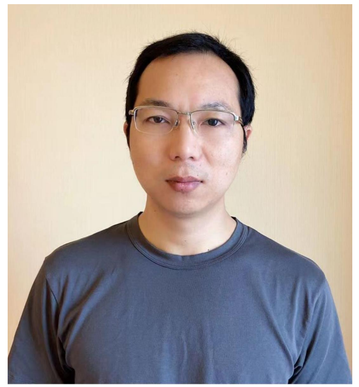
Learning-Augmented Streaming Algorithms for Correlation Clustering

Yinhao Dong
University of Science and Technology of China (USTC)

Joint work with

Shan Jiang
USTC

Shi Li
Nanjing University



Pan Peng
USTC

Correlation Clustering

Input: graph $G = (V, E = E^+ \cup E^-)$

Output: clustering $\mathscr{C}$ of V

Goal: minimize the number of edges in disagreement

	u, v in same cluster of $\mathscr{C}$	u, v in different clusters of $\mathscr{C}$
$(u, v) \in E^+$	agreement	*disagreement*
$(u, v) \in E^-$	*disagreement*	agreement

Correlation Clustering

Input: graph $G = (V, E = E^+ \cup E^-)$

Output: clustering $\mathscr{C}$ of V

Goal: minimize the number of edges in disagreement

	u, v in same cluster of $\mathscr{C}$	u, v in different clusters of $\mathscr{C}$
$(u, v) \in E^+$	agreement	*disagreement*
$(u, v) \in E^-$	*disagreement*	agreement

- Most commonly studied version: G is a complete graph,
  i.e., $E = \binom{V}{2}$

Correlation Clustering

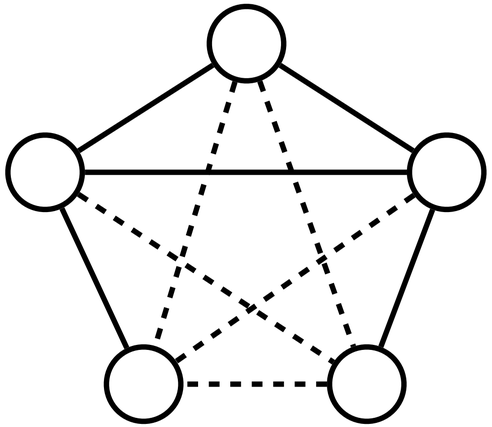
Input: graph $G = (V, E = E^+ \cup E^-)$

Output: clustering $\mathscr{C}$ of V

Goal: minimize the number of edges in disagreement

	u, v in same cluster of $\mathscr{C}$	u, v in different clusters of $\mathscr{C}$
$(u, v) \in E^+$	agreement	*disagreement*
$(u, v) \in E^-$	*disagreement*	agreement

- Most commonly studied version: G is a complete graph, i.e., $E = \binom{V}{2}$



Correlation Clustering

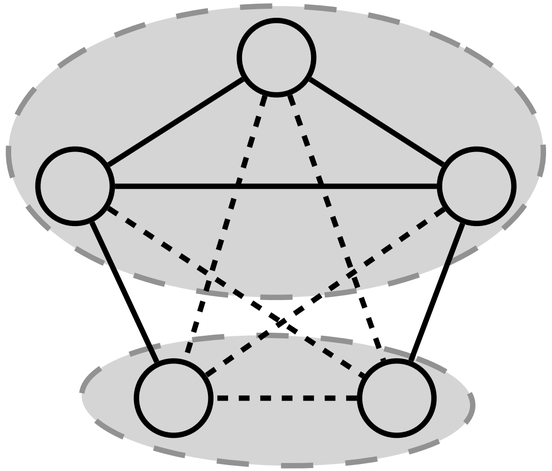
Input: graph $G = (V, E = E^+ \cup E^-)$

Output: clustering $\mathscr{C}$ of V

Goal: minimize the number of edges in disagreement

	u, v in same cluster of $\mathscr{C}$	u, v in different clusters of $\mathscr{C}$
$(u, v) \in E^+$	agreement	*disagreement*
$(u, v) \in E^-$	*disagreement*	agreement

- Most commonly studied version: G is a complete graph, i.e., $E = \binom{V}{2}$



Correlation Clustering

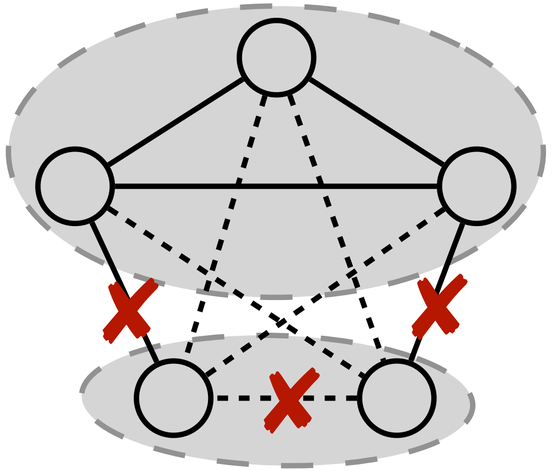
Input: graph $G = (V, E = E^+ \cup E^-)$

Output: clustering $\mathscr{C}$ of V

Goal: minimize the number of edges in disagreement

	u, v in same cluster of $\mathscr{C}$	u, v in different clusters of $\mathscr{C}$
$(u, v) \in E^+$	agreement	*disagreement*
$(u, v) \in E^-$	*disagreement*	agreement

- Most commonly studied version: G is a complete graph, i.e., $E = \binom{V}{2}$



Correlation Clustering

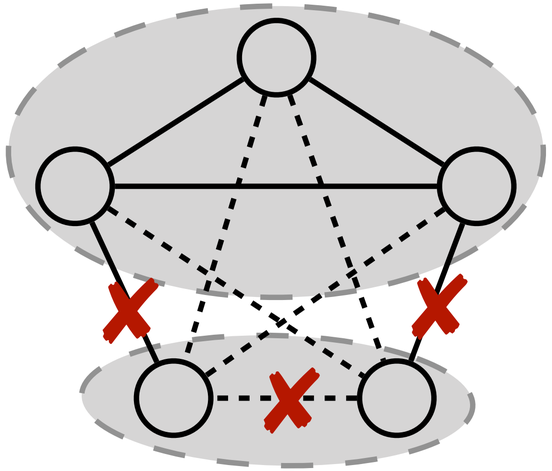
Input: graph $G = (V, E = E^+ \cup E^-)$

Output: clustering $\mathscr{C}$ of V

Goal: minimize the number of edges in disagreement

	u, v in same cluster of $\mathscr{C}$	u, v in different clusters of $\mathscr{C}$
$(u, v) \in E^+$	agreement	*disagreement*
$(u, v) \in E^-$	*disagreement*	agreement

- Most commonly studied version: G is a complete graph, i.e., $E = \binom{V}{2}$
- We consider both complete and general unweighted graphs in this work



Prior Results (Offline)

Prior Results (Offline)

- **On Complete Graphs**
  - APX-hard [Charikar, Guruswami, Wirth, 2005]
  - $(24/23 - \epsilon)$-hardness [Cao, Cohen-Addad, Lee, Li, Newman, Vogl, 2024]
  - Best-known approx. ratio: 1.437 via LP rounding [Cao, Cohen-Addad, Lee, Li, Newman, Vogl, 2024] [Cao, Cohen-Addad, Lee, Li, Lolck, Newman, Thorup, Vogl, Yan, Zhang, 2025]

Prior Results (Offline)

- **On Complete Graphs**
  - APX-hard [Charikar, Guruswami, Wirth, 2005]
  - $(24/23 - \epsilon)$-hardness [Cao, Cohen-Addad, Lee, Li, Newman, Vogl, 2024]
  - Best-known approx. ratio: 1.437 via LP rounding [Cao, Cohen-Addad, Lee, Li, Newman, Vogl, 2024] [Cao, Cohen-Addad, Lee, Li, Lolck, Newman, Thorup, Vogl, Yan, Zhang, 2025]
- **On General Graphs**
  - Equivalent to Min-Multicut and thus APX-hard [Bansal, Blum, Chawla, 2004]
  - Best-known approx. ratio: $O(\log n)$ via ball-growing based LP rounding [Charikar, Guruswami, Wirth, 2005] [Demaine, Emanuel, Fiat, Immorlica, 2006]

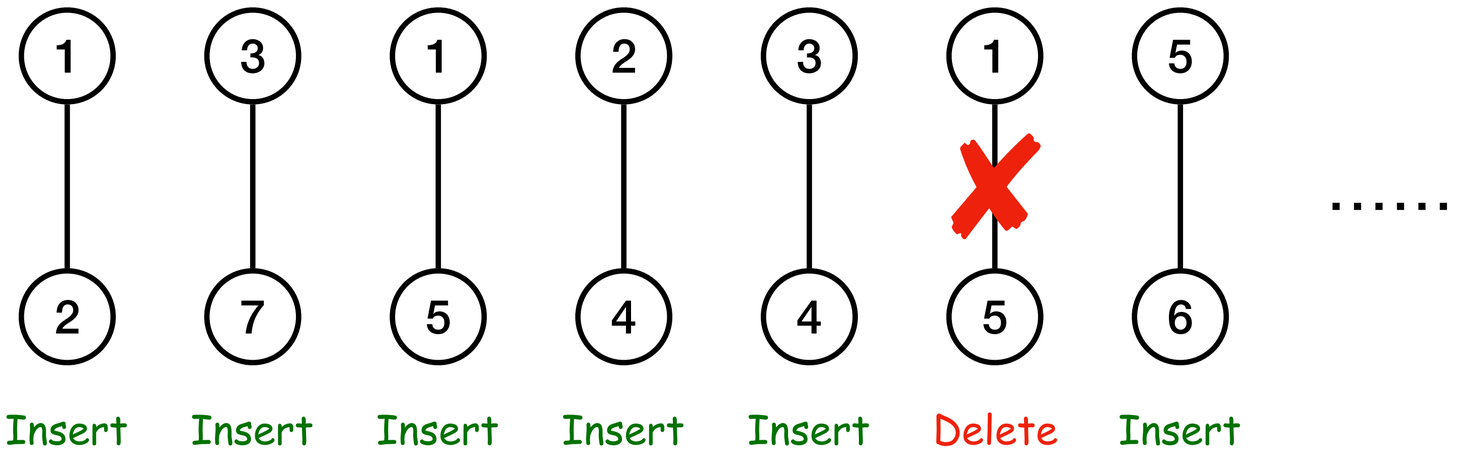
Streaming Model

Streaming Model

- **Graph Stream:** The input graph is presented as a sequence of edge insertions and deletions.

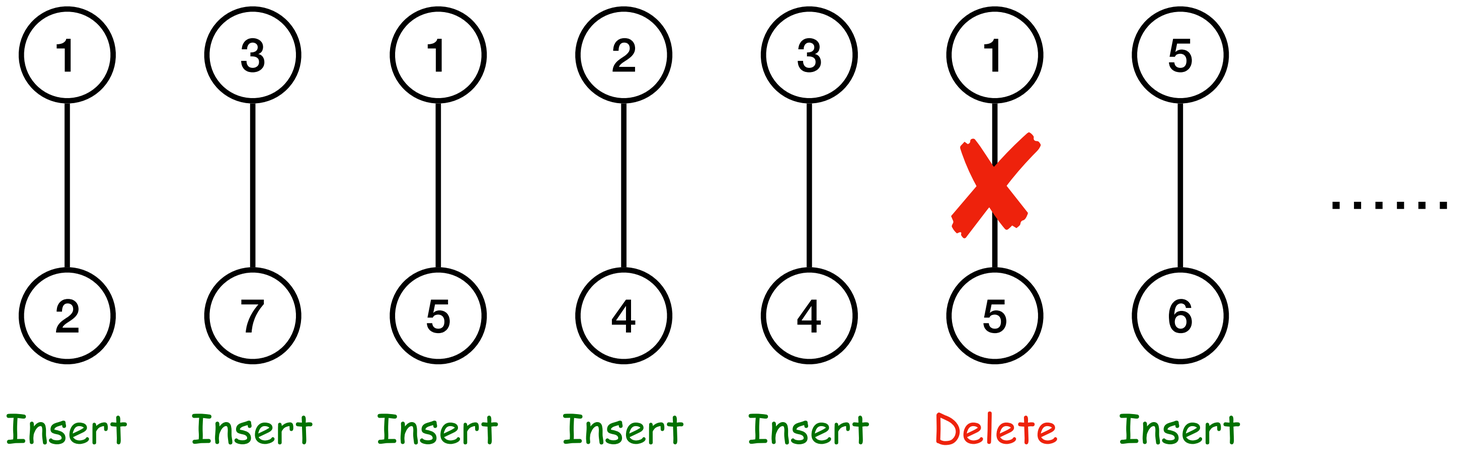
Streaming Model

- **Graph Stream:** The input graph is presented as a sequence of edge insertions and deletions.
  - *insertion-only* stream: contains only edge insertions
  - *dynamic* stream: contains both edge insertions and deletions



Streaming Model

- **Graph Stream:** The input graph is presented as a sequence of edge insertions and deletions.
  - *insertion-only* stream: contains only edge insertions
  - *dynamic* stream: contains both edge insertions and deletions
- **Goal:** scan the stream in (ideally) **one pass**, and find the solution at the end of the stream **using small space**



Correlation Clustering in Dynamic Streams

Correlation Clustering in Dynamic Streams

- Since outputting the clustering requires $\Omega(n)$ space, we consider semi-streaming model: $\tilde{O}(n)$ space is allowed

Correlation Clustering in Dynamic Streams

- Since outputting the clustering requires $\Omega(n)$ space, we consider semi-streaming model: $\tilde{O}(n)$ space is allowed
- **Best-known approximation-space trade-offs on complete graphs**

Correlation Clustering in Dynamic Streams

- Since outputting the clustering requires $\Omega(n)$ space, we consider semi-streaming model: $\tilde{O}(n)$ space is allowed
- **Best-known approximation-space trade-offs on complete graphs**
  - $(3+\epsilon)$-approx., $\tilde{O}(\epsilon^{-1}n)$ total space [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]

Correlation Clustering in Dynamic Streams

- Since outputting the clustering requires $\Omega(n)$ space, we consider semi-streaming model: $\tilde{O}(n)$ space is allowed
- **Best-known approximation-space trade-offs on complete graphs**
  - $(3+\epsilon)$-approx., $\tilde{O}(\epsilon^{-1}n)$ total space [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
  - $(\alpha_{\text{BEST}}+\epsilon)$-approx., $\tilde{O}(\epsilon^{-2}n)$ space during the stream, $\text{poly}(n)$ space for post-processing [Assadi, Khanna, Putterman, 2025]

Correlation Clustering in Dynamic Streams

- Since outputting the clustering requires $\Omega(n)$ space, we consider semi-streaming model: $\tilde{O}(n)$ space is allowed
- **Best-known approximation-space trade-offs on complete graphs**
  - $(3+\epsilon)$-approx., $\tilde{O}(\epsilon^{-1}n)$ total space [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
    best approx. ratio of any poly-time classical algorithm
  - $(\alpha_{\mathrm{BEST}}+\epsilon)$-approx., $\tilde{O}(\epsilon^{-2}n)$ space during the stream, $\mathrm{poly}(n)$ space for post-processing [Assadi, Khanna, Putterman, 2025]

Correlation Clustering in Dynamic Streams

- Since outputting the clustering requires $\Omega(n)$ space, we consider semi-streaming model: $\tilde{O}(n)$ space is allowed
- **Best-known approximation-space trade-offs on complete graphs**
  - $(3+\epsilon)$-approx., $\tilde{O}(\epsilon^{-1}n)$ total space [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
  - best approx. ratio of any poly-time classical algorithm
  - $(\alpha_{\text{BEST}}+\epsilon)$-approx., $\tilde{O}(\epsilon^{-2}n)$ space during the stream, $\text{poly}(n)$ space for post-processing [Assadi, Khanna, Putterman, 2025]
- **Best-known approximation-space trade-off on general graphs**
  - $O(\log|E^-|)$-approx., $\tilde{O}(\epsilon^{-2}n+|E^-|)$ total space [Ahn, Cormode, Guha, McGregor, Wirth, 2015]

Learning-Augmented Algorithms

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

Learning-Augmented Algorithms

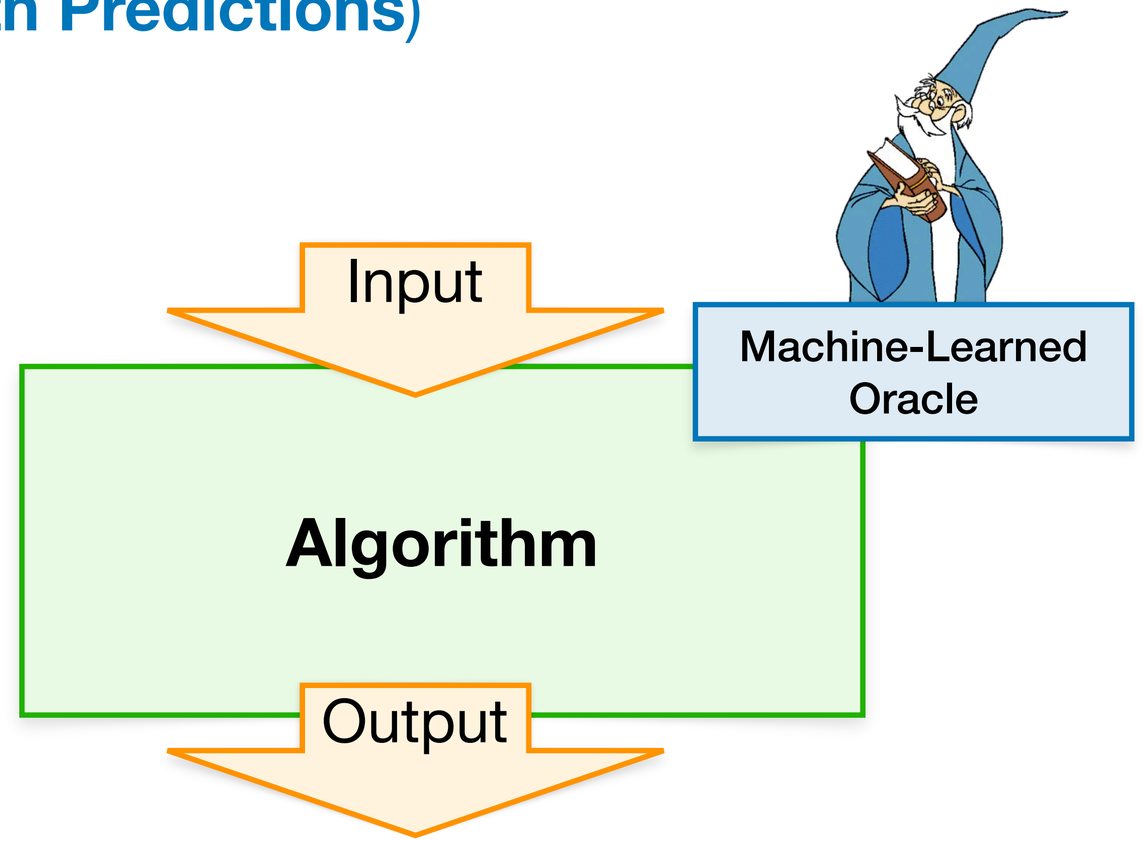
(a.k.a. **Algorithms with Predictions**)

- **Motivation:** Use ML techniques in classical algorithms to improve their performance beyond *worst-case* bounds

Learning-Augmented Algorithms

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

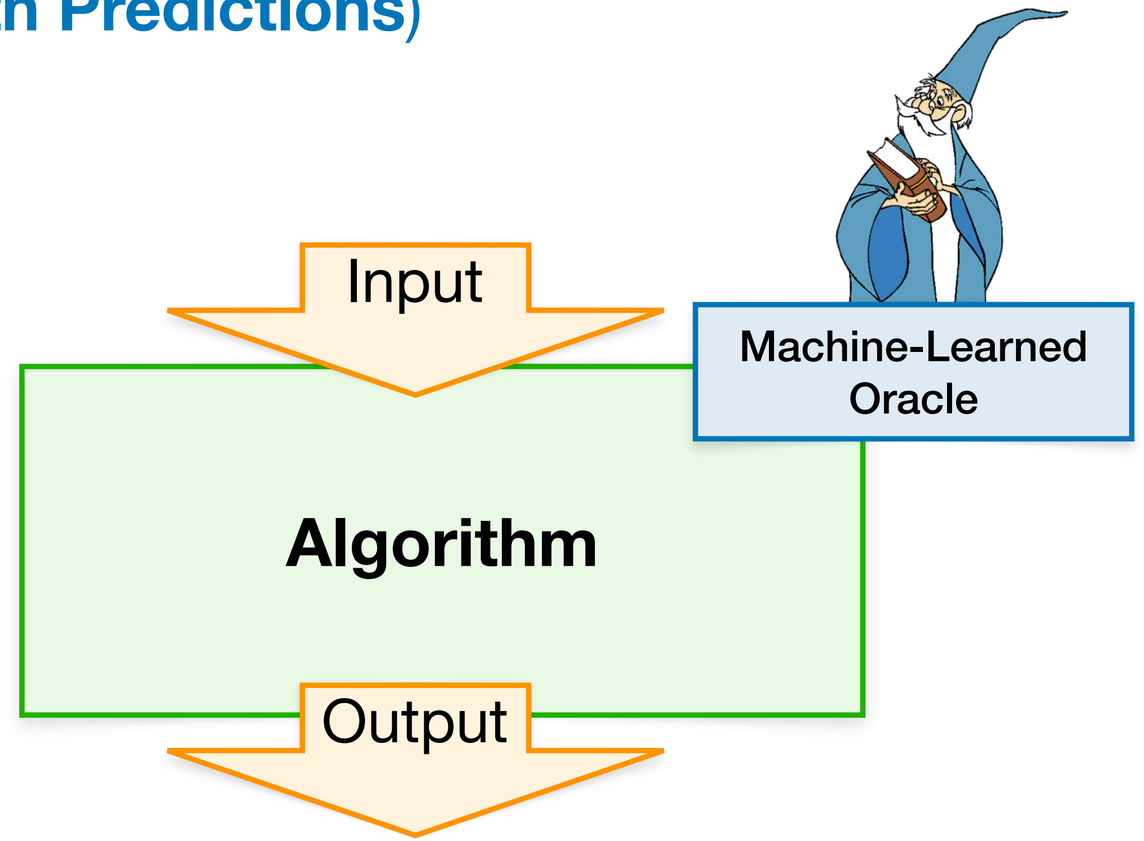
- **Motivation:** Use ML techniques in classical algorithms to improve their performance beyond *worst-case* bounds
- **Assumption:** The algorithm has oracle access to an (untrusted) predictor



Learning-Augmented Algorithms

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

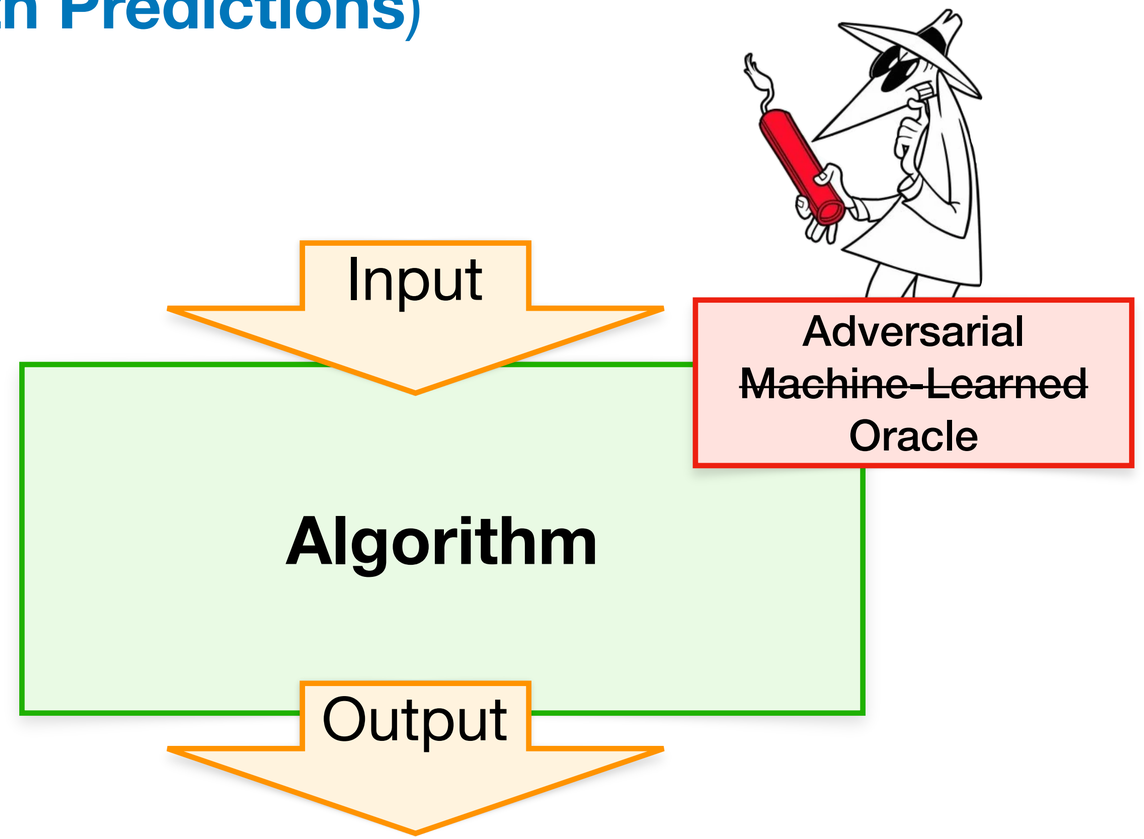
- **Motivation:** Use ML techniques in classical algorithms to improve their performance beyond *worst-case* bounds
- **Assumption:** The algorithm has oracle access to an (untrusted) predictor
- **Goals:**
  - High prediction quality $\Longrightarrow$ significantly outperforms the best-known classical (worst-case) algorithm



Learning-Augmented Algorithms

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

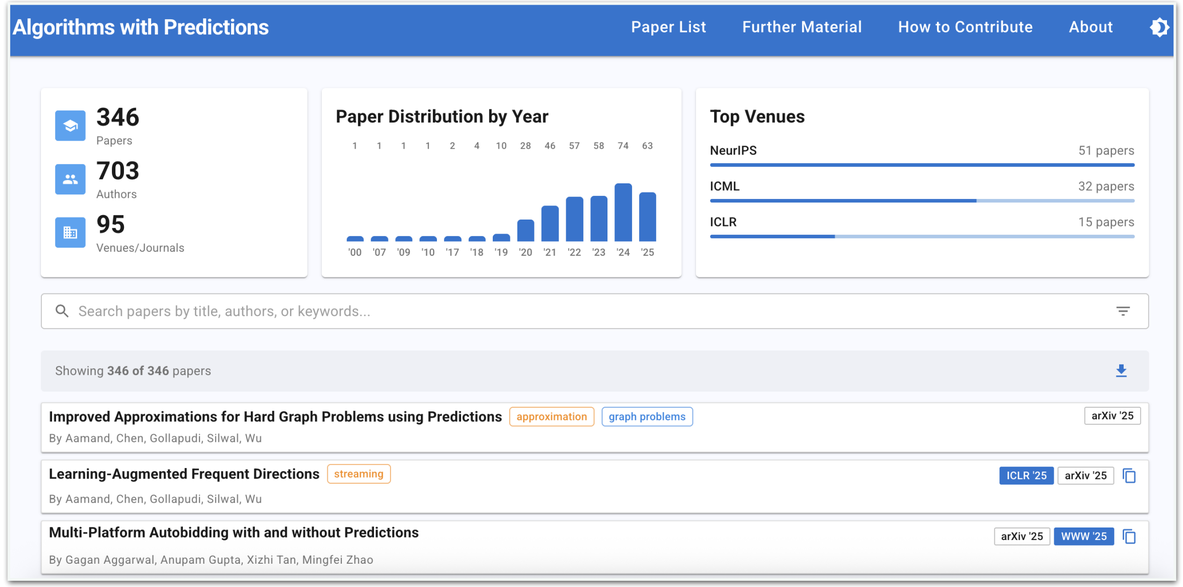
- **Motivation:** Use ML techniques in classical algorithms to improve their performance beyond *worst-case* bounds
- **Assumption:** The algorithm has oracle access to an (untrusted) predictor
- **Goals:**
  - High prediction quality $\Longrightarrow$ significantly outperforms the best-known classical (worst-case) algorithm
  - Low prediction quality $\Longrightarrow$ performs no worse than the best-known classical (worst-case) algorithm



Learning-Augmented Algorithms

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

- **Motivation:** Use ML techniques in classical algorithms to improve their performance beyond *worst-case* bounds
- **Assumption:** The algorithm has oracle access to an (untrusted) predictor
- **Goals:**
  - High prediction quality $\Longrightarrow$ significantly outperforms the best-known classical (worst-case) algorithm
  - Low prediction quality $\Longrightarrow$ performs no worse than the best-known classical (worst-case) algorithm



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

Our Prediction Model

Our Prediction Model

- Oracle access to pairwise distance $d_{uv} \in [0,1]$ between any $u, v \in V$

Our Prediction Model

- Oracle access to pairwise distance $d_{uv} \in [0,1]$ between any $u, v \in V$
- **Arises in many scenarios: multiple graphs on the same vertex set**
  - Healthcare: disease network, provider network, clinical trial network
  - Biology: protein-protein interaction network, gene co-expression network, signaling pathway network
  - Temporal graphs: same vertices, different edges over time

Our Prediction Model

- Oracle access to pairwise distance $d_{uv} \in [0,1]$ between any $u, v \in V$
- **Arises in many scenarios: multiple graphs on the same vertex set**
  - Healthcare: disease network, provider network, clinical trial network
  - Biology: protein-protein interaction network, gene co-expression network, signaling pathway network
  - Temporal graphs: same vertices, different edges over time
- **Observation:** Two vertices similar in one network are likely similar in another — cluster structure can thus be extracted

Our Prediction Model

β-level predictor ($\beta \geq 1$): predicts pairwise distance $d_{uv} \in [0,1]$ between any $u, v \in V$ such that

(1) $d_{uv} + d_{vw} \geq d_{uw}$ for all $u, v, w \in V$ (triangle inequality)

(2) $$\sum_{(u,v)\in E^+} d_{uv} + \sum_{(u,v)\in E^-} (1 - d_{uv}) \leq \beta \cdot \text{OPT}$$

Our Prediction Model

β-level predictor ($\beta \geq 1$): predicts pairwise distance $d_{uv} \in [0,1]$ between any $u, v \in V$ such that

(1) $d_{uv} + d_{vw} \geq d_{uw}$ for all $u, v, w \in V$ (triangle inequality)

(2) $$\sum_{(u,v)\in E^+} d_{uv} + \sum_{(u,v)\in E^-} (1 - d_{uv}) \leq \beta \cdot \text{OPT}$$

- Inspired by the metric LP formulation of Correlation Clustering
- Smaller $\beta \Longrightarrow$ higher quality
- Can be implemented in practice!

$$
\begin{aligned}
\min \quad & \sum_{(u,v)\in E^+} x_{uv} + \sum_{(u,v)\in E^-} (1 - x_{uv}) & \\
\text{s.t.} \quad & x_{uw} + x_{wv} \geq x_{uv} & \forall u, v, w \in V \\
& x_{uv} \in [0,1] & \forall (u,v) \in \binom{V}{2} \\
& x_{uu} = 0 & \forall u \in V
\end{aligned}
$$

Our Results

Setting	Best-known approx.-space trade-offs (without predictions)	Our results (with predictions)
Complete graphs, Dynamic streams	$(3+\epsilon)$-approx. $\tilde{O}(\epsilon^{-1}n)$ total space [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]	$(\min\{2.06\beta,3\}+\epsilon)$-approx. $\tilde{O}(\epsilon^{-2}n)$ total space [**D.**, Jiang, Li, Peng, 2025] better approx.-space tradeoff
	$(\alpha_{\text{BEST}}+\epsilon)$-approx. $\tilde{O}(\epsilon^{-2}n)$ space during the stream $\text{poly}(n)$ space for post-processing [Assadi, Khanna, Putterman, 2025]	
General graphs, Dynamic streams	$O(\log \lvert E^- \rvert)$-approx. $\tilde{O}(\epsilon^{-2}n + \lvert E^- \rvert)$ total space [Ahn, Cormode, Guha, McGregor, Wirth, 2015]	$O(\beta \log \lvert E^- \rvert)$-approx. $\tilde{O}(\epsilon^{-2}n)$ total space [**D.**, Jiang, Li, Peng, 2025] better space complexity

Our Algorithm for Complete Graphs: Key Insight

Our Algorithm for Complete Graphs: Key Insight

- For a long time, $(3+\epsilon)$-approx. is a natural target in streaming

Our Algorithm for Complete Graphs: Key Insight

- For a long time, $(3+\epsilon)$-approx. is a natural target in streaming
  - The seminal 3-approx. combinatorial PIVOT algorithm [Ailon, Charikar, Neuman, 2008] can be efficiently simulated in streaming [Chakrabarty, Makarychev, 2023] [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]

Our Algorithm for Complete Graphs: Key Insight

- For a long time, $(3+\epsilon)$-approx. is a natural target in streaming
  - The seminal 3-approx. combinatorial PIVOT algorithm [Ailon, Charikar, Neuman, 2008] can be efficiently simulated in streaming [Chakrabarty, Makarychev, 2023] [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
  - A combinatorial 1.847-approx. algorithm [Cohen-Addad, Lolck, Pilipczuk, Thorup, Yan, Zhang, 2024]: works only in insertion-only streams, far from practical

Our Algorithm for Complete Graphs: Key Insight

- For a long time, $(3+\epsilon)$-approx. is a natural target in streaming
  - The seminal 3-approx. combinatorial PIVOT algorithm [Ailon, Charikar, Neuman, 2008] can be efficiently simulated in streaming [Chakrabarty, Makarychev, 2023] [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
  - A combinatorial 1.847-approx. algorithm [Cohen-Addad, Lolck, Pilipczuk, Thorup, Yan, Zhang, 2024]: works only in insertion-only streams, far from practical
- However, there are several works breaking 3-approx. barrier in the offline setting
  - 2.06 [Chawla, Makarychev, Schramm, Yaroslavtsev, 2015]: metric LP
  - 1.994 [Cohen-Addad, Lee, Neuman, 2022]: Sherali-Adams LP relaxation hierarchy
  - 1.73 [Cohen-Addad, Lee, Li, Neuman, 2023]: Sherali-Adams LP relaxation hierarchy + Preclustering
  - 1.437 [Cao, Cohen-Addad, Lee, Li, Neuman, Vogl, 2024] [Cao, Cohen-Addad, Lee, Li, Lolck, Newman, Thorup, Vogl, Yan, Zhang, 2025]: cluster LP

Our Algorithm for Complete Graphs: Key Insight

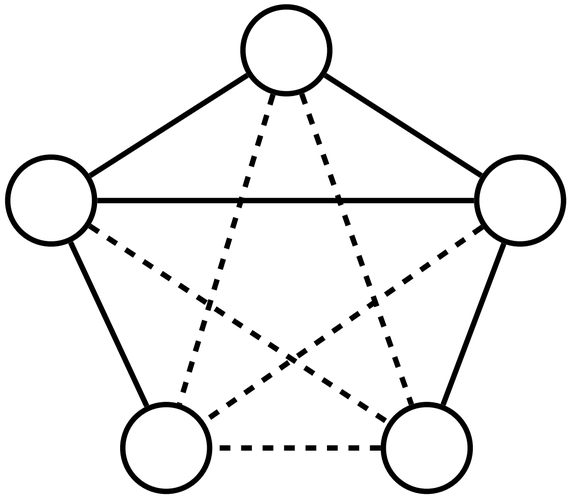
- For a long time, $(3+\epsilon)$-approx. is a natural target in streaming
  - The seminal 3-approx. combinatorial PIVOT algorithm [Ailon, Charikar, Neuman, 2008] can be efficiently simulated in streaming [Chakrabarty, Makarychev, 2023] [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
  - A combinatorial 1.847-approx. algorithm [Cohen-Addad, Lolck, Pilipczuk, Thorup, Yan, Zhang, 2024]: works only in insertion-only streams, far from practical
- However, there are several works breaking 3-approx. barrier in the offline setting
  - 2.06 [Chawla, Makarychev, Schramm, Yaroslavtsev, 2015]: metric LP
  - 1.994 [Cohen-Addad, Lee, Neuman, 2022]: Sherali-Adams LP relaxation hierarchy
  - 1.73 [Cohen-Addad, Lee, Li, Neuman, 2023]: Sherali-Adams LP relaxation hierarchy + Preclustering
  - 1.437 [Cao, Cohen-Addad, Lee, Li, Neuman, Vogl, 2024] [Cao, Cohen-Addad, Lee, Li, Lolck, Newman, Thorup, Vogl, Yan, Zhang, 2025]: cluster LP
  - All these algorithms are based on LP rounding — difficult to implement in streaming 😭

Our Algorithm for Complete Graphs: Key Insight

- For a long time, $(3+\epsilon)$-approx. is a natural target in streaming
  - The seminal 3-approx. combinatorial PIVOT algorithm [Ailon, Charikar, Neuman, 2008] can be efficiently simulated in streaming [Chakrabarty, Makarychev, 2023] [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
  - A combinatorial 1.847-approx. algorithm [Cohen-Addad, Lolck, Pilipczuk, Thorup, Yan, Zhang, 2024]: works only in insertion-only streams, far from practical
- However, there are several works breaking 3-approx. barrier in the offline setting
  - 2.06 [Chawla, Makarychev, Schramm, Yaroslavtsev, 2015]: metric LP
  - 1.994 [Cohen-Addad, Lee, Neuman, 2022]: Sherali-Adams LP relaxation hierarchy
  - 1.73 [Cohen-Addad, Lee, Li, Neuman, 2023]: Sherali-Adams LP relaxation hierarchy + Preclustering
  - 1.437 [Cao, Cohen-Addad, Lee, Li, Neuman, Vogl, 2024] [Cao, Cohen-Addad, Lee, Li, Lolck, Newman, Thorup, Vogl, Yan, Zhang, 2025]: cluster LP
  - All these algorithms are based on LP rounding — difficult to implement in streaming 😭
- **We can use "pairwise distance" predictions to bypass this bottleneck! 😄**

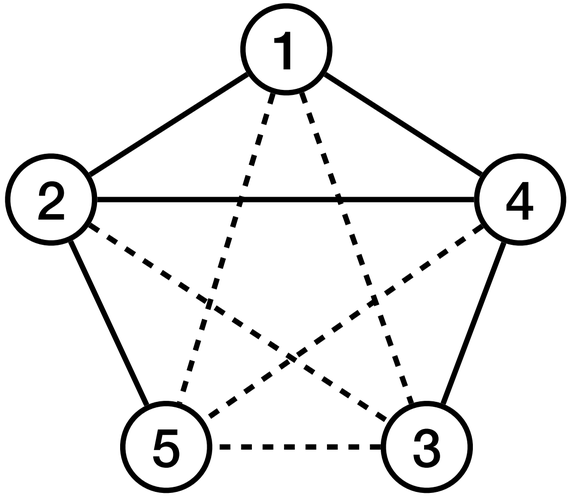
3-Approx. PIVOT Algorithm [Ailon, Charikar, Neuman, 2008]

1. Pick a random permutation $V \to \{1, \ldots, n\}$ over the vertices.
2. $V' \leftarrow V, \mathscr{C} \leftarrow \emptyset$
3. **while** $V' \neq \emptyset$ **do**
   - Let $p \in V'$ be the vertex with the smallest rank. Mark p as **pivot**.
   - Initialize $C \leftarrow \{p\}$.
   - Add $v \in V'$ to C if $(p, v) \in E^+$.
   - $V' \leftarrow V' \backslash C, \mathscr{C} \leftarrow \mathscr{C} \cup \{C\}$
4. **return** $\mathscr{C}$



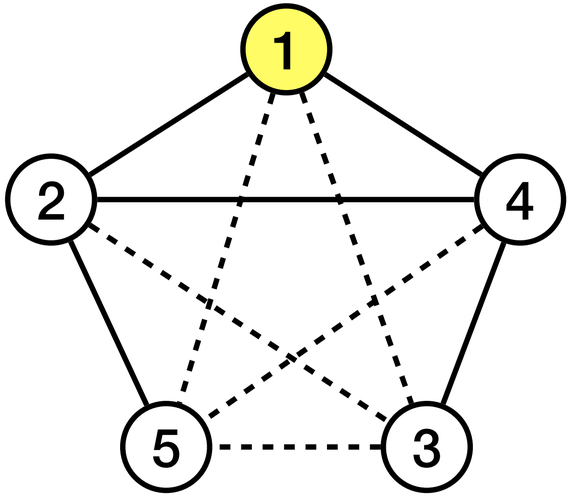
3-Approx. PIVOT Algorithm [Ailon, Charikar, Neuman, 2008]

1. Pick a random permutation $V \to \{1, \ldots, n\}$ over the vertices.
2. $V' \leftarrow V, \mathscr{C} \leftarrow \emptyset$
3. **while** $V' \neq \emptyset$ **do**
   - Let $p \in V'$ be the vertex with the smallest rank. Mark p as **pivot**.
   - Initialize $C \leftarrow \{p\}$.
   - Add $v \in V'$ to C if $(p, v) \in E^+$.
   - $V' \leftarrow V' \backslash C, \mathscr{C} \leftarrow \mathscr{C} \cup \{C\}$
4. **return** $\mathscr{C}$



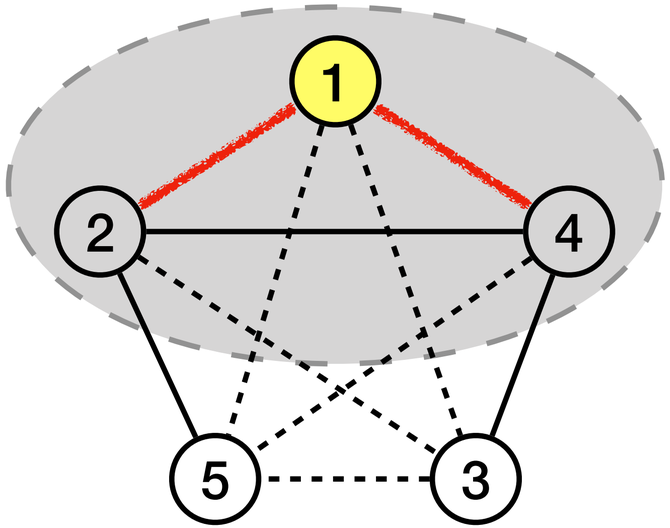
3-Approx. PIVOT Algorithm [Ailon, Charikar, Neuman, 2008]

1. Pick a random permutation $V \to \{1, \ldots, n\}$ over the vertices.
2. $V' \leftarrow V, \mathscr{C} \leftarrow \emptyset$
3. **while** $V' \neq \emptyset$ **do**
   - Let $p \in V'$ be the vertex with the smallest rank. Mark p as **pivot**.
   - Initialize $C \leftarrow \{p\}$.
   - Add $v \in V'$ to C if $(p, v) \in E^+$.
   - $V' \leftarrow V' \backslash C, \mathscr{C} \leftarrow \mathscr{C} \cup \{C\}$
4. **return** $\mathscr{C}$



3-Approx. PIVOT Algorithm [Ailon, Charikar, Neuman, 2008]

1. Pick a random permutation $V \to \{1, \ldots, n\}$ over the vertices.
2. $V' \leftarrow V, \mathscr{C} \leftarrow \emptyset$
3. **while** $V' \neq \emptyset$ **do**
   - Let $p \in V'$ be the vertex with the smallest rank. Mark p as **pivot**.
   - Initialize $C \leftarrow \{p\}$.
   - Add $v \in V'$ to C if $(p, v) \in E^+$.
   - $V' \leftarrow V' \backslash C, \mathscr{C} \leftarrow \mathscr{C} \cup \{C\}$
4. **return** $\mathscr{C}$



2.06-Approx. LP Rounding Algorithm

[Chawla, Makarychev, Schramm, Yaroslavtsev, 2015]

2.06-Approx. LP Rounding Algorithm

[Chawla, Makarychev, Schramm, Yaroslavtsev, 2015]

1. Solve the **metric LP** of Correlation Clustering and obtain the optimal solution $\{x_{uv}\}_{u,v \in V}$.
2. Pick a random permutation $V \to \{1, \ldots, n\}$ over the vertices.
3. $V' \leftarrow V, \mathscr{C} \leftarrow \varnothing$
4. **while** $V' \neq \varnothing$ **do**
   - Let $p \in V'$ be the vertex with the smallest rank. Mark p as **pivot**.
   - Initialize $C \leftarrow \{p\}$.
   - Add $v \in V'$ to C independently w.p. $1 - f(x_{pv})$.
   - $V' \leftarrow V' \backslash C, \mathscr{C} \leftarrow \mathscr{C} \cup \{C\}$
5. **return** $\mathscr{C}$

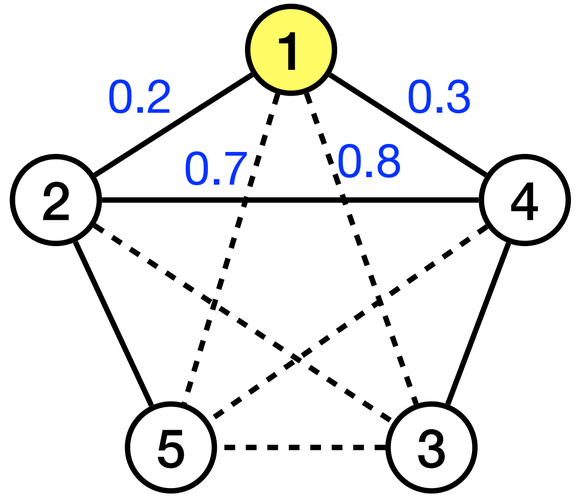
$$
\begin{aligned}
\min \quad & \sum_{(u,v) \in E^+} x_{uv} + \sum_{(u,v) \in E^-} (1 - x_{uv}) \\
\text{s.t.} \quad & x_{uw} + x_{wv} \geq x_{uv} && \forall u, v, w \in V \\
& x_{uv} \in [0, 1] && \forall (u, v) \in \binom{V}{2} \\
& x_{uu} = 0 && \forall u \in V
\end{aligned}
$$

2.06-Approx. LP Rounding Algorithm

[Chawla, Makarychev, Schramm, Yaroslavtsev, 2015]

1. Solve the **metric LP** of Correlation Clustering and obtain the optimal solution $\{x_{uv}\}_{u,v \in V}$.
2. Pick a random permutation $V \to \{1, \ldots, n\}$ over the vertices.
3. $V' \leftarrow V, \mathscr{C} \leftarrow \varnothing$
4. **while** $V' \neq \varnothing$ **do**
   - Let $p \in V'$ be the vertex with the smallest rank. Mark p as **pivot**.
   - Initialize $C \leftarrow \{p\}$.
   - Add $v \in V'$ to C independently w.p. $1 - f(x_{pv})$.
   - $V' \leftarrow V' \backslash C, \mathscr{C} \leftarrow \mathscr{C} \cup \{C\}$
5. **return** $\mathscr{C}$

$$
\begin{aligned}
\min \quad & \sum_{(u,v) \in E^+} x_{uv} + \sum_{(u,v) \in E^-} (1 - x_{uv}) \\
\text{s.t.} \quad & x_{uw} + x_{wv} \geq x_{uv} && \forall u, v, w \in V \\
& x_{uv} \in [0, 1] && \forall (u, v) \in \binom{V}{2} \\
& x_{uu} = 0 && \forall u \in V
\end{aligned}
$$



Our Streaming Algorithm for Complete Graphs

1. **During the stream:**
   - Maintain a truncated subgraph G' of G (refer to [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]).
2. **After the stream:**
   - Run the 3-approx. combinatorial algorithm (PIVOT) on G', then assign unclustered vertices and obtain clustering $\mathscr{C}_1$ on G.
   - Run the 2.06-approx. LP rounding algorithm on G' (**use predictions d_{uv} to replace metric LP solution x_{uv}**), then assign unclustered vertices and obtain clustering $\mathscr{C}_2$ on G.
   - **return** the clustering with the lower cost between $\mathscr{C}_1$ and $\mathscr{C}_2$

Our Streaming Algorithm for Complete Graphs

1. **During the stream:**
   - Maintain a truncated subgraph G' of G (refer to [Cambus, Kuhn, Lindy, Pai, Uitt
2. **After the stream:**
   - Run the 3-approx. combinatorial algorithm (PIVOT) on G, then assign unclustered vertices and obtain clustering $\mathscr{C}_1$ on G.
   - Run the 2.06-approx. LP rounding algorithm on G' (**use predictions d_{uv} to replace metric LP solution x_{uv}**), then assign unclustered vertices and obtain clustering $\mathscr{C}_2$ on G.
   - **return** the clustering with the lower cost between $\mathscr{C}_1$ and $\mathscr{C}_2$

Lemma [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]**:**

$$\text{cost}_G(\mathscr{C}_1) \leq (3+\epsilon) \cdot \text{OPT}$$

Our Streaming Algorithm for Complete Graphs

1. **During the stream:**
   - Maintain a truncated subgraph G' of G (refer to [Cambus, Kuhn, Lindy, Pai, Uitt
2. **After the stream:**
   - Run the 3-approx. combinatorial algorithm (PIVOT) on G', then assign unclustered vertices and obtain clustering $\mathscr{C}_1$ on G.
   - Run the 2.06-approx. LP rounding algorithm on G' (**use predictions d_{uv} to replace metric LP solution x_{uv}**), then assign unclustered vertices and obtain clustering $\mathscr{C}_2$ on G.
   - **return** the clustering with the lower cost bet en $\mathscr{C}_1$ and $\mathscr{C}_2$

Lemma [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]**:**

$$\text{cost}_G(\mathscr{C}_1) \leq (3+\epsilon)\cdot \text{OPT}$$

Lemma [**D.**, Jiang, Li, Peng, 2025]**:**

$$\text{cost}_G(\mathscr{C}_2) \leq (2.06\beta+\epsilon)\cdot \text{OPT}$$

Our Streaming Algorithm for Complete Graphs

1. **During the stream:**
   - Maintain a truncated subgraph G' of G (refer to [Cambus, Kuhn, Lindy, Pai, Uitt
2. **After the stream:**
   - Run the 3-approx. combinatorial algorithm (PIVOT) on G', then assign unclustered vertices and obtain clustering $\mathscr{C}_1$ on G.
   - Run the 2.06-approx. LP rounding algorithm on G' (**use predictions d_{uv} to replace metric LP solution x_{uv}**), then assign unclustered vertices and obtain clustering $\mathscr{C}_2$ on G.
   - **return** the clustering with the lower cost bet en $\mathscr{C}_1$ and $\mathscr{C}_2$

Lemma [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]**:**
$\text{cost}_G(\mathscr{C}_1) \le (3+\epsilon)\cdot \text{OPT}$

Lemma [**D.**, Jiang, Li, Peng, 2025]**:**
$\text{cost}_G(\mathscr{C}_2) \le (2.06\beta+\epsilon)\cdot \text{OPT}$

Theorem [**D.**, Jiang, Li, Peng, 2025]**:**

β-level predictor

↓ w.p. $\ge 1 - 1/n^2$

$(\min\{2.06\beta,\ 3\}+\epsilon)$-approx.

$\tilde{O}(n)$ words of **total space**,

works in dynamic streams

Our Streaming Algorithm for Complete Graphs

1. **During the stream:**
   - Maintain a truncated subgraph G' of G (refer to [Cambus, Kuhn, Lindy, Pai, Uitt
2. **After the stream:**
   - Run the 3-approx. combinatorial algorithm (PIVOT) on G', then assign unclustered vertices and obtain clustering $\mathscr{C}_1$ on G.
   - Run the 2.06-approx. LP rounding algorithm on G' (**use predictions d_{uv} to replace metric LP solution x_{uv}**), then assign unclustered vertices and obtain clustering $\mathscr{C}_2$ on G.
   - **return** the clustering with the lower cost bet en $\mathscr{C}_1$ and $\mathscr{C}_2$

Lemma [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]**:**

$\text{cost}_G(\mathscr{C}_1) \le (3+\epsilon) \cdot \text{OPT}$

Lemma [**D.**, Jiang, Li, Peng, 2025]**:**

$\text{cost}_G(\mathscr{C}_2) \le (2.06\beta+\epsilon) \cdot \text{OPT}$

Theorem [**D.**, Jiang, Li, Peng, 2025]**:**

β-level predictor

↓ w.p. $\ge 1 - 1/n^2$

$(\min\{2.06\beta,\ 3\} + \epsilon)$-approx.

$\tilde{O}(n)$ words of **total space**,

works in dynamic streams

Remarks:

- Better than 3-approx. under good prediction quality
- Simple and efficient
- Do not consider the space for the predictor

Our Streaming Algorithm for General Graphs

Our Streaming Algorithm for General Graphs

- Recall that the best-known streaming algorithm for general graphs is a $O(\log |E^-|)$-approx. while using $\tilde{O}(\epsilon^{-2}n + |E^-|)$ total space [Ahn, Cormode, Guha, McGregor, Wirth, 2015]

Our Streaming Algorithm for General Graphs

- Recall that the best-known streaming algorithm for general graphs is a $O(\log |E^-|)$-approx. while using $\tilde{O}(\epsilon^{-2}n + |E^-|)$ total space [Ahn, Cormode, Guha, McGregor, Wirth, 2015]

  - **During the stream:** Sparsify the positive subgraph $G^+ := (V, E^+)$ to H^+, and store E^-

  - **After the stream:** Use the stored information to solve an LP, and run a ball-growing based LP rounding algorithm on H^+

Our Streaming Algorithm for General Graphs

- Recall that the best-known streaming algorithm for general graphs is a $O(\log |E^-|)$-approx. while using $\tilde{O}(\epsilon^{-2}n + |E^-|)$ total space [Ahn, Cormode, Guha, McGregor, Wirth, 2015]
  - **During the stream:** Sparsify the positive subgraph $G^+ := (V, E^+)$ to H^+, and store E^-
  - **After the stream:** Use the stored information to solve an LP, and run a ball-growing based LP rounding algorithm on H^+
- **We can use predictions to guide the rounding algorithm, thus avoiding storing E^- and leading to better space complexity!**

Our Streaming Algorithm for General Graphs

1. **During the stream:**
   - Maintain a spectral sparsifier H^+ for $G^+ := (V, E^+)$.
2. **After the stream:** perform ball-growing
   - $V' \leftarrow V, \mathscr{C} \leftarrow \emptyset$
   - while $V' \neq \emptyset$ do
     - Pick an arbitrary vertex $u \in V'$. Initialize $r_u \leftarrow 0$.
     - Increase r_u and grow a ball $B(u, r_u)$ **using predictions d_{uv} as distance metric**, until a certain condition is satisfied.
     - $V' \leftarrow V' \backslash B(u, r_u), \mathscr{C} \leftarrow \mathscr{C} \cup \{B(u, r_u)\}$
   - **return** the resulting clustering

Our Streaming Algorithm for General Graphs

1. **During the stream:**
   - Maintain a spectral sparsifier H^+ for $G^+ := (V, E^+)$.
2. **After the stream:** perform ball-growing
   - $V' \leftarrow V, \mathscr{C} \leftarrow \emptyset$
   - while $V' \neq \emptyset$ do
     - Pick an arbitrary vertex $u \in V'$. Initialize $r_u \leftarrow 0$.
     - Increase r_u and grow a ball $B(u, r_u)$ **using predictions d_{uv} as distance metric**, until a certain condition is satisfied.
     - $V' \leftarrow V' \backslash B(u, r_u), \mathscr{C} \leftarrow \mathscr{C} \cup \{B(u, r_u)\}$
   - **return** the resulting clustering

Theorem [D., Jiang, Li, Peng, 2025]**:**

β-level predictor

⬇ w.p. $\geq 1 - 1/n^2$

$O(\beta \log |E^-|)$-approx.

$\tilde{O}(n)$ words of **total space**,

works in dynamic streams

Our Streaming Algorithm for General Graphs

1. **During the stream:**
   - Maintain a spectral sparsifier H^+ for $G^+ := (V, E^+)$.
2. **After the stream:** perform ball-growing
   - $V' \leftarrow V, \mathscr{C} \leftarrow \emptyset$
   - while $V' \neq \emptyset$ do
     - Pick an arbitrary vertex $u \in V'$. Initialize $r_u \leftarrow 0$.
     - Increase r_u and grow a ball $B(u, r_u)$ **using predictions d_{uv} as distance metric**, until a certain condition is satisfied.
     - $V' \leftarrow V' \backslash B(u, r_u), \mathscr{C} \leftarrow \mathscr{C} \cup \{B(u, r_u)\}$
   - **return** the resulting clustering

Theorem [D., Jiang, Li, Peng, 2025]**:**

β-level predictor

⬇ w.p. $\geq 1 - 1/n^2$

$O(\beta \log |E^-|)$-approx.

$\tilde{O}(n)$ words of **total space**,

works in dynamic streams

Remarks:

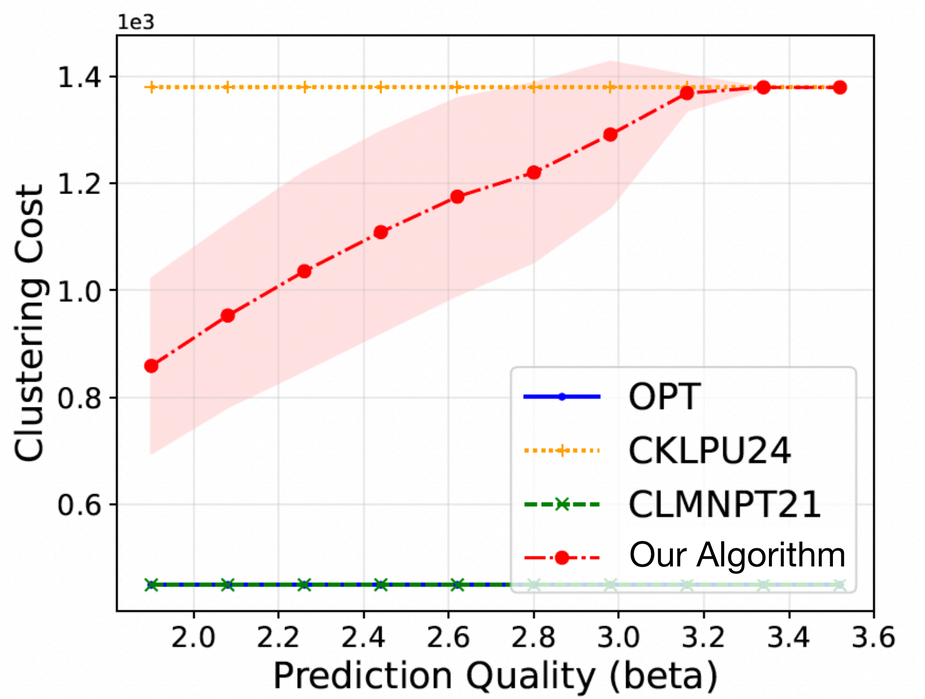
- Close to $O(\log |E^-|)$-approx. under good prediction quality
- Better space complexity
- Do not consider the space for the predictor

Experimental Setting

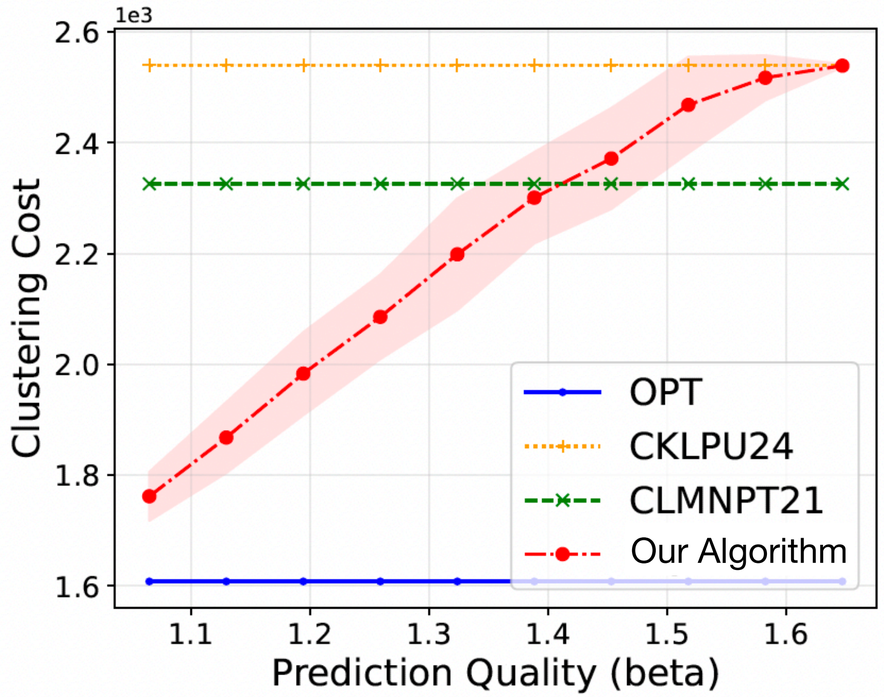
- **Datasets:**
  - Synthetic datasets: Generated from the Stochastic Block Model (SBM)
  - Real-world datasets: EmailCore, Facebook, LastFM, DBLP from SNAP Collection
- **Predictor:** Noisy predictor, Spectral embedding, Binary classifier
- **Baselines:**
  - $(3+\epsilon)$-approx. algorithm without predictions [Cambus, Kuhn, Lindy, Pai, Uitto, 2024]
  - Agreement decomposition-based algorithm [Cohen-Addad, Lattanzi, Mitrović, Norouzi-Fard, Parotsidis, Tarnawski, 2021]: 701-approx. in theory, performs well on certain types of graphs

Experimental Results

Synthetic Dataset



Real-World Dataset: Facebook



Conclusion:

- Our algorithm (with predictions) outperforms its non-learning counterpart (CKLPU24) under high prediction quality, while no worse under low prediction quality.
- Our algorithm performs much better in practice than the theoretical guarantee suggests.

Summary

- The first learning-augmented streaming algorithms for Correlation Clustering on both **complete** and **general** graphs (in dynamic streams)
  - For complete graphs: β-level "pairwise distance" predictor $\Longrightarrow$ $(\min\{2.06\beta, 3\} + \epsilon)$-approx., $\tilde{O}(\epsilon^{-2}n)$ total space
    better approximation-space tradeoff
  - For general graphs: β-level "pairwise distance" predictor $\Longrightarrow$ $O(\beta \log |E^-|)$-approx., $\tilde{O}(\epsilon^{-2}n)$ total space
    better space complexity

Summary

- The first learning-augmented streaming algorithms for Correlation Clustering on both **complete** and **general** graphs (in dynamic streams)
  - For complete graphs: β-level "pairwise distance" predictor $\Longrightarrow$ $(\min\{2.06\beta, 3\} + \epsilon)$-approx., $\tilde{O}(\epsilon^{-2}n)$ total space
  - For general graphs: β-level "pairwise distance" predictor $\Longrightarrow$ $O(\beta \log |E^-|)$-approx., $\tilde{O}(\epsilon^{-2}n)$ total space

better approximation-space tradeoff

better space complexity

- **Open Problems**
  - $(\alpha_{\text{BEST}} + \epsilon)$-approx., $\tilde{O}(n)$ **total space** for complete graphs?
  - Better-than-$O(\log |E^-|)$-approx. for general graphs?

Summary

- The first learning-augmented streaming algorithms for Correlation Clustering on both **complete** and **general** graphs (in dynamic streams)
  - For complete graphs: β-level "pairwise distance" predictor $\Longrightarrow$ $(\min\{2.06\beta, 3\} + \epsilon)$-approx., $\tilde{O}(\epsilon^{-2}n)$ total space
  - For general graphs: β-level "pairwise distance" predictor $\Longrightarrow$ $O(\beta \log |E^-|)$-approx., $\tilde{O}(\epsilon^{-2}n)$ total space

better approximation-space tradeoff

better space complexity

- **Open Problems**
  - $(\alpha_{\text{BEST}} + \epsilon)$-approx., $\tilde{O}(n)$ **total space** for complete graphs?
  - Better-than-$O(\log |E^-|)$-approx. for general graphs?

Thank you!