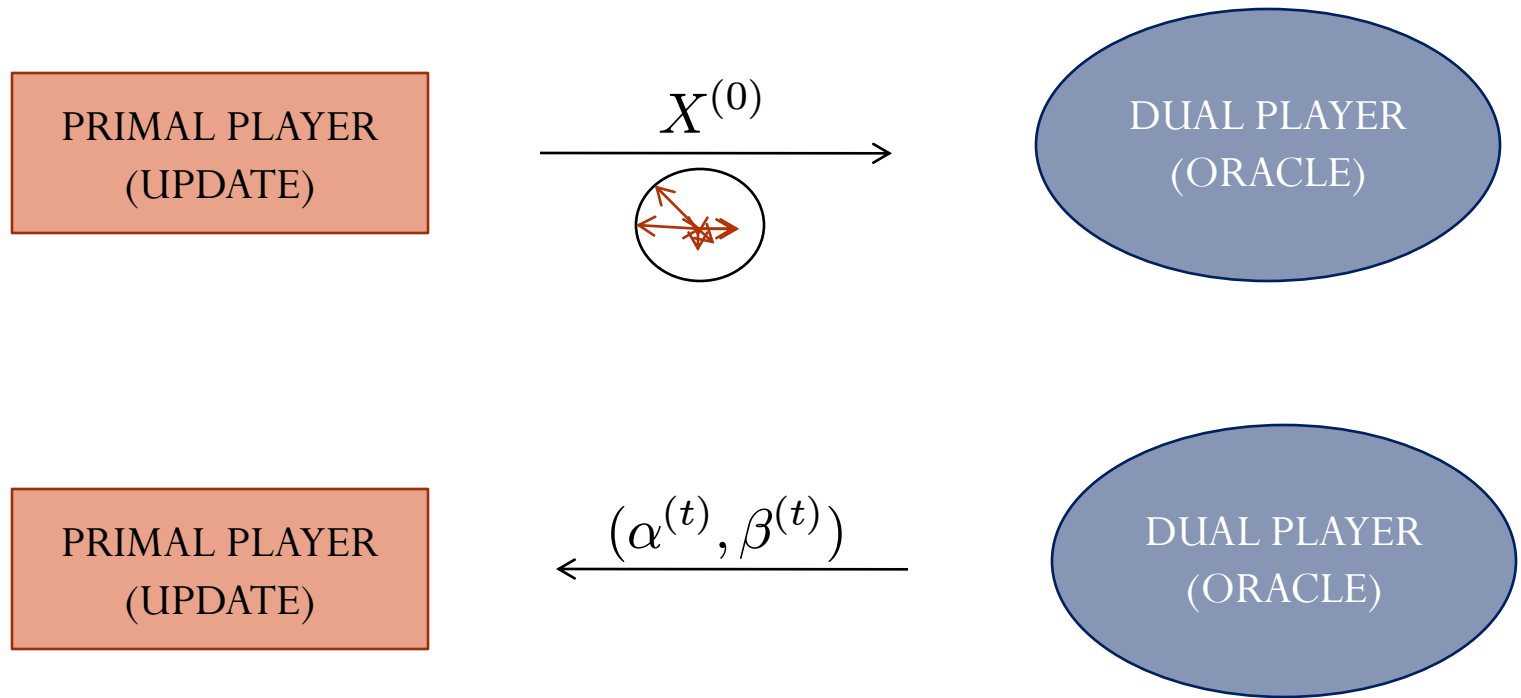
# Arora-Kale Framework



If feasible, **primal** has won.

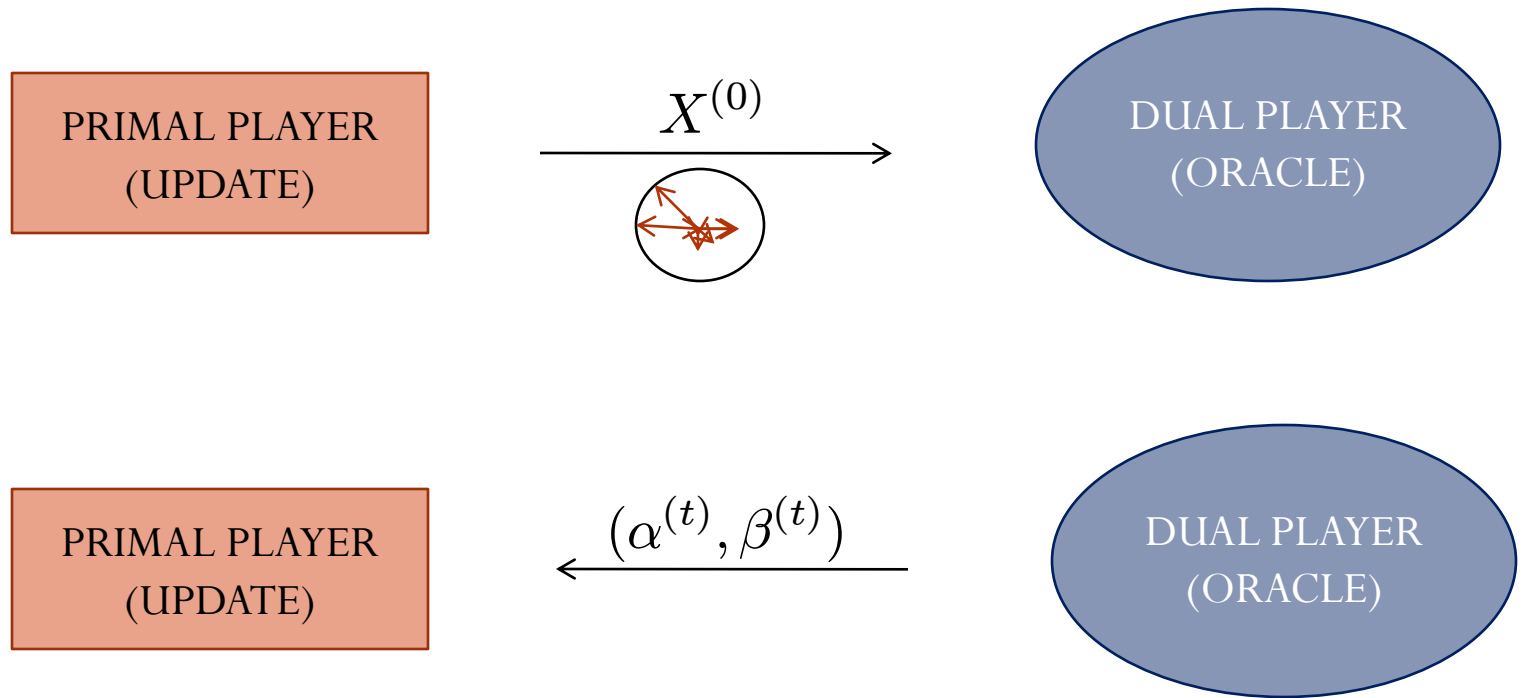
Otherwise, **dual** reports **broken constraints**.

# Arora-Kale Framework

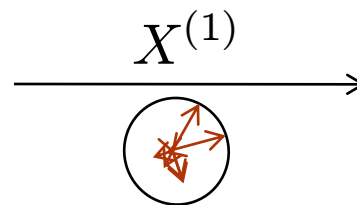


**Primal** updates solution to try and **fix constraints** reported by **dual**.

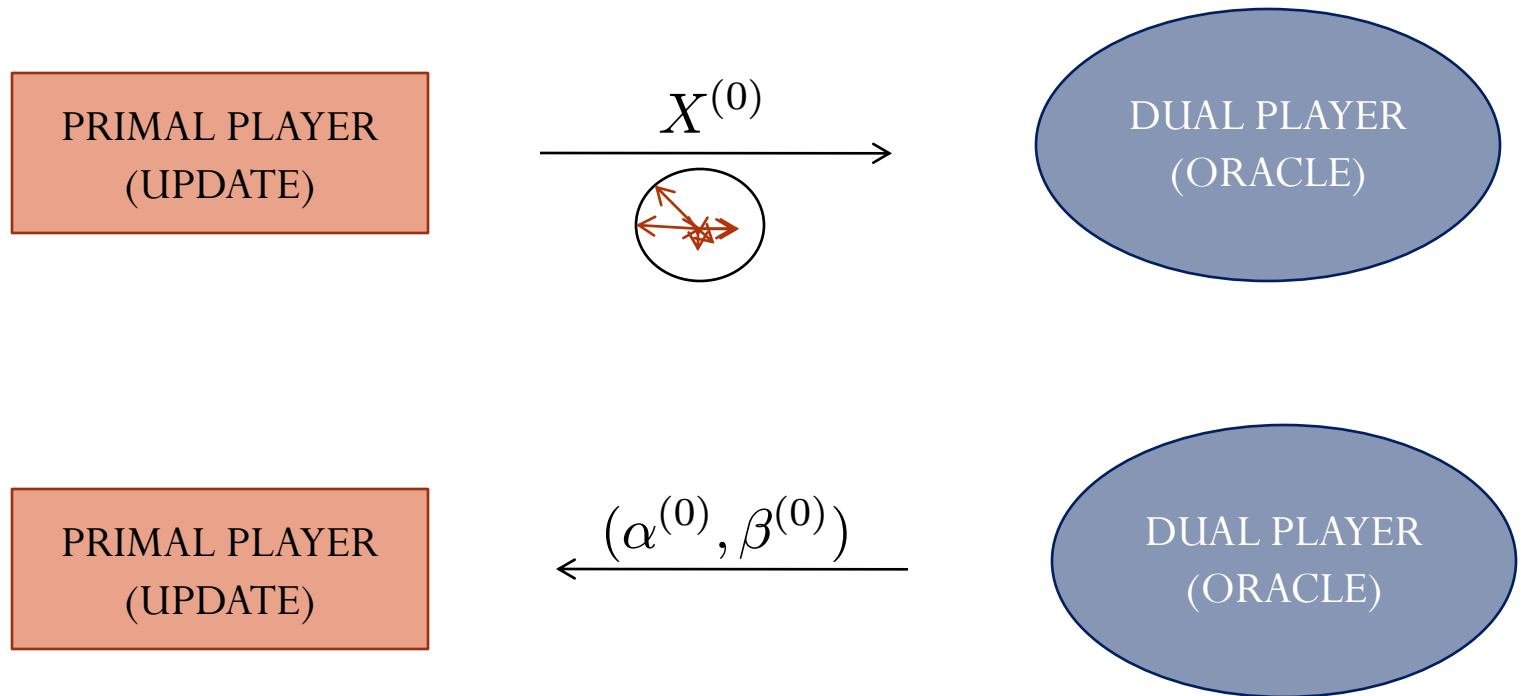
# Arora-Kale Framework



**Primal** updates solution to try and **fix constraints** reported by **dual**.



# Arora-Kale Framework



**Primal** updates solution to try and **fix constraints** reported by **dual**.

**MATRIX EXPONENTIAL UPDATE**

# Matrix Exponential Update

- How does update work?

Consider **current dual solution**:

$$\alpha = \frac{1}{t} \sum_{i=0}^t \alpha^{(i)}, \quad \beta = \frac{1}{t} \sum_{i=0}^t \beta^{(i)}.$$

**Update:**

$$X^{(t+1)} = Z \cdot e^{-t \left( L + \sum_{i \in V} \beta_i L(\text{Star}_i) - \alpha L(K_n) \right)}$$

# Matrix Exponential Update

- How does update work?

Consider **current dual solution**:

$$\alpha = \frac{1}{t} \sum_{i=0}^t \alpha^{(i)}, \quad \beta = \frac{1}{t} \sum_{i=0}^t \beta^{(i)}.$$

**Update:**

$$X^{(t+1)} = Z \cdot e^{-t \left( L + \sum_{i \in V} \beta_i L(\text{Star}_i) - \alpha L(K_n) \right)}$$

**Constraint in Dual SDP**

# Matrix Exponential Update

- How does update work?

Consider **current dual solution**:

$$\alpha = \frac{1}{t} \sum_{i=0}^t \alpha^{(i)}, \quad \beta = \frac{1}{t} \sum_{i=0}^t \beta^{(i)}.$$

**Update:**

$$X^{(t+1)} = Z' \cdot e^{-t \left( L + \sum_{i \in V} \beta_i L(\text{Star}_i) \right)}$$

# Matrix Exponential Update

- How does update work?

Consider **current dual solution**:

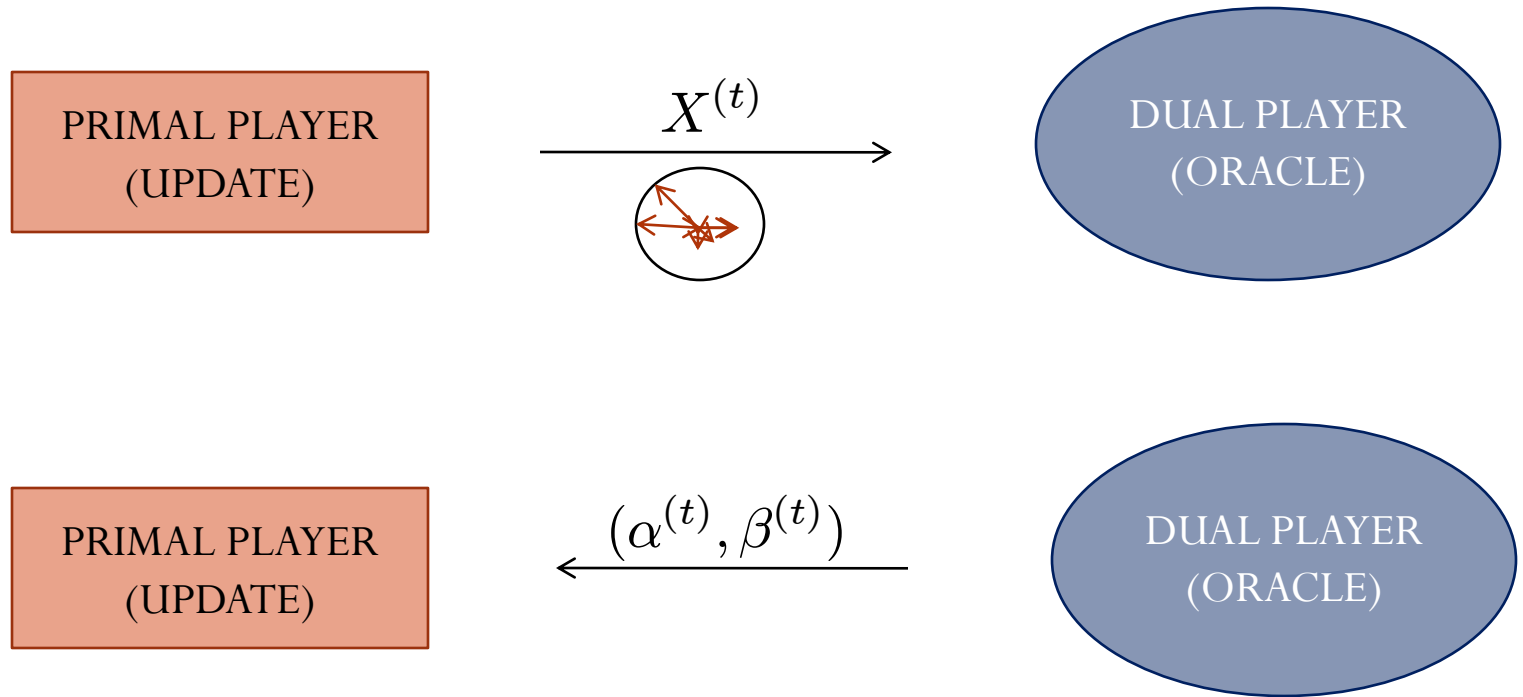
$$\alpha = \frac{1}{t} \sum_{i=0}^t \alpha^{(i)}, \quad \beta = \frac{1}{t} \sum_{i=0}^t \beta^{(i)}.$$

**Update:**

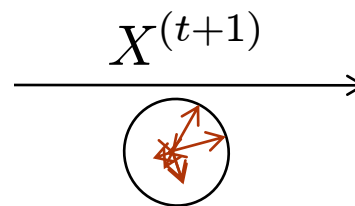
$$X^{(t+1)} = Z' \cdot e^{-t \left( L + \sum_{i \in V} \beta_i L(\text{Star}_i) \right)}$$

Heat kernel random walk on modified graph  $G_t$

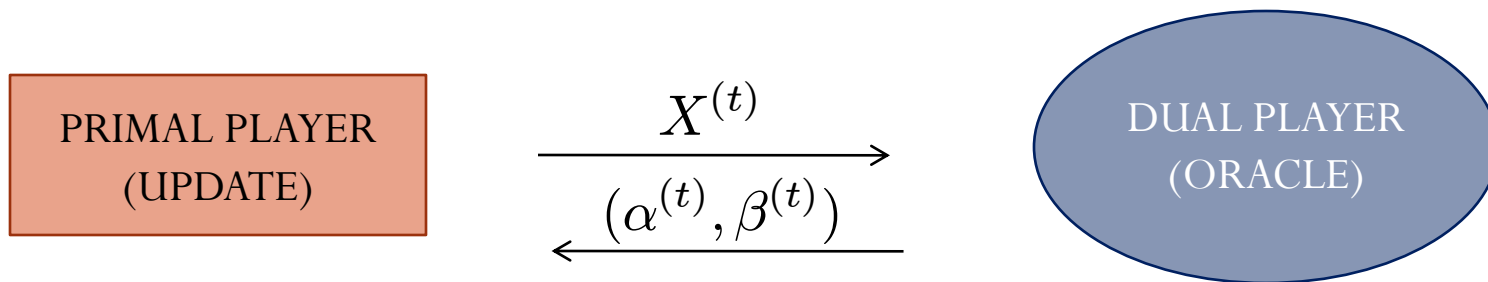
# Our Algorithm in SDP Language



**Primal** updates solution to try and **fix constraints** reported by **dual**.



# Our Algorithm in SDP Language



$X^{(t)}$   $\longrightarrow$  Embedding given by  $O(\log n)$  random walks on  $G_t$

If feasible, one dimension contains a **sparse balanced cut**.

Otherwise:

$\mathbb{E}_{\{i,j\} \in E_G} \|v_i^{(t)} - v_j^{(t)}\|^2 \geq \gamma$   $\longrightarrow$  Walks are not sufficiently mixed.

$\mathbb{E}_{j \in V} \|v_i^{(t)} - v_j^{(t)}\|^2 \geq \frac{1}{b} \cdot \frac{1}{2m}$   $\longrightarrow$  Sparse unbalanced cut containing  $i$ .

# Our Algorithm in SDP Language

Update:

$$X^{(t+1)} = Z' \cdot e^{-t(L + \sum_{i \in V} \beta_i L(\text{Star}_i))}$$

$X^{(t)}$   $\longrightarrow$  Embedding given by  $O(\log n)$  random walks on  $G_t$

If feasible, one dimension contains a **sparse balanced cut**.

Otherwise:

$\mathbb{E}_{\{i,j\} \in E_G} \|v_i^{(t)} - v_j^{(t)}\|^2 \geq \gamma$   $\longrightarrow$  Walks are not sufficiently mixed.

Proceed to next iteration.

$\mathbb{E}_{j \in V} \|v_i^{(t)} - v_j^{(t)}\|^2 \geq \frac{1}{b} \cdot \frac{1}{2m}$   $\longrightarrow$  Sparse unbalanced cut containing  $i$ .

Add  $\text{Star}_i$  to current graph.

# Final Remarks

- Hedging and Approximate Computation
  - Connection to recursive eigenvector method
  - Random walks as approximate eigenvector
  - Hedging : star addition is “soft” removal
- New Approach in Designing Fast Algorithm
  - Empirical evaluation
  - Apply to other problems
  - SDP as a tool for the analysis of algorithms

# Final Remarks

- Hedging and Approximate Computation
  - Connection to recursive eigenvector method
  - Random walks as approximate eigenvector
  - Hedging : star addition is “soft” removal
- New Approach in Designing Fast Algorithm
  - Empirical evaluation
  - Apply to other problems
  - SDP as a tool for the analysis of algorithms

**THANK YOU**