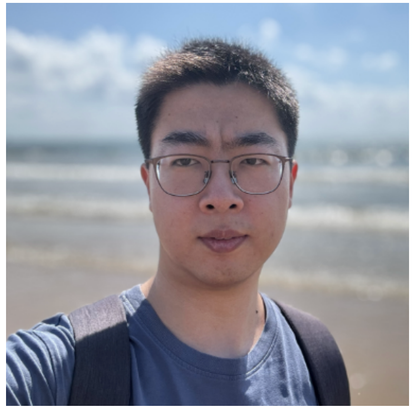
Learning-Augmented Streaming Algorithms for Approximating Max-DICUT

Xiaoyang Xu
University of California, Santa Barbara

Based on joint work with



Yinhao Dong
USTC

Pan Peng
USTC

Ali Vakilian
Virginia Tech

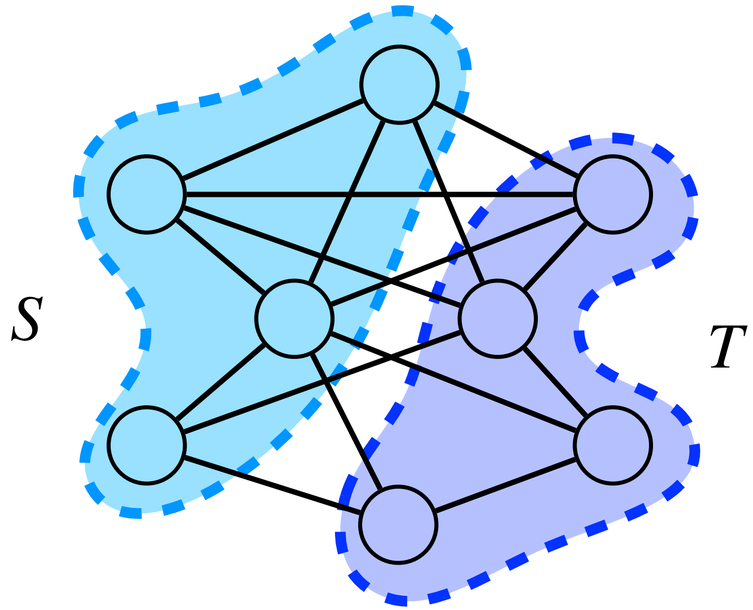
Santhoshini Velusamy
University of Waterloo

Max-CUT

Input: An undirected, unweighted graph $G = (V, E)$

Goal: Find a bipartition of V into S and T that maximizes the cut value

$$|E(S,T)| = |\{(u,v) \in E : u \in S \wedge v \in T\}|$$



Max-CUT

Input: An undirected, unweighted graph $G = (V, E)$

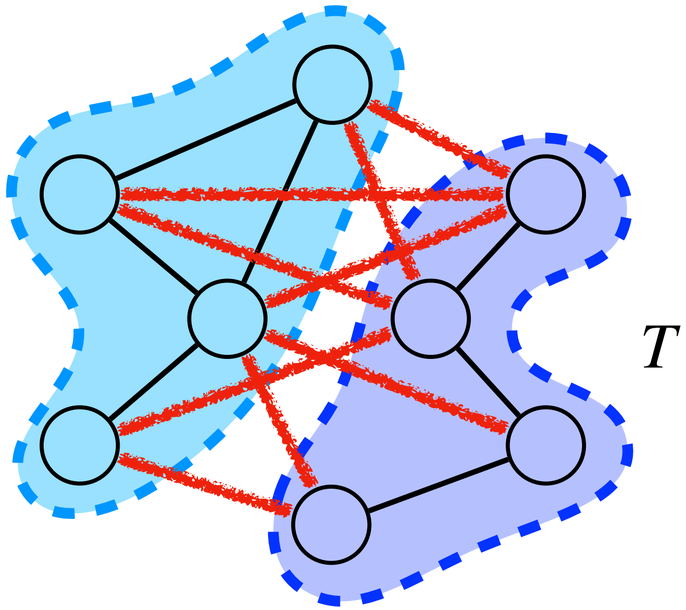
Goal: Find a bipartition of V into S and T that maximizes the cut value

$$|E(S,T)| = |\{(u,v) \in E : u \in S \wedge v \in T\}|$$

- **NP**-hard
- α-approximation: outputs a **cut** (S, T), s.t.

$$\alpha \cdot \text{OPT} \leq |E(S,T)| \leq \text{OPT}$$

- SDP rounding: 0.878-approx [Goemans, Williamson, JACM'95]
  - Unique Games Conjecture $\Longrightarrow$ best possible [Khot, Kindler, Mossel, O'Donnell, SICOMP'07]



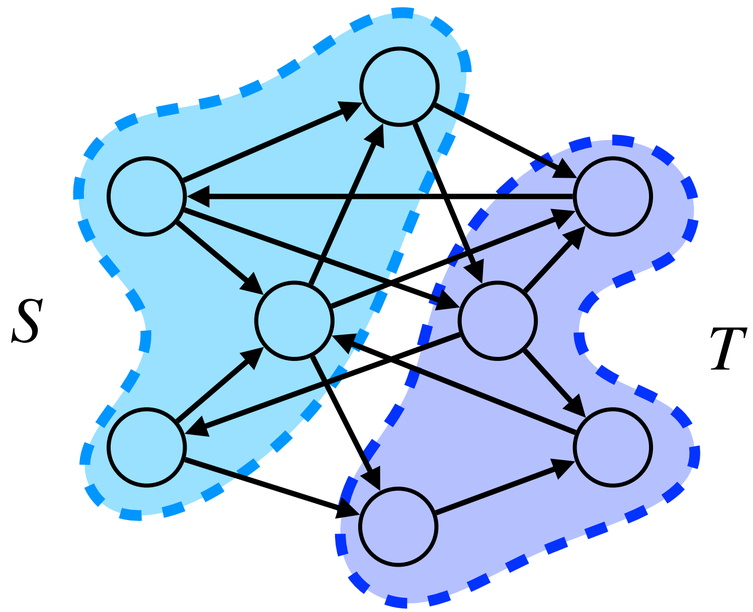
Max-DICUT

Input: A directed, unweighted graph $G = (V, E)$

Goal: Find an ordered bipartition of V into S and T that maximizes the dicut value

$$|E(S,T)| = |\{(u,v) \in E : u \in S \wedge v \in T\}|$$

(only counting the number of edges that go in one direction)



Max-DICUT

Input: A directed, unweighted graph $G = (V, E)$

Goal: Find an ordered bipartition of V into S and T that maximizes the dicut value

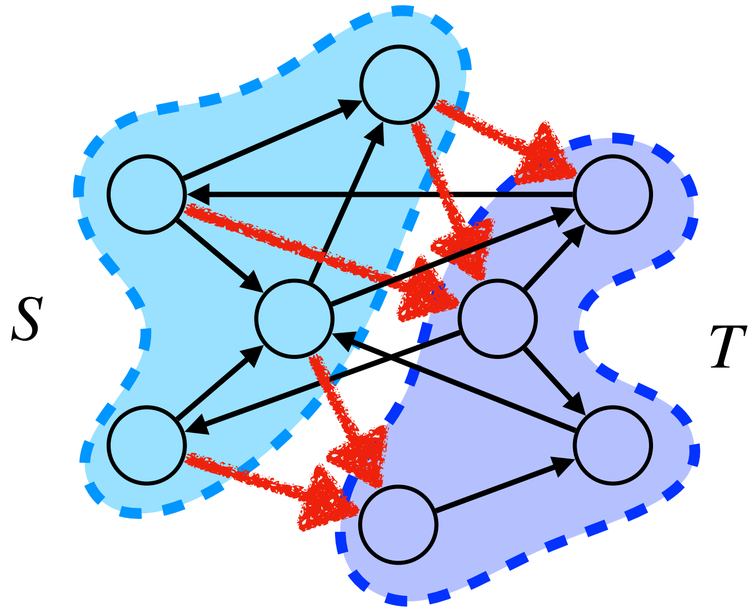
$$|E(S,T)| = |\{(u,v) \in E : u \in S \wedge v \in T\}|$$

(only counting the number of edges that go in one direction)

- **NP**-hard
- α-approximation: outputs a **dicut** (S, T), s.t.

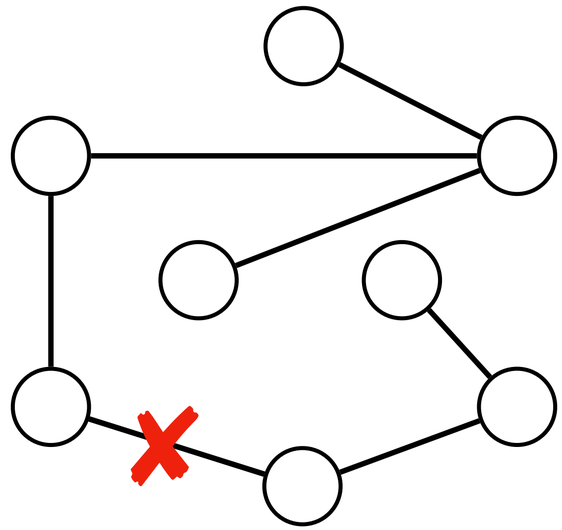
$$\alpha \cdot \text{OPT} \leq |E(S,T)| \leq \text{OPT}$$

- 0.87446-approx possible; 0.87461-approx impossible, under Unique Games Conjecture
  [Brakensiek, Huang, Potechin, Zwick, FOCS'23]



Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions
- **Goal:** in **one pass**, using **small space**



Max-(DI)CUT in Streaming

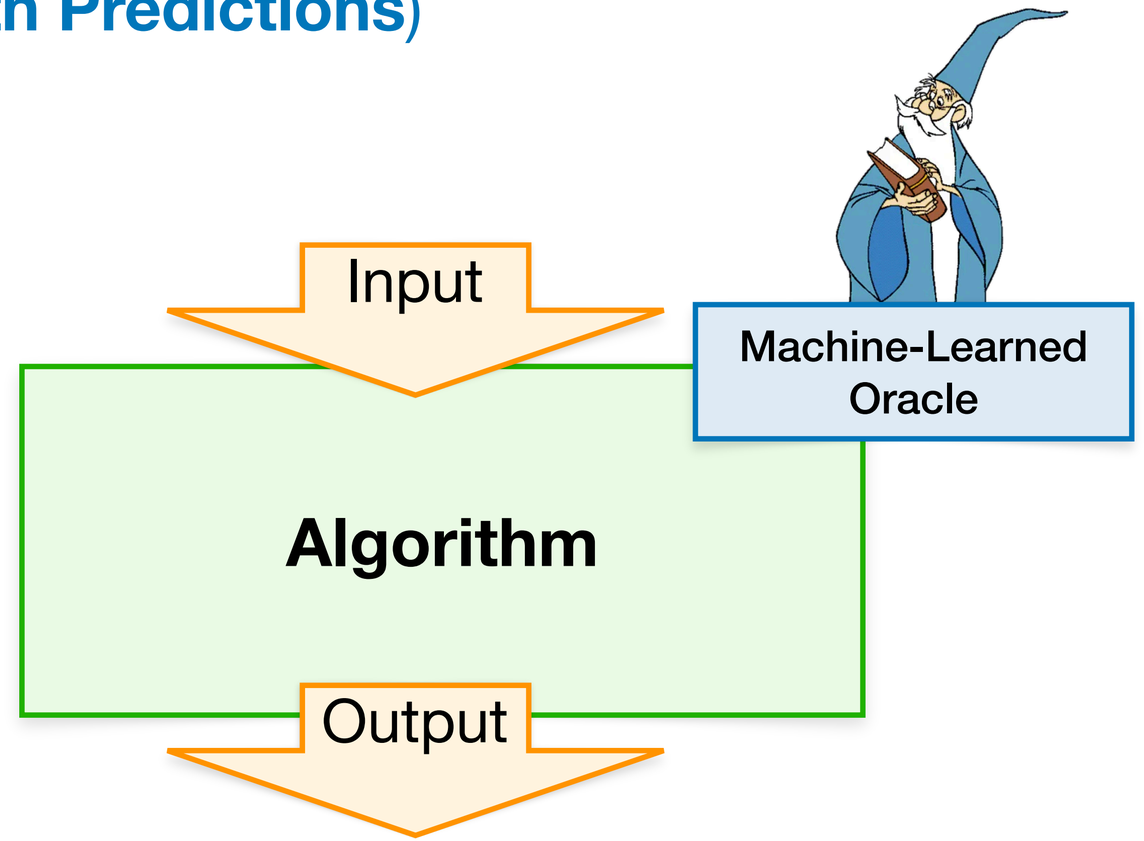
	Max-CUT	Max-DICUT
Solution	$1-\eta$ $\widetilde{O}(n)$ cut sparsifier	$1-\eta$ $\widetilde{O}(n)$ additive cut sparsifier
Value	$1/2$ $O(\log n)$ trivial	$2/5$ $O(\log n)$ [Guruswami, Velingker, Velusamy, APPROX'17]
		$4/9$ $O(\log n)$ [Chou, Golovnev, Velusamy, FOCS'20]
		0.485 $\widetilde{O}(\sqrt{n})$ [Saxena, Singer, Sudan, Velusamy, FOCS'23]
		$1/2-\eta$ $n^{1-\Omega_\eta(1)}$ [Azarmehr, Behnezhad, Ferrante, Saneian, STOC'26], building upon [Saxena, Singer, Sudan, Velusamy, FOCS'23, SODA'25]
Lower Bound	$1/2+\eta$ $\Omega_\eta(n)$ [Kapralov, Krachun, STOC'19], building upon [Kapralov, Khanna, Sudan, SODA'15] [Kogan, Krauthgamer, ITCS'15] [Kapralov, Khanna, Sudan, Velingker, SODA'17]	$4/9+\eta$ $\Omega_\eta(\sqrt{n})$ [Chou, Golovnev, Velusamy, FOCS'20]

What if we can obtain extra information about the input?

Learning-Augmented Algorithms

(a.k.a. **Algorithms with Predictions**)

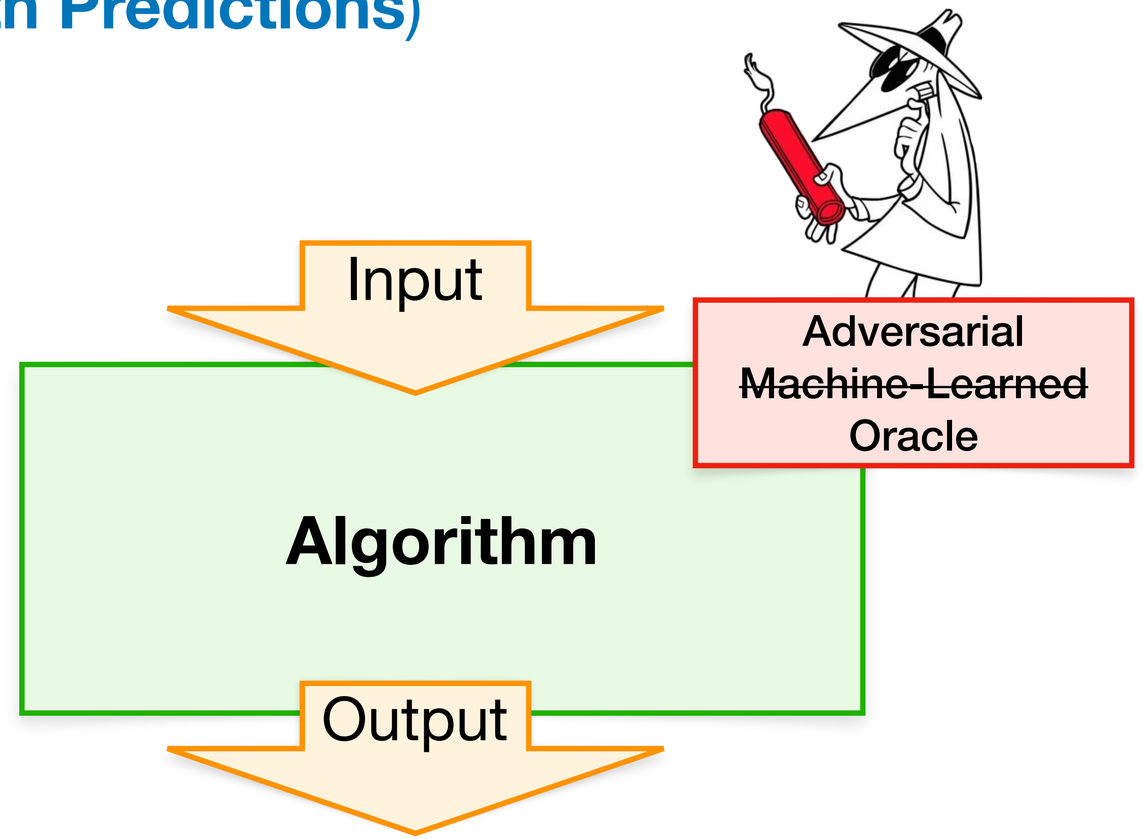
- The algorithm has access to a **learned oracle** providing a certain type of **predictions** about the input instance
- **Goals:**
  - **Consistency:** High prediction quality
    $\Longrightarrow$ Better performance than the best-known classical algorithm



Learning-Augmented Algorithms

(a.k.a. **Algorithms with Predictions**)

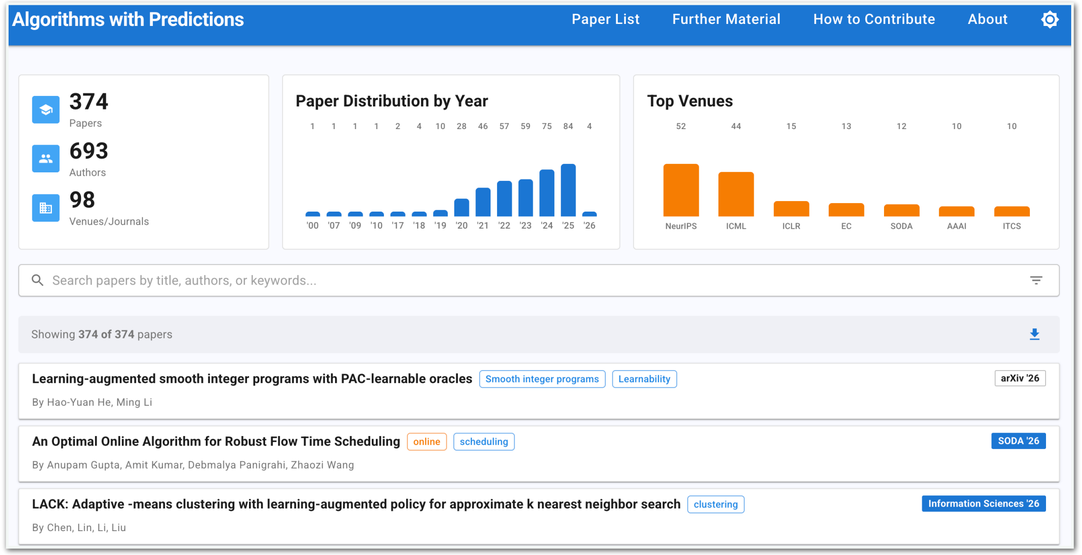
- The algorithm has access to a **learned oracle** providing a certain type of **predictions** about the input instance
- **Goals:**
  - **Consistency:** High prediction quality
    $\Longrightarrow$ Better performance than the best-known classical algorithm
  - **Robustness:** Low prediction quality
    $\Longrightarrow$ Performs no worse than the best-known classical algorithm



Learning-Augmented Algorithms

(a.k.a. **Algorithms with Predictions**)

- The algorithm has access to a **learned oracle** providing a certain type of **predictions** about the input instance
- **Goals:**
  - **Consistency:** High prediction quality
    $\Longrightarrow$ Better performance than the best-known classical algorithm
  - **Robustness:** Low prediction quality
    $\Longrightarrow$ Performs no worse than the best-known classical algorithm

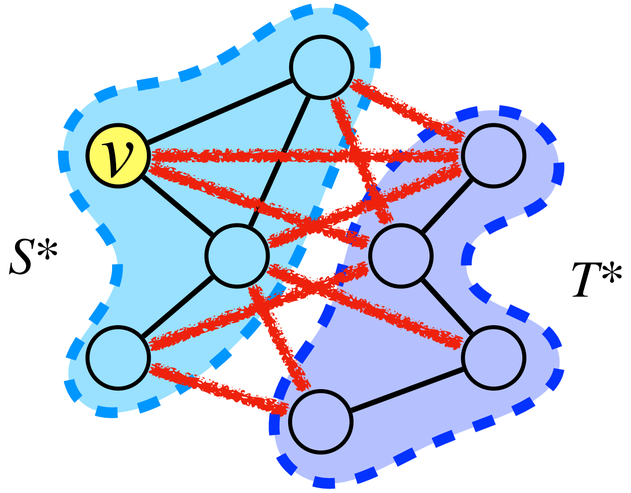


An up-to-date repository of publications
(https://algorithms-with-predictions.github.io)

How to define predictions?

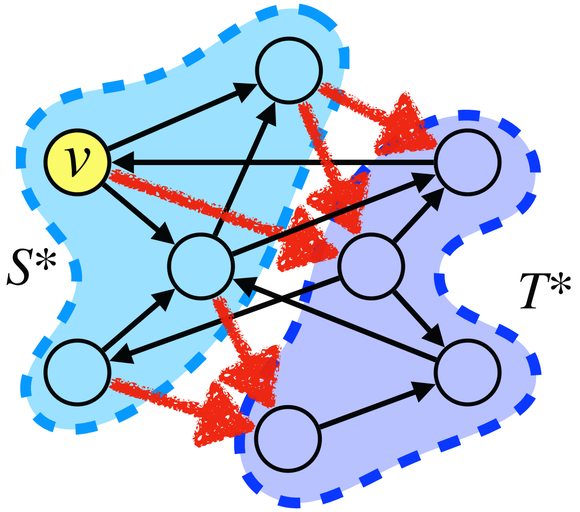
Our Prediction Model

- **ϵ-accurate predictions** [Cohen-Addad, d'Orsi, Gupta, Lee, Panigrahi, NeurIPS'24] [Braverman, Dharangutte, Shah, Wang, APPROX'24] [Ghoshal, Makarychev, Makarychev, SODA'25]
  - Let $x^* \in \{-1,1\}^n$ denote some fixed but unknown **optimal solution**
  - Oracle access to a **prediction vector** $Y \in \{-1,1\}^n$
  - Each entry Y_v is independently correct with probability $1/2 + \epsilon$



$$\forall v \in V, \quad \Pr[Y_v = x^*_v] = \frac{1}{2} + \epsilon$$

Only slightly better than a random guess!



Max-DICUT:

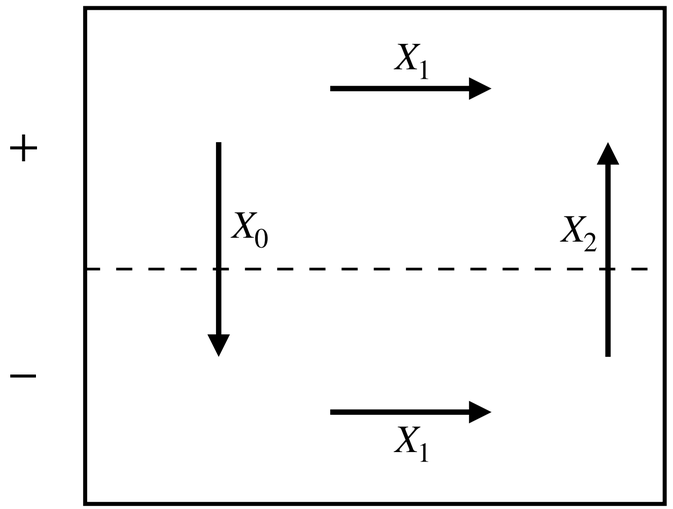
$1-\eta$ approx vs poly$(1/\eta, 1/\epsilon)$ space

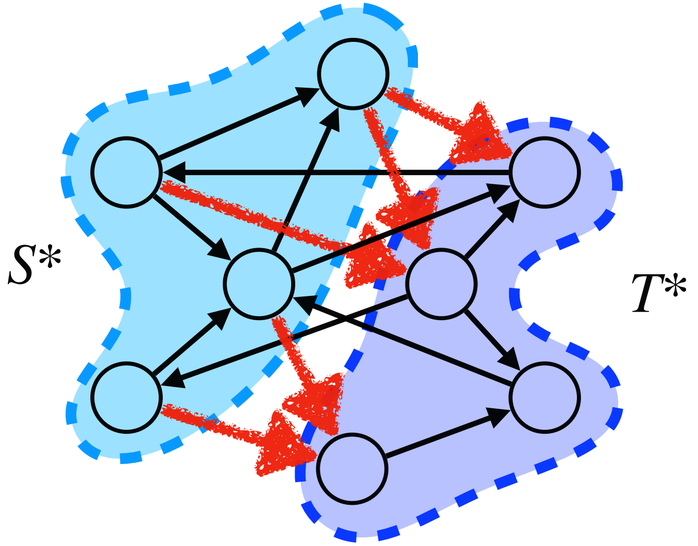
Predictions Give Us Much More

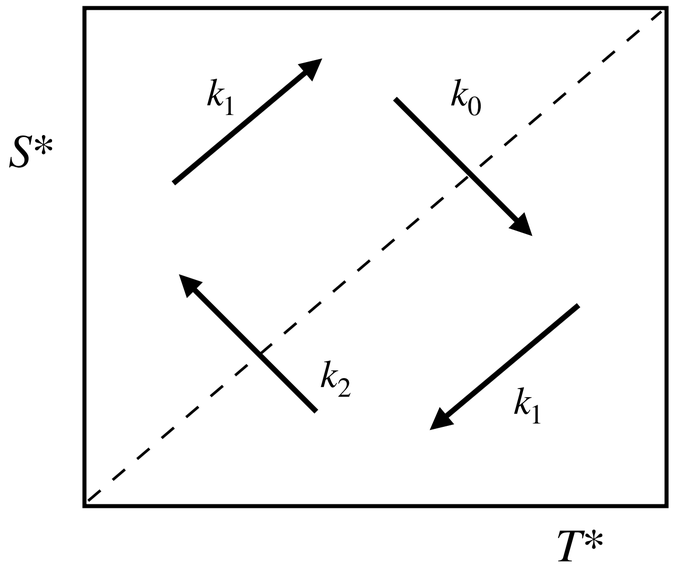
Assuming the max degree is not too large; prediction Y and optimal solution (S^*, T^*)

- $X_0 =$ # edges with $(+, -)$ predictions
- $X_1 =$ # edges with $(+, +), (-, -)$ predictions
- $X_2 =$ # edges with $(-, +)$ predictions

- $k_0 =$ # edges from S^* to T^*
- $k_1 =$ # edges inside S^* or T^*
- $k_2 =$ # edges from T^* to S^*
- Note that OPT $= k_0$

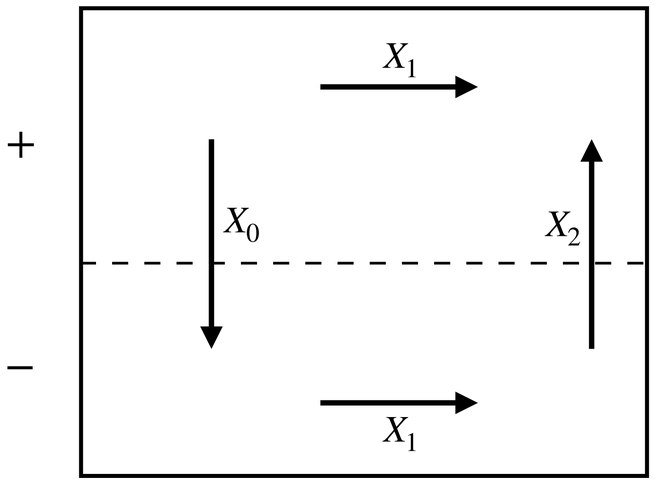


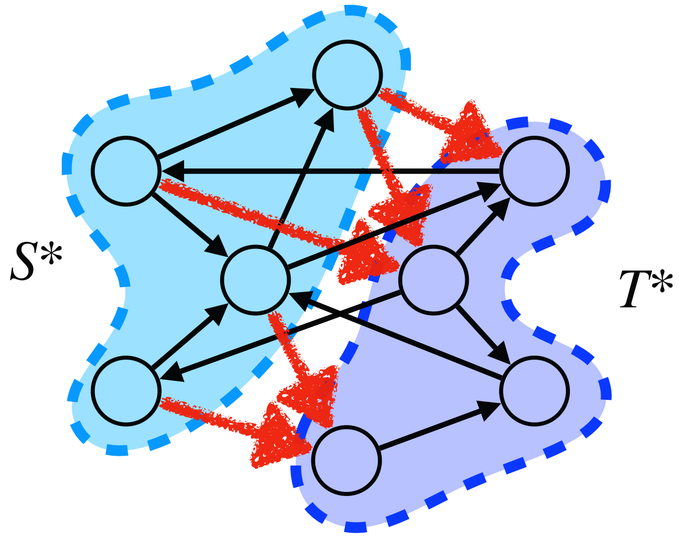


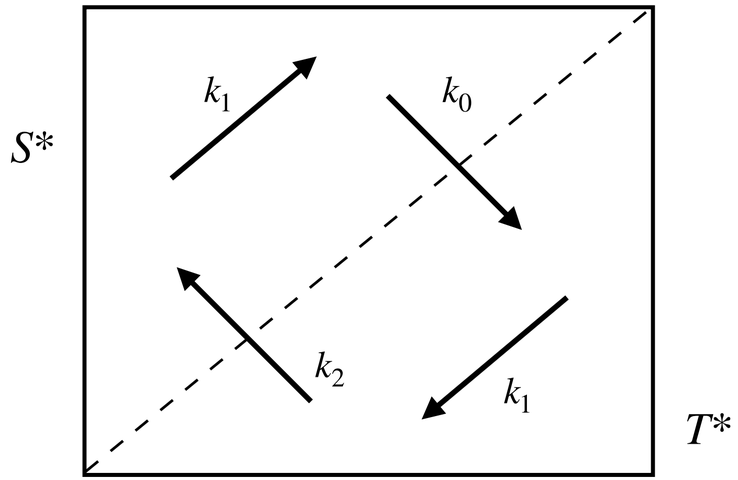


Predictions Give Us Much More

Assuming the max degree is not too large; prediction Y and optimal solution (S^*, T^*)







Observation:

The expectations $(\mathbb{E}[X_0], \mathbb{E}[X_1], \mathbb{E}[X_2])$ are a linear transformation of (k_0, k_1, k_2)

The "decoding" approach:

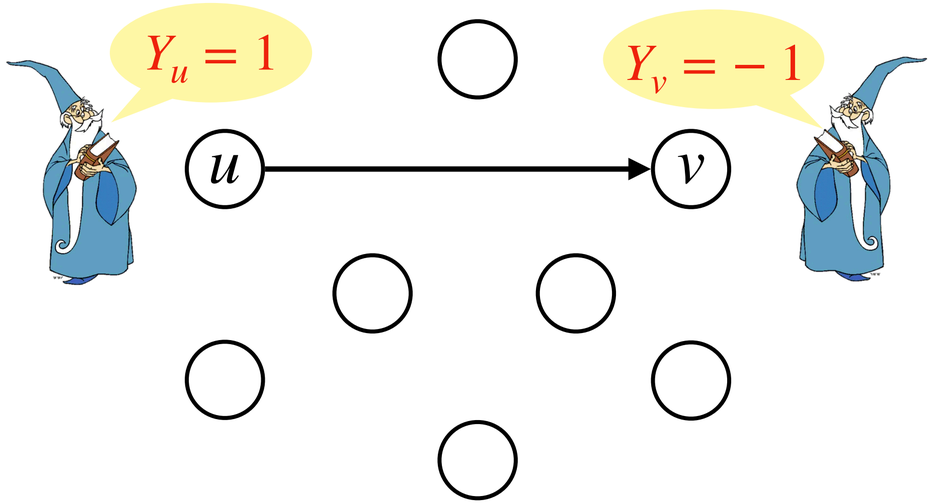
- Solve the linear system
- Define unbiased estimator $\widetilde{k}_0 := \alpha_0(\epsilon) \cdot X_0 + \alpha_1(\epsilon) \cdot X_1 + \alpha_2(\epsilon) \cdot X_2$

Our Algorithm for Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \text{poly}(\epsilon, \eta) \cdot m$

Algorithm for Low-Degree Graphs

- initially $X_0, X_1, X_2 = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u = +1, Y_v = -1$ then $X_0 = X_0 + 1$
  - if $Y_u = Y_v$ then $X_1 = X_1 + 1$
  - if $Y_u = -1, Y_v = +1$ then $X_2 = X_2 + 1$
- **return** $\alpha_0(\epsilon) \cdot X_0 + \alpha_1(\epsilon) \cdot X_1 + \alpha_2(\epsilon) \cdot X_2$



Use the "decoding" approach:

ϵ-accurate predictions $\xrightarrow{\text{w.h.p.}}$ $(1-\eta)$-approx, $\text{poly}(1/\epsilon, 1/\eta)$ words of space

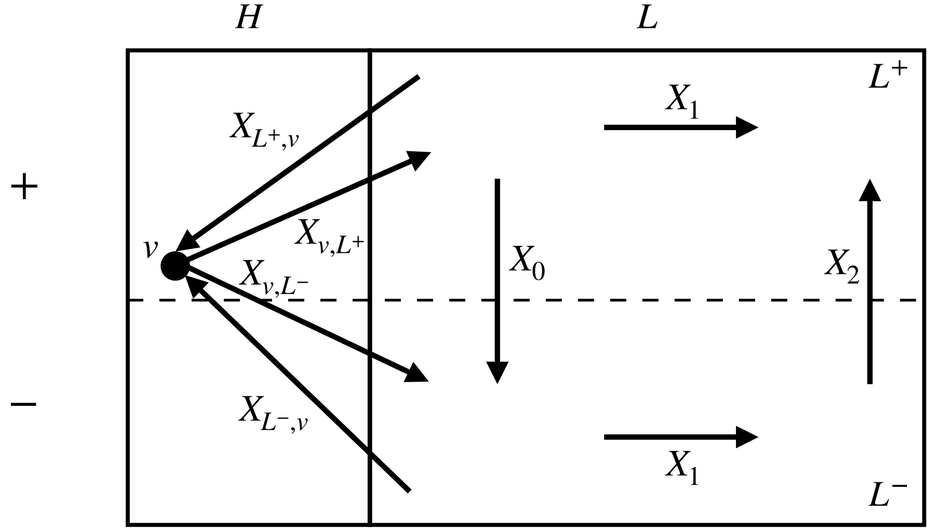
Decoding the Contribution of High-Degree Vertices

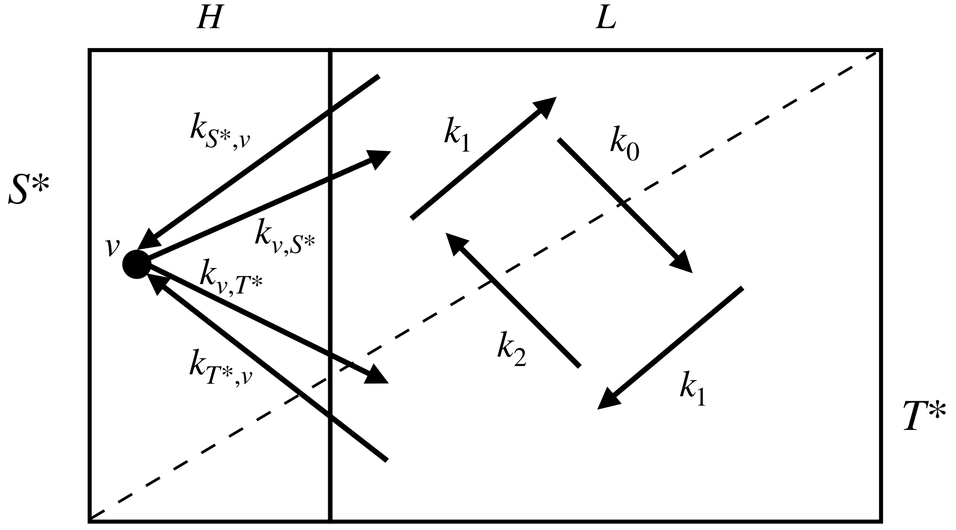
Divide vertices: (high-degree) H and (low-degree) L; Consider $v \in H$

- $X_{L^+,v}$ = # edges from L^+ to v
- $X_{L^-,v}$ = # edges from L^- to v
- X_{v,L^+} = # edges from v to L^+
- X_{v,L^-} = # edges from v to L^-

Here, $L^+ = \{u \in L : Y_u = +1\}$, $L^- = \{u \in L : Y_u = -1\}$

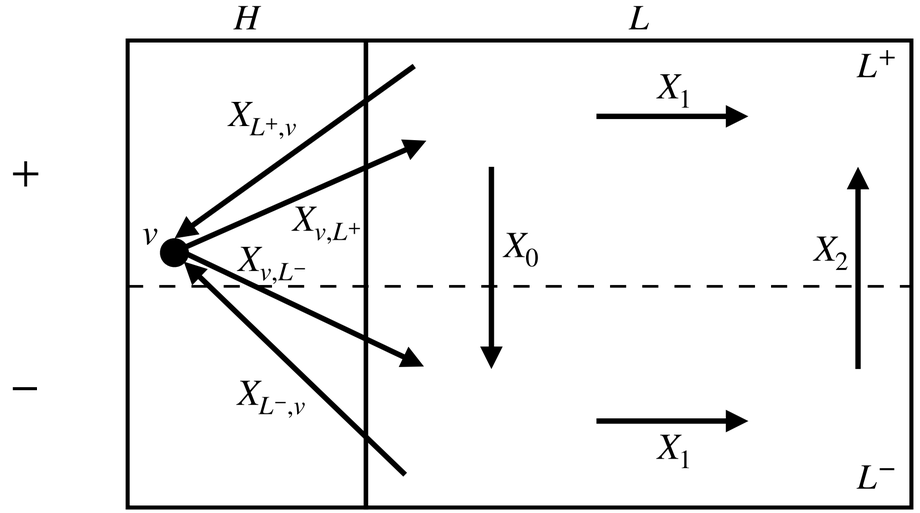
- $k_{S^*,v}$ = # edges from $S^* \cap L$ to v
- $k_{T^*,v}$ = # edges from $T^* \cap L$ to v
- k_{v,S^*} = # edges from v to $S^* \cap L$
- k_{v,T^*} = # edges from v to $T^* \cap L$

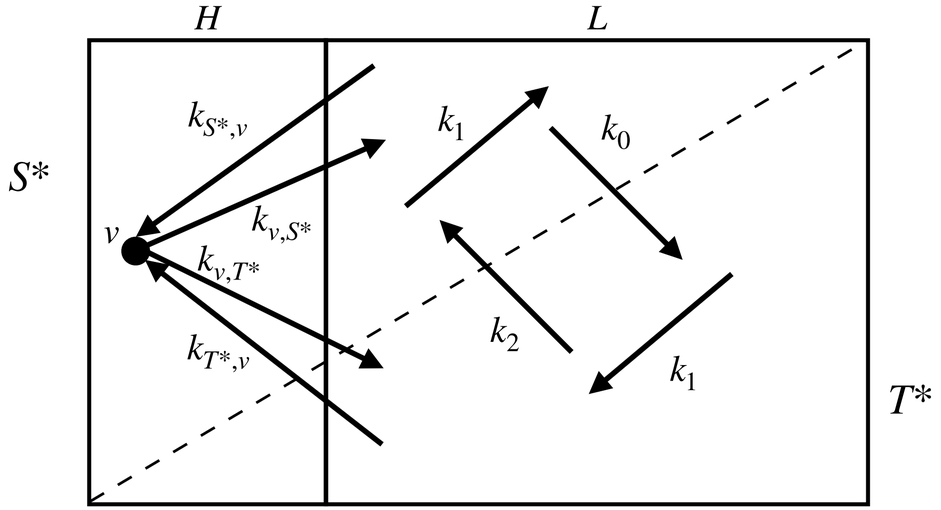




Decoding the Contribution of High-Degree Vertices

Divide vertices: (high-degree) H and (low-degree) L; Consider $v \in H$





Observation:

The expectations $(\mathbb{E}[X_{L^+,v}], \mathbb{E}[X_{L^-,v}])$ are a linear transformation of $(k_{S^*,v}, k_{T^*,v})$ (similarly for the outgoing direction)

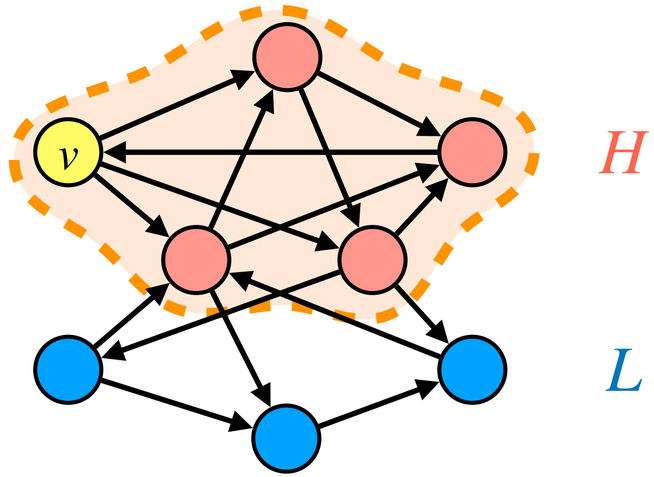
Again, **"decoding" approach:**

- Solve the linear system
- Define unbiased estimators $\widetilde{k}_{S^*,v}$ and $\widetilde{k}_{v,T^*}$, for each $v \in H$

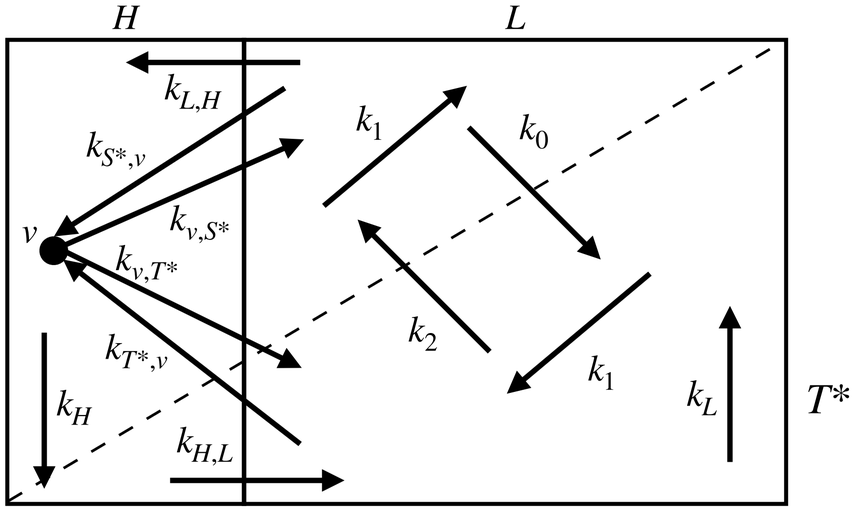
High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Use the decoding approach on $G[L]$, compute the dicut value $\widetilde{k}_L$ of $G[L]$
- Use the second decoding on vertices v in H to obtain $\widetilde{k}_{S^*,v}$ and $\widetilde{k}_{v,T^*}$
- Output $\widetilde{k}_L + \sum_{v \in H} \max\{\widetilde{k}_{S^*,v}, \widetilde{k}_{v,T^*}\} - \frac{\eta m}{16}$



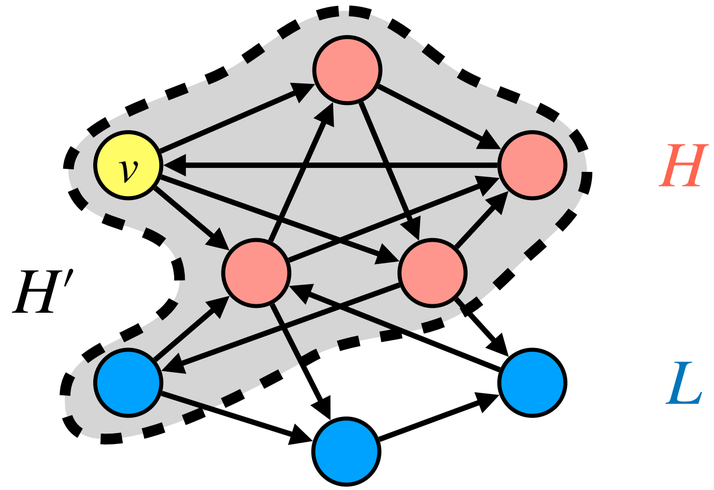
- $\mathrm{OPT} = k_L + k_H + k_{L,H} + k_{H,L}$
- Since $|H| = |k_H| = \mathrm{poly}(1/\epsilon)$, we can safely drop them in the calculation



Streaming Implementation

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Use the decoding approach on $G[L]$, compute the dicut value $\widetilde{k}_L$ of $G[L]$
- Use the second decoding on vertices v in H to obtain $\widetilde{k}_{S^*,v}$ and $\widetilde{k}_{v,T^*}$
- Output $\widetilde{k}_L + \sum_{v \in H} \max\{\widetilde{k}_{S^*,v}, \widetilde{k}_{v,T^*}\} - \frac{\eta m}{16}$



Streaming Algorithm

- **<u>Streaming Phase</u>:**
  - Perform **reservoir sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch**
- **<u>Post-Processing Phase</u>:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute the decodings approximately

Summary

- **Streaming Max-(DI)CUT with predictions**
  - **Takeaway:** ϵ-accurate predictions can help bypass the classical $(1/2+\eta)$ -approx vs $\Omega_\eta(n)$ space trade-off, and can get even $(1-\eta)$-approx vs $\text{poly}(1/\eta, 1/\epsilon)$ space
- **Open Problems**
  - Extension to weighted graphs?
  - Other graph problems in streaming?
  - Extension to any Max-CSP? (Currently, we can solve Boolean Max-2CSP and only bounded degree Max-kCSP)

Thank you!