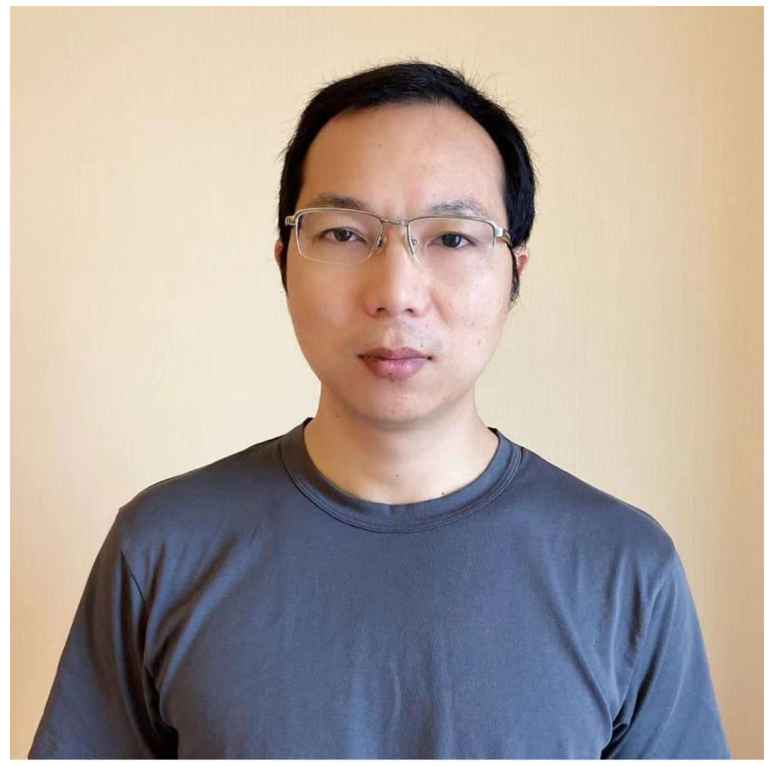
Learning-Augmented Streaming Algorithms for Approximating Max-Cut

Yinhao Dong (董寅灏)

University of Science and Technology of China (USTC)

Based on joint work with



Pan Peng

USTC

Ali Vakilian

TTIC -> Virginia Tech

第六届 CCF 理论计算机科学博士生论坛

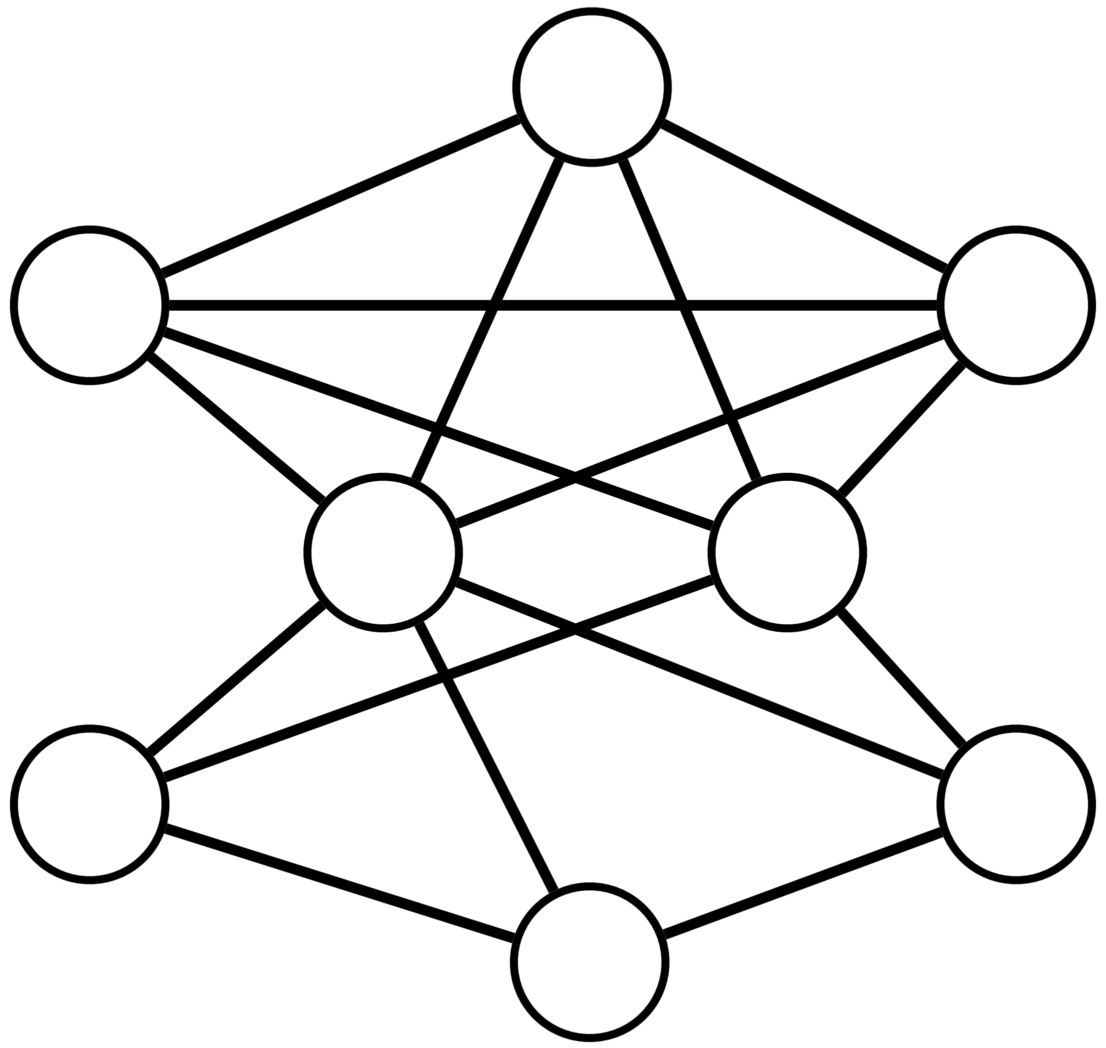
中南大学 · 2025 年 11 月

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

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

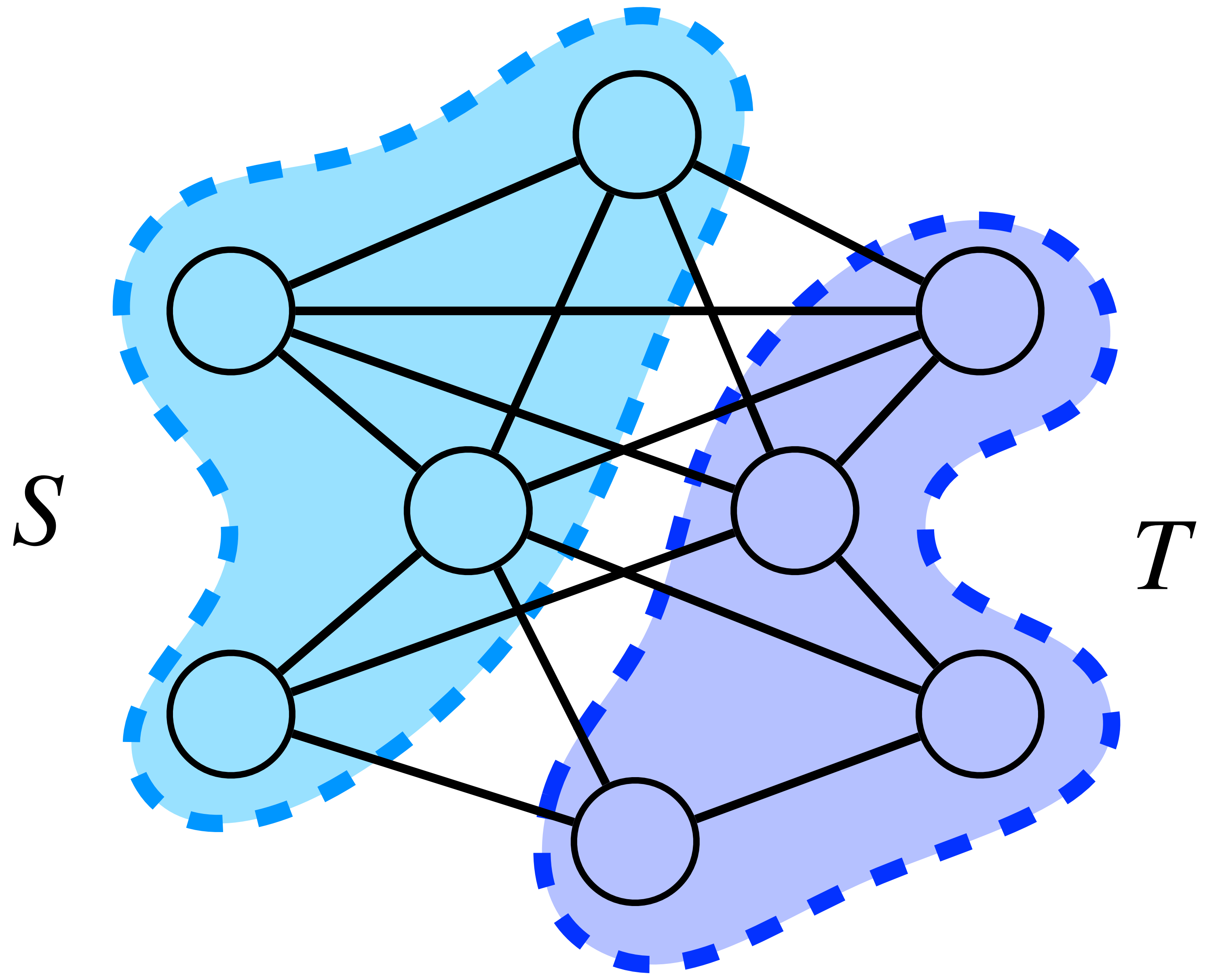


Max-Cut

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

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

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

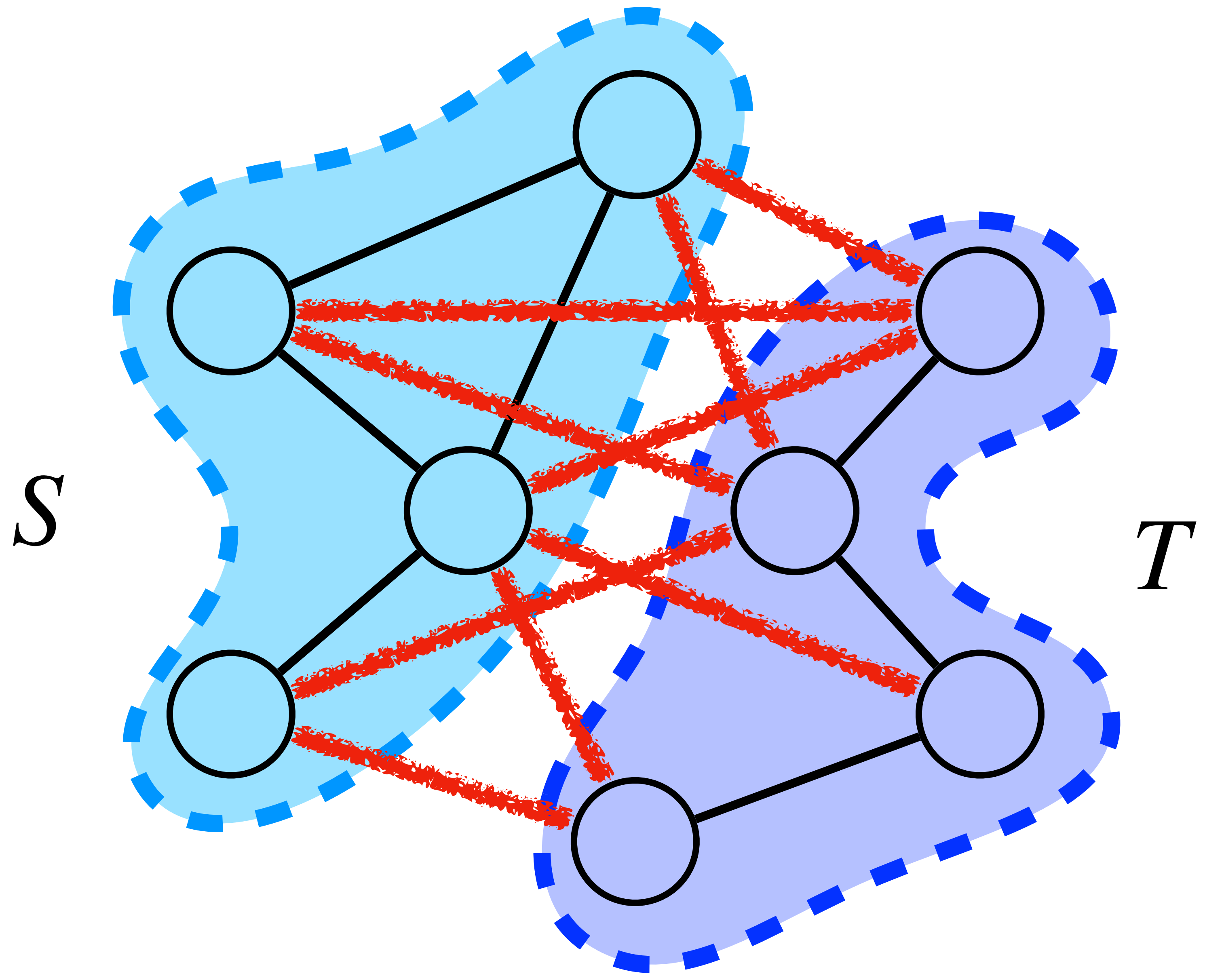


Max-Cut

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

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

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



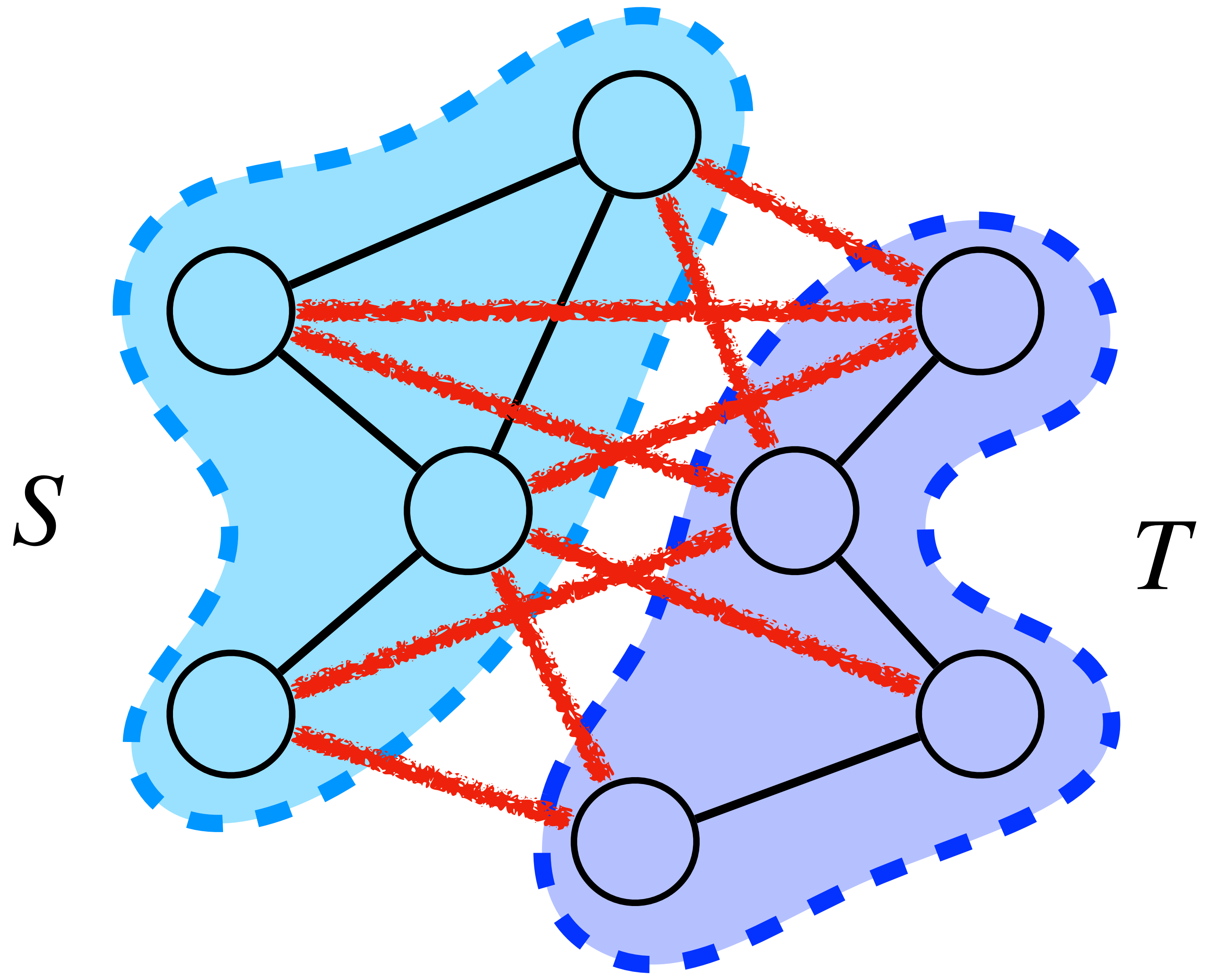
Max-Cut

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



Max-Cut

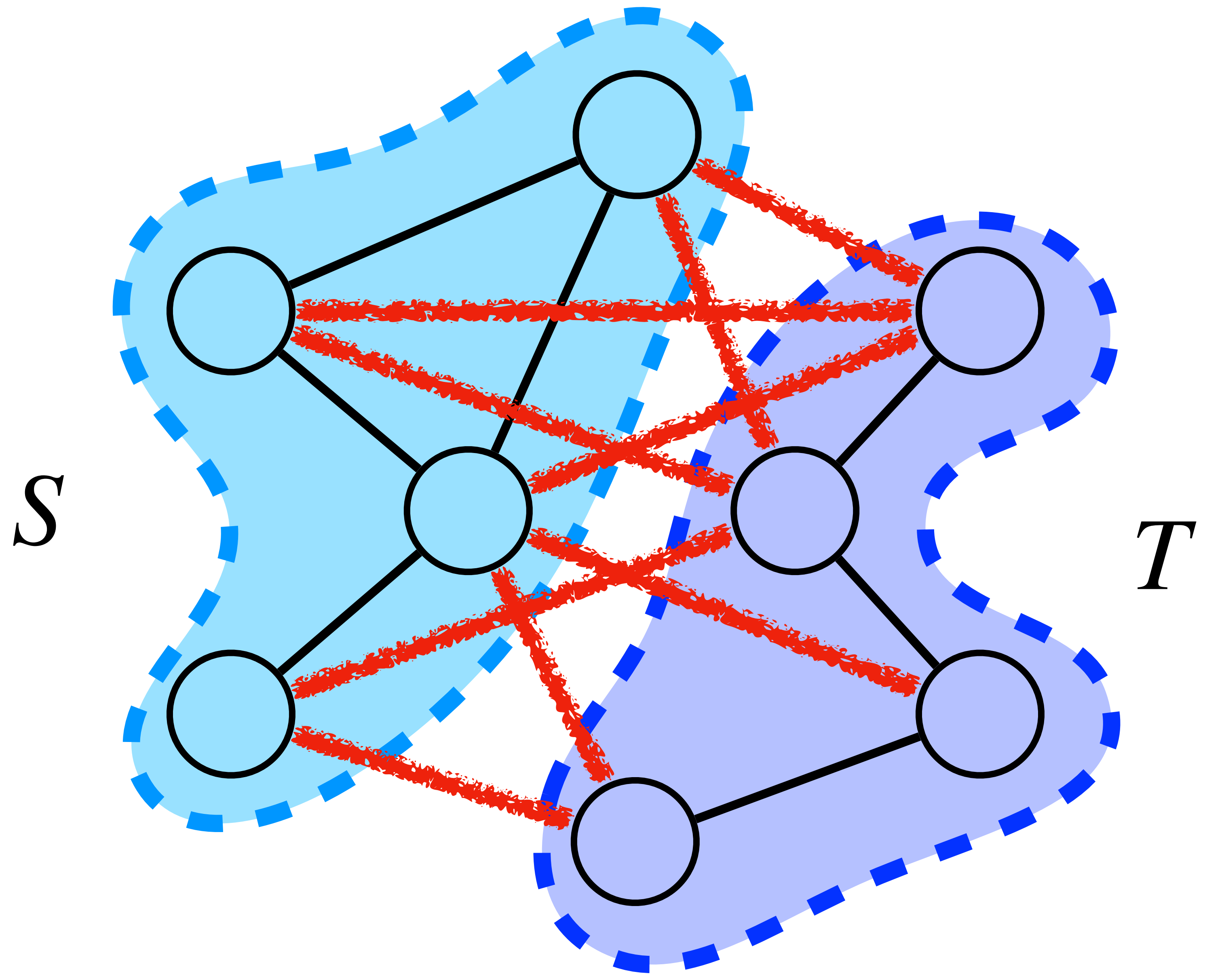
Input: An undirected, unweighted graph $G = (V, E)$

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.

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



Max-Cut

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

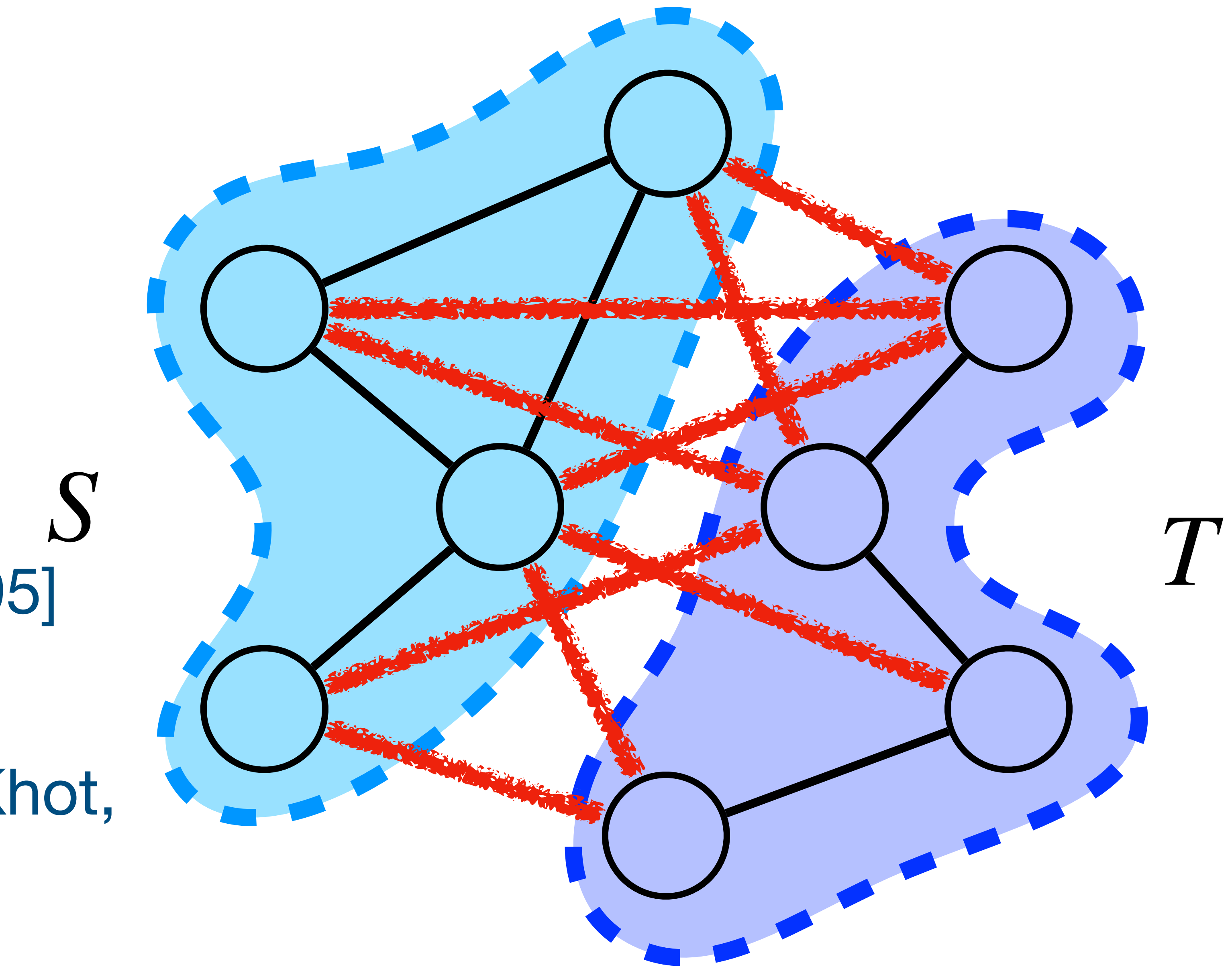
Goal: Find a *bipartition* of V into S and T that maximizes the cut value

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

- **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-Cut

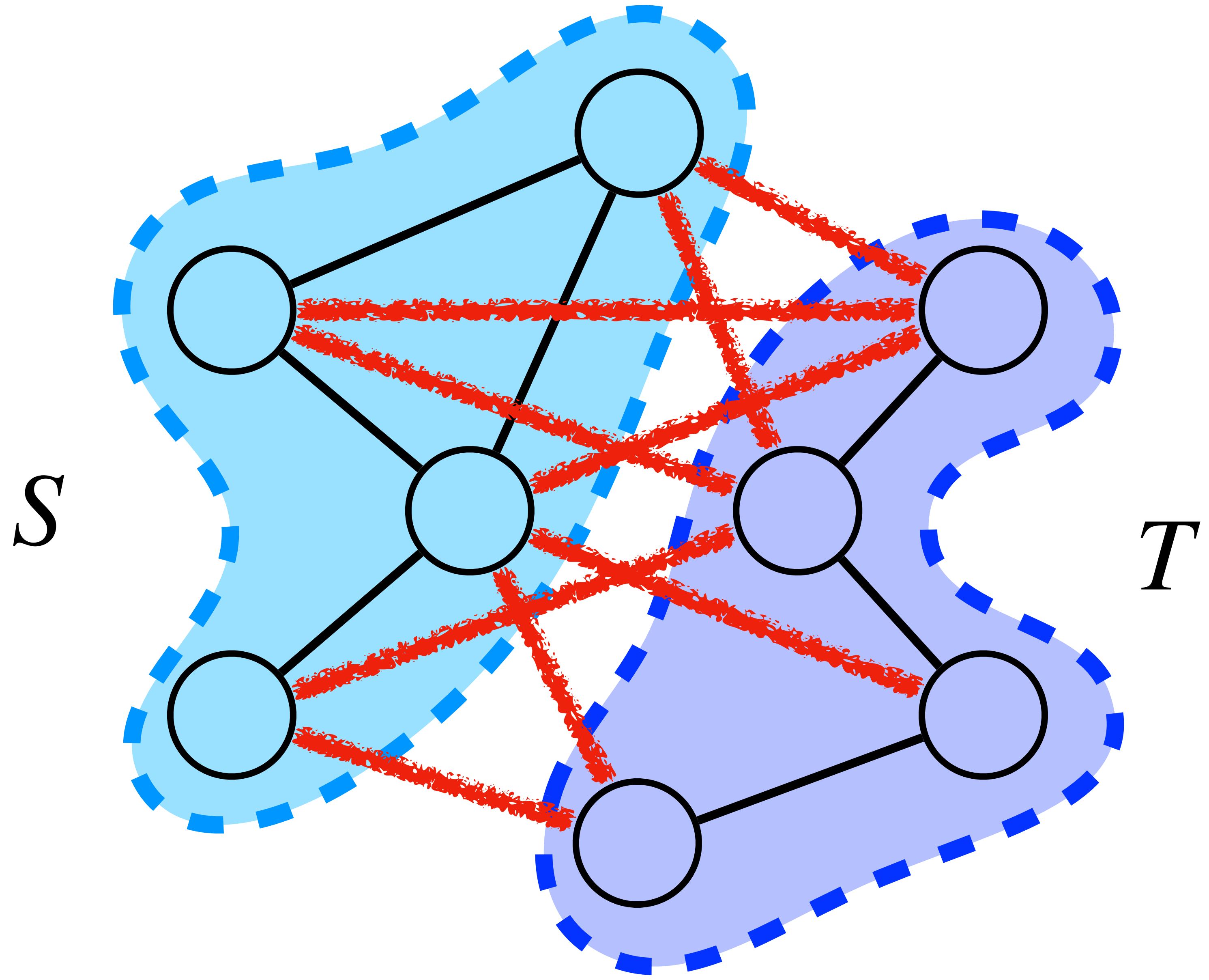
Input: An undirected, unweighted graph $G = (V, E)$

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

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

Estimation Task:

Compute an *estimate* v of the Max-Cut value



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

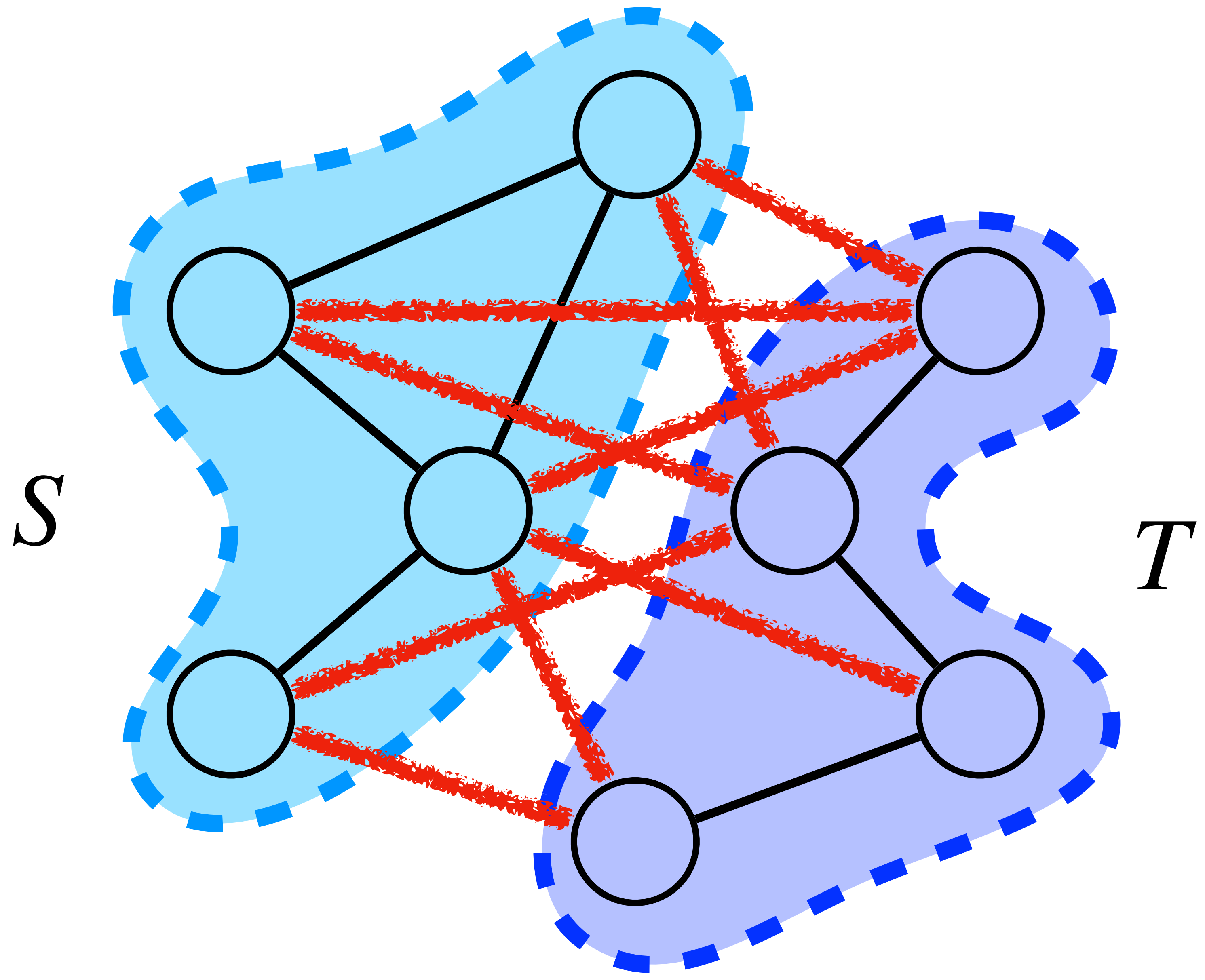
$$|E(S, T)| = |\{(u, v) \in E : u \in S \wedge v \in T\}|$$

Estimation Task:

Compute an *estimate* v of the Max-Cut value

- α-approximation: outputs a **value** v, s.t.

$$\alpha \cdot \text{OPT} \leq v \leq \text{OPT}$$



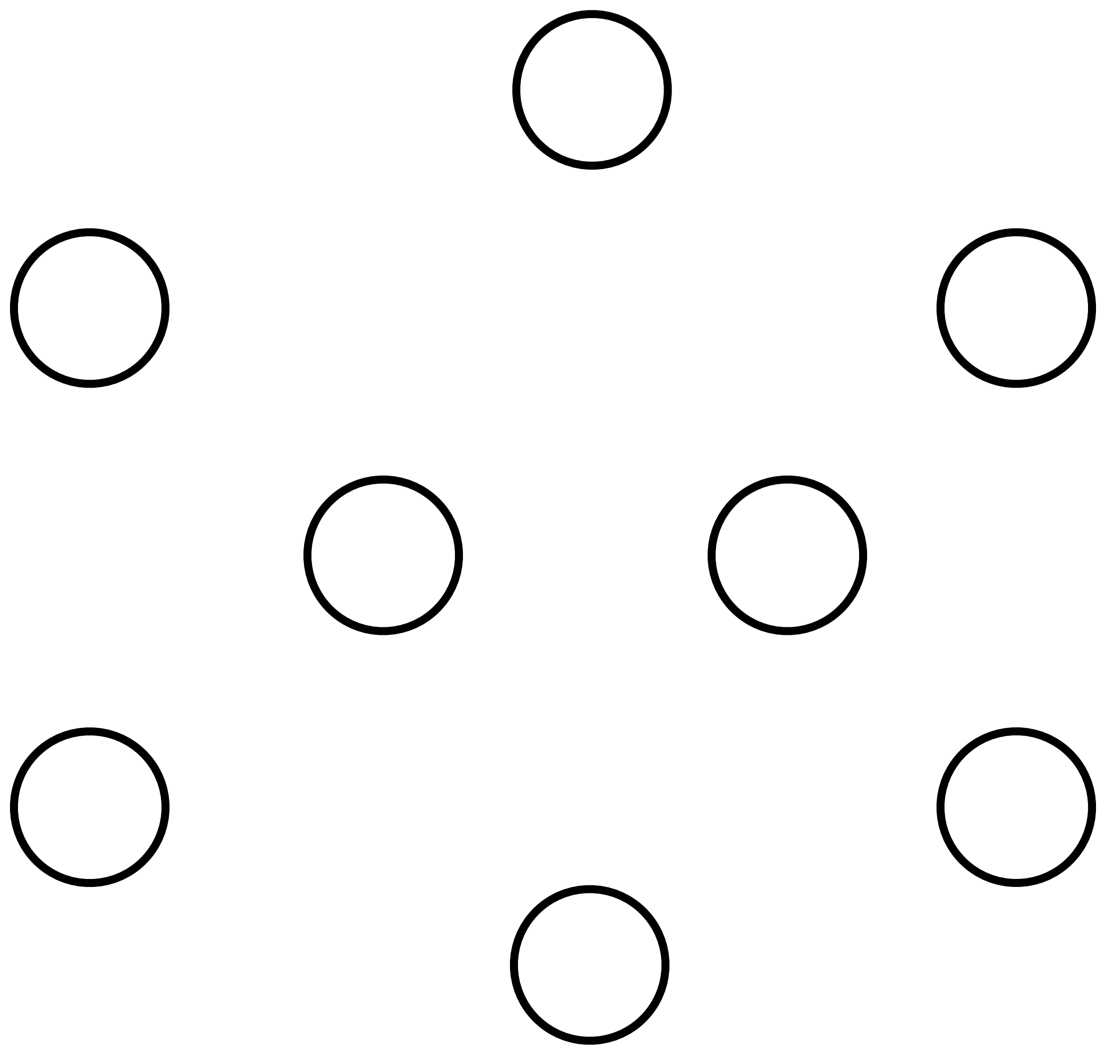
Graph Streaming Model

Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions

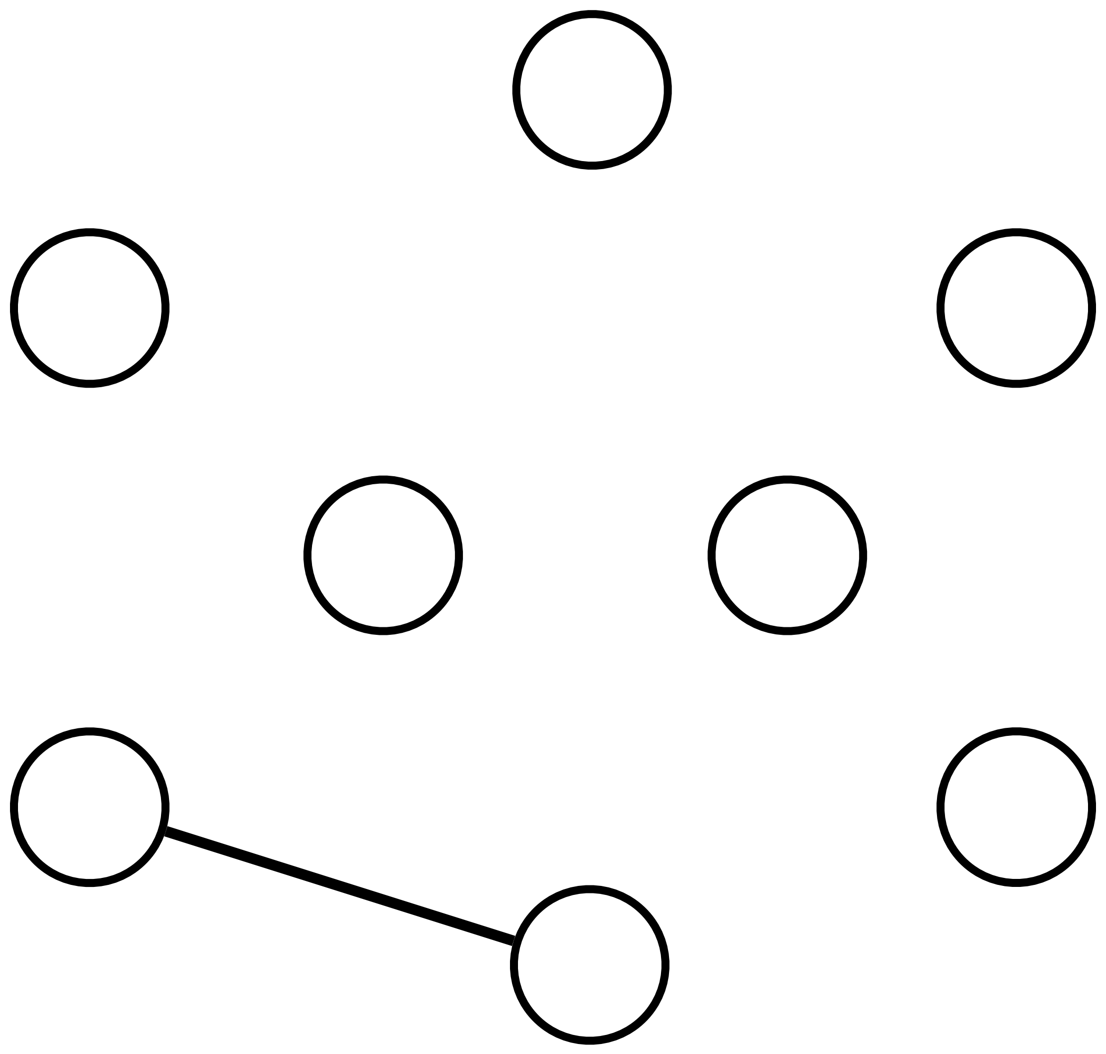
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



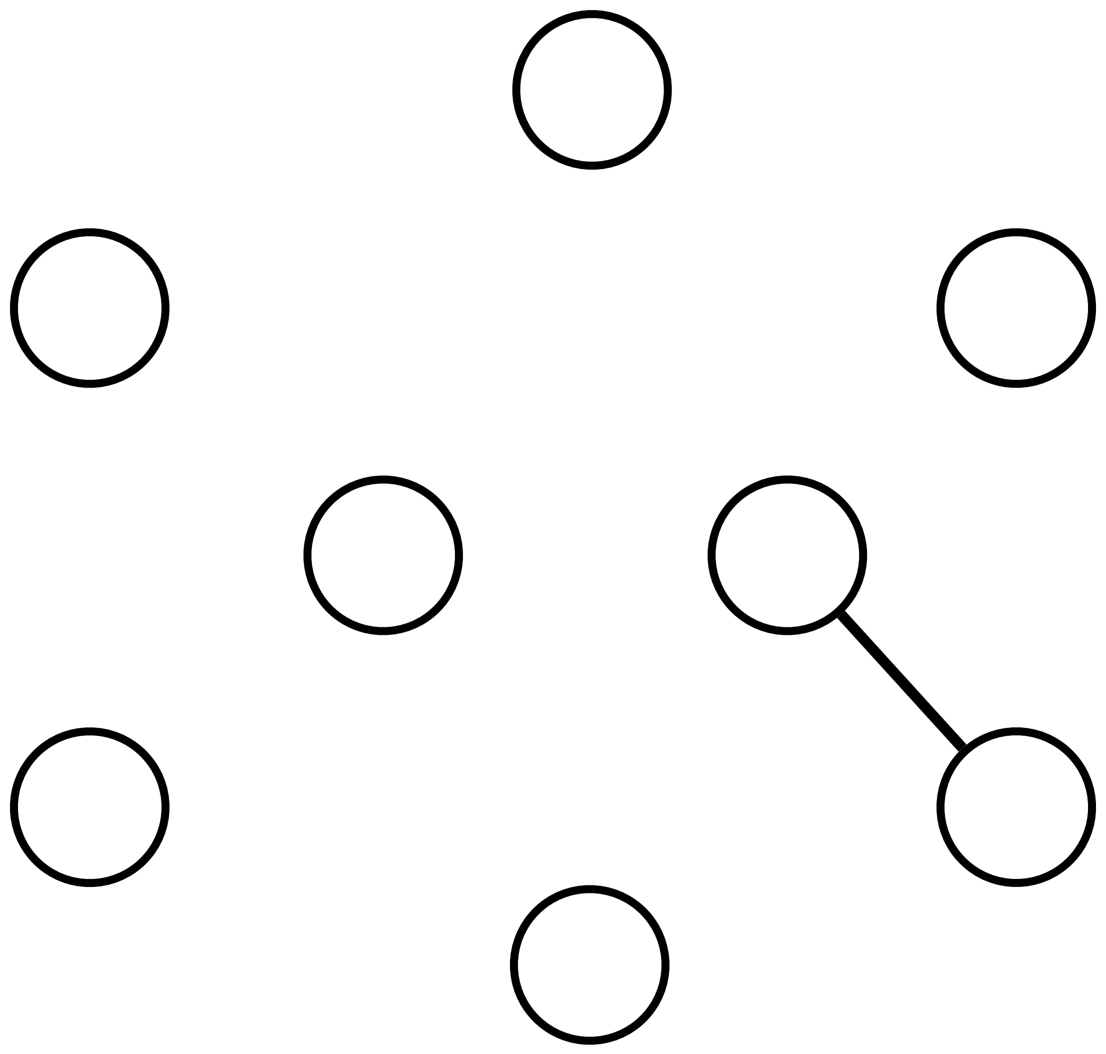
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



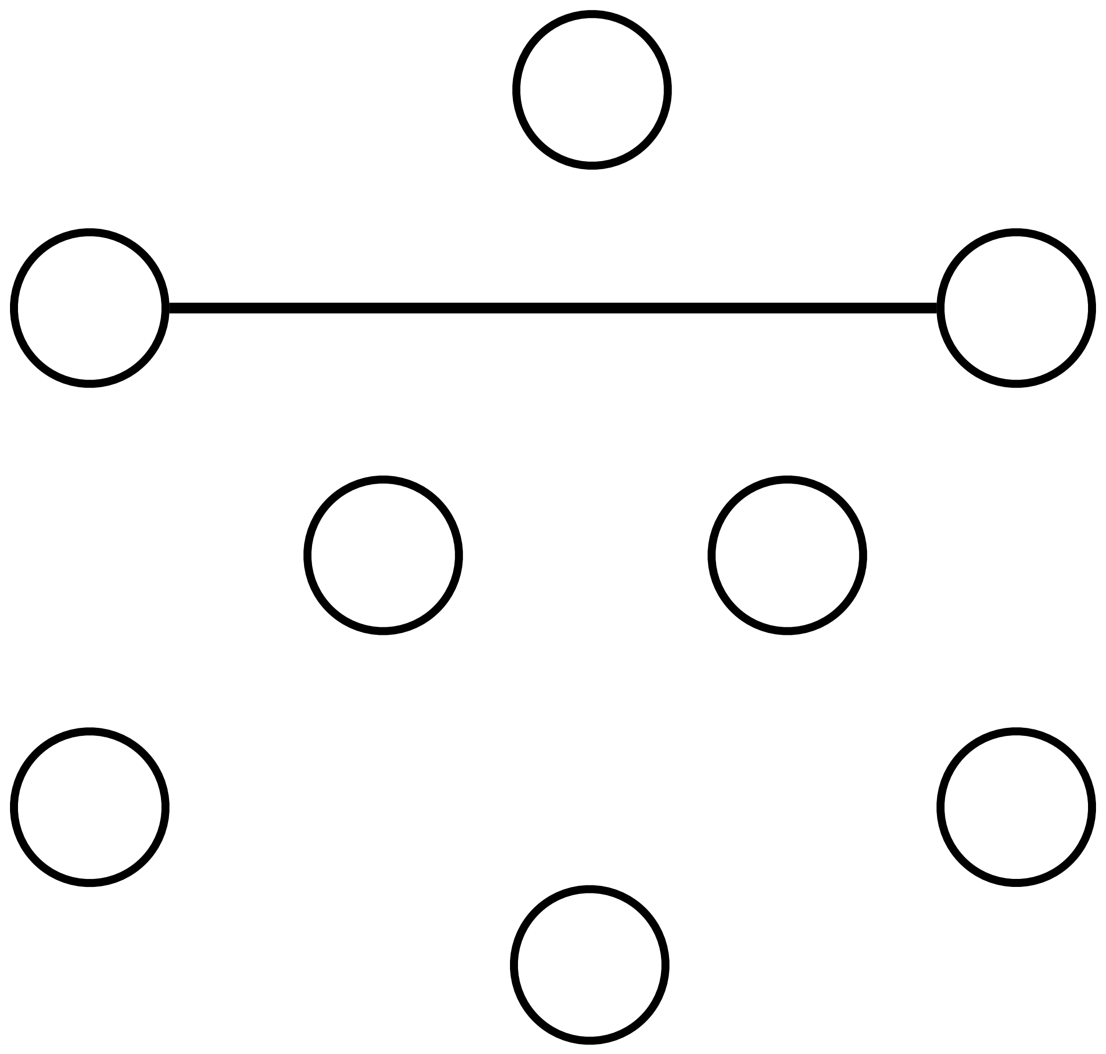
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



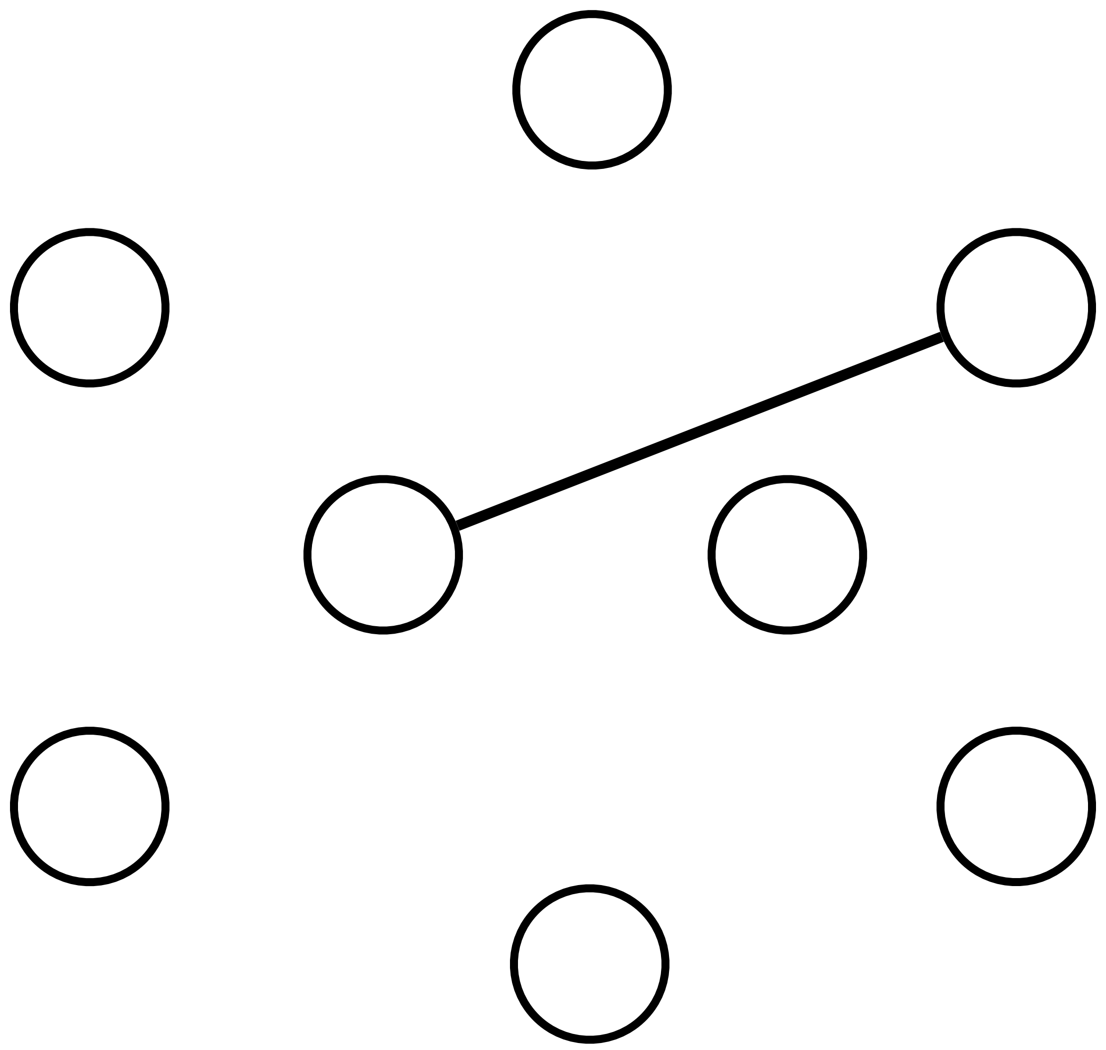
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



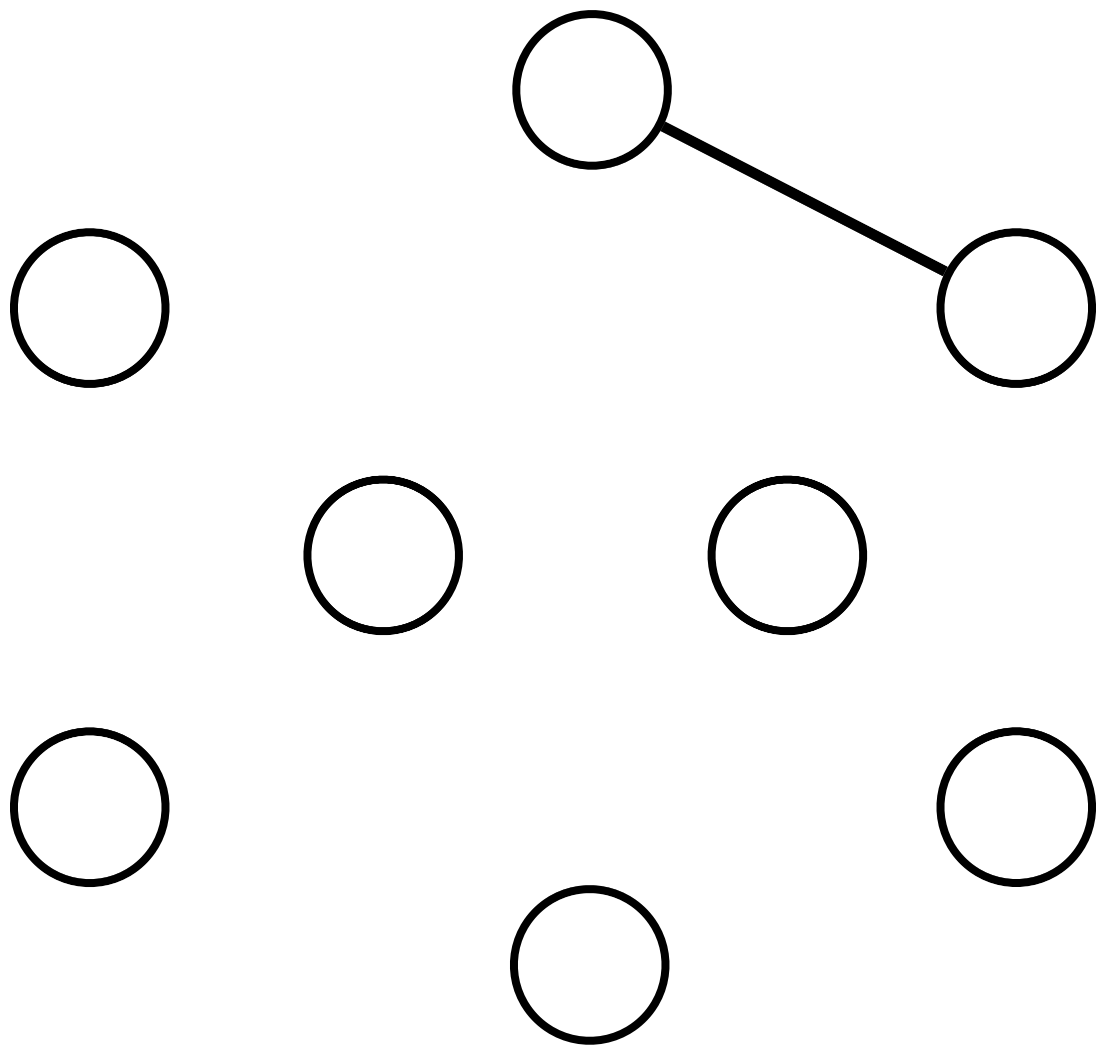
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



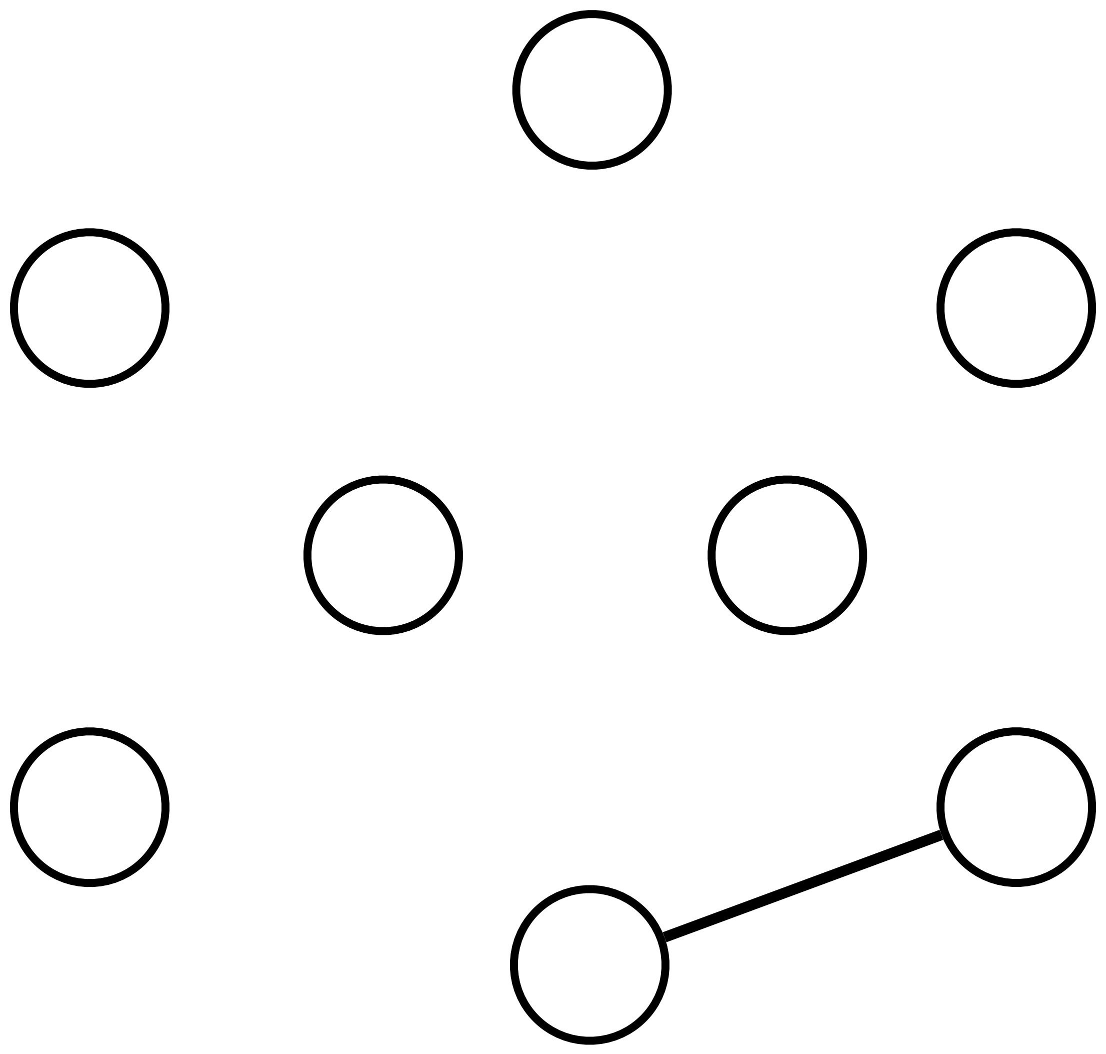
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



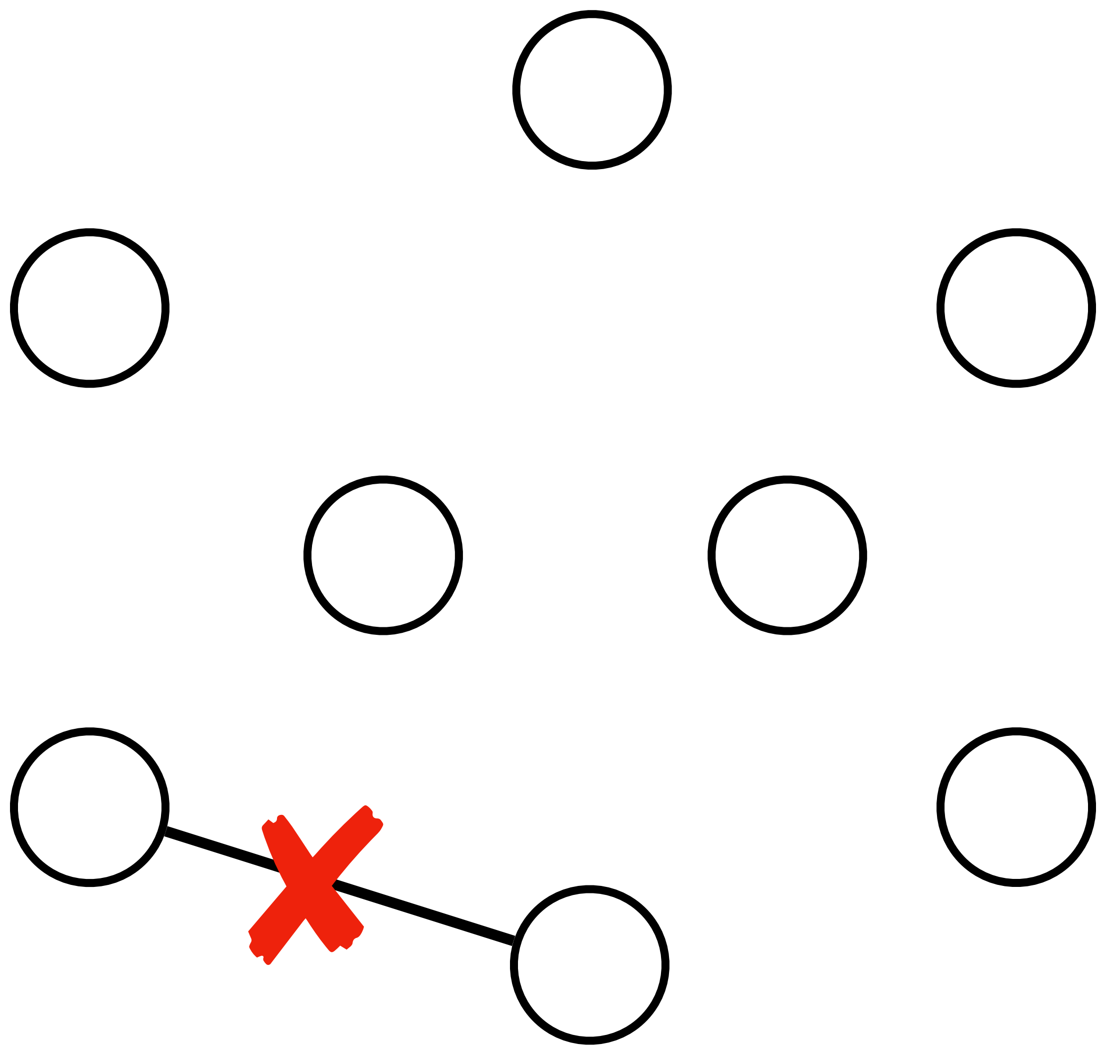
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



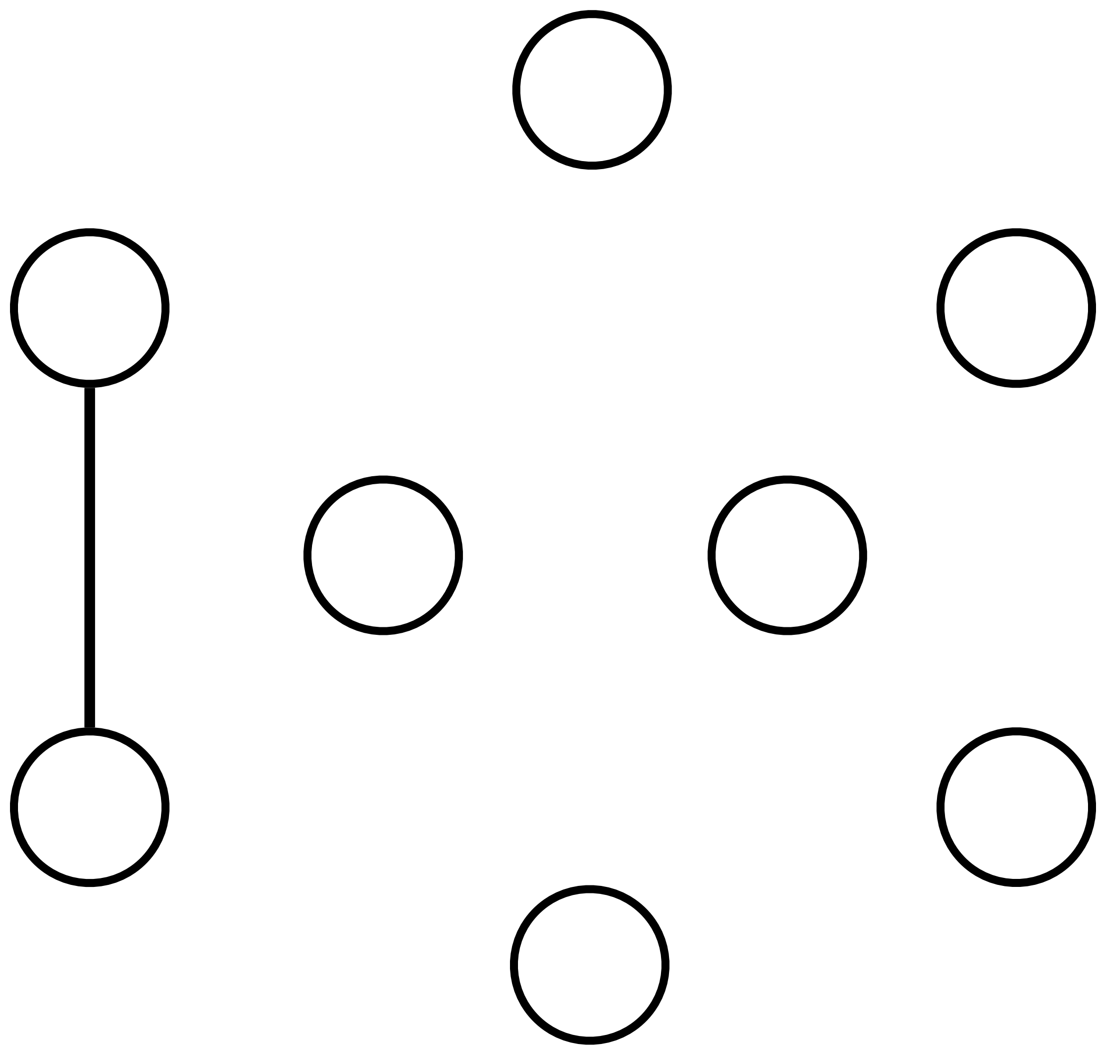
Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions



Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions

Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions
  - *insertion-only* streams: contains of edge insertions only
  - *dynamic* streams: has both insertions and deletions

Graph Streaming Model

- Input graph is presented as a sequence of edge insertions and deletions
    - *insertion-only* streams: contains of edge insertions only
    - *dynamic* streams: has both insertions and deletions
- **Goal:** in **one pass**, using **small space**, compute the solution

Estimating Max-Cut Value in Streaming

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

Goal: Compute an *estimate* of the Max-Cut value using small space

Estimating Max-Cut Value in Streaming

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

Goal: Compute an *estimate* of the Max-Cut value using small space

- A trivial $1/2$-approximation using $O(\log n)$ bits of space: count the number of edges m and output $m/2$

Estimating Max-Cut Value in Streaming

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

Goal: Compute an *estimate* of the Max-Cut value using small space

- A trivial $1/2$-approximation using $O(\log n)$ bits of space: count the number of edges m and output $m/2$
- A series of works explored the approximation ratio vs space trade-offs
  [Kapralov, Khanna, Sudan, SODA'15] [Kogan, Krauthgamer, ITCS'15] [Kapralov, Khanna, Sudan, Velingker, SODA'17] [Kapralov, Krachun, STOC'19]

Estimating Max-Cut Value in Streaming

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

Goal: Compute an *estimate* of the Max-Cut value using small space

- A trivial $1/2$-approximation using $O(\log n)$ bits of space: count the number of edges m and output $m/2$
- A series of works explored the approximation ratio vs space trade-offs
[Kapralov, Khanna, Sudan, SODA'15] [Kogan, Krauthgamer, ITCS'15] [Kapralov, Khanna, Sudan, Velingker, SODA'17] [Kapralov, Krachun, STOC'19]
- Any (randomized) algorithm that achieves $(1/2 + \epsilon)$-approximation requires $\Omega_\epsilon(n)$ bits of space [Kapralov, Krachun, STOC'19]

Estimating Max-Cut Value in Streaming

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

Goal: Compute an *estimate* of the Max-Cut value using small space

- A trivial $1/2$-approximation using $O(\log n)$ bits of space: count the number of edges m and output $m/2$
- A series of works explored the approximation ratio vs space trade-offs [Kapralov, Khanna, Sudan, SODA'15] [Kogan, Krauthgamer, ITCS'15] [Kapralov, Khanna, Sudan, Velingker, SODA'17] [Kapralov, Krachun, STOC'19]
- Any (randomized) algorithm that achieves $(1/2 + \epsilon)$-approximation requires $\Omega_\epsilon(n)$ bits of space [Kapralov, Krachun, STOC'19]

What if we can obtain extra information about the input?

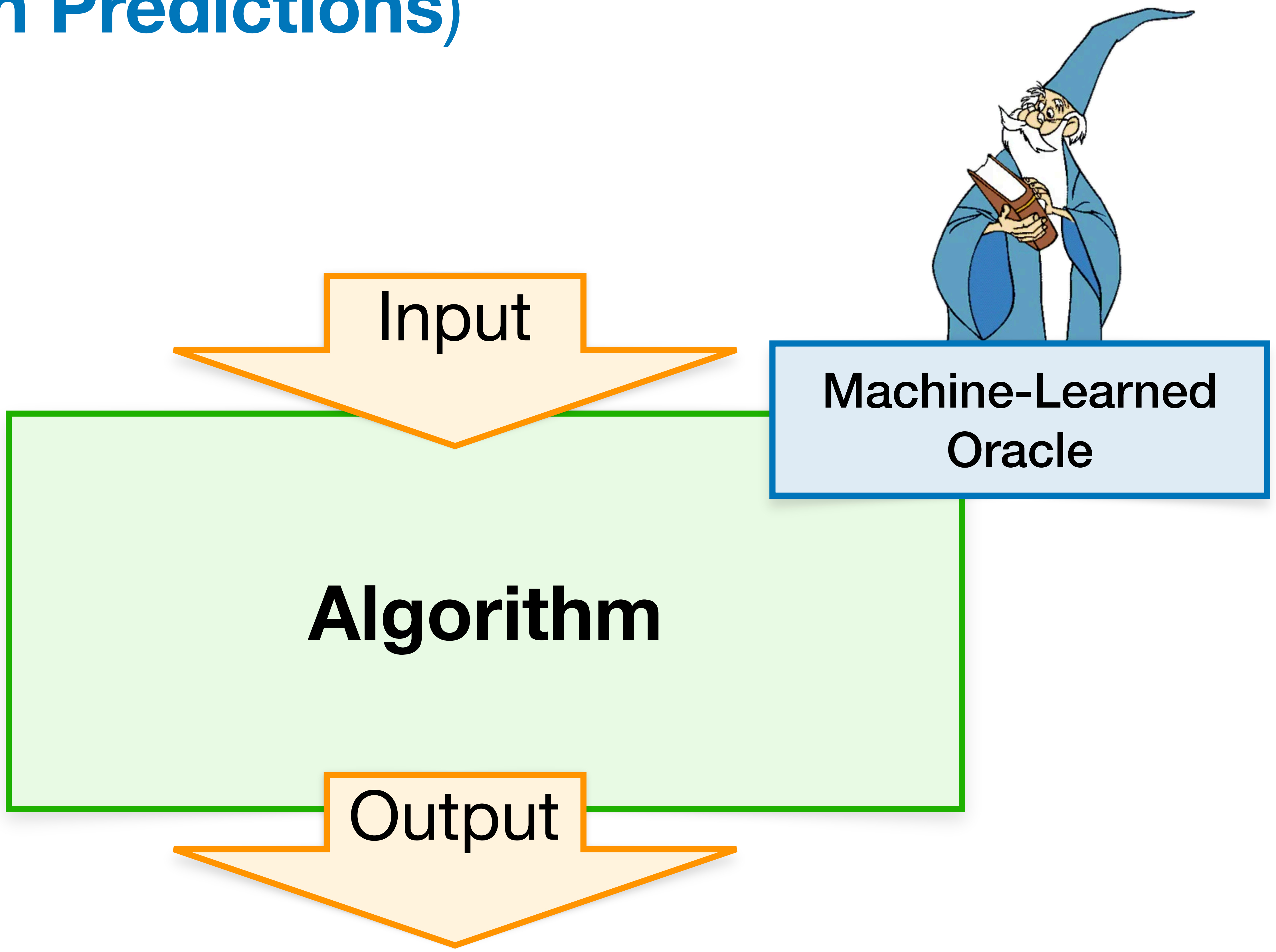
Learning-Augmented Algorithms

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

Learning-Augmented Algorithms

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

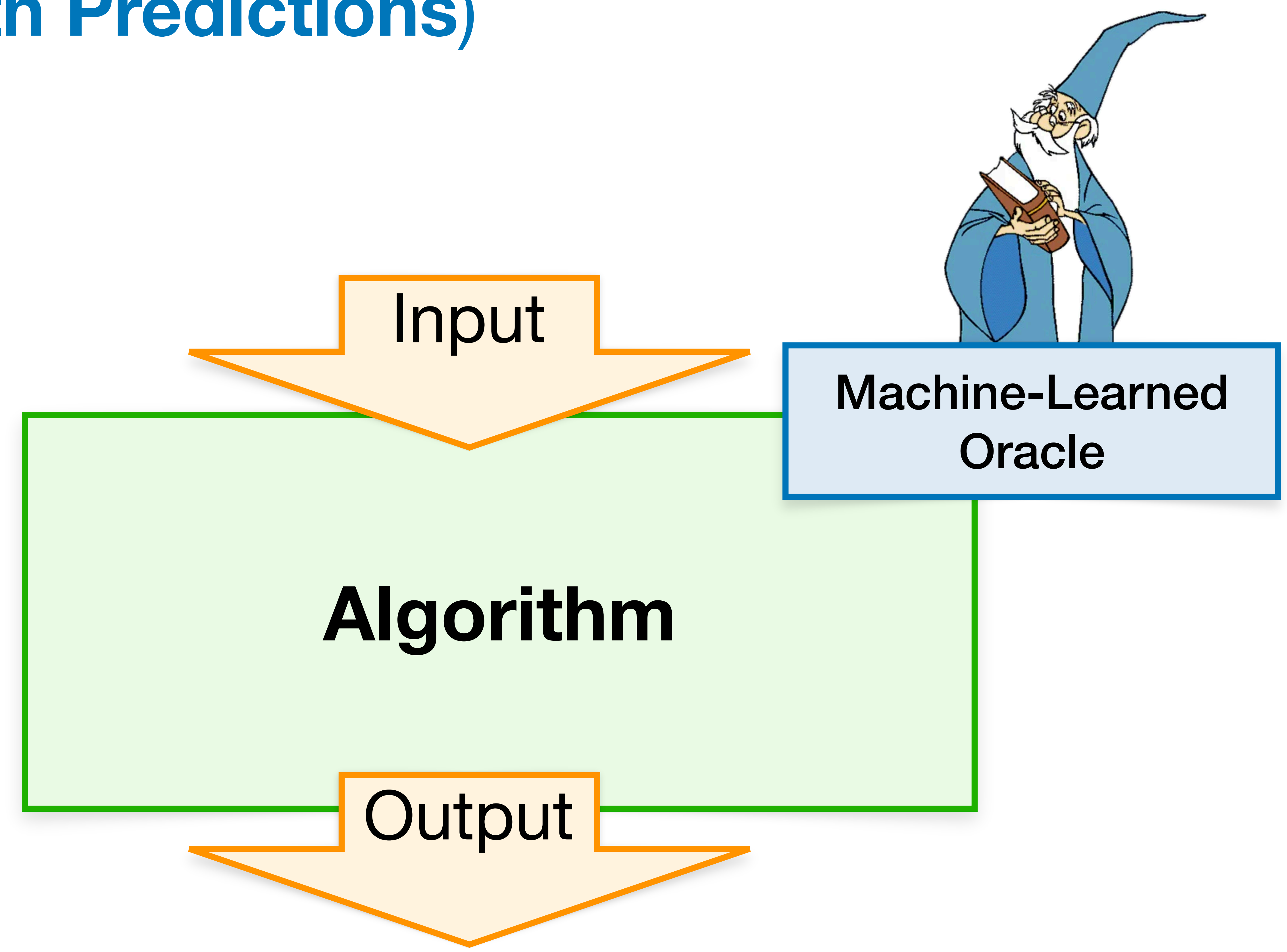
- The algorithm has access to a **learned oracle** providing a certain type of **predictions** about the input instance



Learning-Augmented Algorithms

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

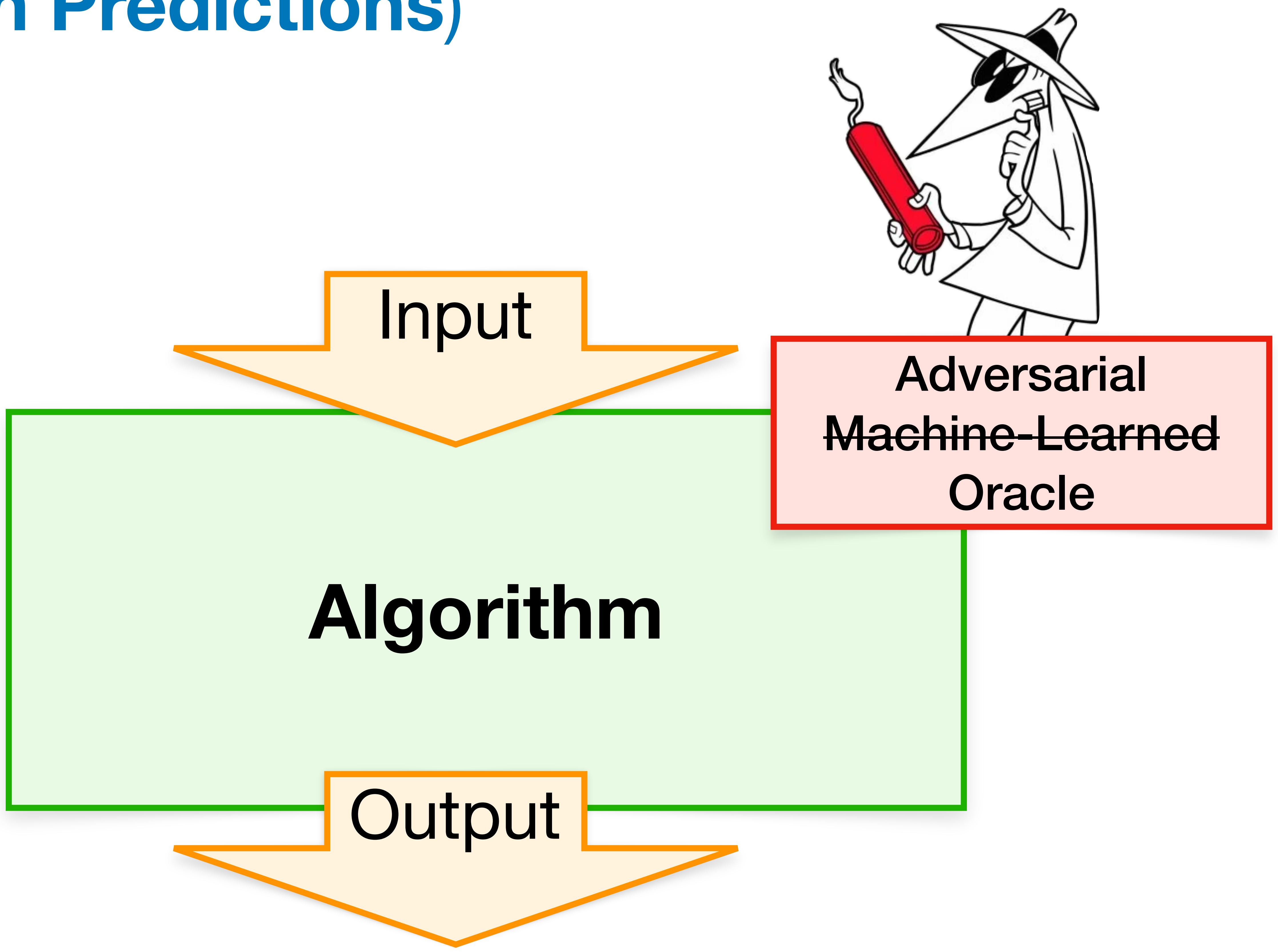
- The algorithm has access to a **learned oracle** providing a certain type of **predictions** about the input instance
- **Goals:**
  - **Consistency:** High prediction quality $\Longrightarrow$ Better performance than the best-known classical algorithm



Learning-Augmented Algorithms

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

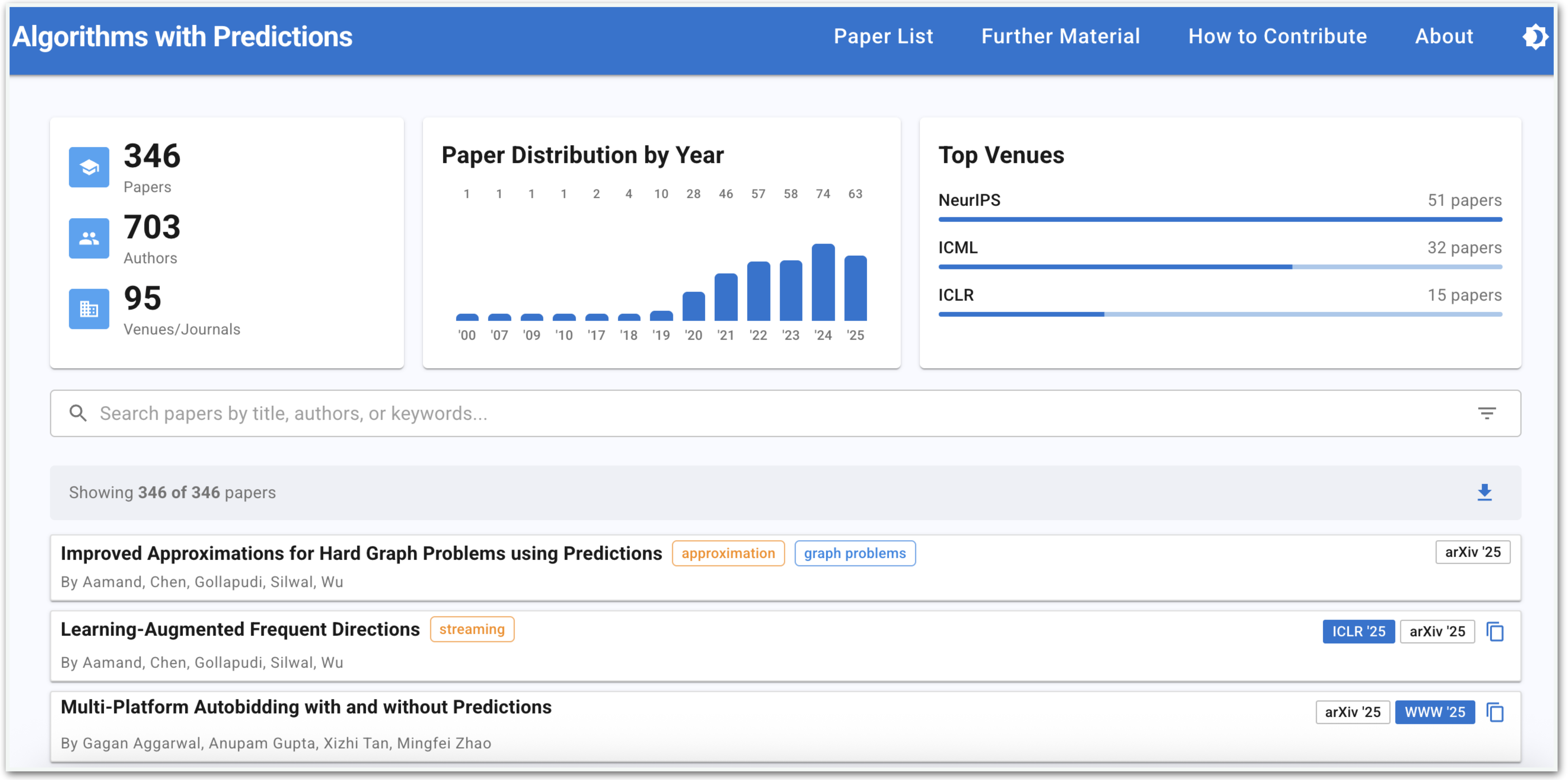
- The algorithm has access to a **learned oracle** providing a certain type of **predictions** about the input instance
- **Goals:**
  - **Consistency:** High prediction quality $\implies$ Better performance than the best-known classical algorithm
  - **Robustness:** Low prediction quality $\implies$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

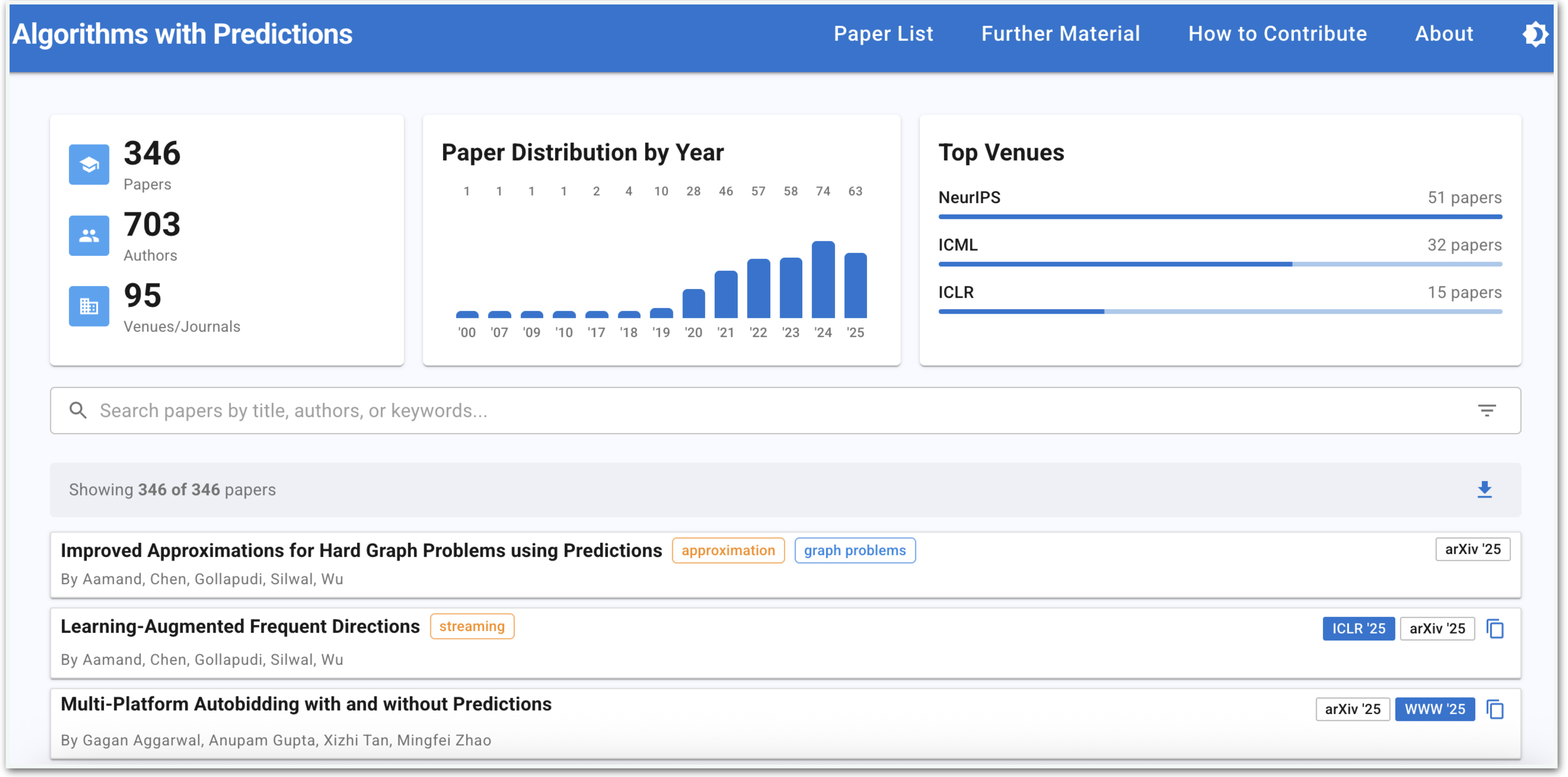


An up-to-date repository of publications (https://algorithms-with-predictions.github.io)

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

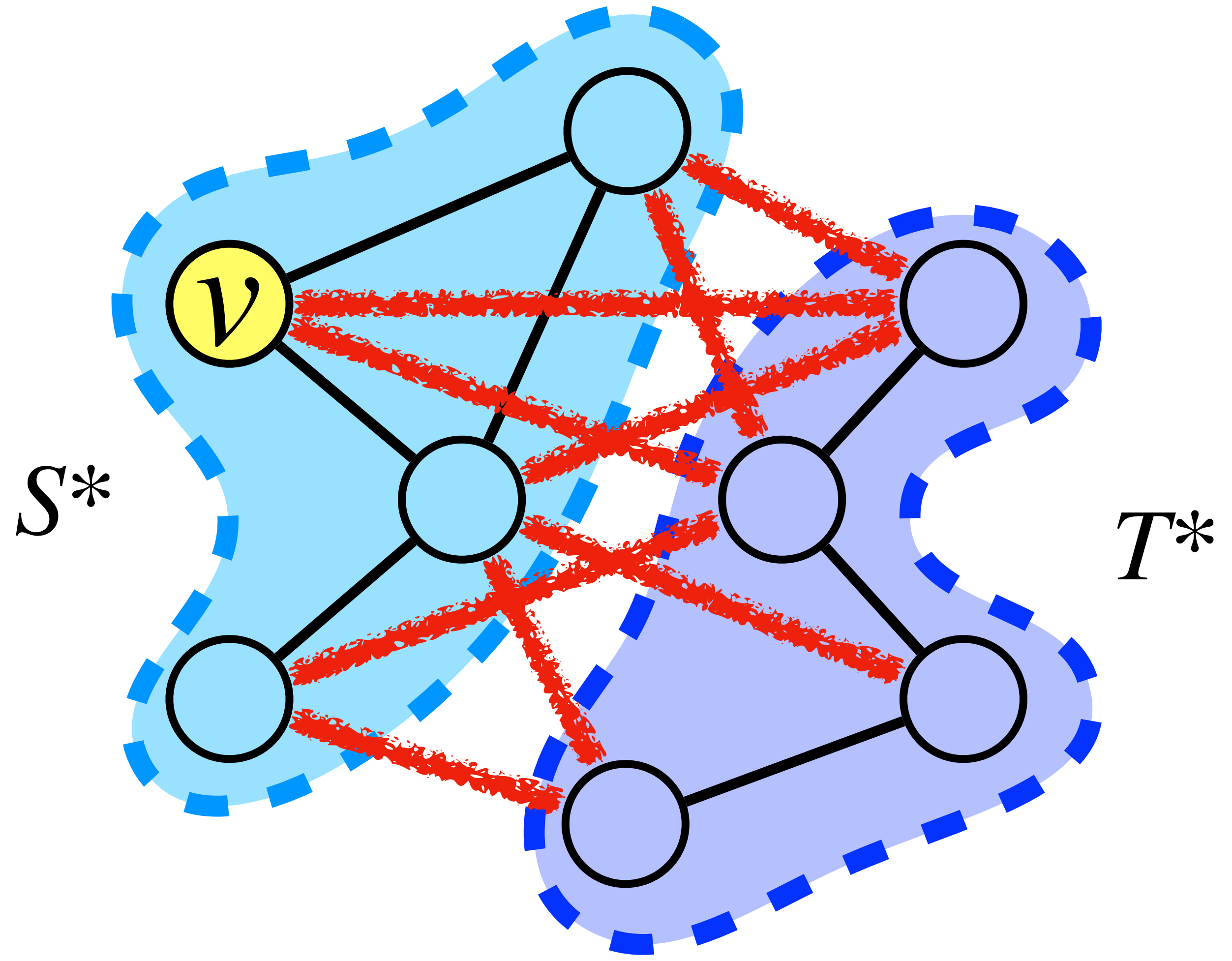
- **ϵ-accurate predictions** [Cohen-Addad, d'Orsi, Gupta, Lee, Panigrahi, NeurIPS'24] [Braverman, Dharangutte, Shah, Wang, APPROX'24] [Ghoshal, Makarychev, Makarychev, SODA'25]

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**

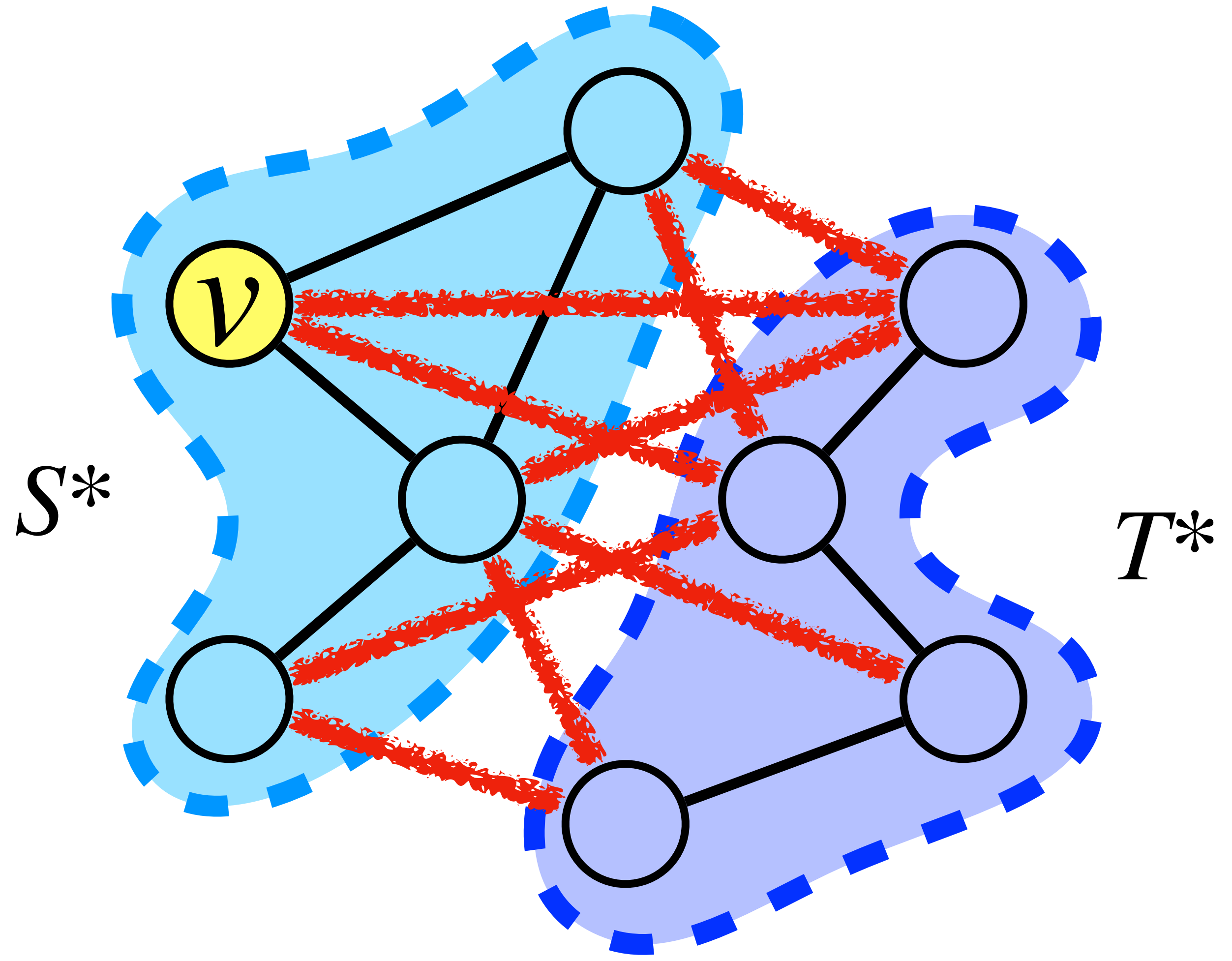
Our Prediction Model

- **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**



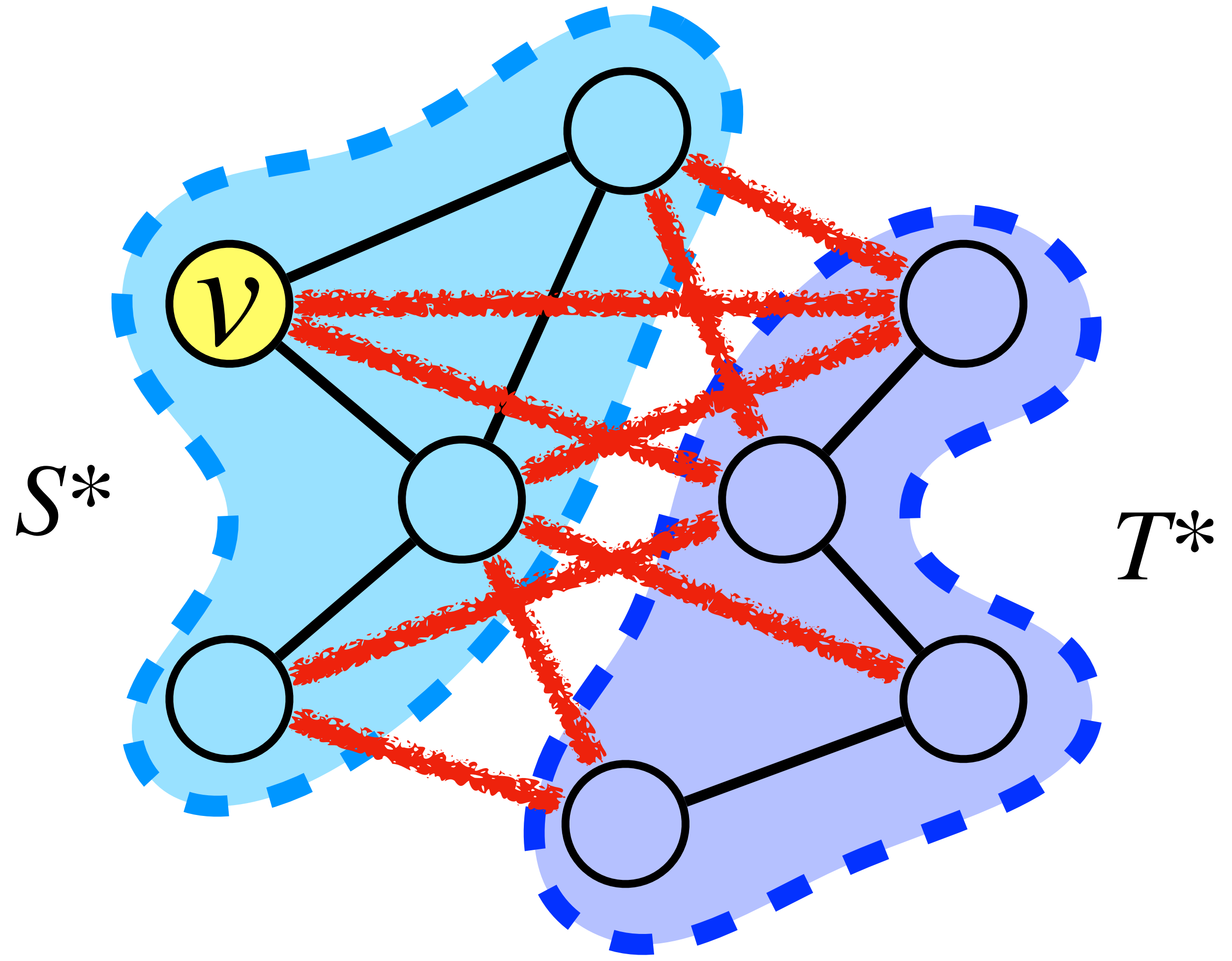
Our Prediction Model

- **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** $Y \in \{-1,1\}^n$



Our Prediction Model

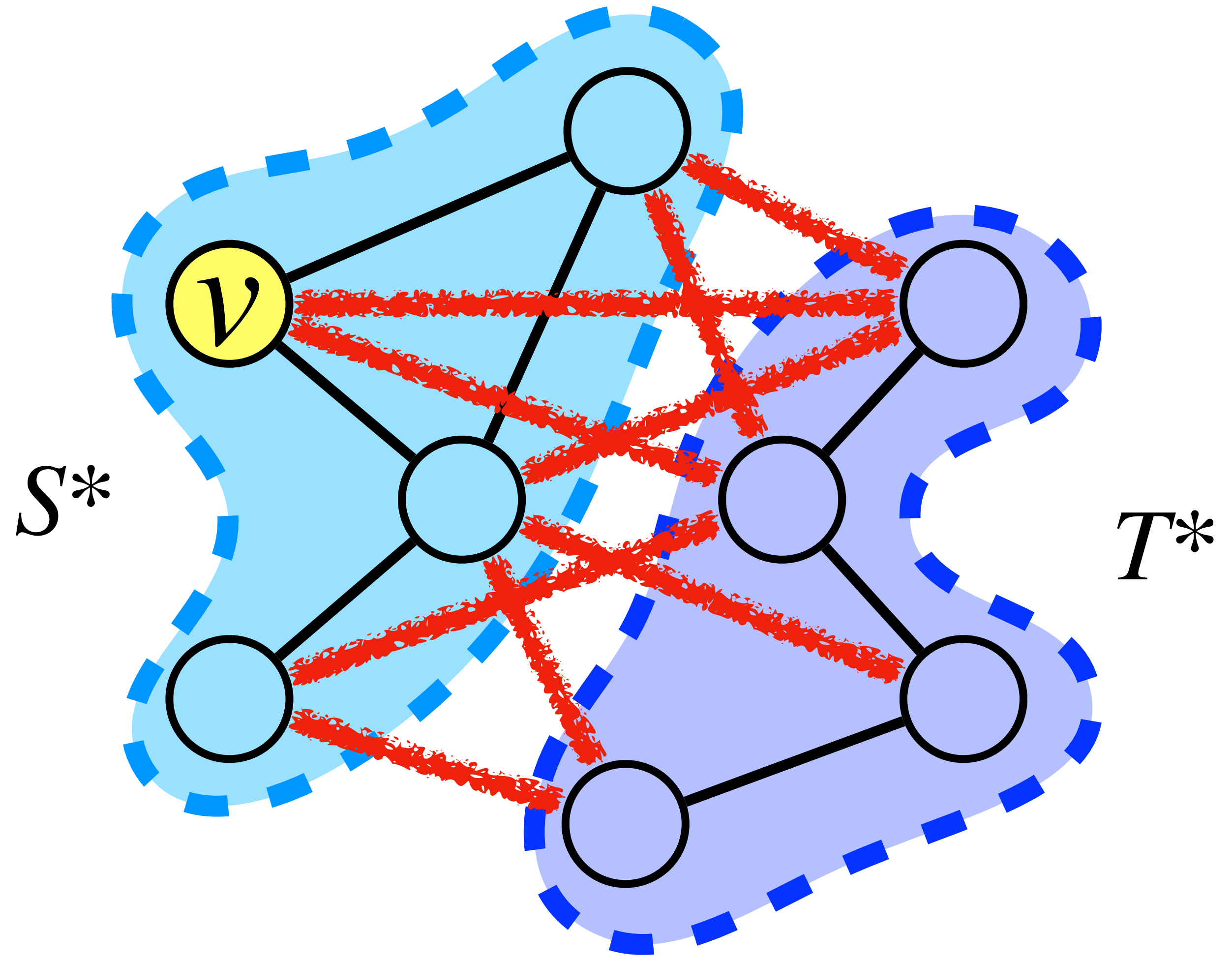
- **ϵ-accurate predictions** [Cohen-Addad, d'Orsi, Gupta, Lee, Panigrahi, NeurIPS'24] [Braverman, Dharangutte, Shah, Wang, APPROX'24] [Ghoshal, Makarychev, Makarychev, SODA'25]
  - Let $x^* \in \{-1,1\}^n$ denote some fixed but unknown **optimal solution**
  - Oracle access to a **prediction vector**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



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

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!

Our Results

Theorem [**D.**, Peng, Vakilian, ITCS'25]**:**
Given ϵ-accurate predictions, there exists a single-pass streaming algorithm achieving a $(1/2 + \Omega(\epsilon^2))$-approx with high probability, that uses

- $\text{poly}(1/\epsilon)$ words of space in **insertion-only streams**, and
- $\text{poly}(1/\epsilon, \log n)$ words of space in **dynamic streams**.

Our Results

Theorem [D., Peng, Vakilian, ITCS'25]**:**
Given ϵ-accurate predictions, there exists a single-pass streaming algorithm achieving a $(1/2 + \Omega(\epsilon^2))$-approx with high probability, that uses

- $\text{poly}(1/\epsilon)$ words of space in **insertion-only streams**, and
- $\text{poly}(1/\epsilon, \log n)$ words of space in **dynamic streams**.

- Bypass the classical $(1/2 + \epsilon)$-approx vs $\Omega_\epsilon(n)$ space trade-off [Kapralov, Krachun, STOC'19]

Our Results

Theorem [D., Peng, Vakilian, ITCS'25]**:**
Given ϵ-accurate predictions, there exists a single-pass streaming algorithm achieving a $(1/2 + \Omega(\epsilon^2))$-approx with high probability, that uses

- $\text{poly}(1/\epsilon)$ words of space in **insertion-only streams**, and
- $\text{poly}(1/\epsilon, \log n)$ words of space in **dynamic streams**.

- Bypass the classical $(1/2 + \epsilon)$-approx vs $\Omega_\epsilon(n)$ space trade-off [Kapralov, Krachun, STOC'19]
- No need to store predictions (only oracle access suffices)

Simple Case: Low-Degree Graphs

Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Algorithm for Low-Degree Graphs

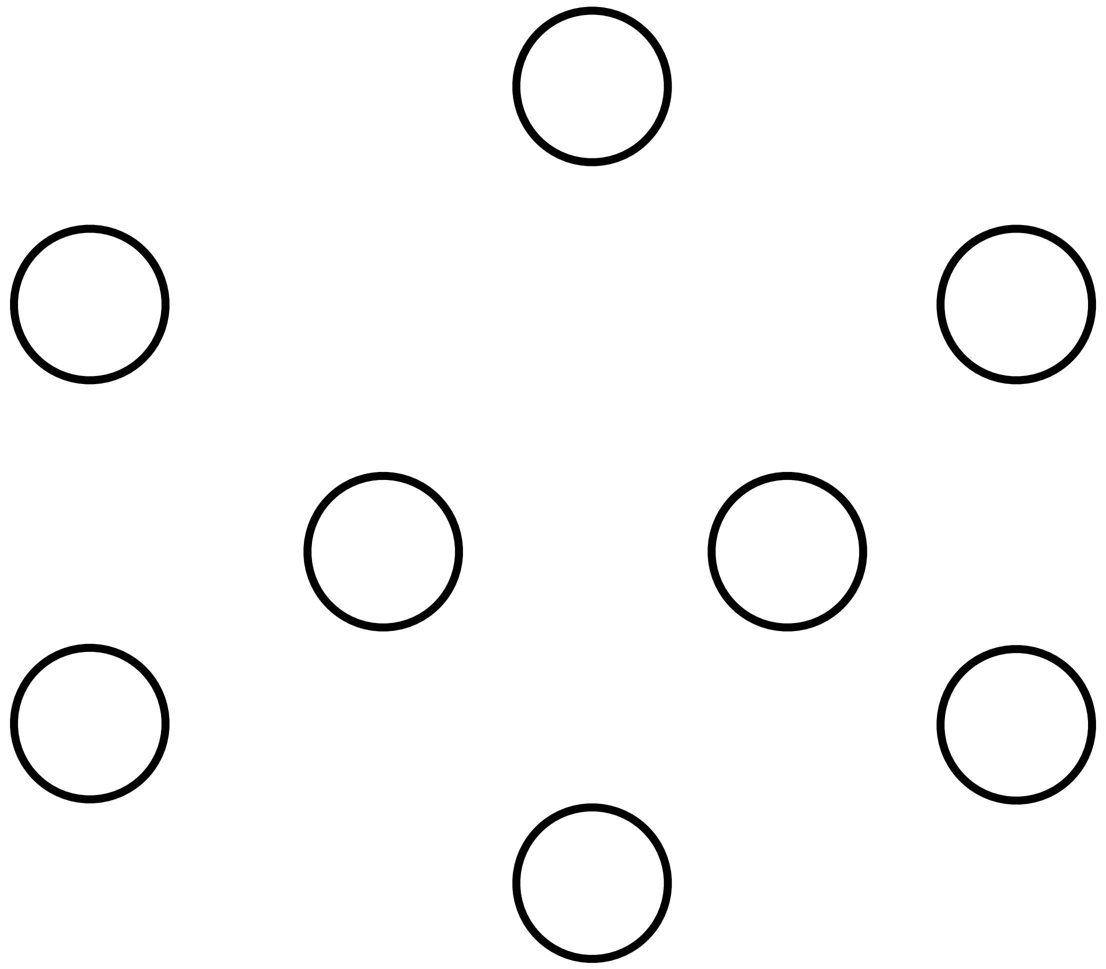
- initially $X = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u \neq Y_v$ then $X = X + 1$
- **return** X

Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Algorithm for Low-Degree Graphs

- initially $X = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u \neq Y_v$ then $X = X + 1$
- **return** X

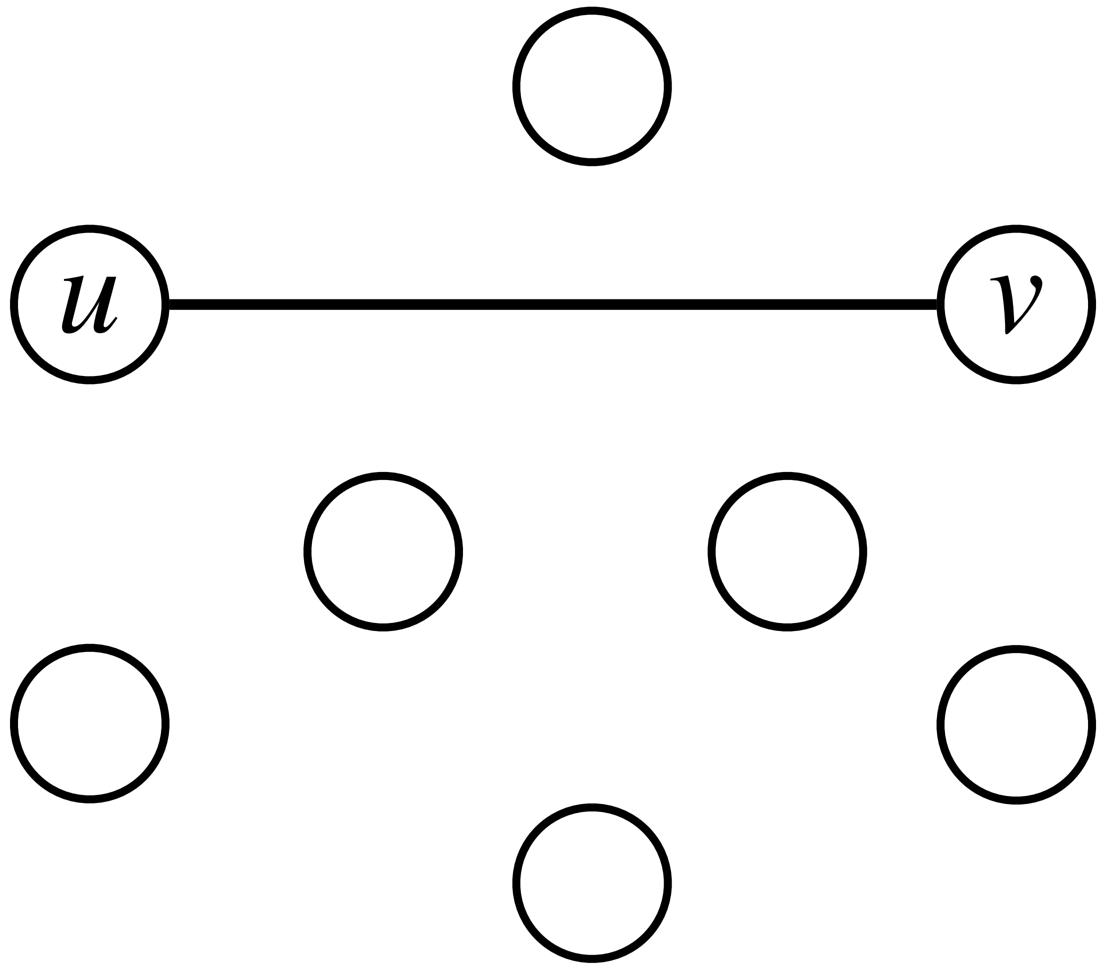


Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Algorithm for Low-Degree Graphs

- initially $X = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u \neq Y_v$ then $X = X + 1$
- **return** X

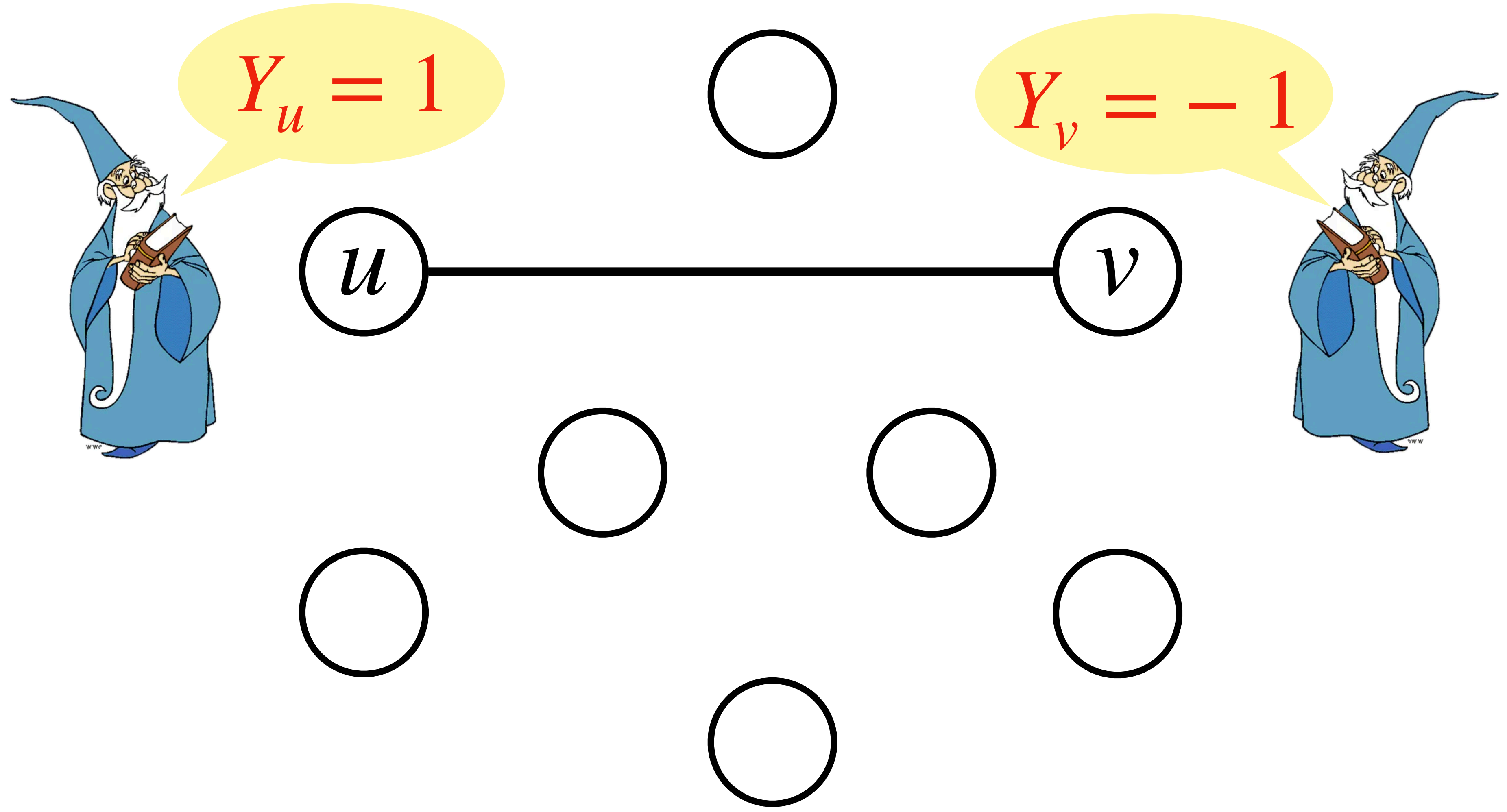


Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Algorithm for Low-Degree Graphs

- initially $X = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u \neq Y_v$ then $X = X + 1$
- **return** X



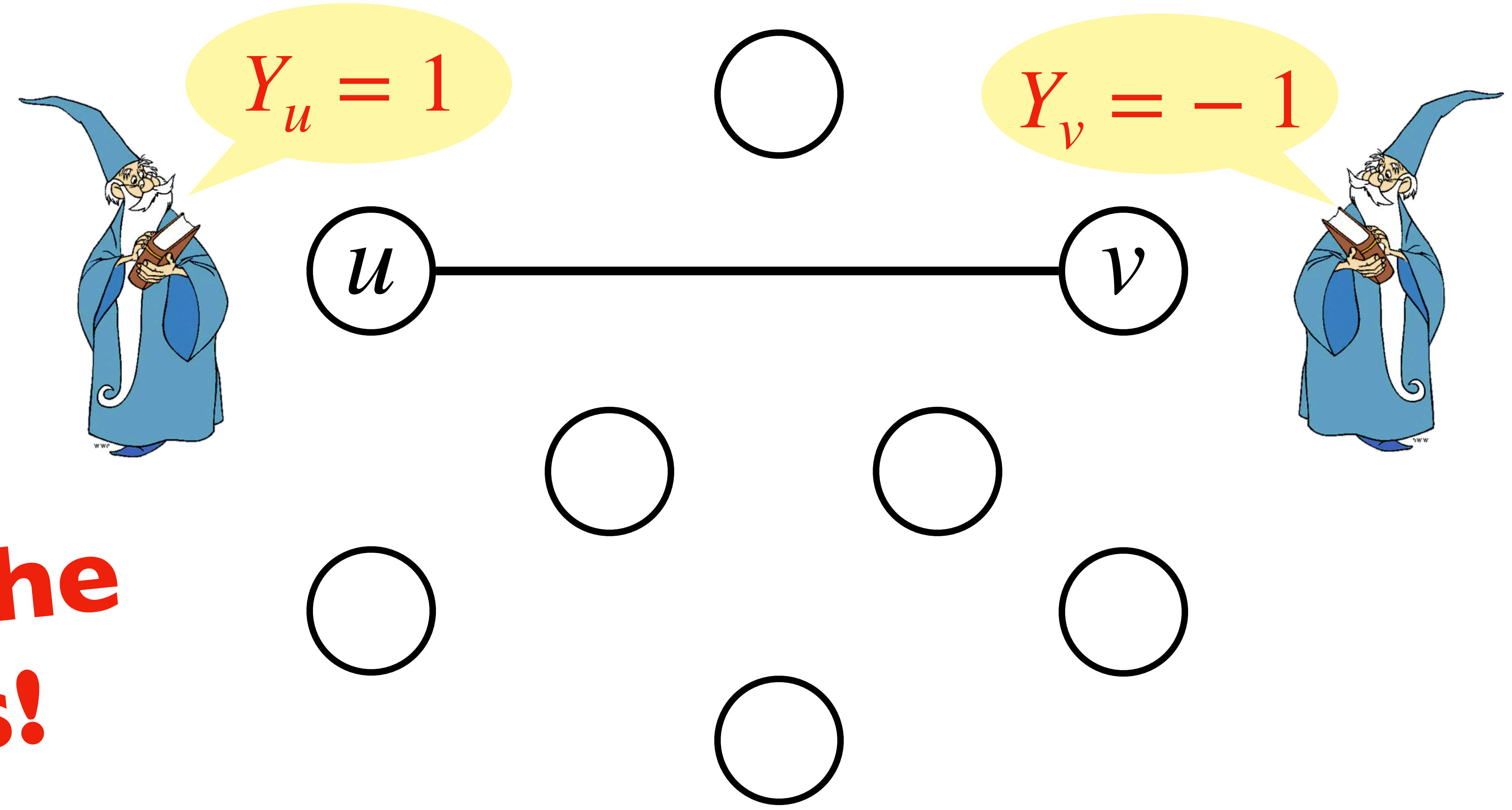
Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Algorithm for Low-Degree Graphs

- initially $X = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u \neq Y_v$ then $X = X + 1$
- **return** X

Just Follow the Predictions!

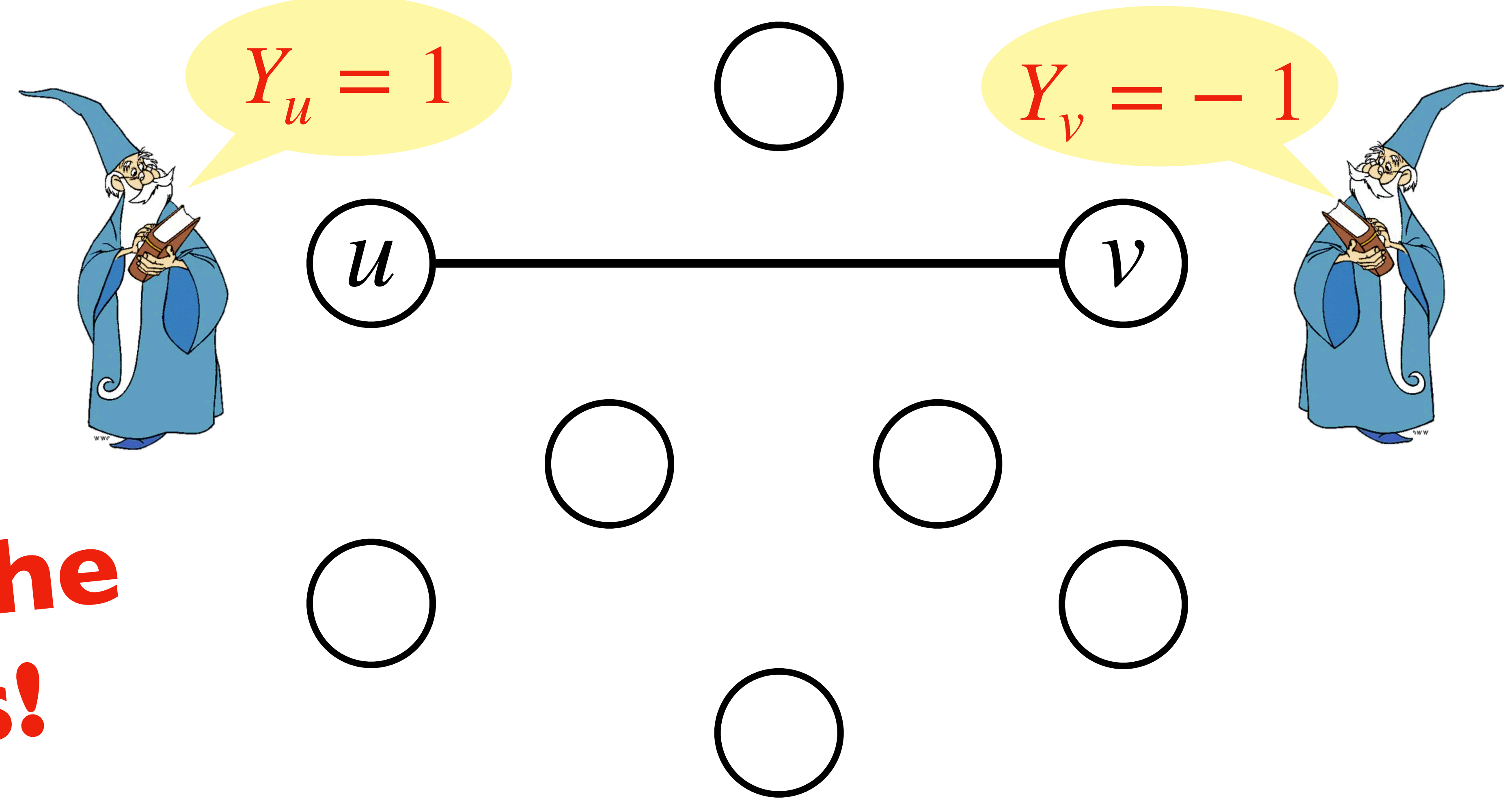


Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Algorithm for Low-Degree Graphs

- initially $X = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u \neq Y_v$ then $X = X + 1$
- **return** X



Just Follow the Predictions!

Observation:

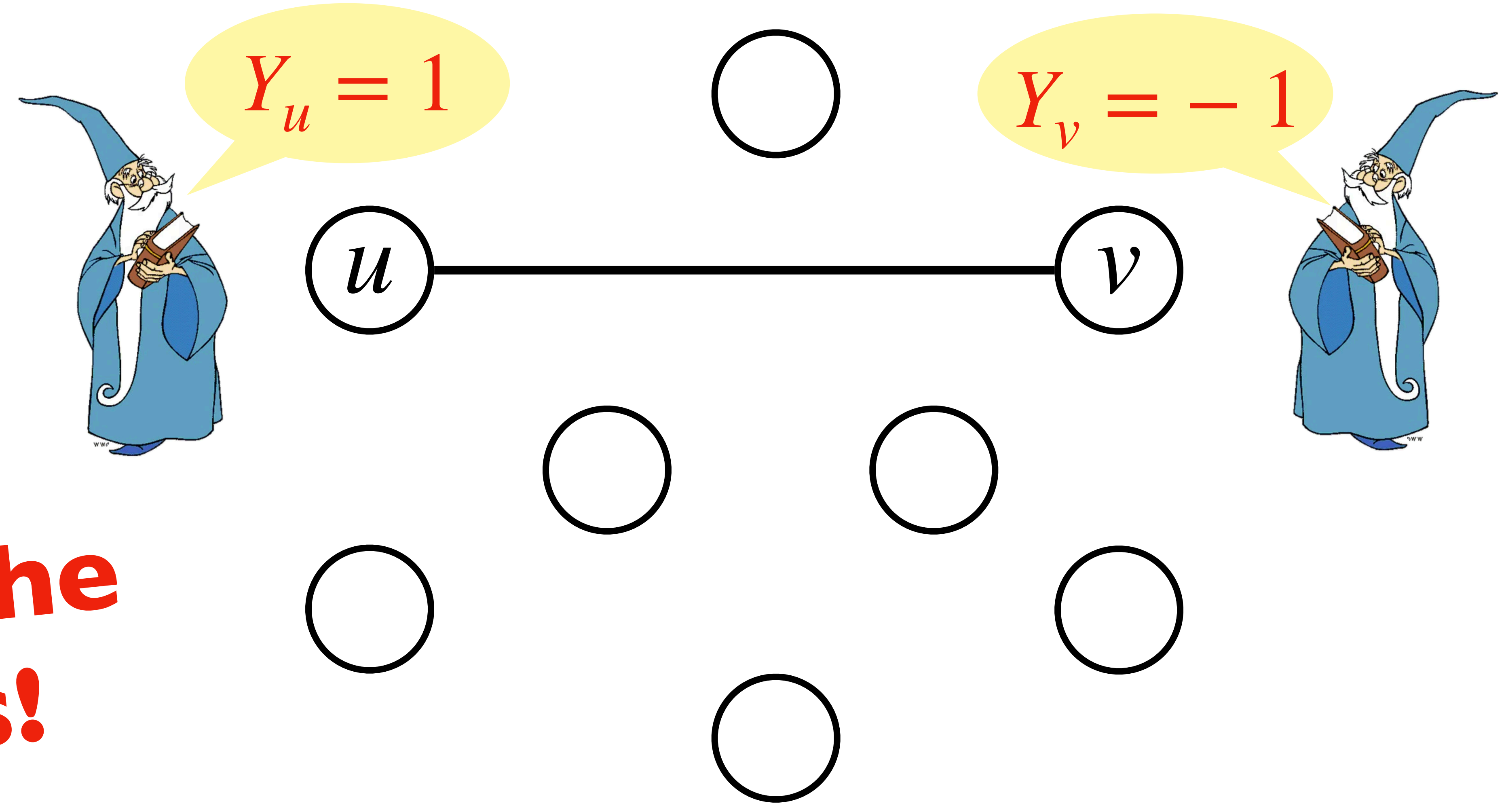
ϵ-accurate predictions $\xrightarrow{\text{w.p.} \geq 2/3}$ $(1/2 + \epsilon^2)$-approx., $O(1)$ words of space

Simple Case: Low-Degree Graphs

Low-Degree Graphs: Graphs with maximum degree $\Delta = \Theta(\epsilon^2 m)$

Algorithm for Low-Degree Graphs

- initially $X = 0$
- **for each** edge (u, v) in the stream **do**:
  - $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  - if $Y_u \neq Y_v$ then $X = X + 1$
- **return** X



Just Follow the Predictions!

Observation:

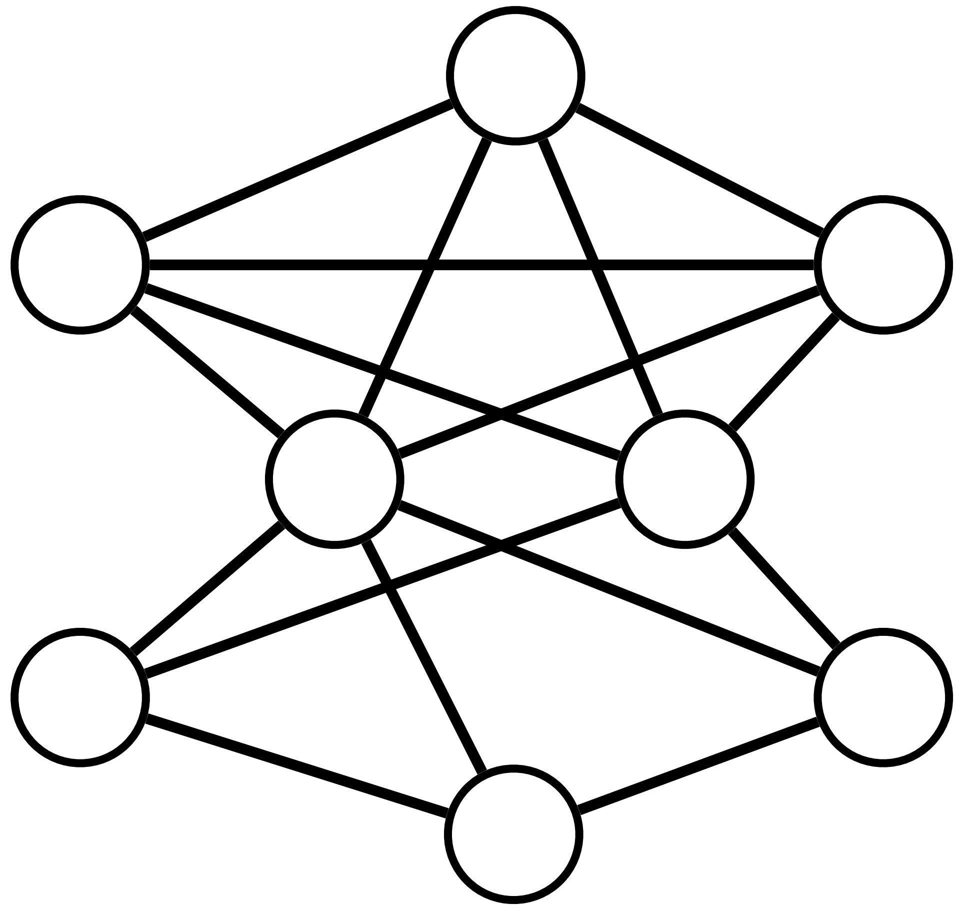
ϵ-accurate predictions $\xrightarrow{\text{w.p.} \geq 2/3}$ $(1/2 + \epsilon^2)$-approx., $O(1)$ words of space

What to do in general graphs?

High-Level Overview of Our Algorithm

High-Level Overview of Our Algorithm

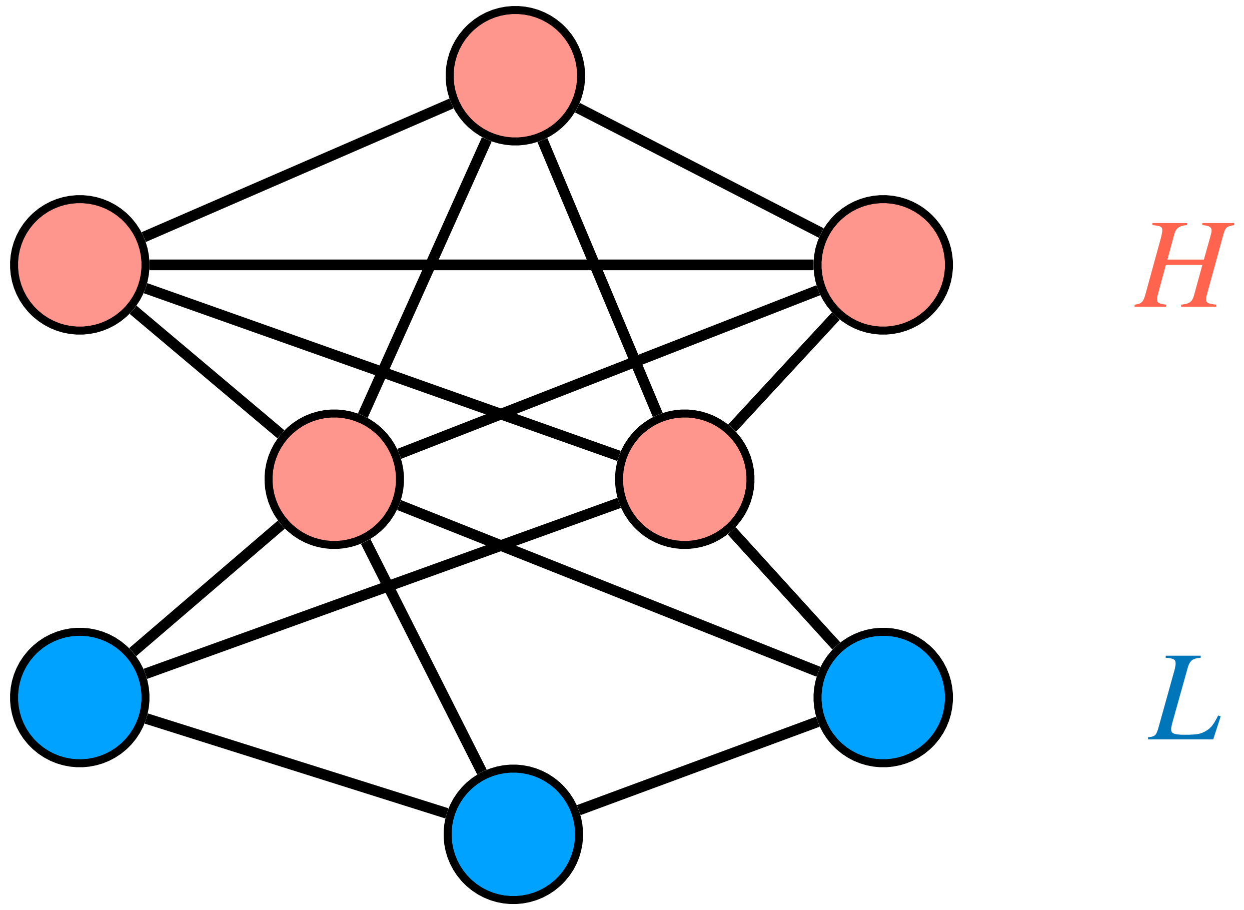
Offline Algorithm for General Graphs



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

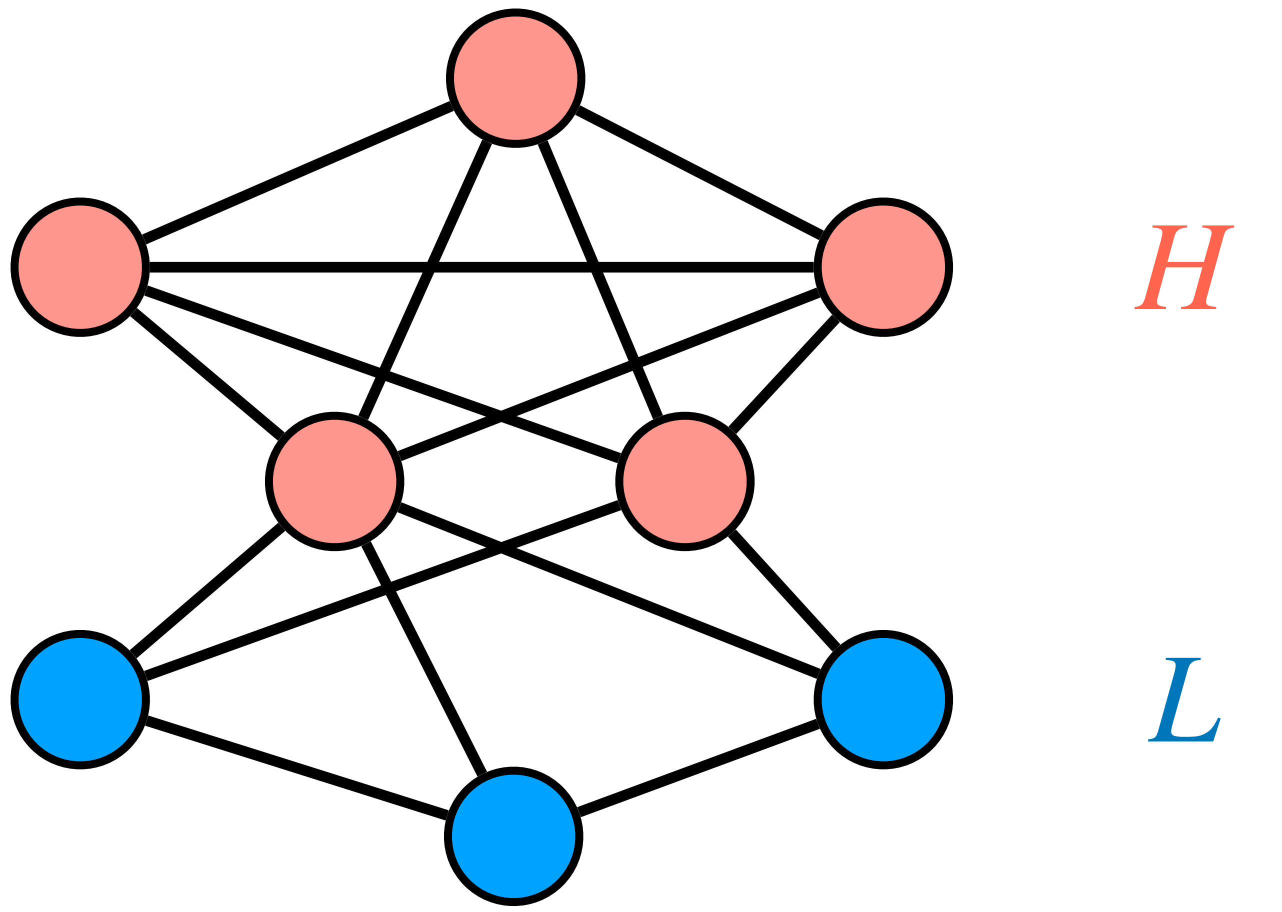
- Divide V into H (high-degree) and L (low-degree)



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)

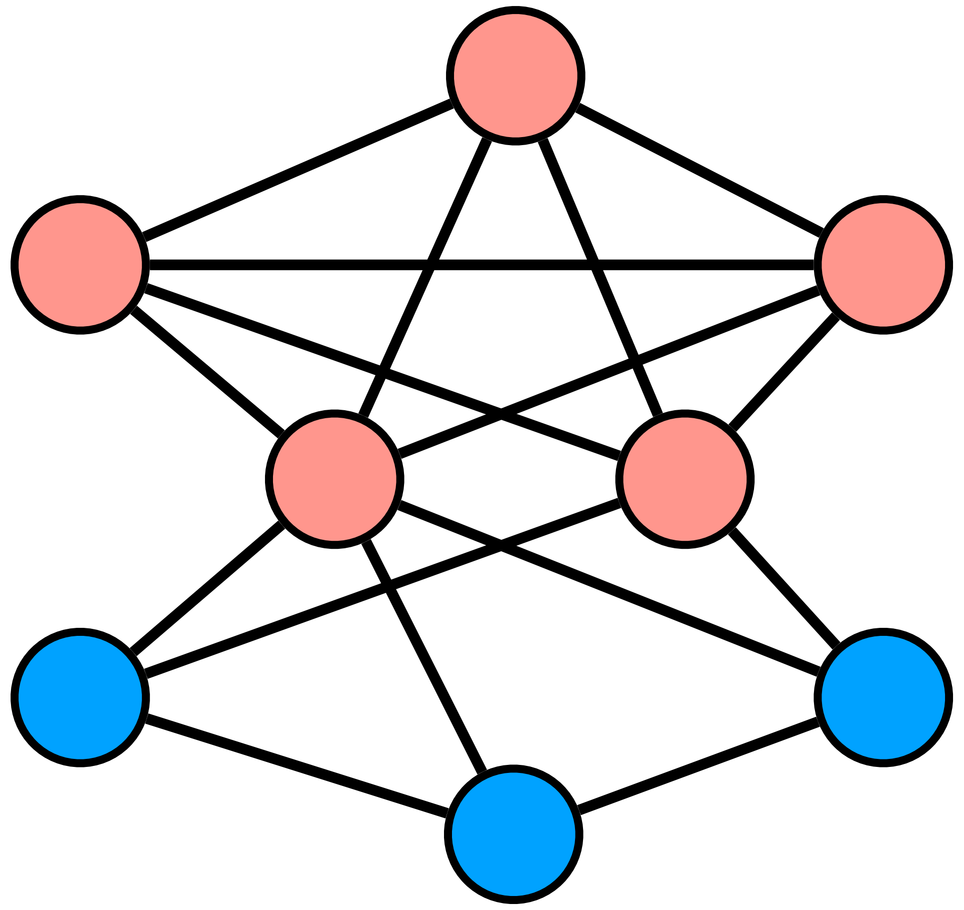


- Degree threshold is $\Theta(\epsilon^2 m)$, so
  - $|H| \leq 1/\epsilon^2$, and $|E(H)| < |H|^2 \leq 1/\epsilon^4$

High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:

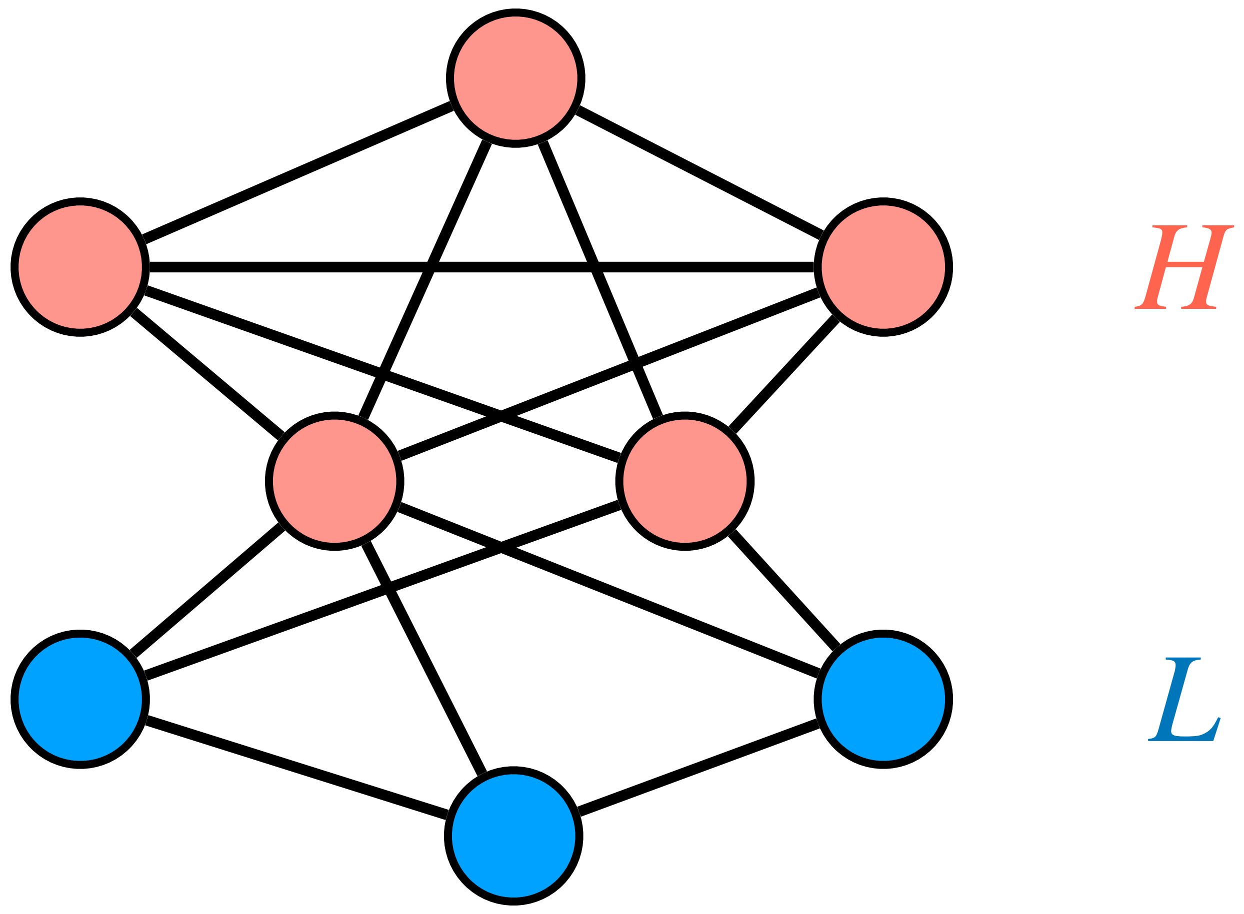


- Degree threshold is $\Theta(\epsilon^2 m)$, so
  - $|H| \leq 1/\epsilon^2$, and $|E(H)| < |H|^2 \leq 1/\epsilon^4$

High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

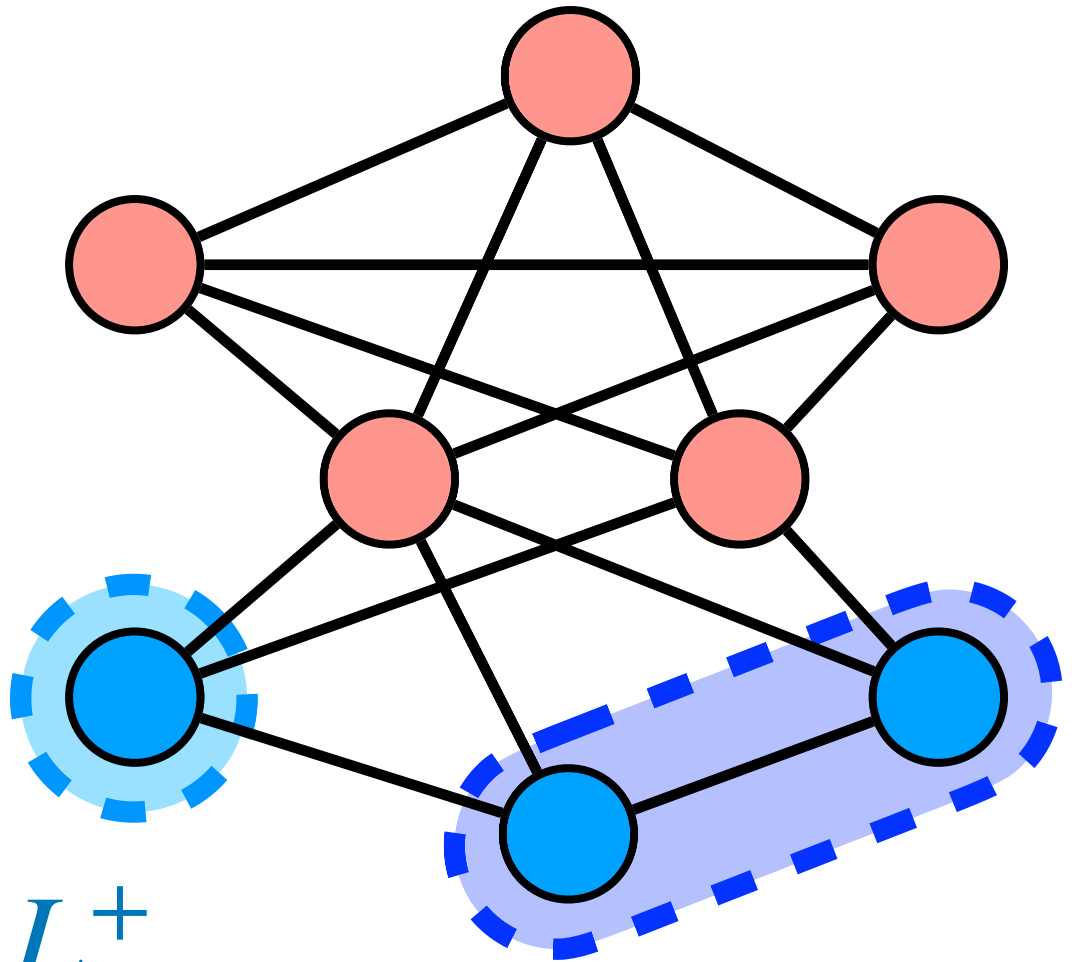
- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

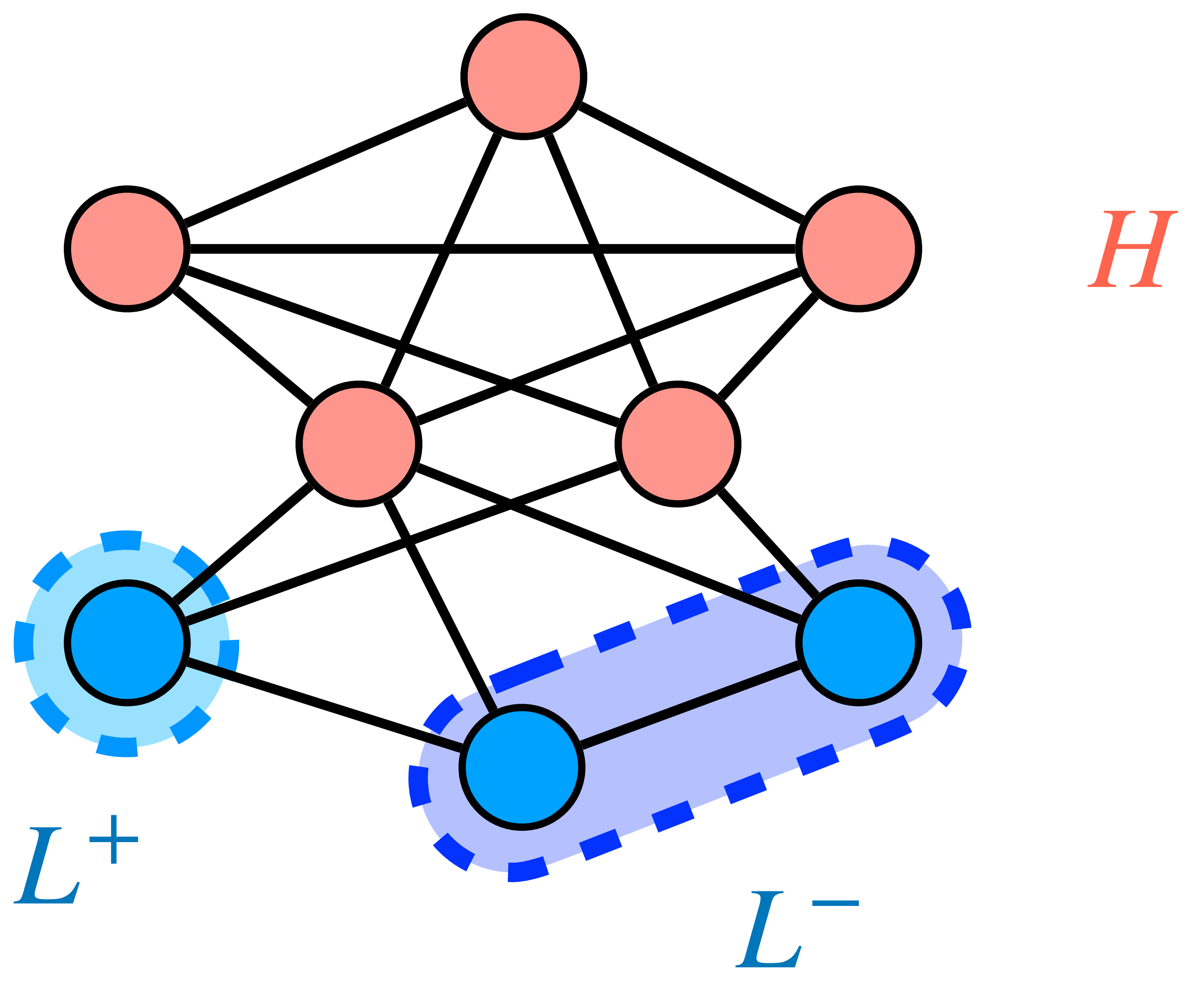
- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)



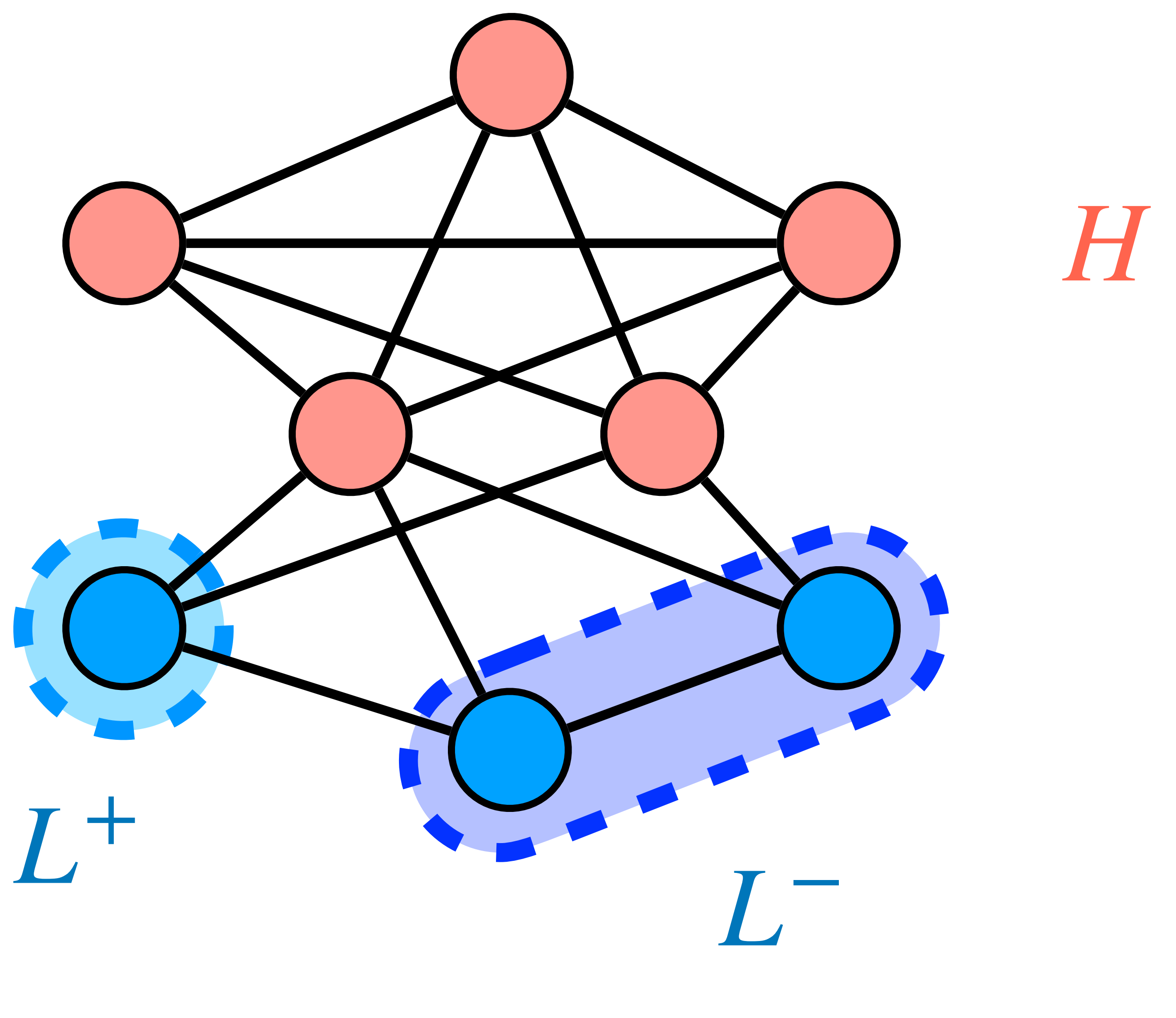
Observation on Low-Degree Graphs:

ϵ-accurate predictions $\xrightarrow{\text{w.p.} \geq 2/3}$ $(1/2 + \epsilon^2)$-approx., $O(1)$ words of space

High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

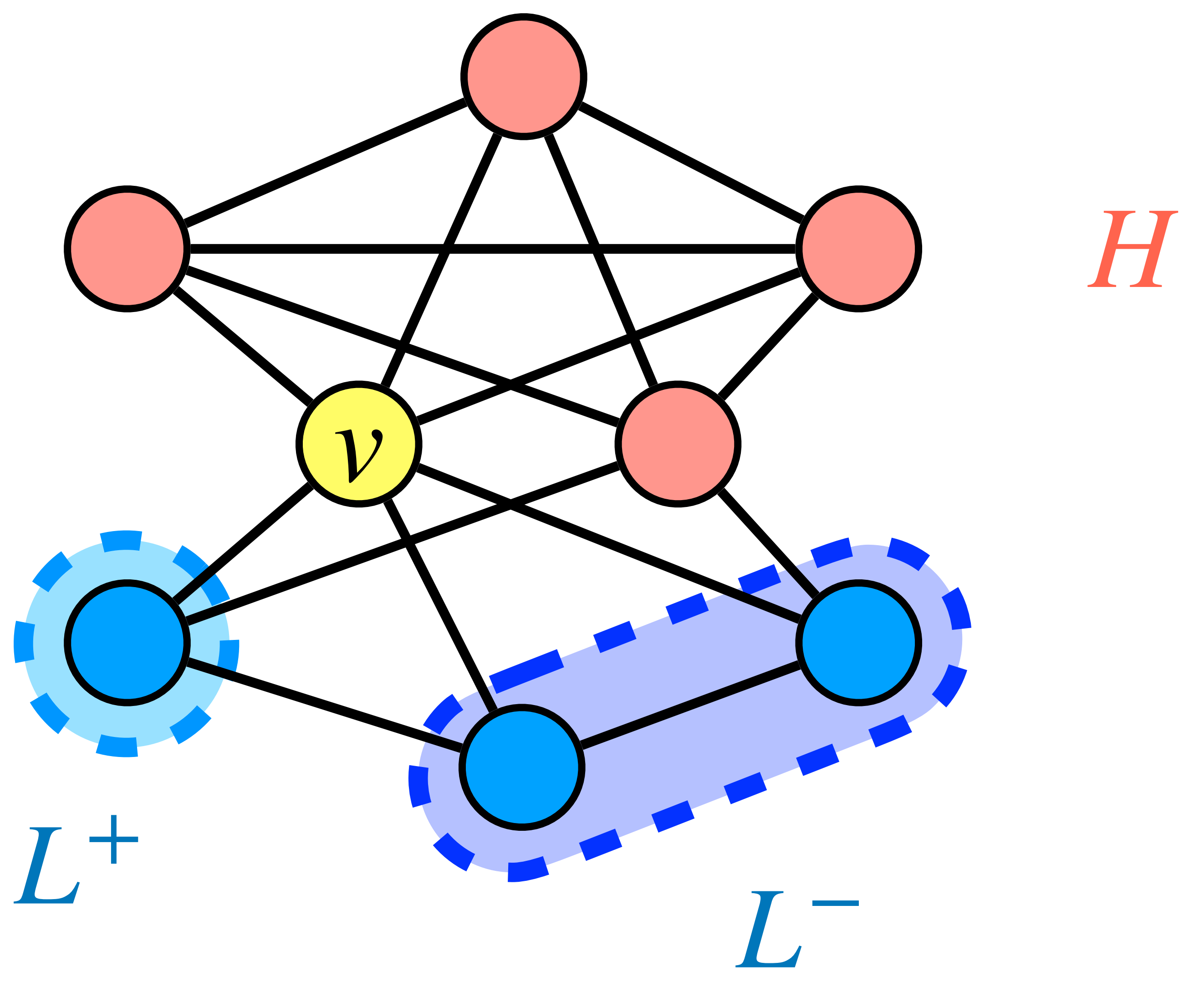
- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

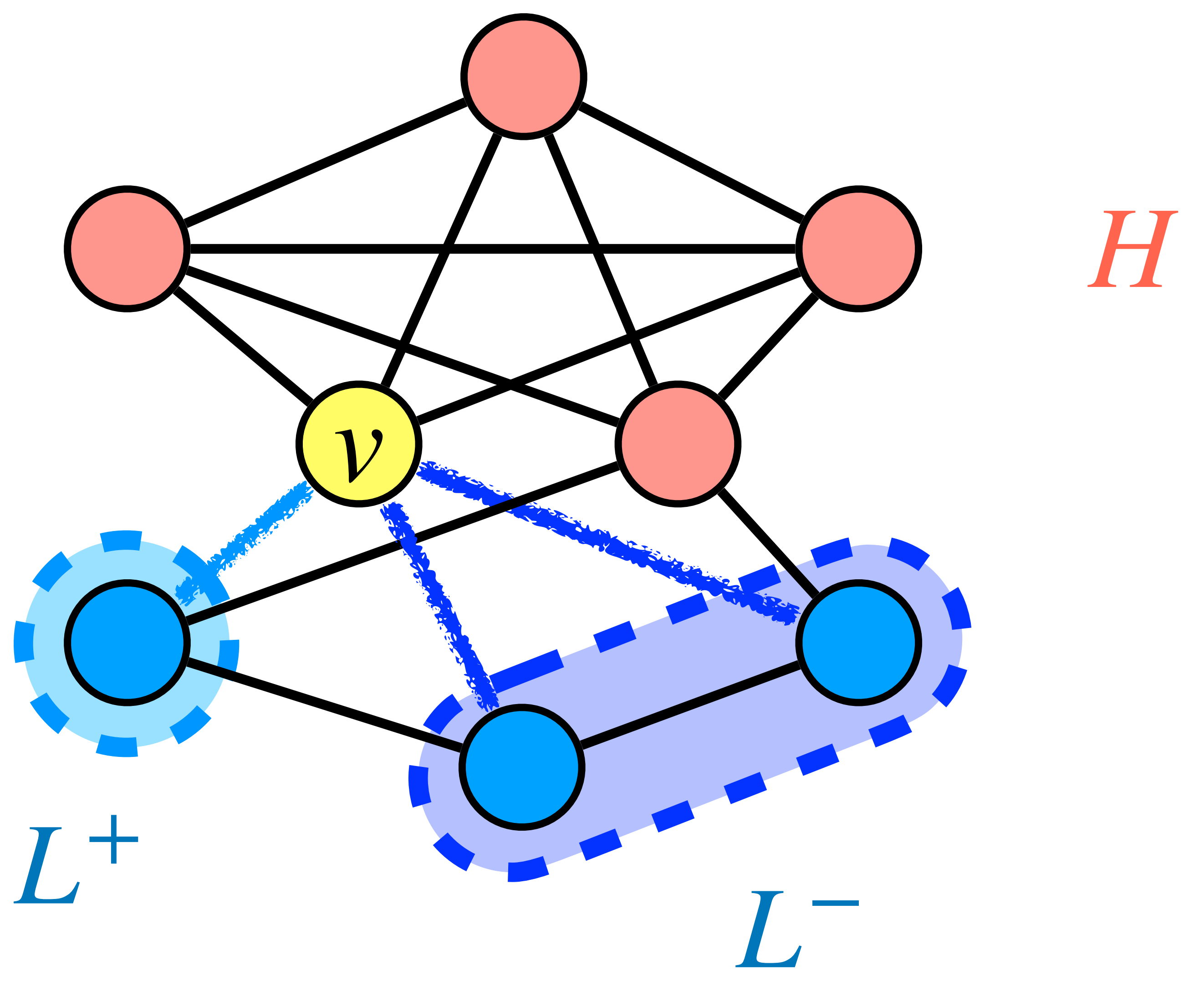
- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

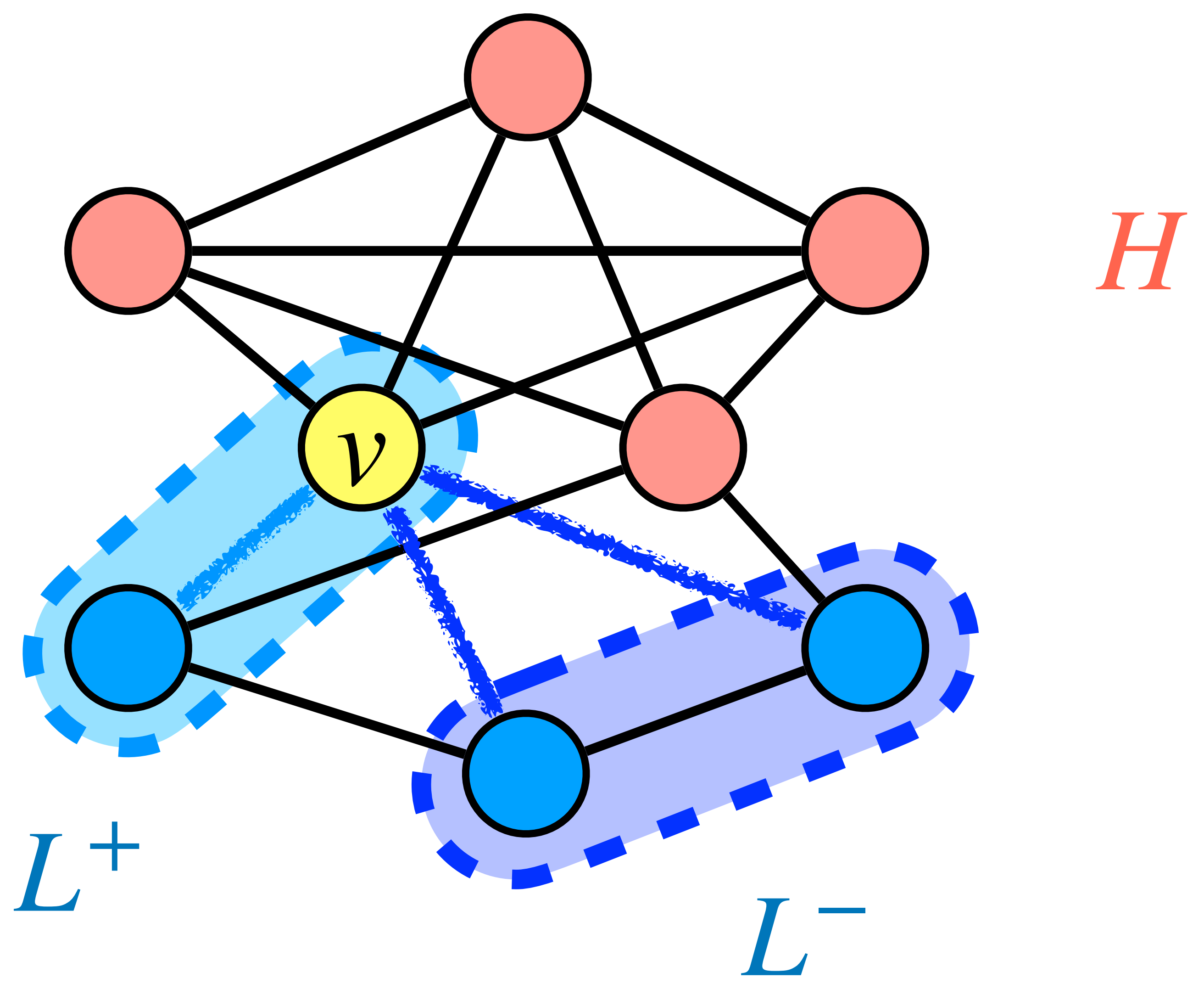
- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

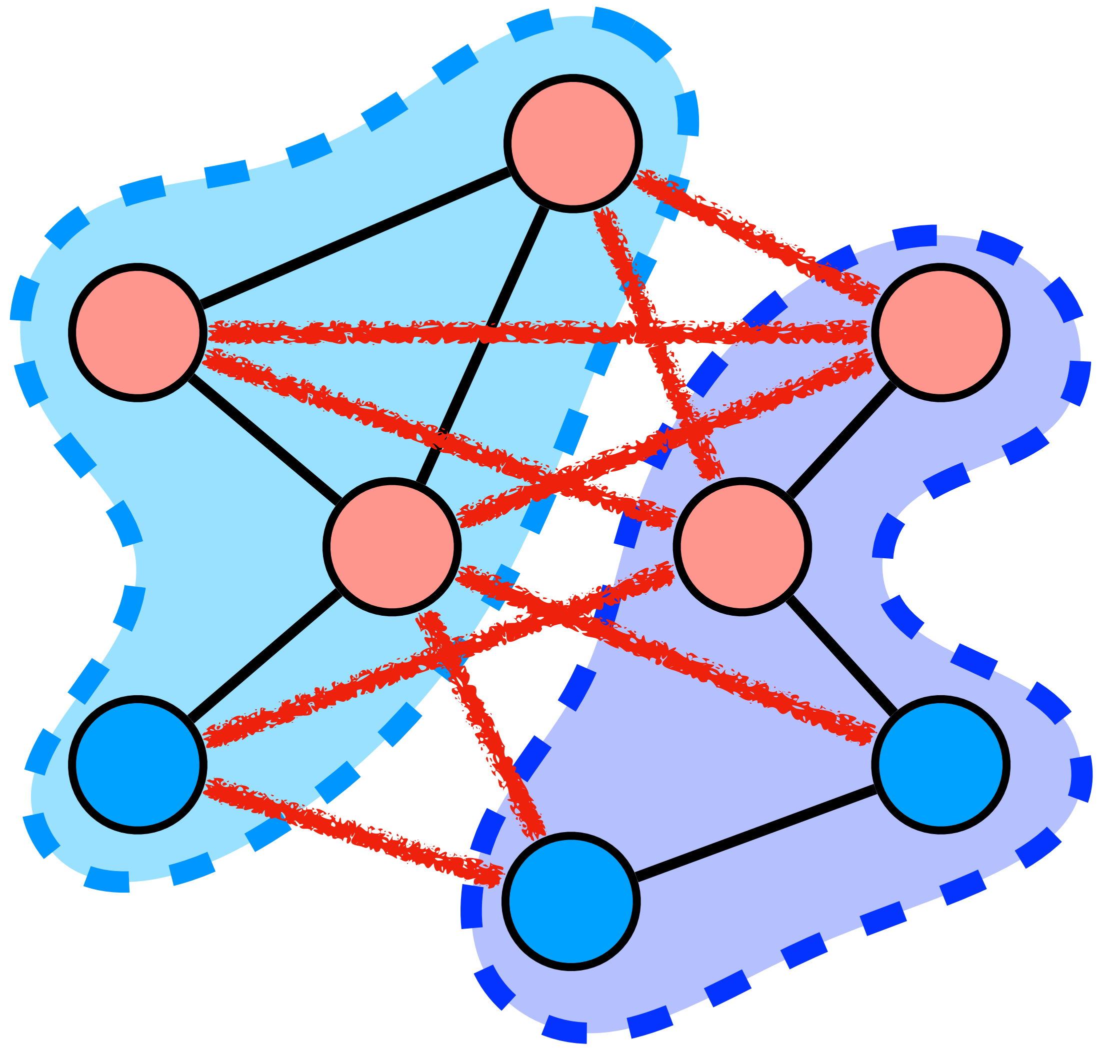
- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

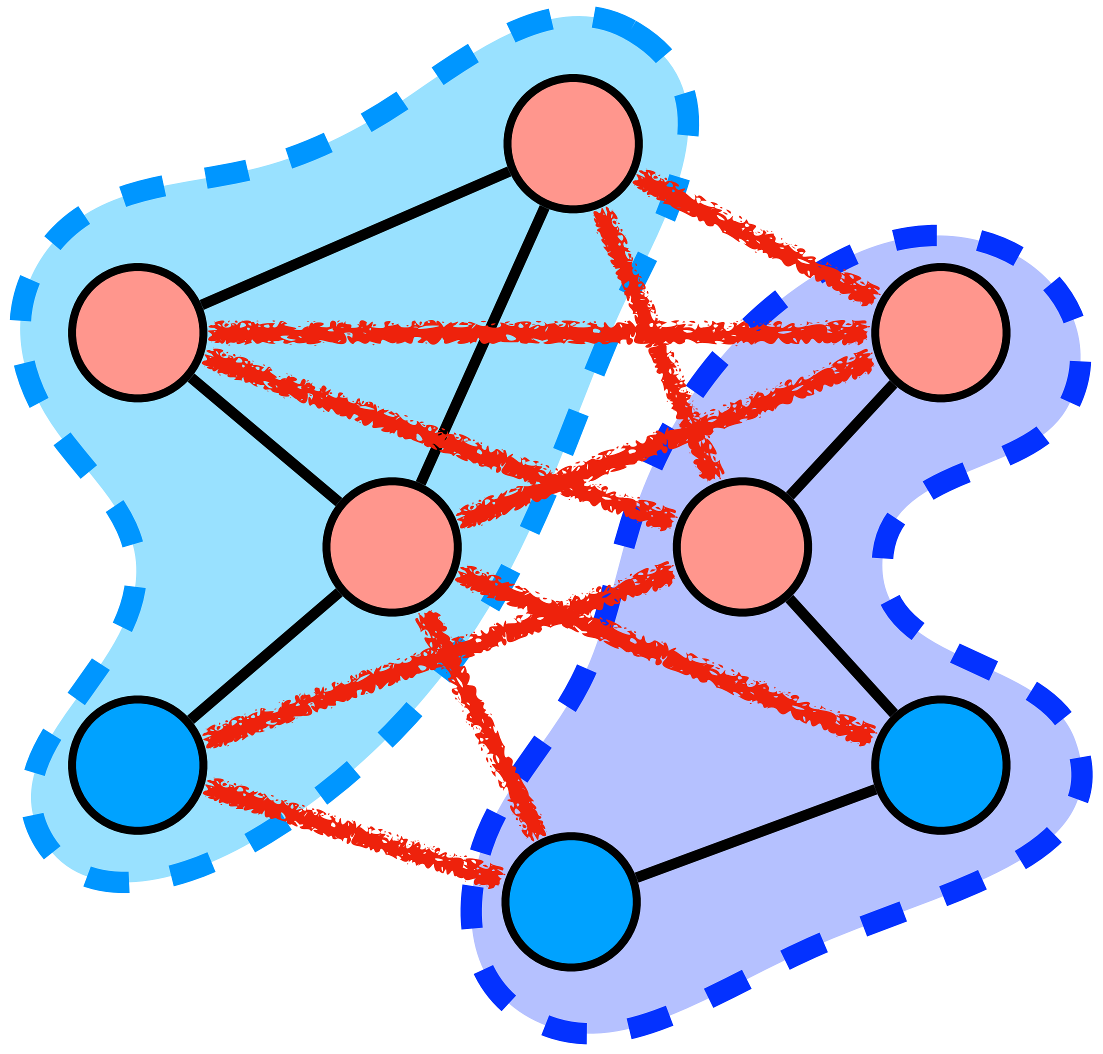
- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach

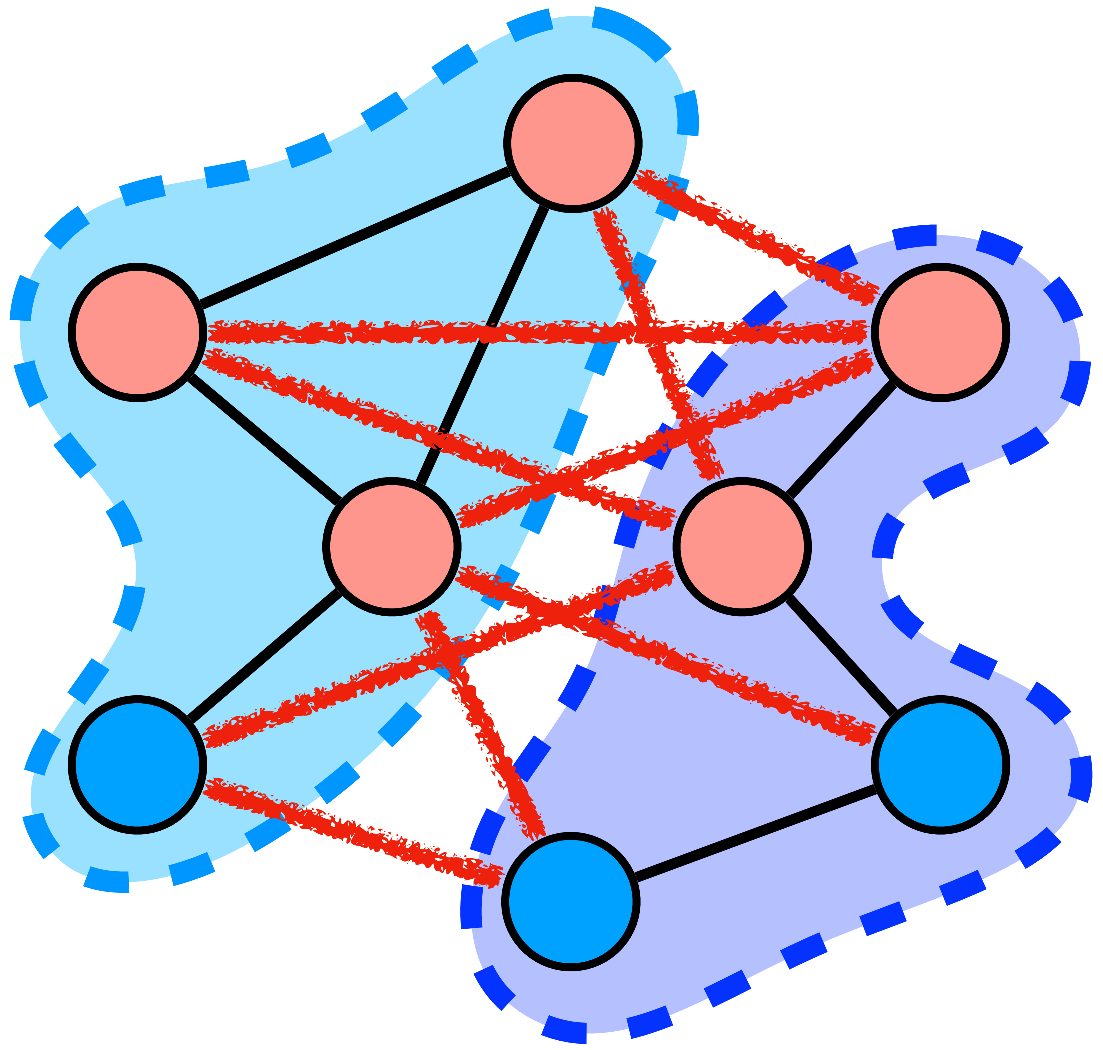


- $$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{\, |E(v, L^+)|, |E(v, L^-)| \,\}$$

High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



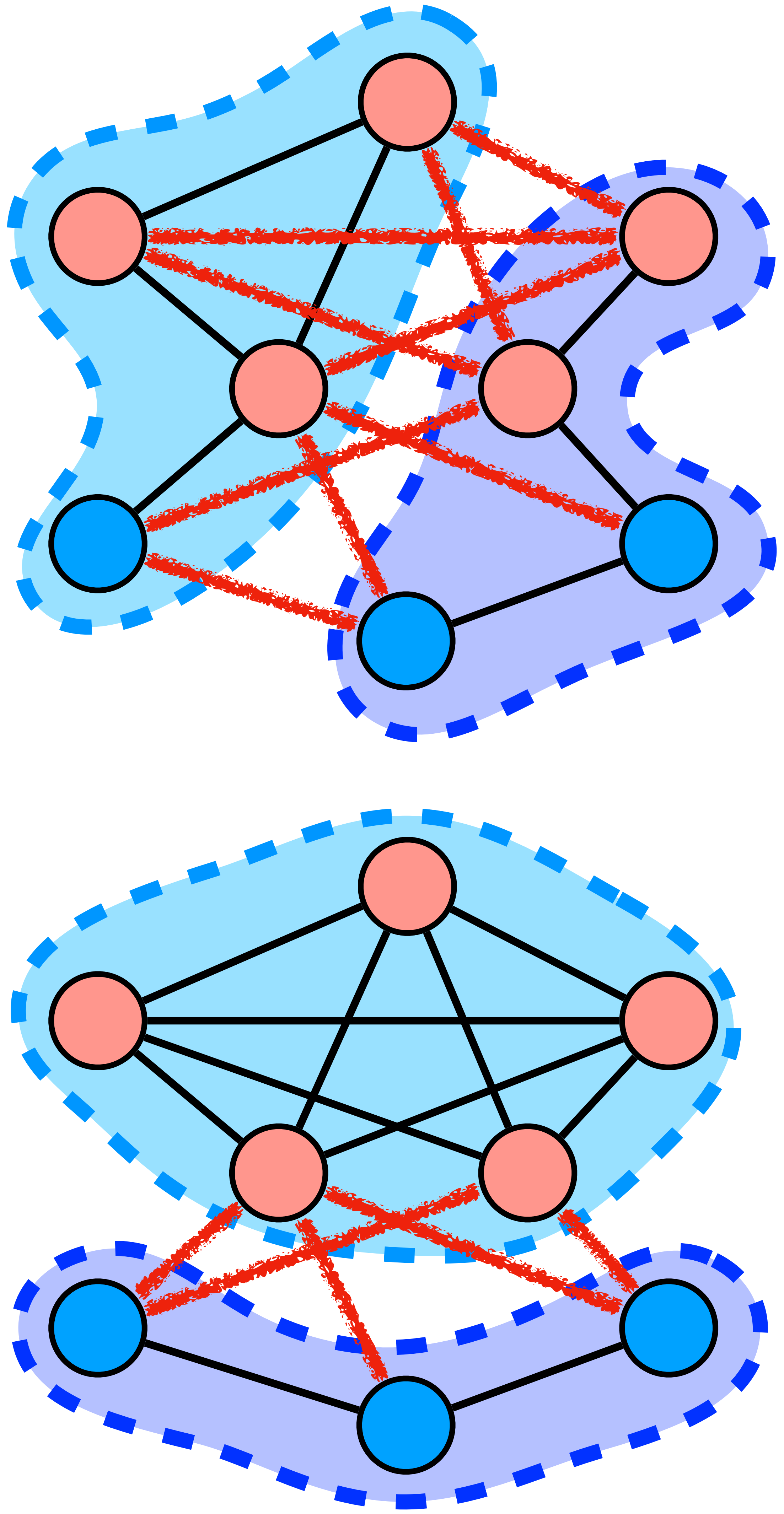
- $$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{ |E(v, L^+)|, |E(v, L^-)| \}$$

High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

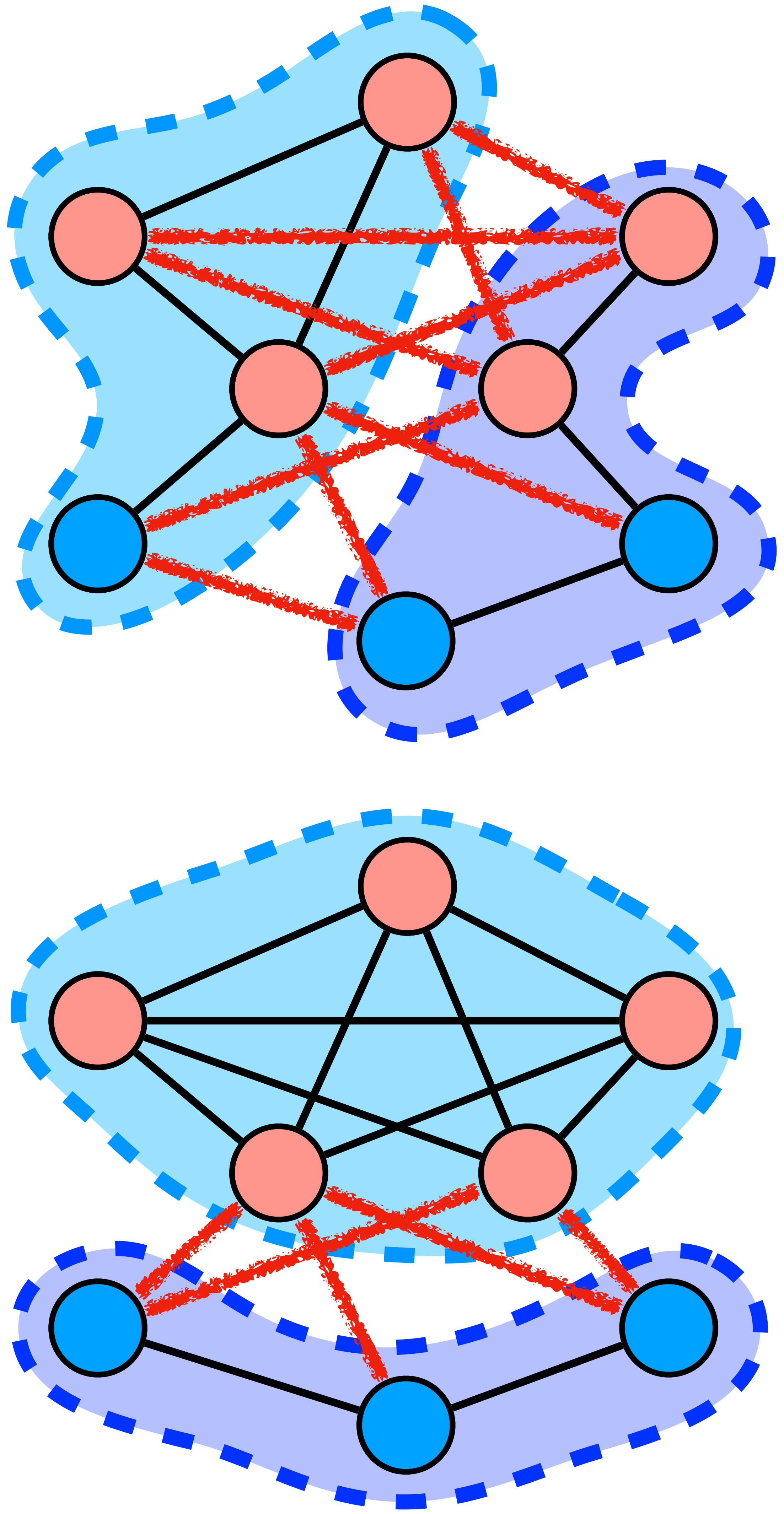
- $$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$$



High-Level Overview of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



- $Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$
- $Z_2 = |E(H, L)|$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot$ OPT
- **If** $Z_2 = |E(H, L)| \geq (1/2 + \epsilon^2) \cdot$ OPT, **DONE**

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot \text{OPT}$
- **If** $Z_2 = |E(H, L)| \geq (1/2 + \epsilon^2) \cdot \text{OPT}$, **DONE**
- **Otherwise**, $|E(H, L)| < (1/2 + \epsilon^2) \cdot \text{OPT}$, it suffices to prove $Z_1 \geq (1/2 + \Omega(\epsilon^2)) \cdot \text{OPT}$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot \text{OPT}$
- **If** $Z_2 = |E(H, L)| \geq (1/2 + \epsilon^2) \cdot \text{OPT}$, **DONE**
- **Otherwise**, $|E(H, L)| < (1/2 + \epsilon^2) \cdot \text{OPT}$, it suffices to prove $Z_1 \geq (1/2 + \Omega(\epsilon^2)) \cdot \text{OPT}$

$$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot \text{OPT}$
- **If** $Z_2 = |E(H, L)| \geq (1/2 + \epsilon^2) \cdot \text{OPT}$, **DONE**
- **Otherwise**, $|E(H, L)| < (1/2 + \epsilon^2) \cdot \text{OPT}$, it suffices to prove $Z_1 \geq (1/2 + \Omega(\epsilon^2)) \cdot \text{OPT}$

$$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$$

$$\geq \left(\frac{1}{2} + \epsilon^2\right) \cdot \text{OPT}_L + \frac{|E(H, L)|}{2}$$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

Observation: $|E(L^+, L^-)| \geq (1/2 + \epsilon^2) \cdot \mathrm{OPT}_L$

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$
- **If** $Z_2 = |E(H, L)| \geq (1/2 + \epsilon^2) \cdot \mathrm{OPT}$, **DONE**
- **Otherwise**, $|E(H, L)| < (1/2 + \epsilon^2) \cdot \mathrm{OPT}$, it suffices to prove $Z_1 \geq (1/2 + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$

$$Z_1 = \underline{|E(L^+, L^-)|} + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$$

$$\geq \underline{\left(\frac{1}{2} + \epsilon^2\right) \cdot \mathrm{OPT}_L} + \frac{|E(H, L)|}{2}$$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot \mathsf{OPT}$
- **If** $Z_2 = |E(H, L)| \geq (1/2 + \epsilon^2) \cdot \mathsf{OPT}$, **DONE**
- **Otherwise**, $|E(H, L)| < (1/2 + \epsilon^2) \cdot \mathsf{OPT}$, it suffices to prove $Z_1 \geq (1/2 + \Omega(\epsilon^2)) \cdot \mathsf{OPT}$

$$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$$

$$\geq \left(\frac{1}{2} + \epsilon^2\right) \cdot \mathsf{OPT}_L + \frac{|E(H, L)|}{2}$$

$$\geq \left(\frac{1}{2} + \epsilon^2\right) \left(1 - \alpha - \epsilon^4\right) \cdot \mathsf{OPT} + \frac{\alpha \cdot \mathsf{OPT}}{2}$$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

Lemma: Suppose $|E(H,L)| = \alpha \cdot \mathrm{OPT}$ where $\alpha < 1/2 + \epsilon^2$, then $\mathrm{OPT}_L > (1 - \alpha - \epsilon^4) \cdot \mathrm{OPT}$

- $\mathrm{OPT} \le \mathrm{OPT}_L + \mathrm{OPT}_H + |E(H,L)|$
- $\mathrm{OPT}_H \le |E(H)| \le 1/\epsilon^4$

- **Goal:** To prove $\max\{Z_1, Z_2\} \ge (1/2 + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$
- **If** $Z_2 = |E(H,L)| \ge (1/2 + \epsilon^2) \cdot \mathrm{OPT}$, **DONE**
- **Otherwise**, $|E(H,L)| < (1/2 + \epsilon^2) \cdot \mathrm{OPT}$, it suffices to prove $Z_1 \ge (1/2 + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$

$$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$$

$$\ge \left(\frac{1}{2} + \epsilon^2\right) \cdot \underline{\mathrm{OPT}_L} + \frac{|E(H,L)|}{2}$$

$$\ge \left(\frac{1}{2} + \epsilon^2\right) \underline{(1 - \alpha - \epsilon^4) \cdot \mathrm{OPT}} + \frac{\alpha \cdot \mathrm{OPT}}{2}$$

Approximation Analysis of Our Algorithm

Offline Algorithm for General Graphs

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

- **Goal:** To prove $\max\{Z_1, Z_2\} \geq (1/2 + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$
- **If** $Z_2 = |E(H, L)| \geq (1/2 + \epsilon^2) \cdot \mathrm{OPT}$, **DONE**
- **Otherwise**, $|E(H, L)| < (1/2 + \epsilon^2) \cdot \mathrm{OPT}$, it suffices to prove $Z_1 \geq (1/2 + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$

$$Z_1 = |E(L^+, L^-)| + \sum_{v \in H} \max\{|E(v, L^+)|, |E(v, L^-)|\}$$

$$\geq \left(\frac{1}{2} + \epsilon^2\right) \cdot \mathrm{OPT}_L + \frac{|E(H, L)|}{2}$$

$$\geq \left(\frac{1}{2} + \epsilon^2\right) \left(1 - \alpha - \epsilon^4\right) \cdot \mathrm{OPT} + \frac{\alpha \cdot \mathrm{OPT}}{2}$$

$$\geq \left(\frac{1}{2} + \Omega(\epsilon^2)\right) \cdot \mathrm{OPT}$$

Implementation of Our Algorithm in Streaming

Implementation of Our Algorithm in Streaming

- Random Order Streams (insertion-only)
- Arbitrary Order Streams (insertion-only)
- Dynamic Streams

Our Algorithm in Random Order Streams

Our Algorithm in Random Order Streams

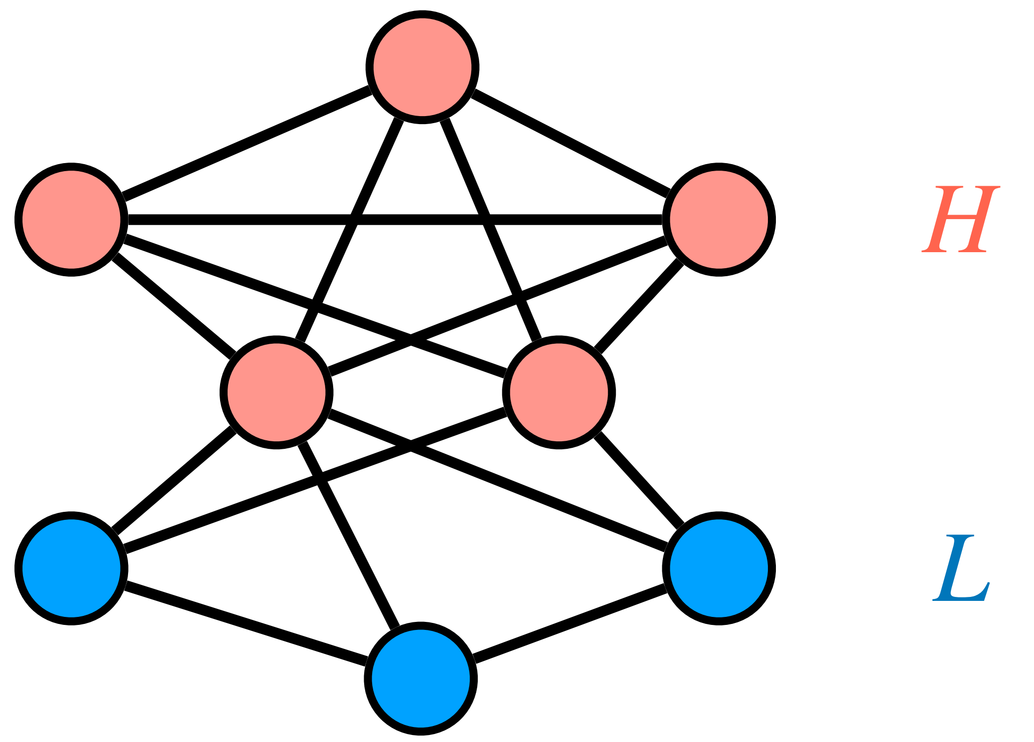
Random Order Stream: Edges of the input graph arrive in a uniformly random order

Our Algorithm in Random Order Streams

Random Order Stream: Edges of the input graph arrive in a uniformly random order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

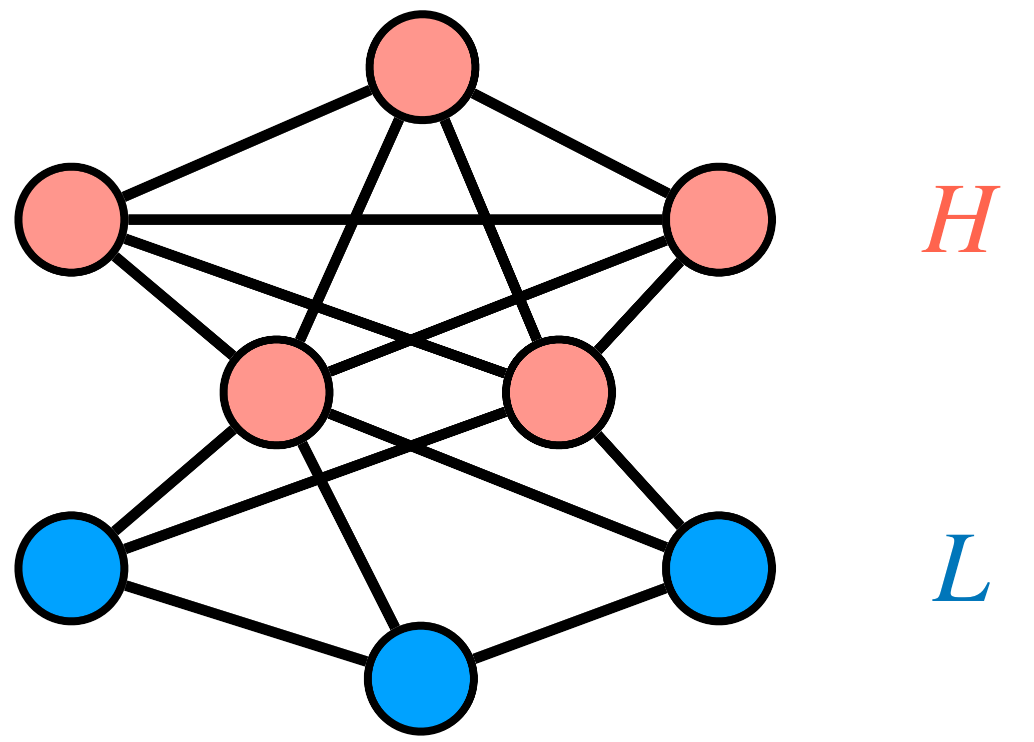


Our Algorithm in Random Order Streams

Random Order Stream: Edges of the input graph arrive in a uniformly random order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Random Order Streams

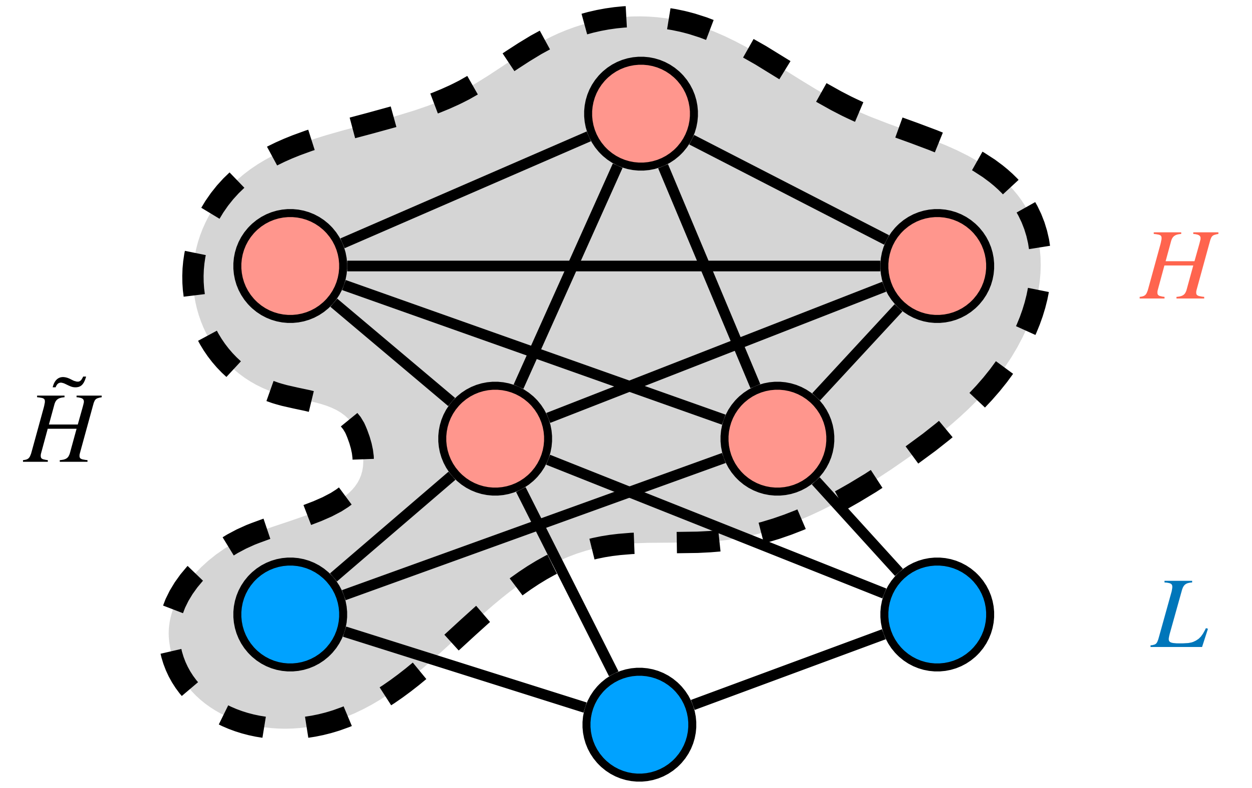
- **Streaming Phase:**
  - **Store the first $\mathrm{poly}(1/\epsilon)$ edges**, detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$
  - Compute Estimate (1) and Estimate (2) w.r.t. $\tilde{H}$ and $\tilde{L}$
  - Store the degree info of $\tilde{H}$ and $E(\tilde{H})$
- **Post-Processing Phase:**
  - Correct H, L and the estimates using degree info of $\tilde{H}$

Our Algorithm in Random Order Streams

Random Order Stream: Edges of the input graph arrive in a uniformly random order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Random Order Streams

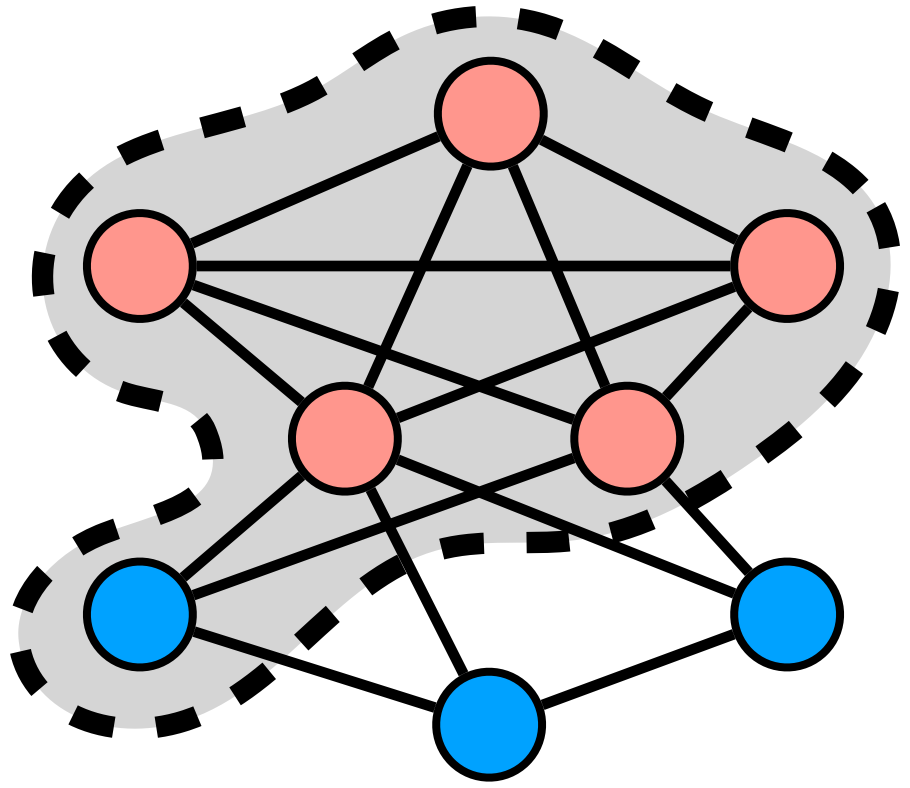
- **Streaming Phase:**
  - **Store the first** $\text{poly}(1/\epsilon)$ **edges**, detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$
  - Compute Estimate (1) and Estimate (2) w.r.t. $\tilde{H}$ and $\tilde{L}$
  - Store the degree info of $\tilde{H}$ and $E(\tilde{H})$
- **Post-Processing Phase:**
  - Correct H, L and the estimates using degree info of $\tilde{H}$

Our Algorithm in Random Order Streams

Random Order Stream: Edges of the input graph arrive in a uniformly random order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Random Order Streams

- **Streaming Phase:**
  - **Store the first** $\mathrm{poly}(1/\epsilon)$ **edges**, detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$
  - Compute Estimate (1) and Estimate (2) w.r.t. $\tilde{H}$ and $\tilde{L}$
  - Store the degree info of $\tilde{H}$ and $E(\tilde{H})$
- **Post-Processing Phase:**
  - Correct H, L and the estimates using degree info of $\tilde{H}$

- **Approx:** $1/2 + \Omega(\epsilon^2)$
- **Space:** $\mathrm{poly}(1/\epsilon)$ words

Our Algorithm in Arbitrary Order Streams

Our Algorithm in Arbitrary Order Streams

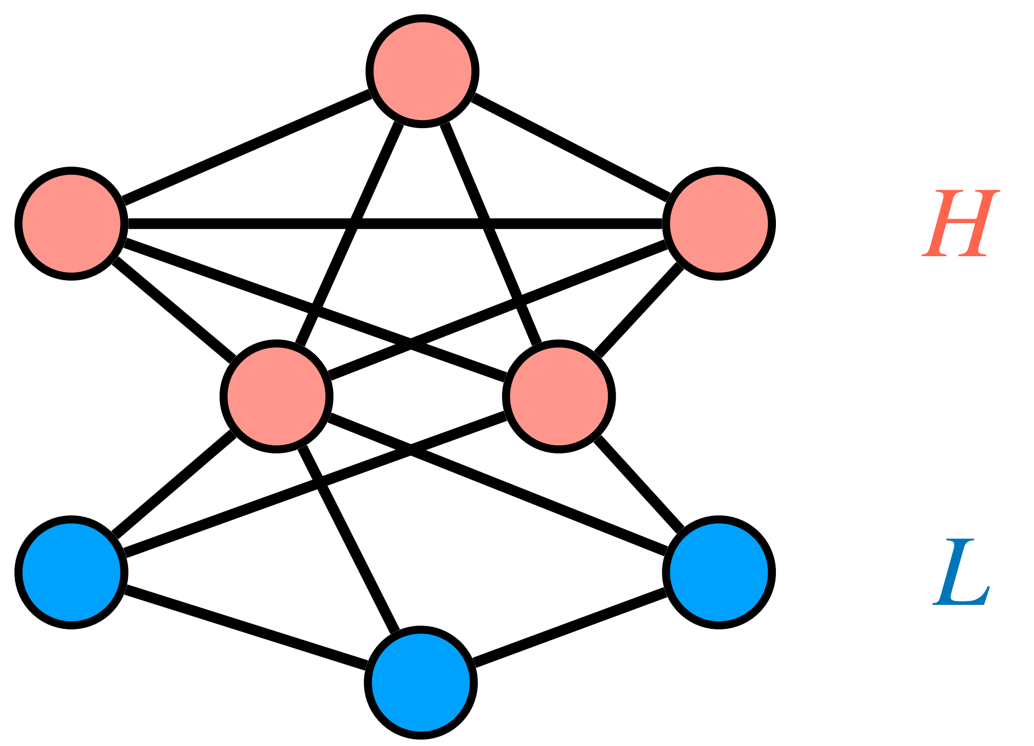
Arbitrary Order Stream: Edges of the input graph arrive in an arbitrary/adversarial order

Our Algorithm in Arbitrary Order Streams

Arbitrary Order Stream: Edges of the input graph arrive in an arbitrary/adversarial order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Arbitrary Order Streams

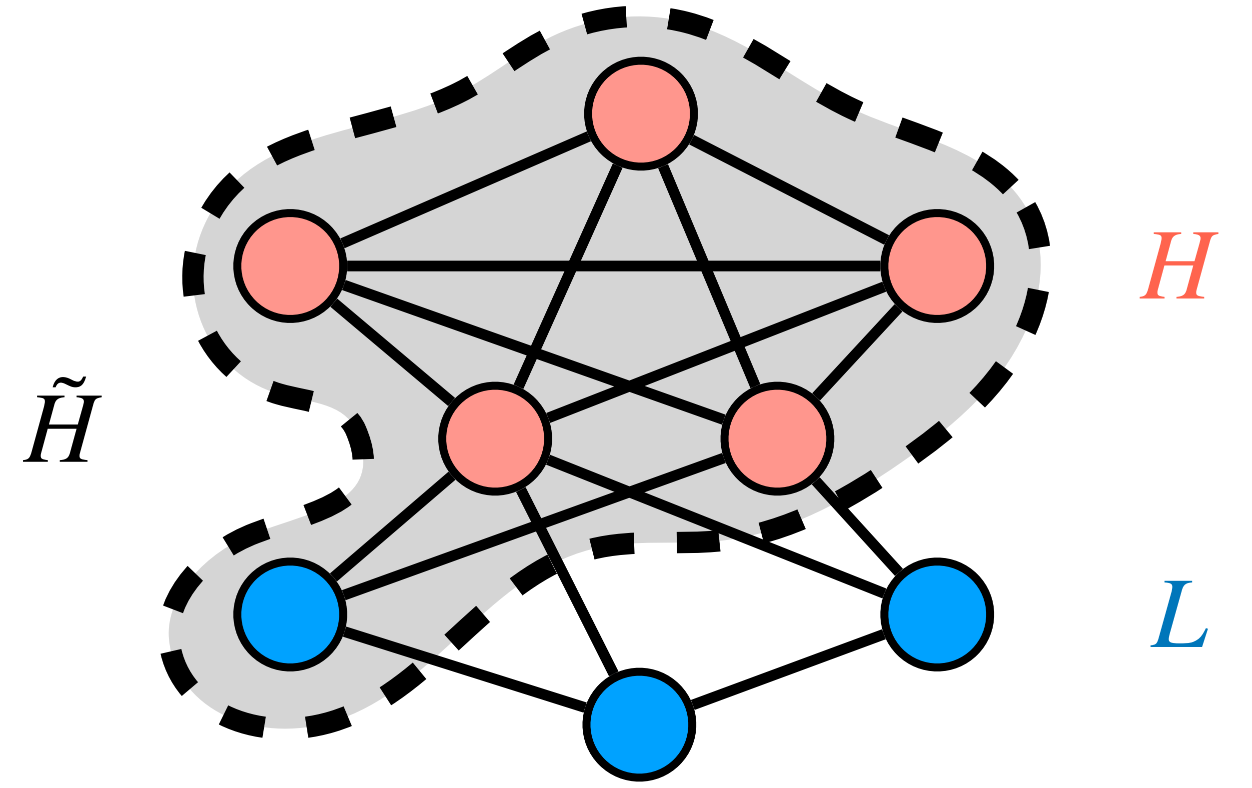
- **Streaming Phase:**
  - Perform **reservoir sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch** (or **Misra-Gries**)
- **Post-Processing Phase:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute Estimate (1) and Estimate (2) approximately

Our Algorithm in Arbitrary Order Streams

Arbitrary Order Stream: Edges of the input graph arrive in an arbitrary/adversarial order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Arbitrary Order Streams

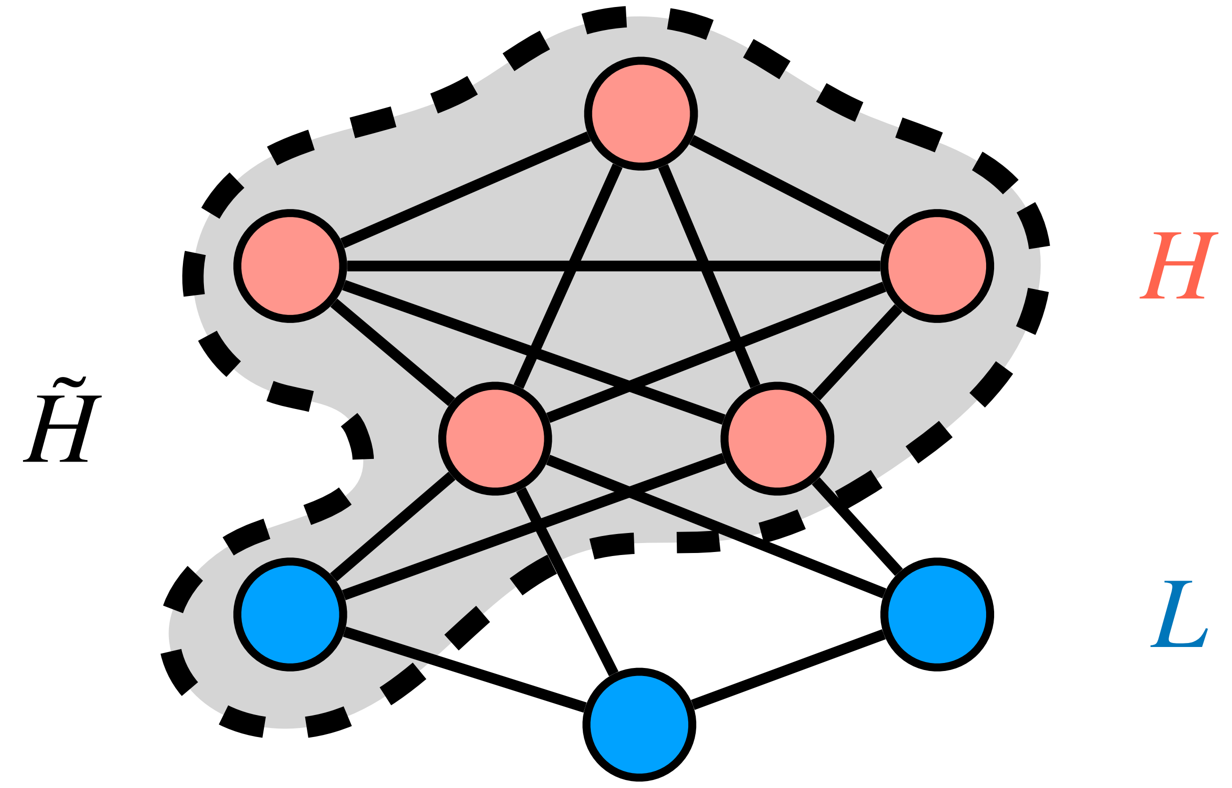
- **Streaming Phase:**
  - Perform **reservoir sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch** (or **Misra-Gries**)
- **Post-Processing Phase:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute Estimate (1) and Estimate (2) approximately

Our Algorithm in Arbitrary Order Streams

Arbitrary Order Stream: Edges of the input graph arrive in an arbitrary/adversarial order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Arbitrary Order Streams

- **Streaming Phase:**
  - Perform **reservoir sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch** (or **Misra-Gries**)
- **Post-Processing Phase:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute Estimate (1) and Estimate (2) approximately

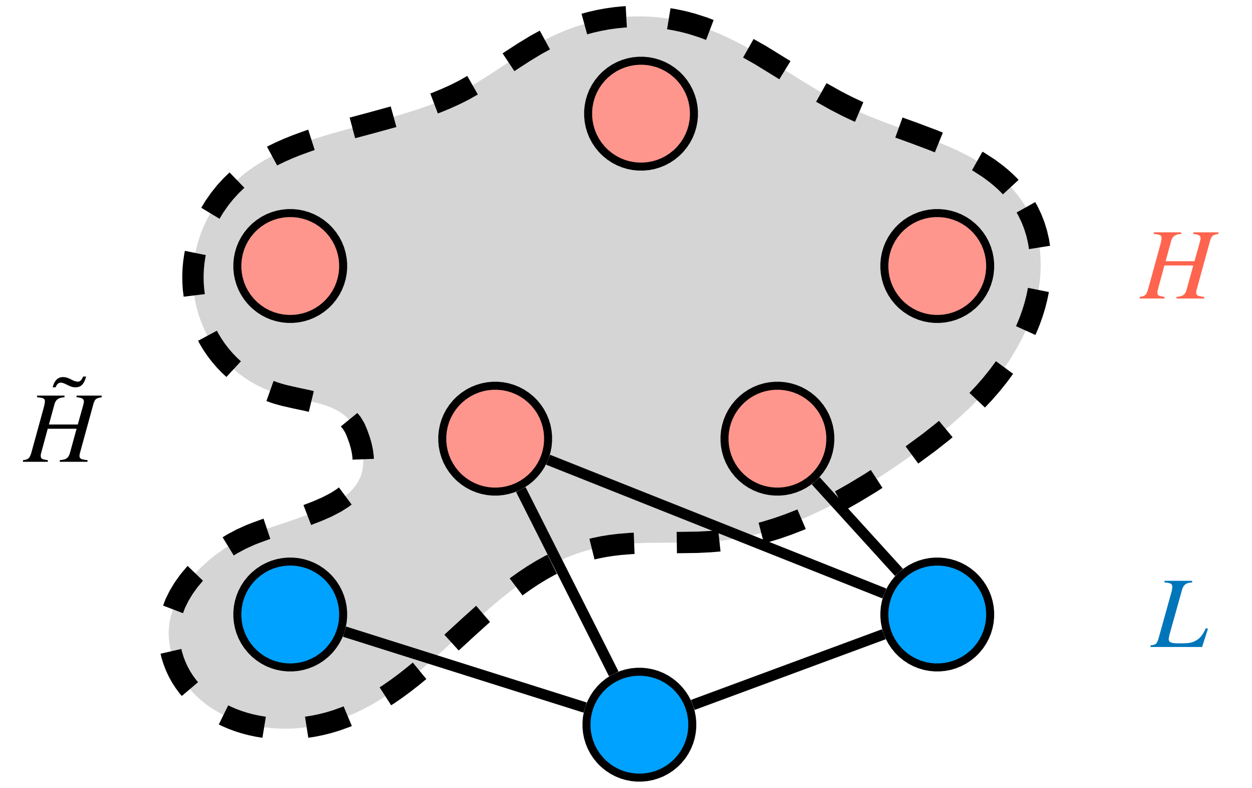
- Since $|\tilde{H}|, |E(\tilde{H})| = \mathrm{poly}(1/\epsilon)$, we can safely drop them in the calculation

Our Algorithm in Arbitrary Order Streams

Arbitrary Order Stream: Edges of the input graph arrive in an arbitrary/adversarial order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Arbitrary Order Streams

- **Streaming Phase:**
  - Perform **reservoir sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch** (or **Misra-Gries**)
- **Post-Processing Phase:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute Estimate (1) and Estimate (2) approximately

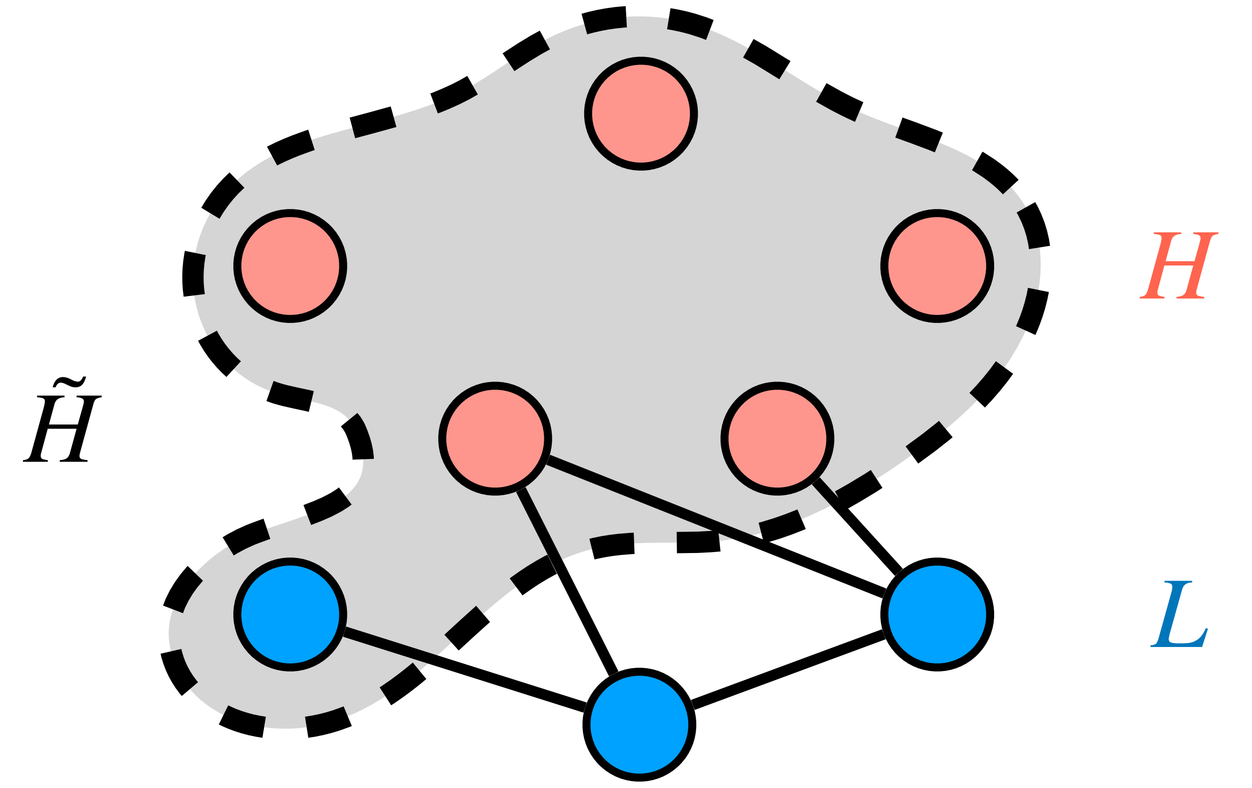
- Since $|\tilde{H}|, |E(\tilde{H})| = \mathrm{poly}(1/\epsilon)$, we can safely drop them in the calculation

Our Algorithm in Arbitrary Order Streams

Arbitrary Order Stream: Edges of the input graph arrive in an arbitrary/adversarial order

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)



Algorithm in Arbitrary Order Streams

- **Streaming Phase:**
  - Perform **reservoir sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch** (or **Misra-Gries**)
- **Post-Processing Phase:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute Estimate (1) and Estimate (2) approximately

- **Approx:** $1/2 + \Omega(\epsilon^2)$
- **Space:** $\mathrm{poly}(1/\epsilon)$ words

Our Algorithm in Dynamic Streams

Our Algorithm in Dynamic Streams

Dynamic Stream: The stream contains both edge insertions and deletions

Our Algorithm in Dynamic Streams

Dynamic Stream: The stream contains both edge insertions and deletions

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

Algorithm in Dynamic Streams

- **Streaming Phase:**
  - Perform ℓ_0**-sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch**
- **Post-Processing Phase:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute Estimate (1) and Estimate (2) approximately

Our Algorithm in Dynamic Streams

Dynamic Stream: The stream contains both edge insertions and deletions

Offline Algorithm

- Divide V into H (high-degree) and L (low-degree)
- Output the maximum of the following Max-Cut estimates:
  - **Candidate Estimate (1):**
    - Follow predictions on $G[L]$, compute the cut value of (L^+, L^-)
    - Extend (L^+, L^-) with vertices in H via a ***greedy*** approach
  - **Candidate Estimate (2):**
    - Compute the cut value of (H, L)

Algorithm in Dynamic Streams

- **Streaming Phase:**
  - Perform ℓ_0**-sampling** to obtain $\mathrm{poly}(1/\epsilon)$ edges
  - Store the degree info of all vertices to V^+ and V^- using **CountMin Sketch**
- **Post-Processing Phase:**
  - Detect $\tilde{H} \supseteq H$ and $\tilde{L} \subseteq L$ from the sampled edges
  - Compute Estimate (1) and Estimate (2) approximately

- **Approx:** $1/2 + \Omega(\epsilon^2)$
- **Space:** $\mathrm{poly}(1/\epsilon, \log n)$ words

Summary

Summary

- **Streaming Max-Cut with predictions**
  - **Takeaway:** ϵ-accurate predictions can help bypass the classical $(1/2+\epsilon)$-approx vs $\Omega_\epsilon(n)$ space trade-off!

Summary

- **Streaming Max-Cut with predictions**
  - **Takeaway:** ϵ-accurate predictions can help bypass the classical $(1/2 + \epsilon)$-approx vs $\Omega_\epsilon(n)$ space trade-off!
- **Open Problems**
  - Extension to weighted graphs?
  - Other graph problems in streaming?
  - Other prediction model? E.g., ϵ-accurate edge predictions [Aamand, Chen, Gollapudi, Silwal, Wu, ICML'25]

Summary

- **Streaming Max-Cut with predictions**
  - **Takeaway:** ϵ-accurate predictions can help bypass the classical $(1/2 + \epsilon)$-approx vs $\Omega_\epsilon(n)$ space trade-off!
- **Open Problems**
  - Extension to weighted graphs?
  - Other graph problems in streaming?
  - Other prediction model? E.g., ϵ-accurate edge predictions [Aamand, Chen, Gollapudi, Silwal, Wu, ICML'25]

Thank you!