

Exponential Quantum Space Advantage for Maximum Directed Cut in the Streaming Setting

Previous title: Exponential Quantum Streaming Advantage for Maximum Directed Cut

John Kallaugh



Sandia National
Laboratories

Ojas Parekh



Sandia National
Laboratories

Nadezhda Voronova



Boston University

Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.



192.168.0.1
10.0.0.2
172.16.20.5
192.0.2.123
203.0.113.5
172.16.0.3
203.0.113.4

Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.



10.0.0.2
172.16.20.5
192.0.2.123
203.0.113.5
172.16.0.3
203.0.113.4
127.0.0.1

Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.



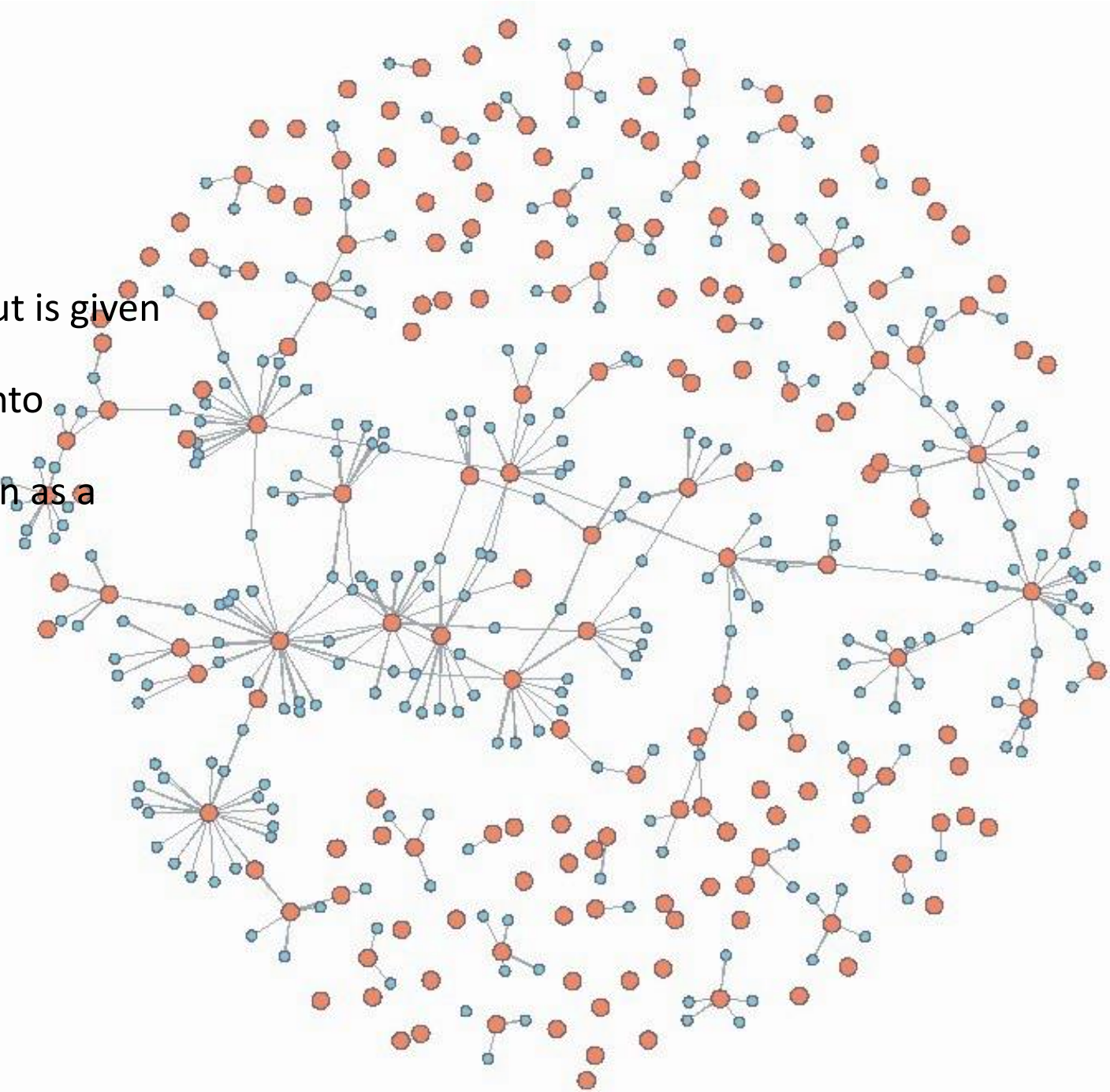
172.16.20.5
192.0.2.123
203.0.113.5
172.16.0.3
203.0.113.4
127.0.0.1
8.8.8.8

Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

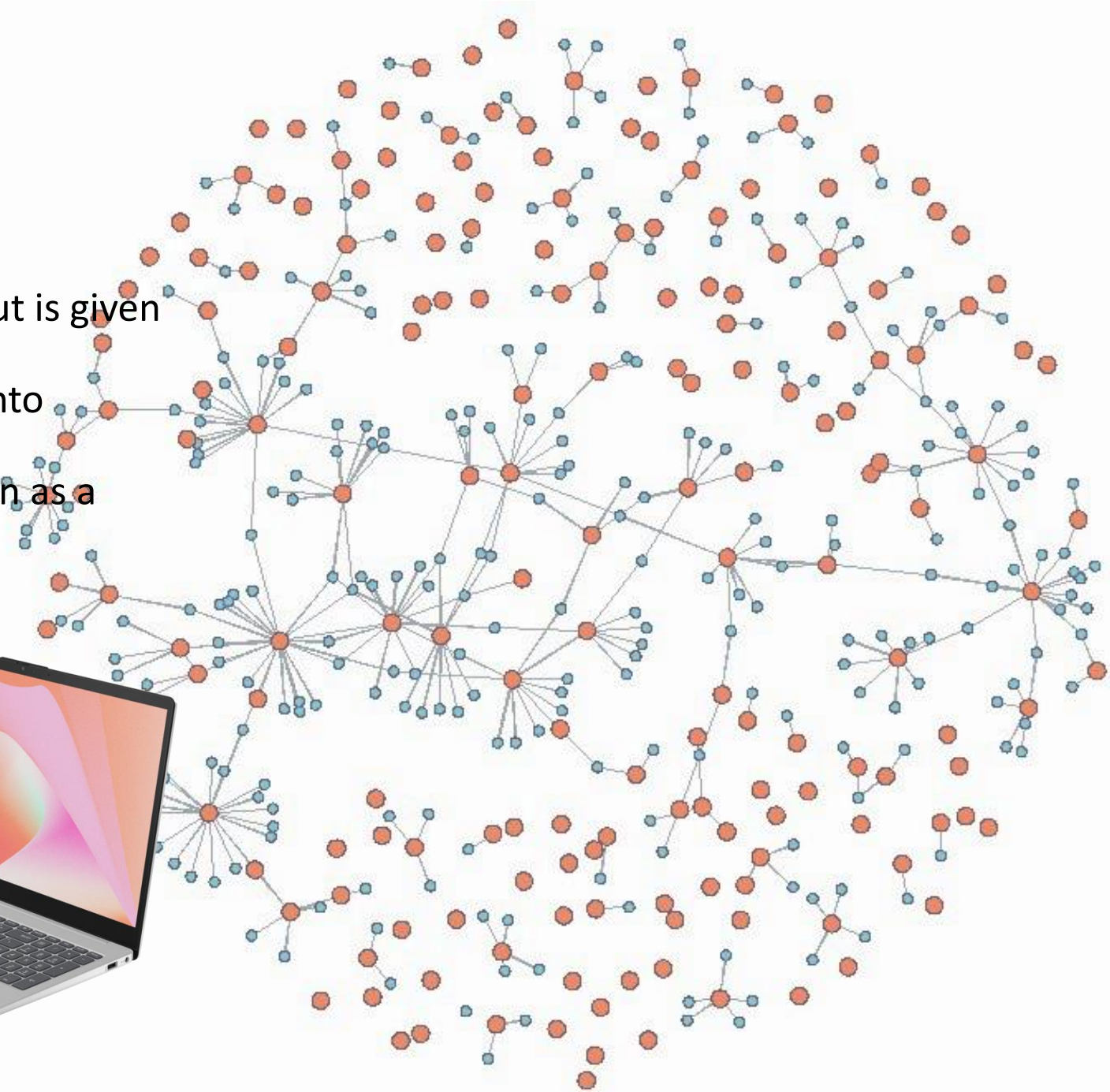


Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

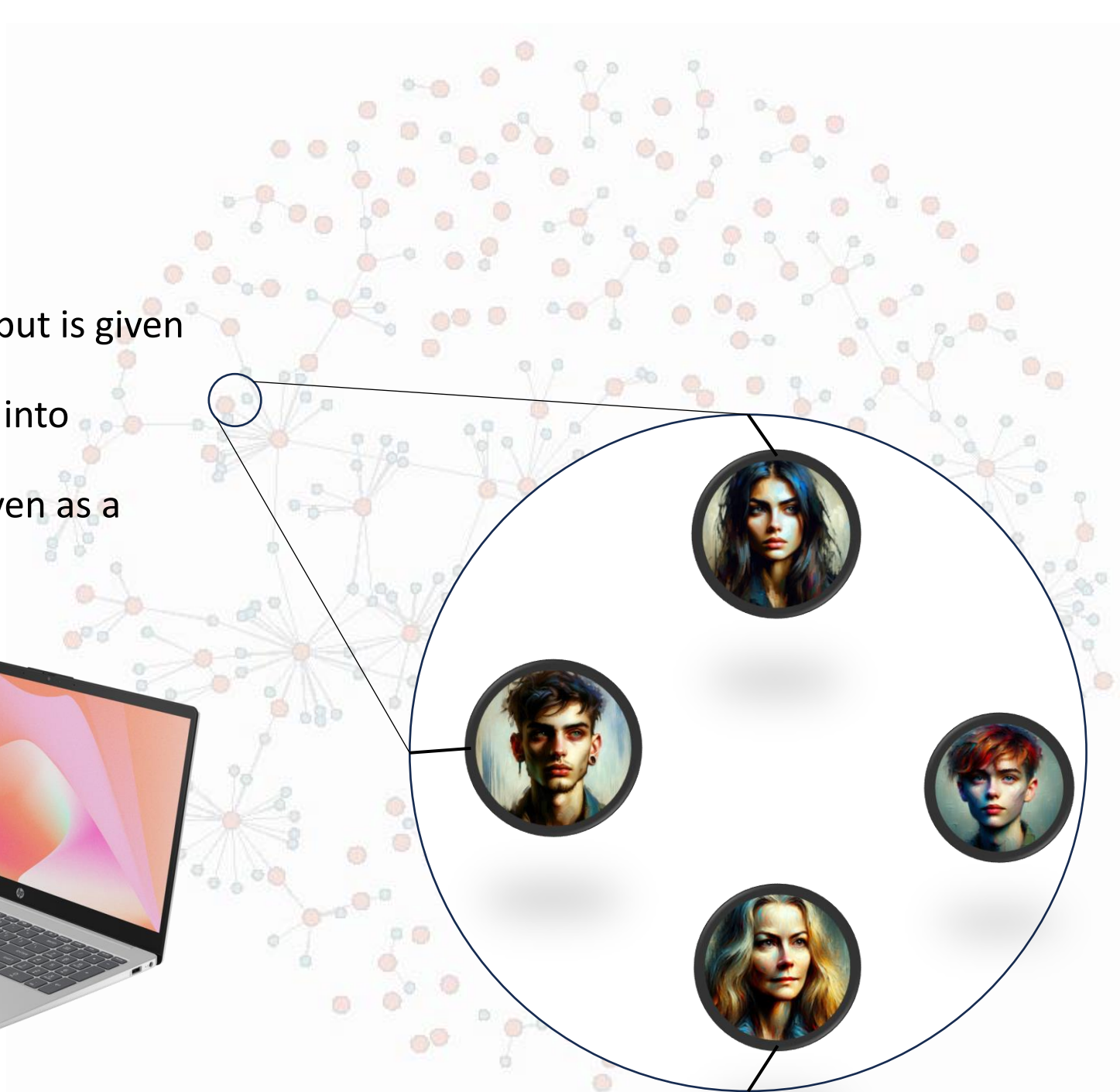


Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

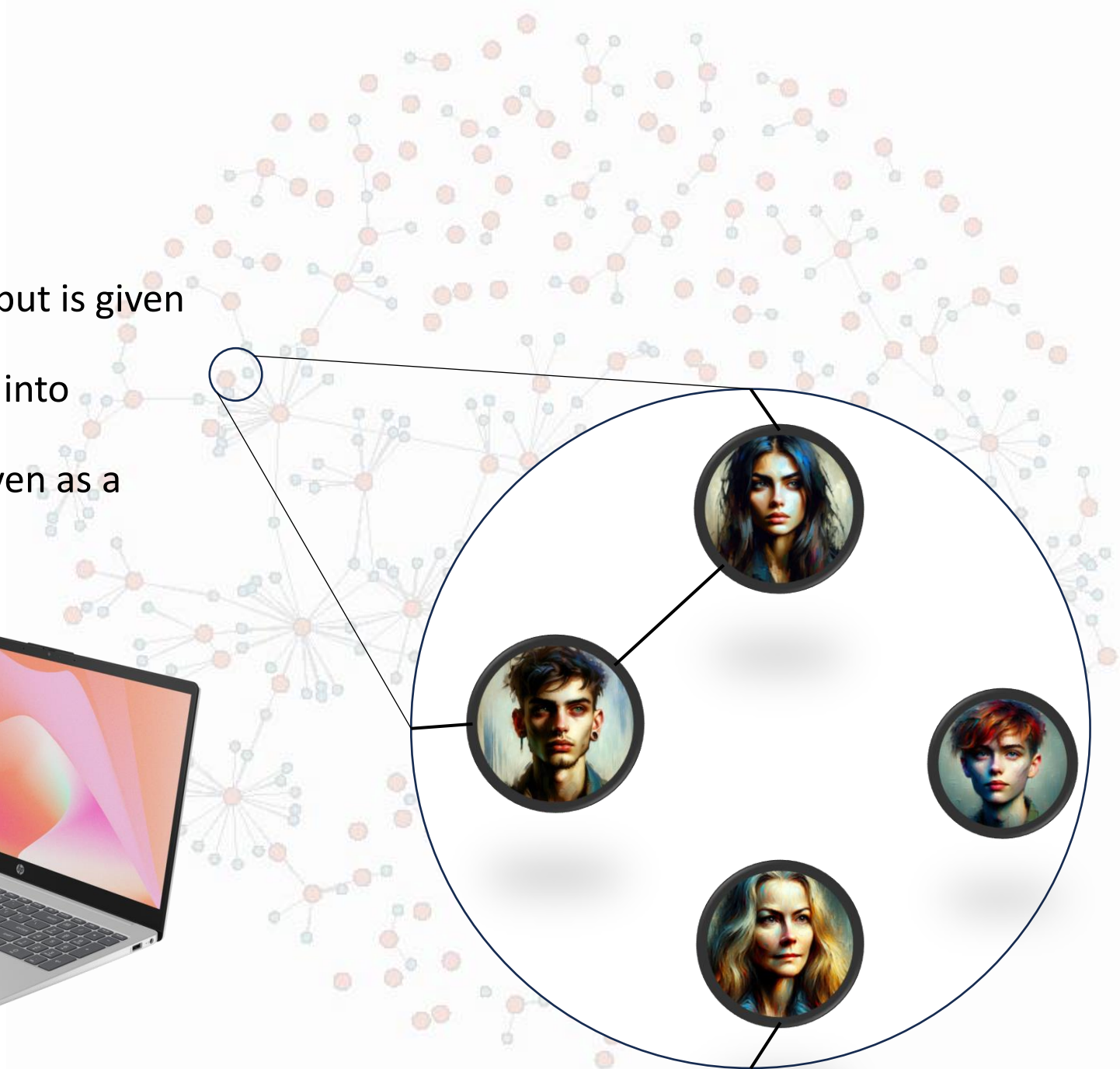


Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

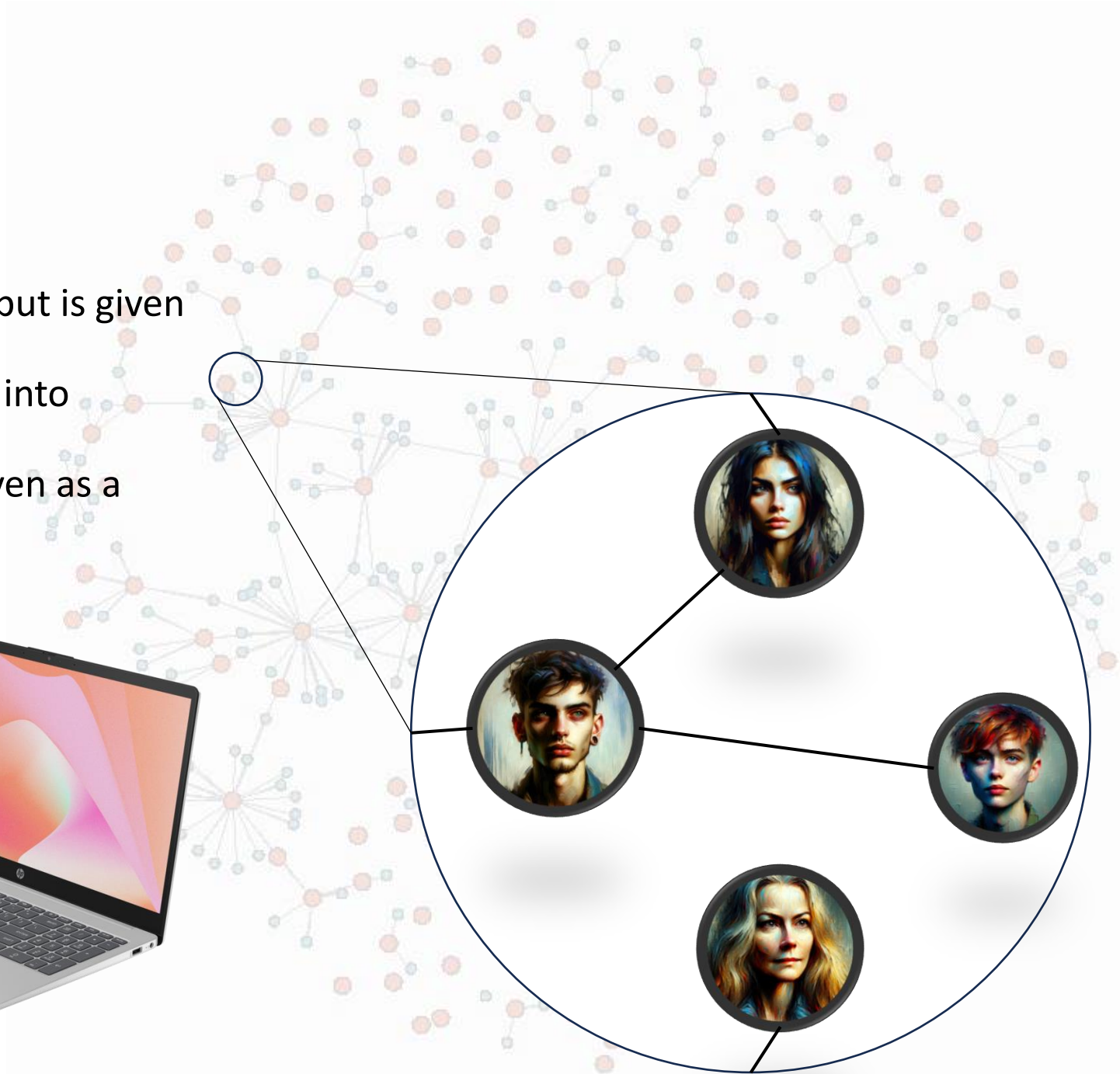


Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

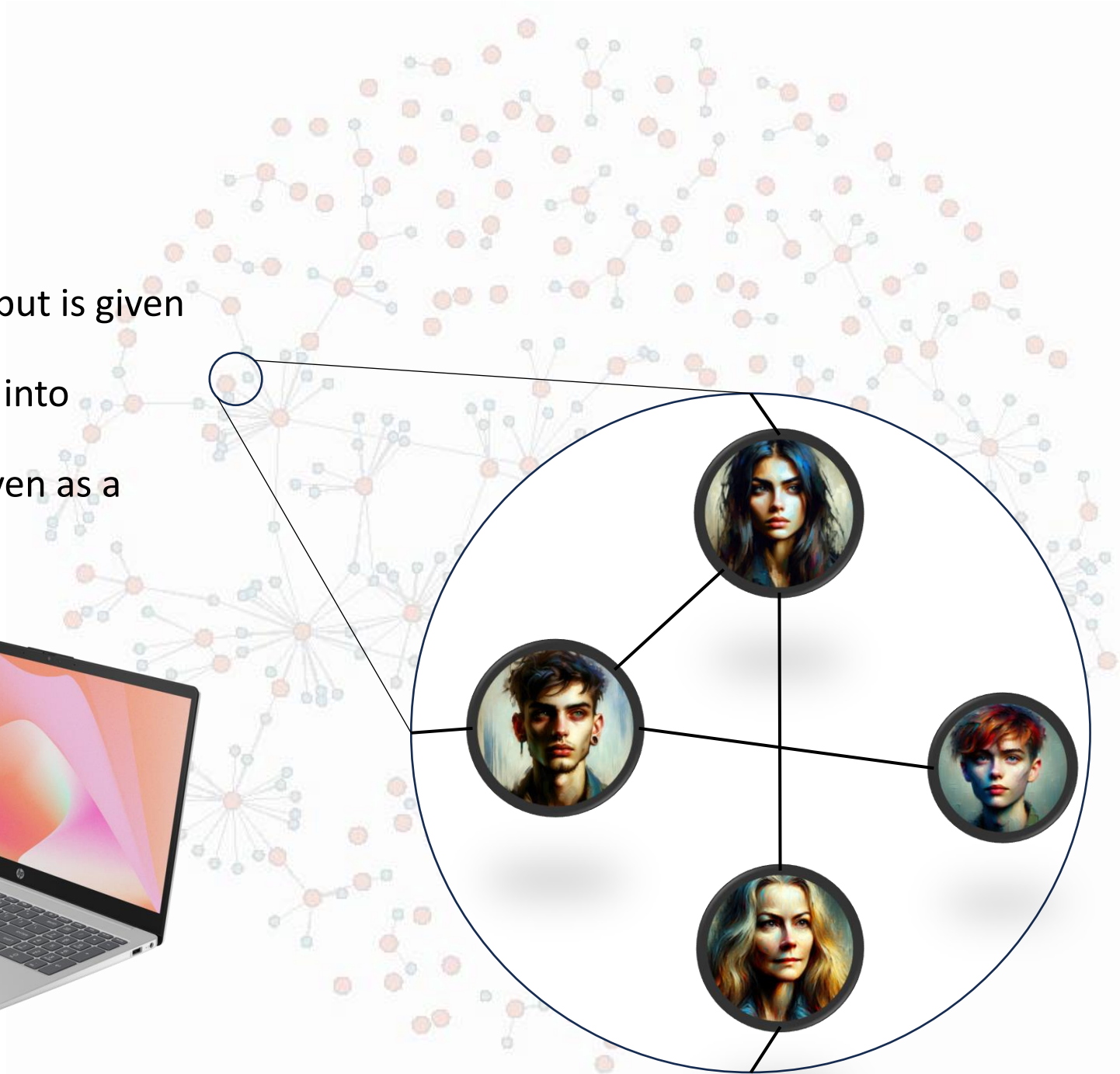


Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

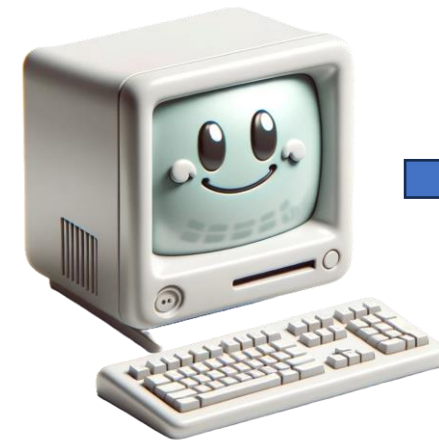


Streaming setting

Algorithms for *very large datasets* where input is given *one piece at a time*.

Very large datasets meaning too large to fit into memory.

One piece at a time meaning the input is given as a sequence of updates.

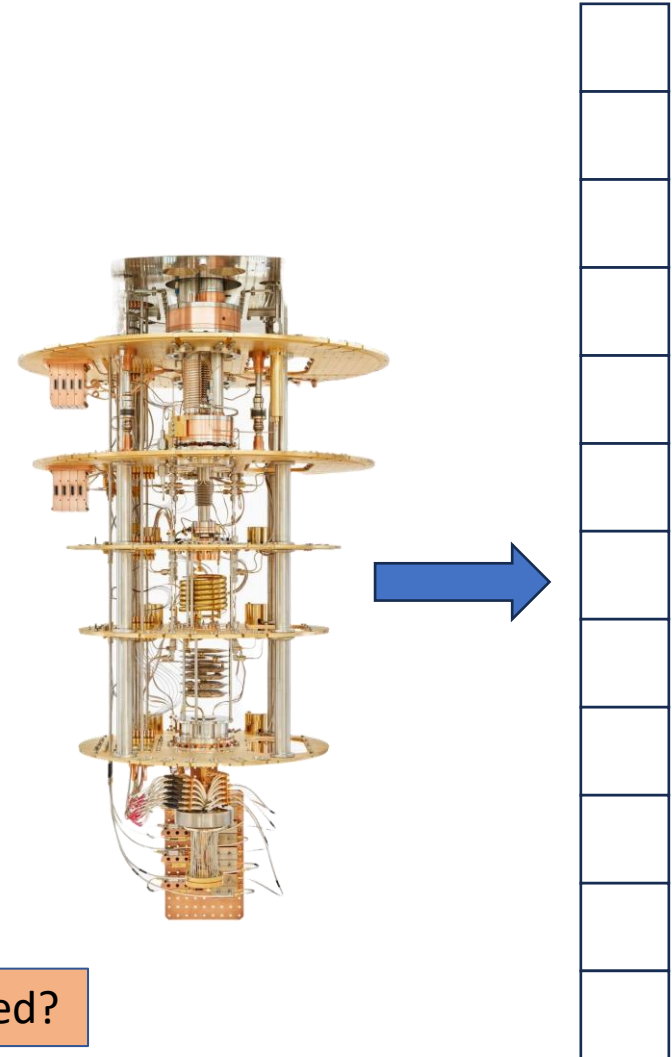


How much space does the computer need?

Goal: sublinear space, preferably polylogarithmic

Quantum streaming setting

Algorithm can store a quantum state.



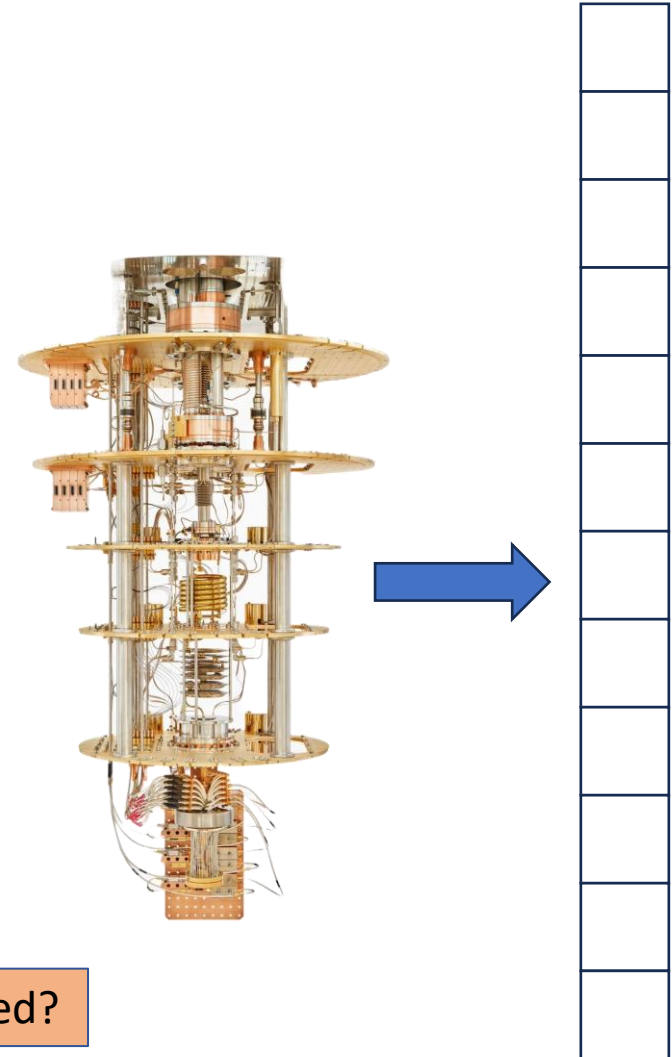
How many qubits does the computer need?

Goal: sublinear space, preferably polylogarithmic

Quantum streaming setting

Algorithm can store a quantum state.

Motivation:



How many qubits does the computer need?

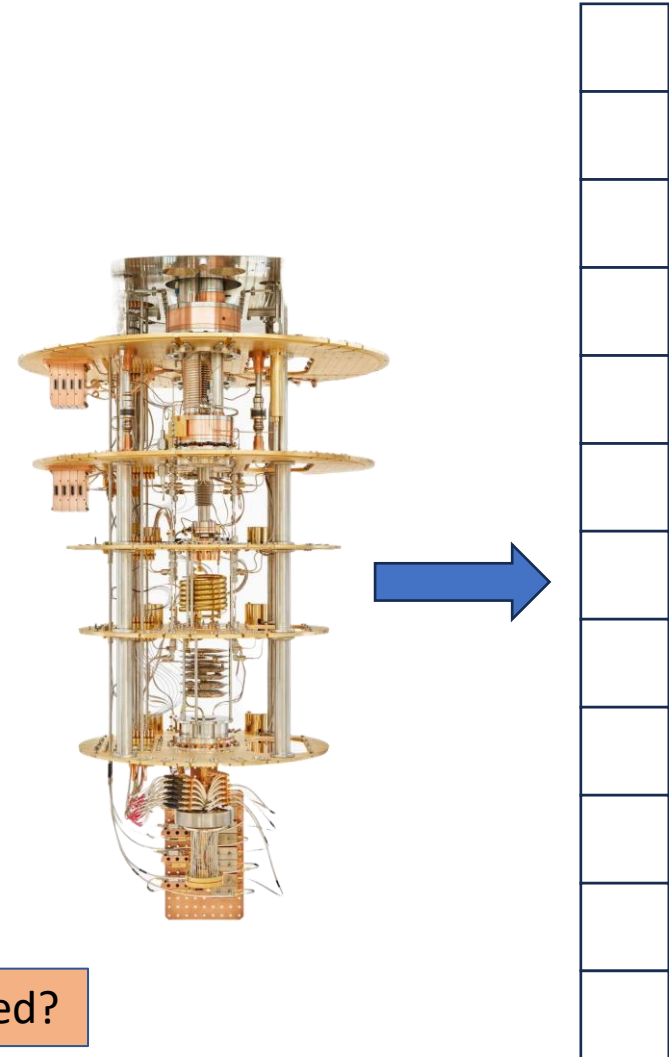
Goal: sublinear space, preferably polylogarithmic

Quantum streaming setting

Algorithm can store a quantum state.

Motivation:

- Optimizing quantum space usage



How many qubits does the computer need?

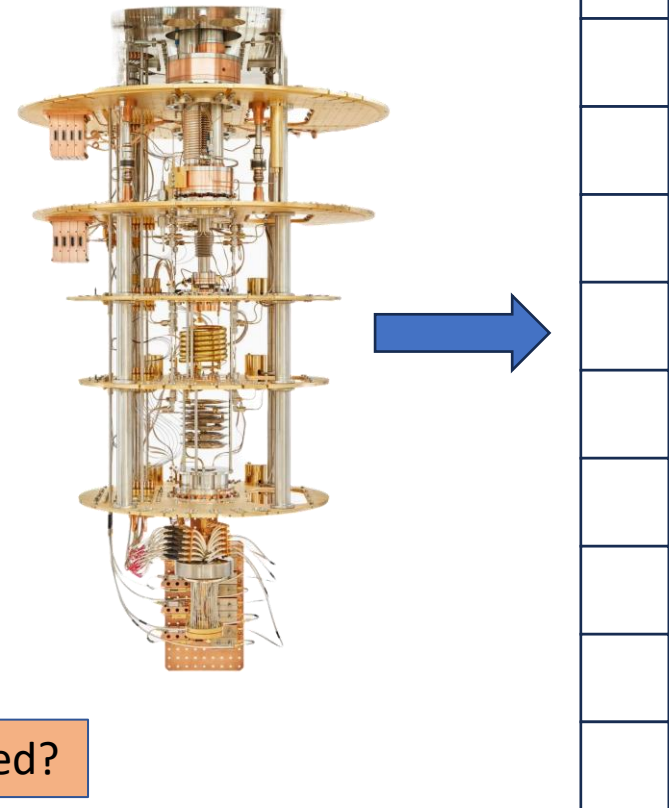
Goal: sublinear space, preferably polylogarithmic

Quantum streaming setting

Algorithm can store a quantum state.

Motivation:

- Optimizing quantum space usage
- Provable quantum advantage



How many qubits does the computer need?

Goal: sublinear space, preferably polylogarithmic

Exponential advantage for natural problem

Previous work	Natural problem	Exponential advantage	Problem

Exponential advantage for natural problem

Previous work	Natural problem	Exponential advantage	Problem
[Le Gall'06]			Variant of DISJ
[Gavinsky, Kempe, Kerenidis, Raz, de Wolf'08]			Boolean Hidden Matching

Exponential advantage for natural problem

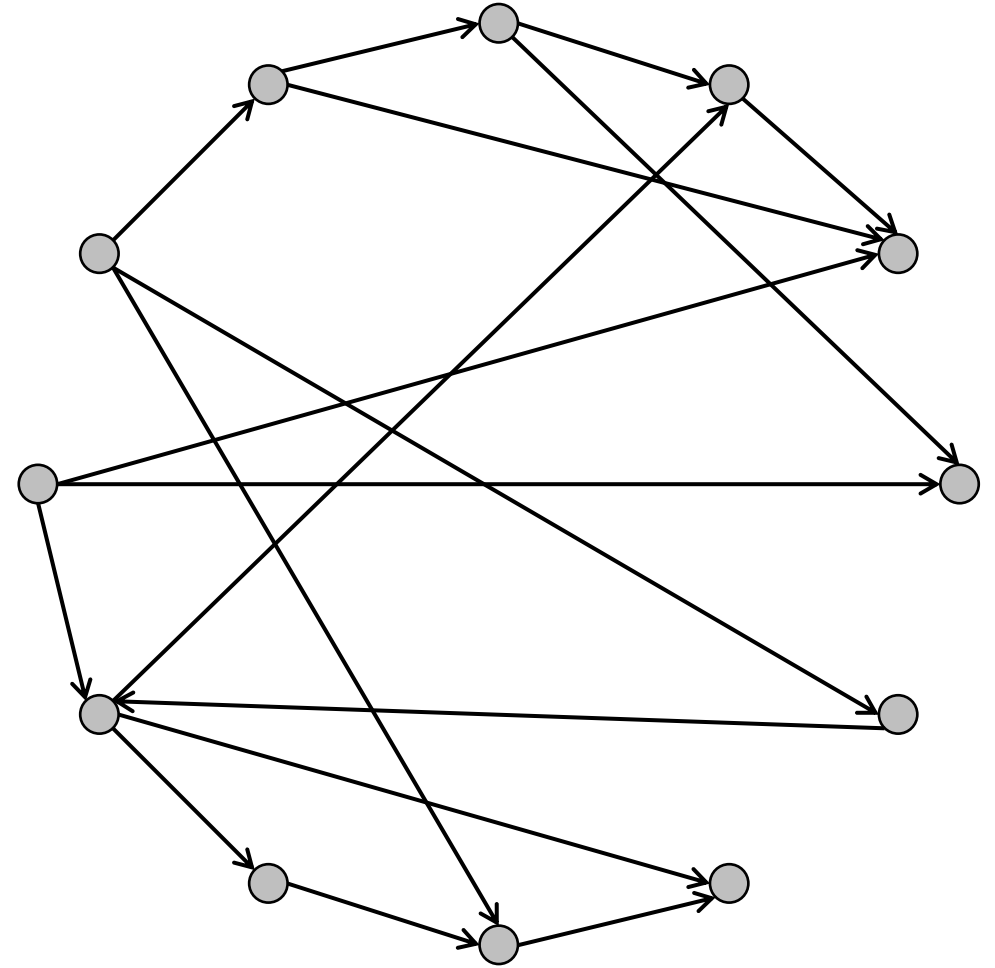
Previous work	Natural problem	Exponential advantage	Problem
[Le Gall'06]			Variant of DISJ
[Gavinsky, Kempe, Kerenidis, Raz, de Wolf'08]			Boolean Hidden Matching
[Montanaro'16]			Moment estimation Multipass setting
[Kallaugher'21]			Triangle counting

Exponential advantage for natural problem

Previous work	Natural problem	Exponential advantage	Problem
[Le Gall'06]			Variant of DISJ
[Gavinsky, Kempe, Kerenidis, Raz, de Wolf'08]			Boolean Hidden Matching
[Montanaro'16]			Moment estimation Multipass setting
[Kallaugher'21]			Triangle counting
This work			Maximum Directed Cut

Maximum Directed Cut

$G = (V, E)$ – directed graph on n vertices

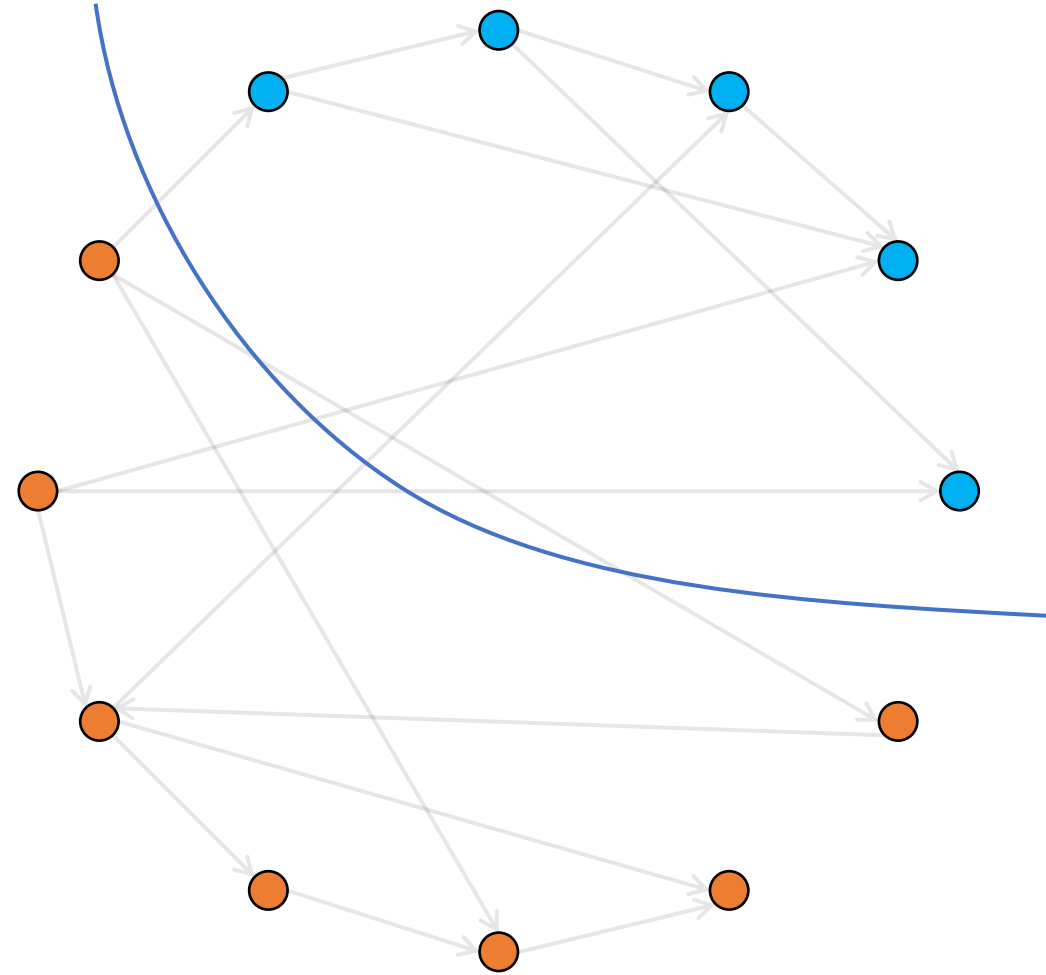


Maximum Directed Cut

$G = (V, E)$ – directed graph on n vertices

Maximum Directed cut: output the **value**

$$OPT = \max_{S \sqcup T = V} |\{(u, v) \in E \mid u \in S, v \in T\}|$$

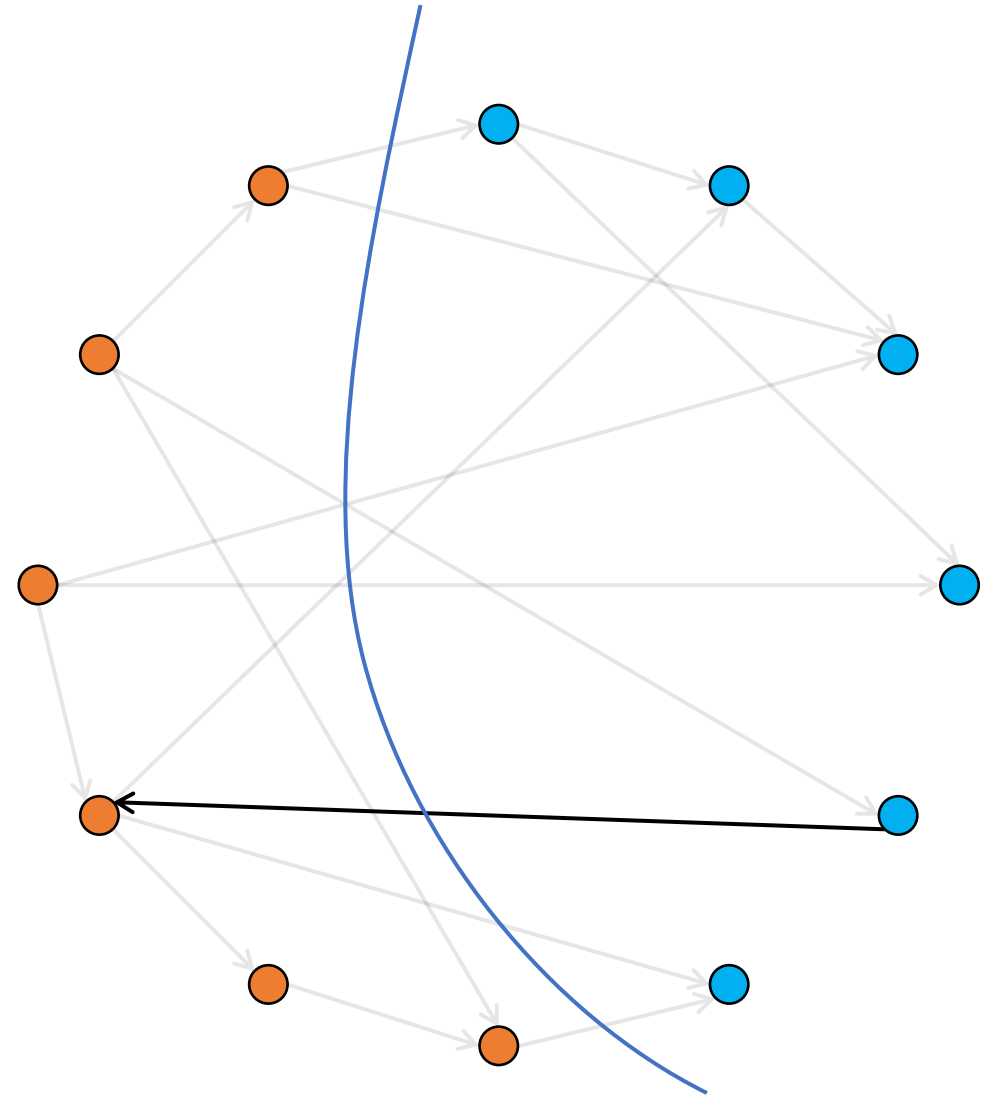


Maximum Directed Cut

$G = (V, E)$ – directed graph on n vertices

Maximum Directed cut: output the **value**

$$OPT = \max_{S \sqcup T = V} |\{(u, v) \in E \mid u \in S, v \in T\}|$$

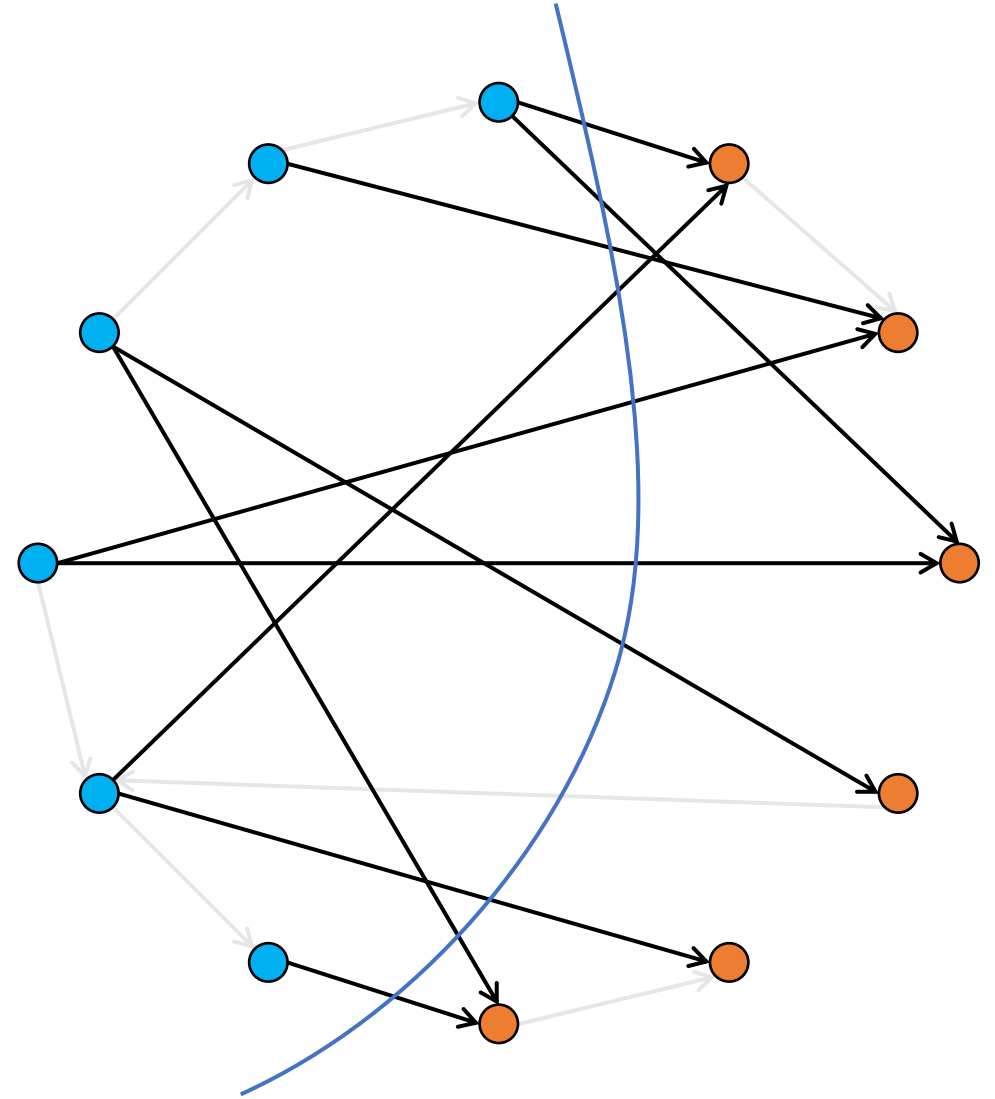


Maximum Directed Cut

$G = (V, E)$ – directed graph on n vertices

Maximum Directed cut: output the **value**

$$OPT = \max_{S \sqcup T = V} |\{(u, v) \in E \mid u \in S, v \in T\}|$$



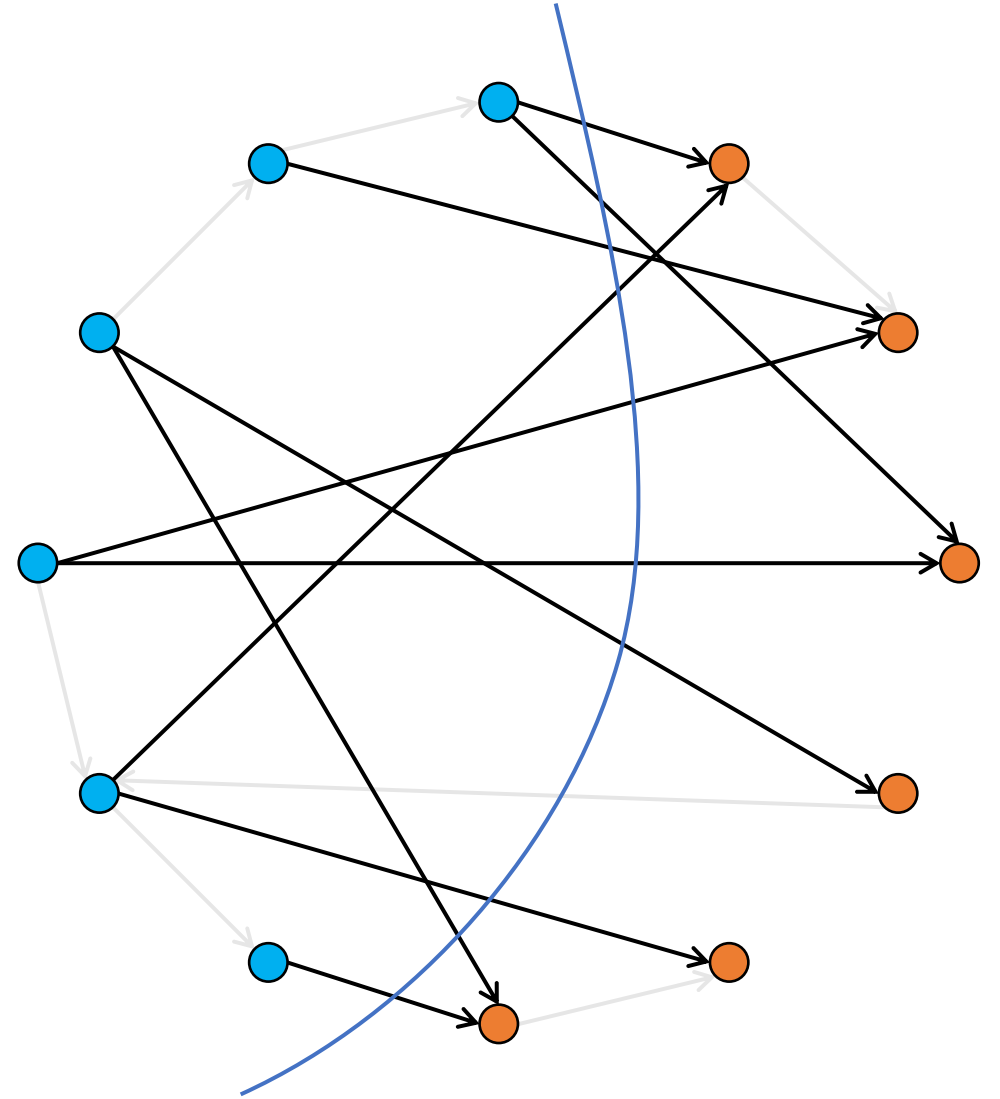
Maximum Directed Cut

$G = (V, E)$ – directed graph on n vertices

Maximum Directed cut: output the **value**

$$OPT = \max_{S \sqcup T = V} |\{(u, v) \in E \mid u \in S, v \in T\}|$$

- NP-hard



Maximum Directed Cut

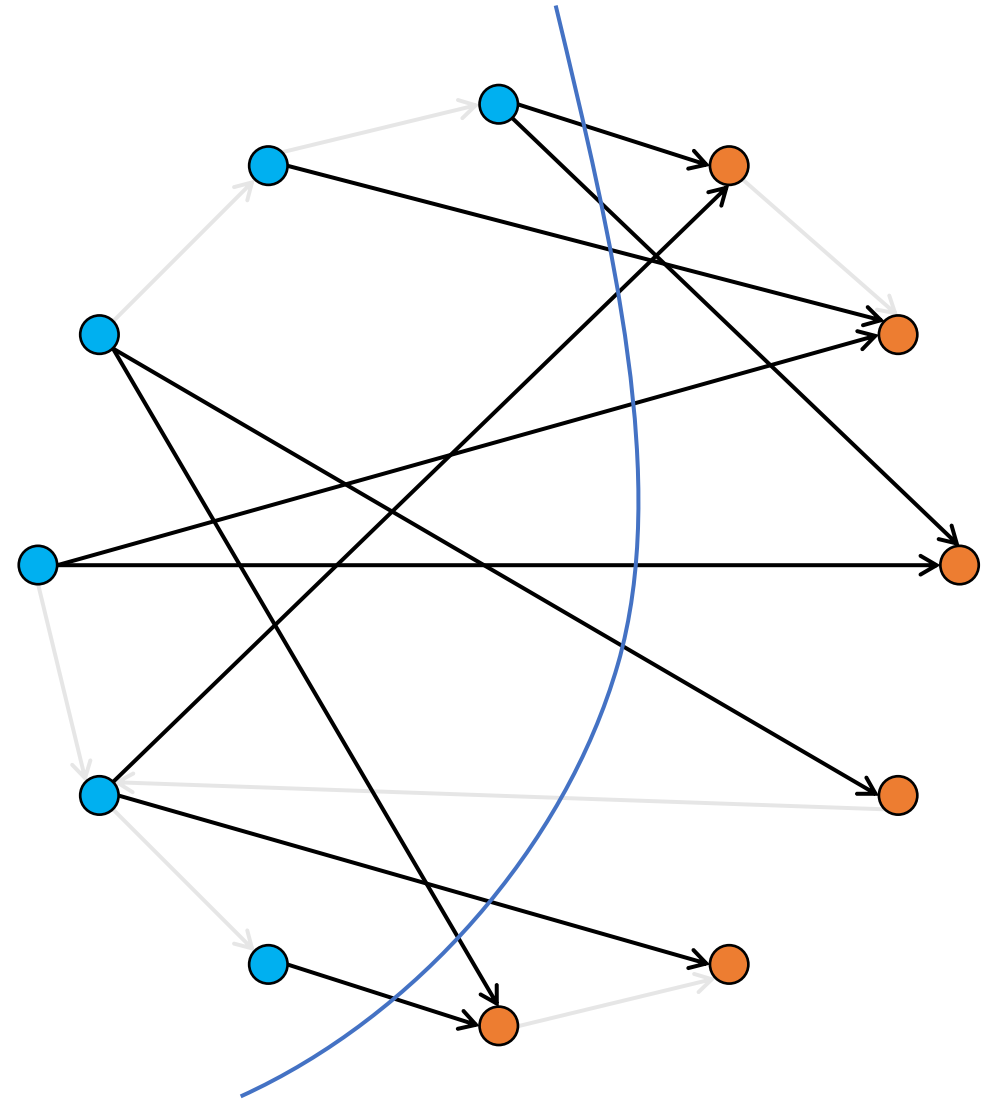
$G = (V, E)$ – directed graph on n vertices

Maximum Directed cut: output the **value**

$$OPT = \max_{S \sqcup T = V} |\{(u, v) \in E \mid u \in S, v \in T\}|$$

α -approximation of OPT is
 $X \in [\alpha \cdot OPT, OPT]$

- NP-hard



Maximum Directed Cut

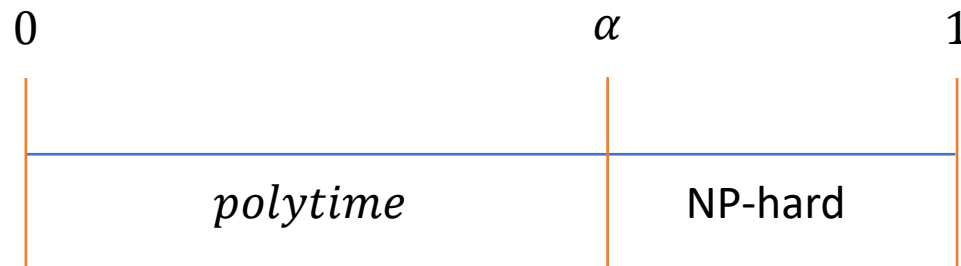
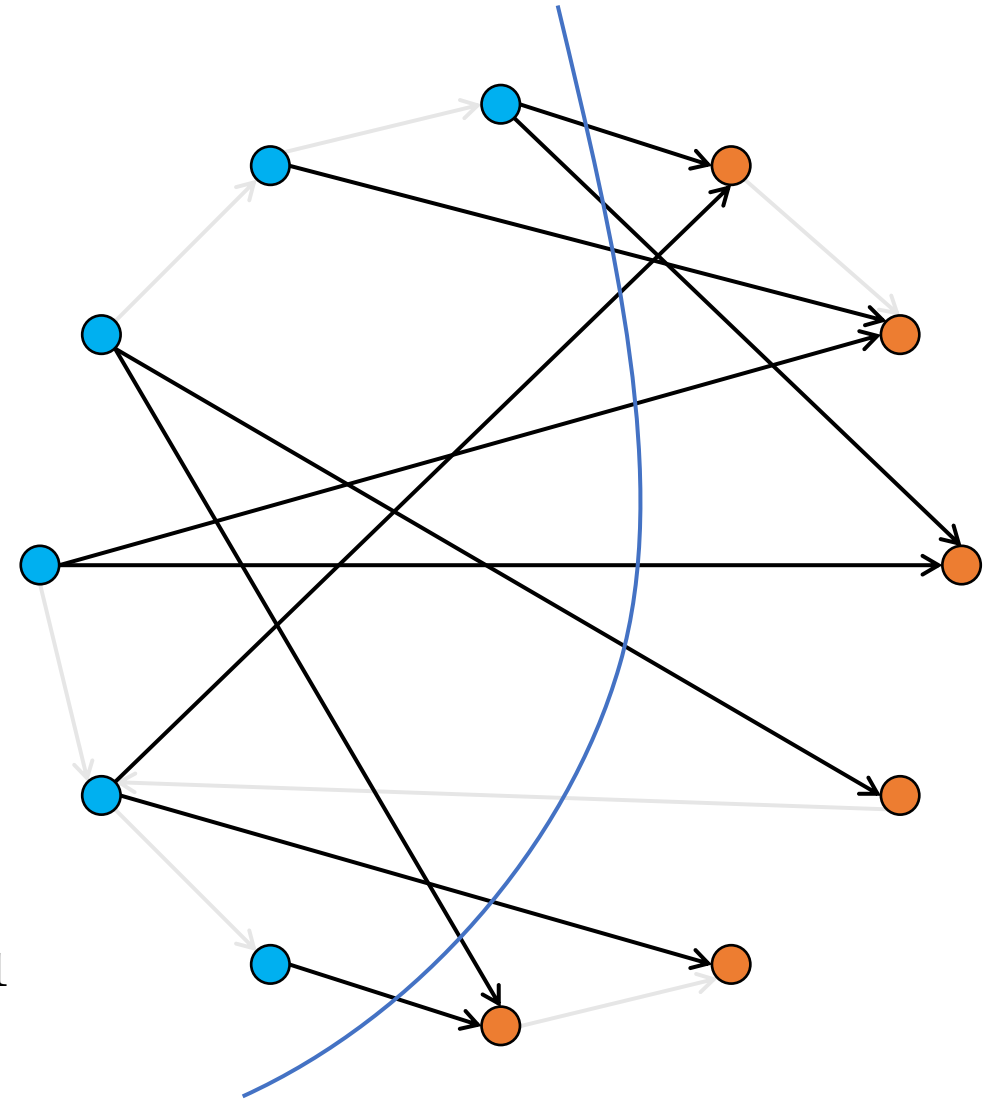
$G = (V, E)$ – directed graph on n vertices

Maximum Directed cut: output the **value**

$$OPT = \max_{S \sqcup T = V} |\{(u, v) \in E \mid u \in S, v \in T\}|$$

α -approximation of OPT is
 $X \in [\alpha \cdot OPT, OPT]$

- NP-hard
- Under UGC there exists α



Maximum Directed Cut

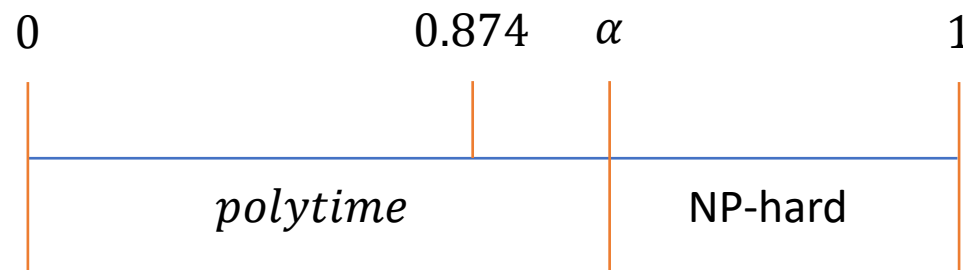
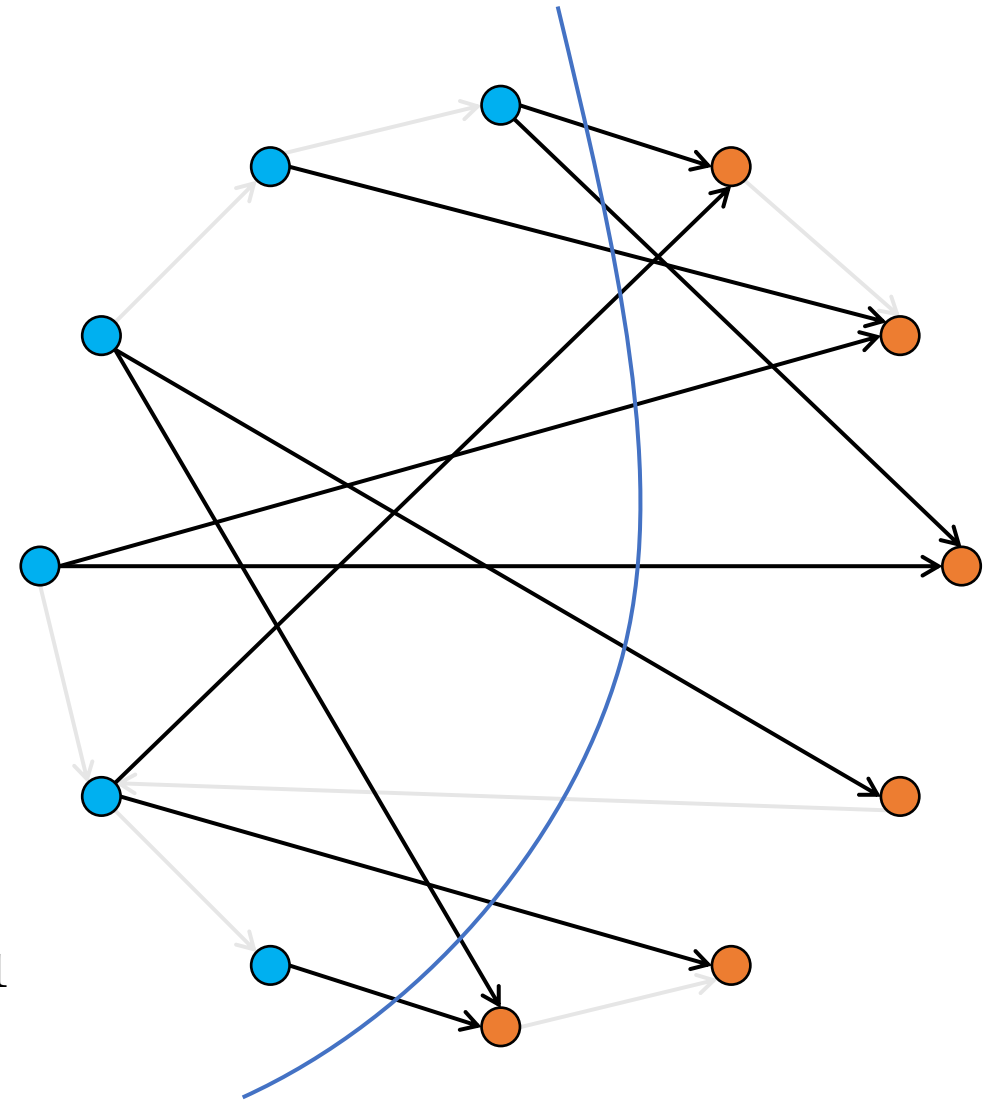
$G = (V, E)$ – directed graph on n vertices

Maximum Directed cut: output the **value**

$$OPT = \max_{S \sqcup T = V} |\{(u, v) \in E \mid u \in S, v \in T\}|$$

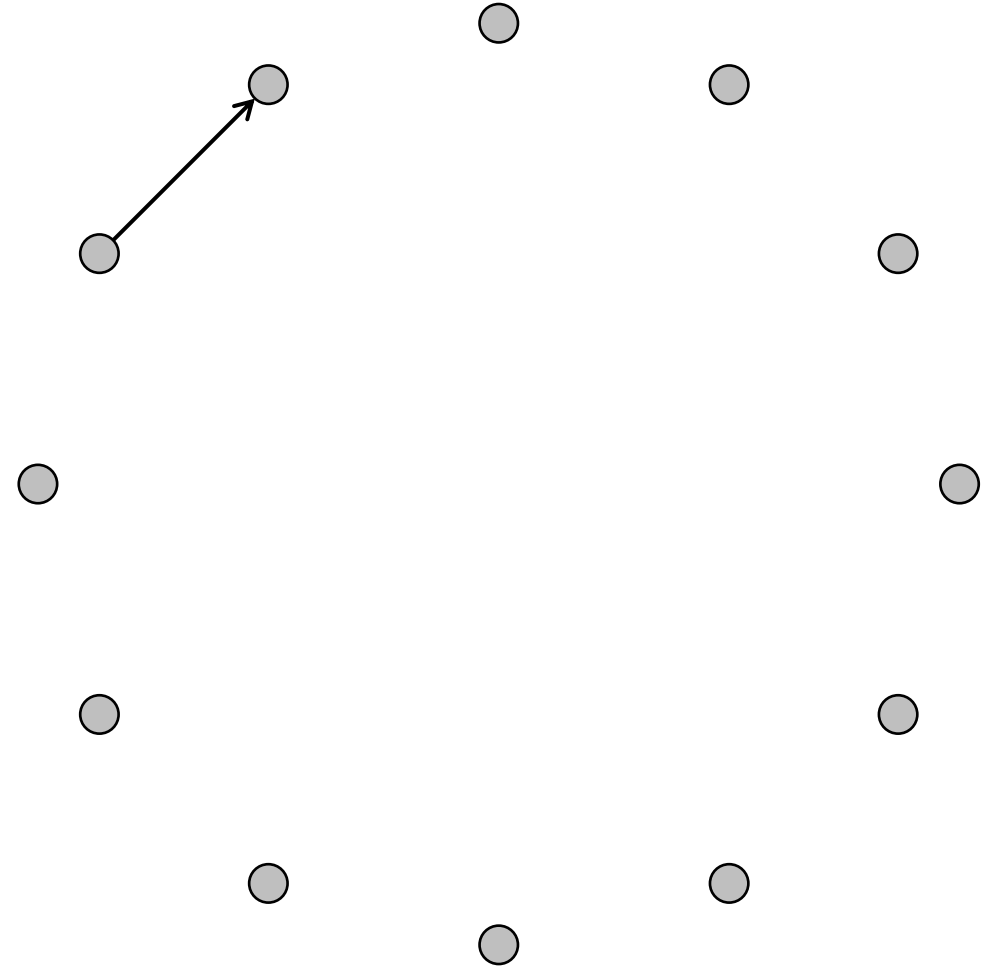
α -approximation of OPT is
 $X \in [\alpha \cdot OPT, OPT]$

- NP-hard
- Under UGC there exists α



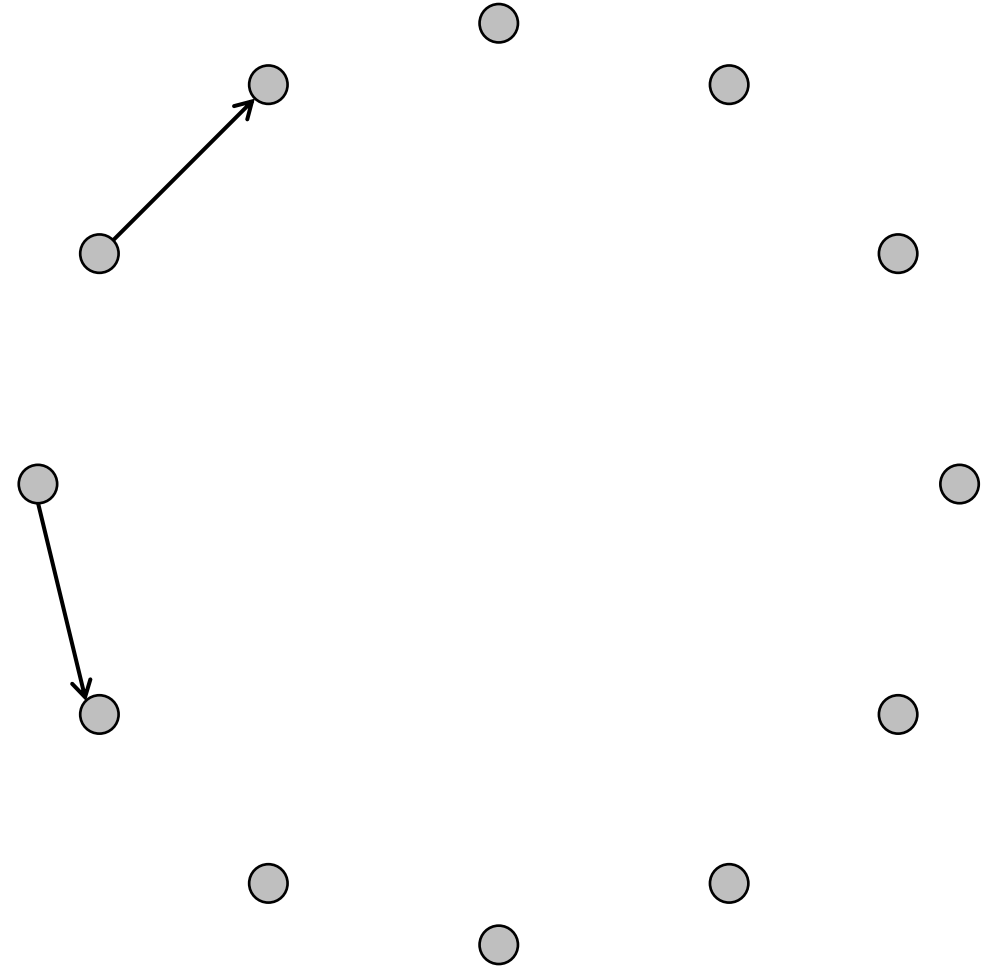
Max DiCut in streaming

$G = (V, E)$ – directed graph on n vertices



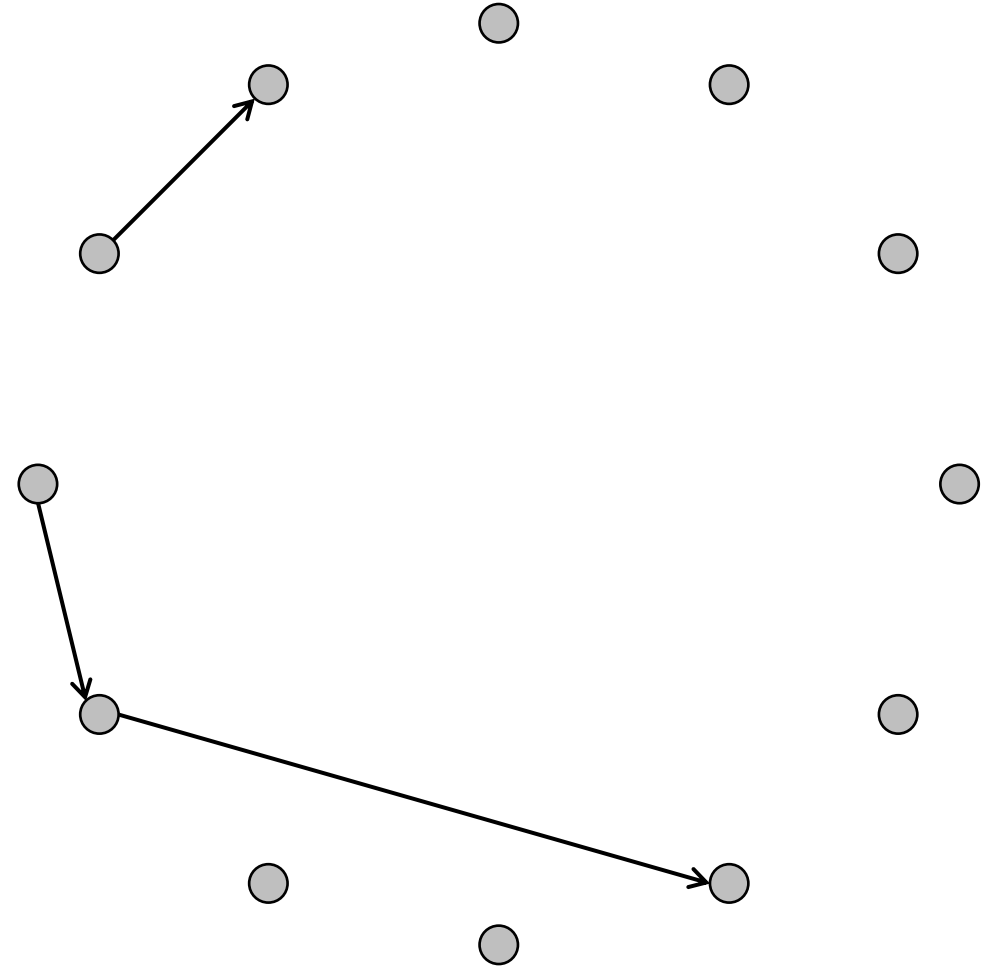
Max DiCut in streaming

$G = (V, E)$ – directed graph on n vertices



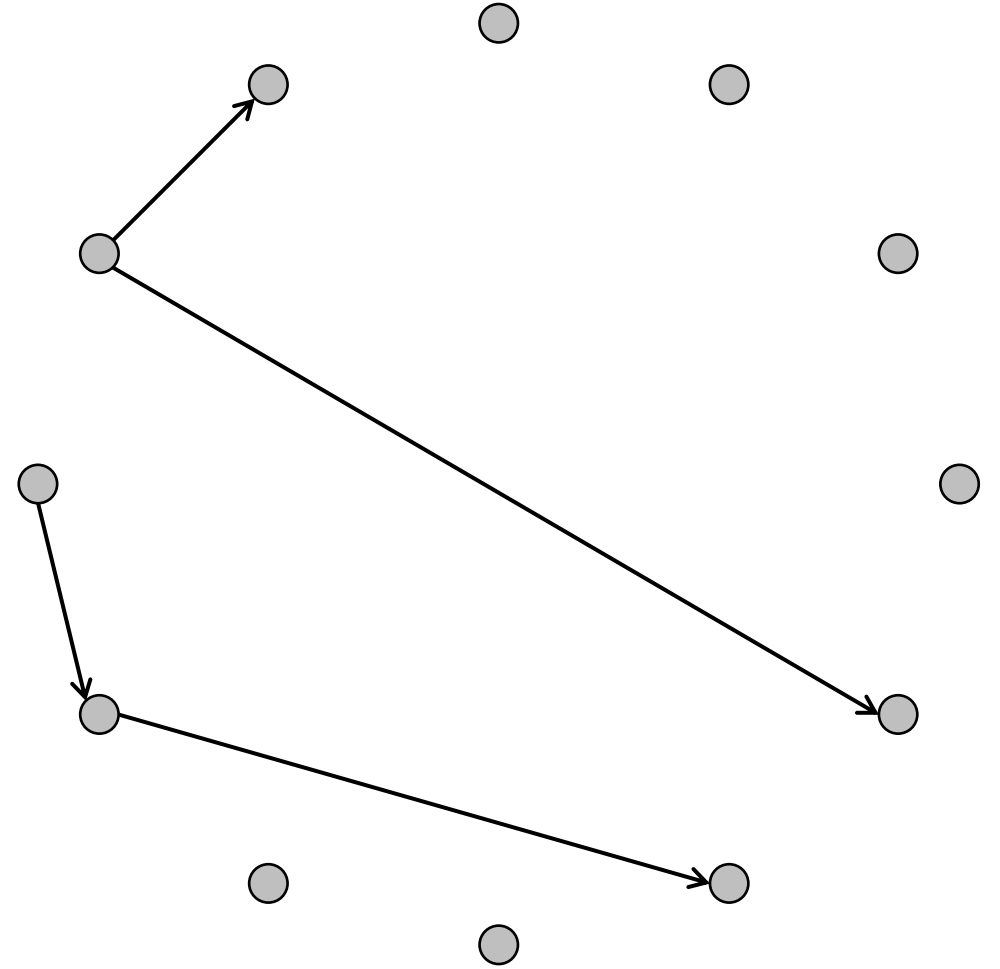
Max DiCut in streaming

$G = (V, E)$ – directed graph on n vertices



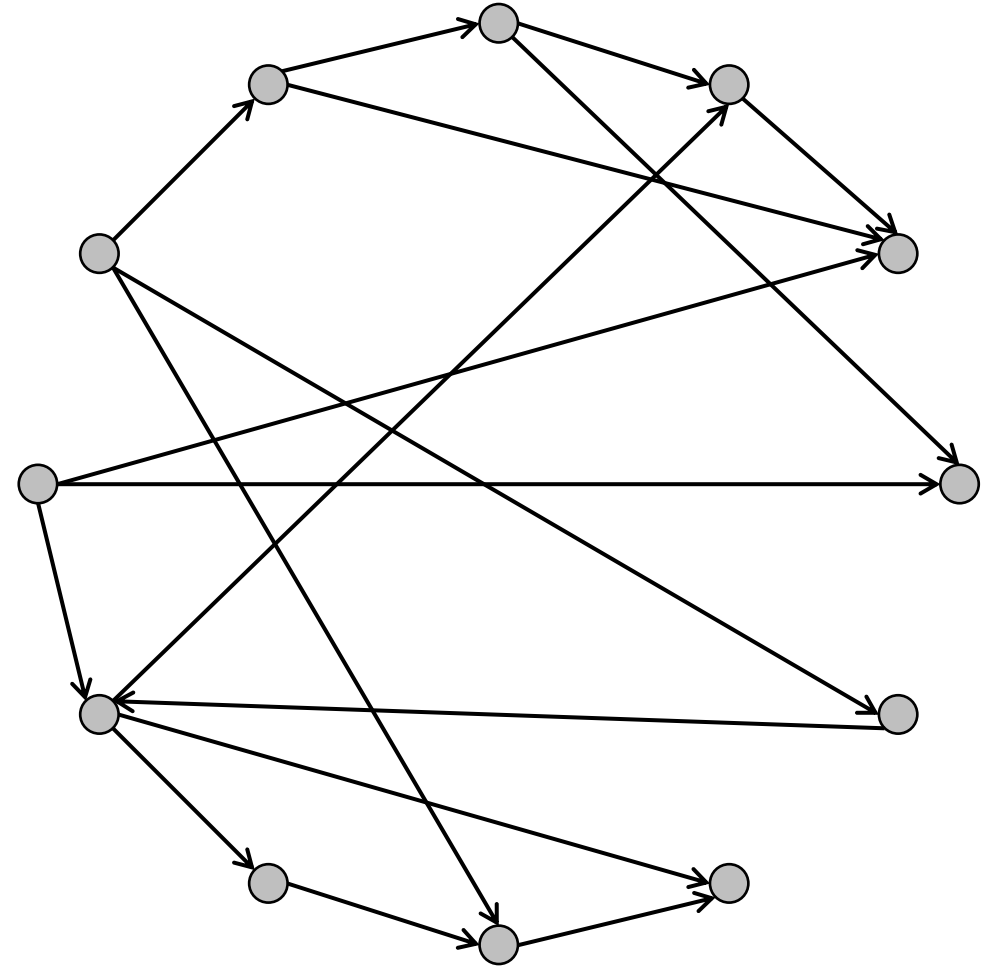
Max DiCut in streaming

$G = (V, E)$ – directed graph on n vertices



Max DiCut in streaming

$G = (V, E)$ – directed graph on n vertices

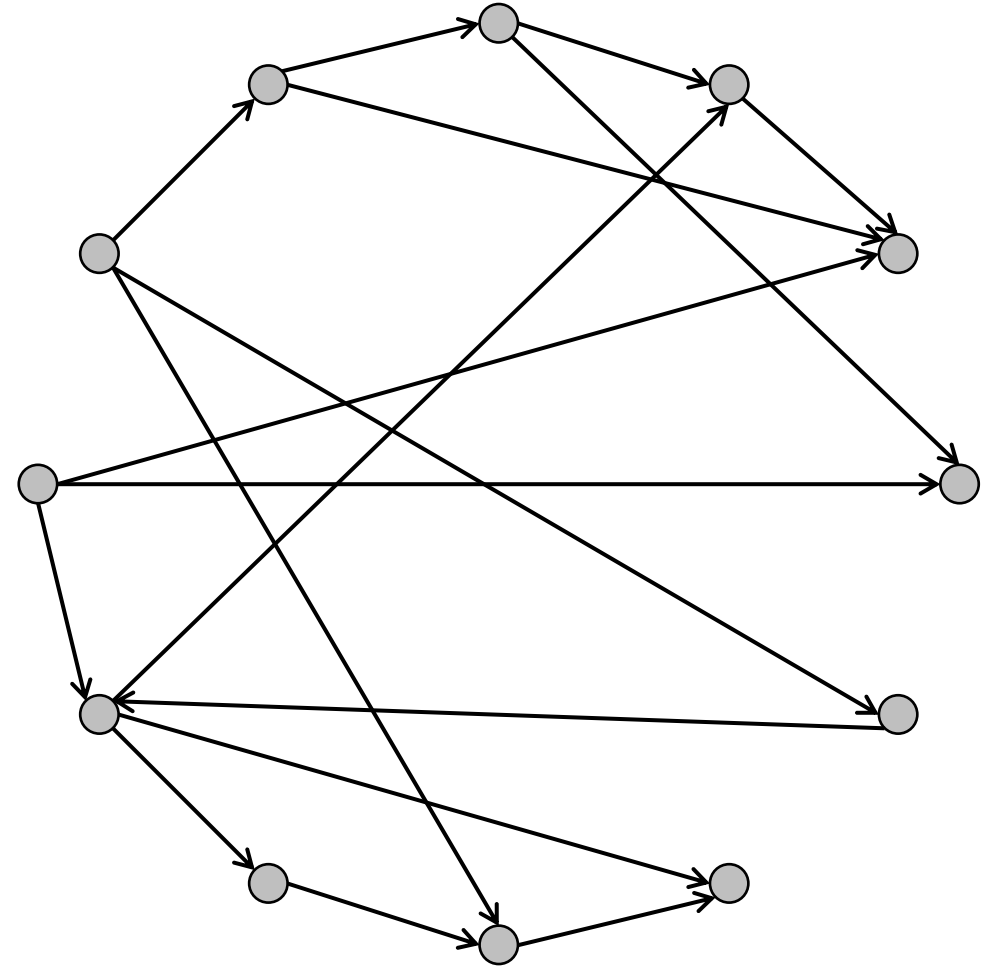


Max DiCut in streaming

$G = (V, E)$ – directed graph on n vertices

How much space is needed for
Maximum Directed Cut approximation?

Goal: sublinear space, preferably $\text{polylog}(n)$



Max DiCut in streaming

polylog space

polynomial space



Max DiCut in streaming

- $1/4$ is trivial

polylog space

polynomial space

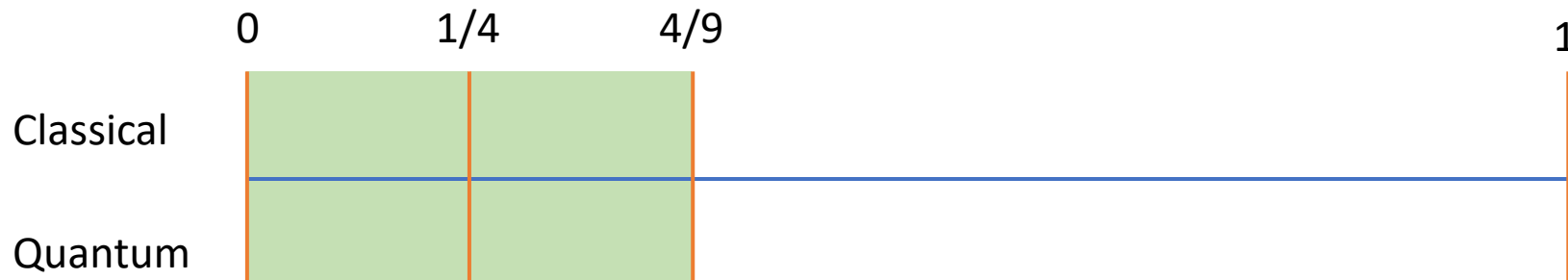


Max DiCut in streaming

- $1/4$ is trivial
- [CGV'20] $4/9$ is achievable in $\text{polylog}(n)$ classical bits

polylog space

polynomial space

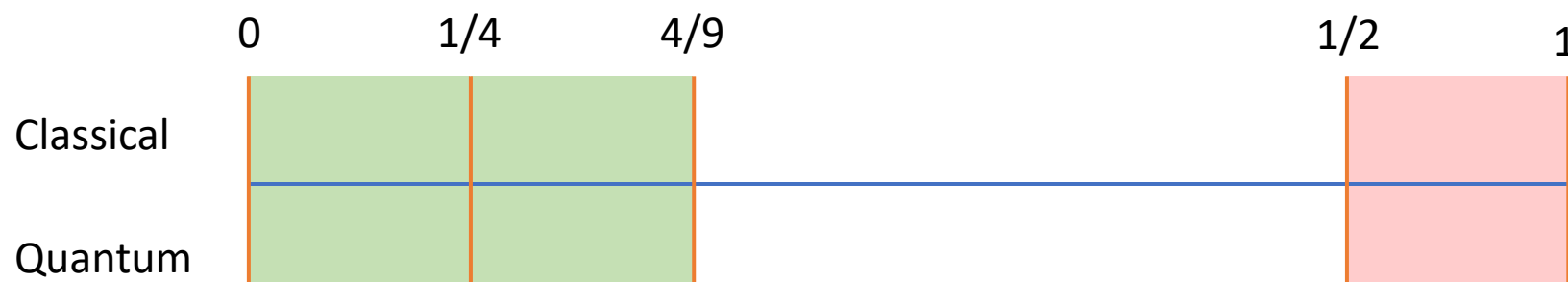


Max DiCut in streaming

- $1/4$ is trivial
- [CGV'20] $4/9$ is achievable in $\text{polylog}(n)$ classical bits
- [KK'18][KP'23] Better than $1/2$ requires $\Omega(n)$ qubits

polylog space

polynomial space

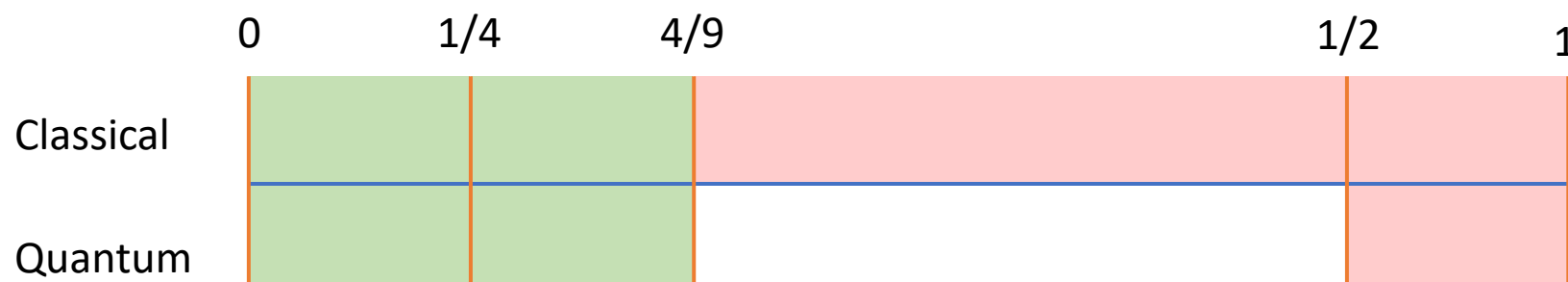


Max DiCut in streaming

- $1/4$ is trivial
- [CGV'20] $4/9$ is achievable in $\text{polylog}(n)$ classical bits
- [KK'18][KP'23] Better than $1/2$ requires $\Omega(n)$ qubits
- [CGV'20] Better than $4/9$ requires $\Omega(\sqrt{n})$ classical bits

polylog space

polynomial space

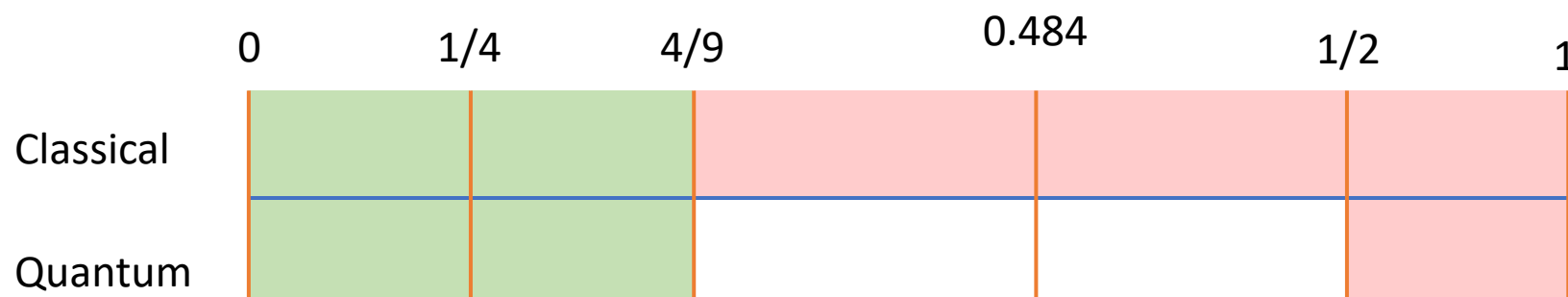


Max DiCut in streaming

- $1/4$ is trivial
- [CGV'20] $4/9$ is achievable in $\text{polylog}(n)$ classical bits
- [KK'18][KP'23] Better than $1/2$ requires $\Omega(n)$ qubits
- [CGV'20] Better than $4/9$ requires $\Omega(\sqrt{n})$ classical bits
- [SSSV'23] 0.484 is achievable in $\tilde{O}(\sqrt{n})$ classical bits

polylog space

polynomial space

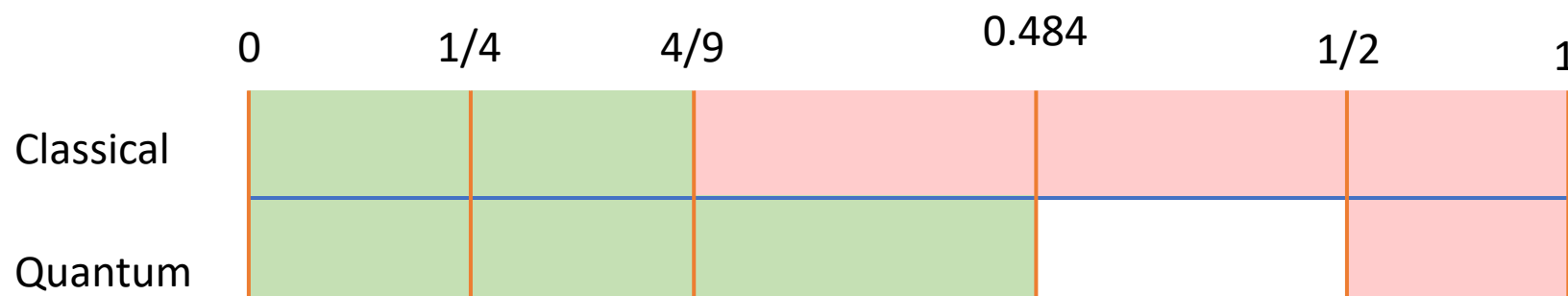


Max DiCut in streaming

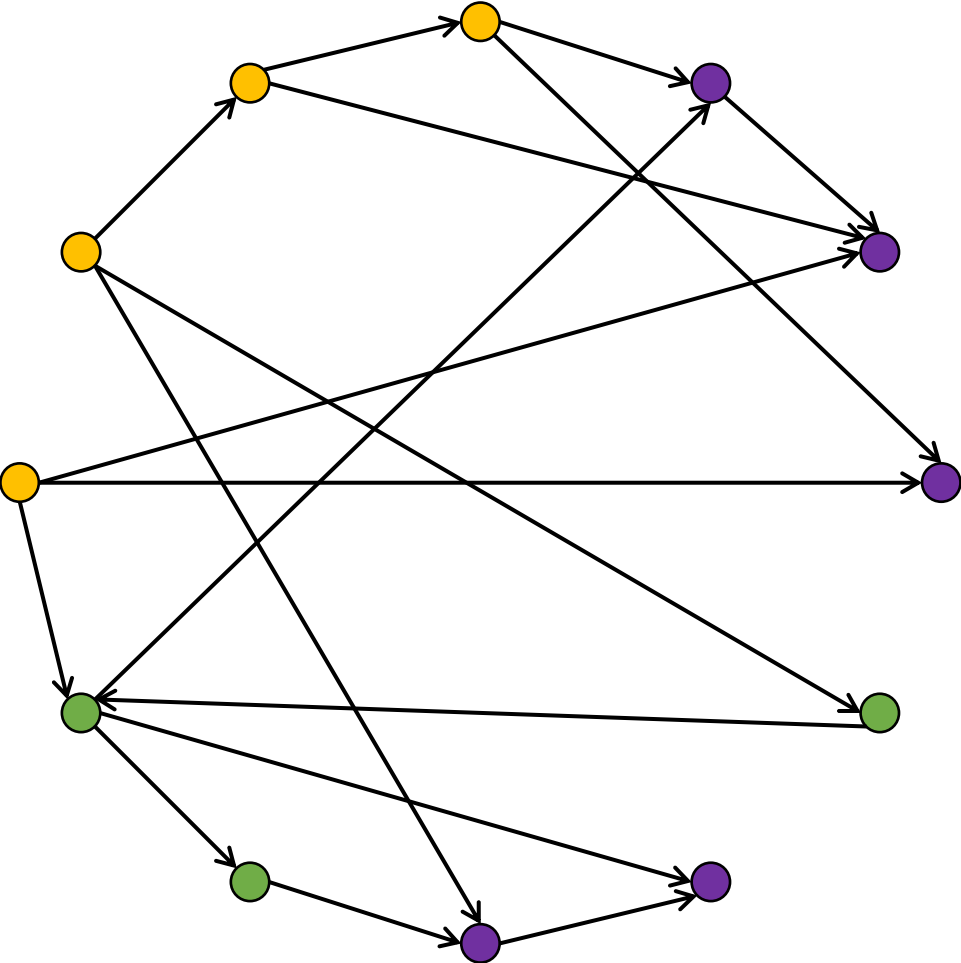
- $1/4$ is trivial
- [CGV'20] $4/9$ is achievable in $\text{polylog}(n)$ classical bits
- [KK'18][KP'23] Better than $1/2$ requires $\Omega(n)$ qubits
- [CGV'20] Better than $4/9$ requires $\Omega(\sqrt{n})$ classical bits
- [SSSV'23] 0.484 is achievable in $\tilde{O}(\sqrt{n})$ classical bits
- **[This work]** 0.484 is achievable in $\text{polylog}(n)$ qubits

polylog space

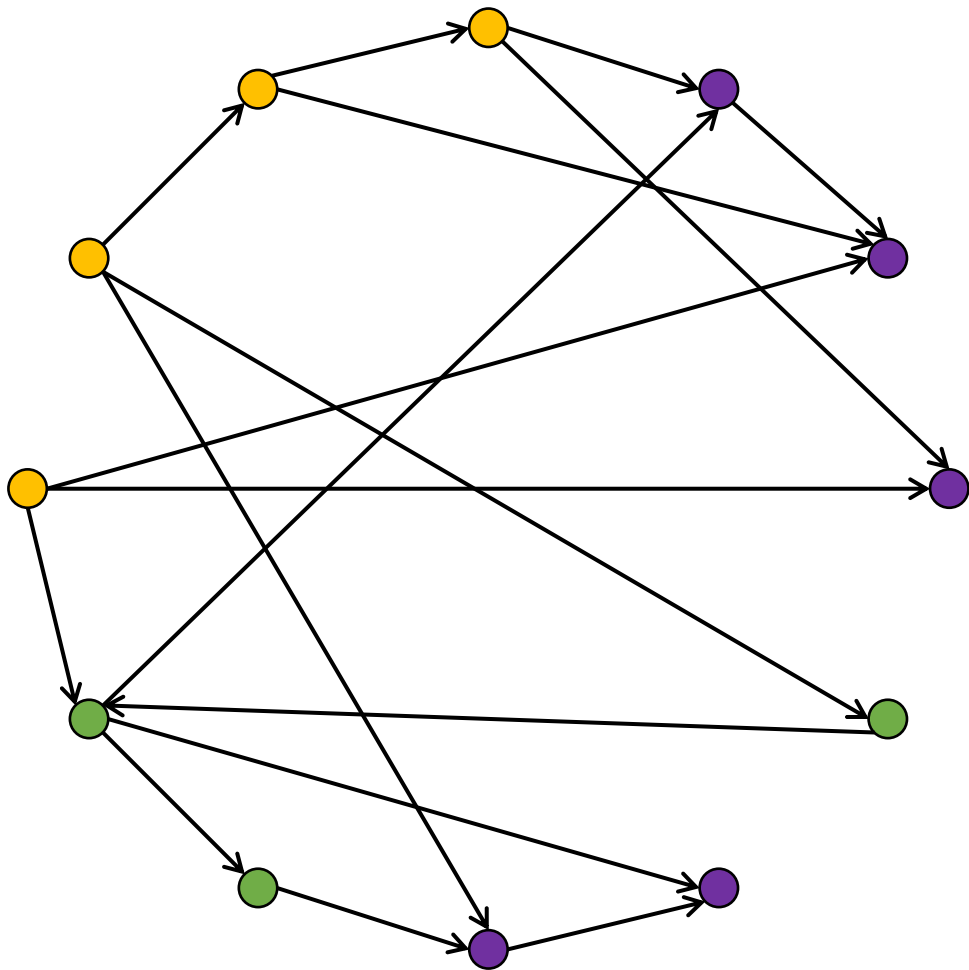
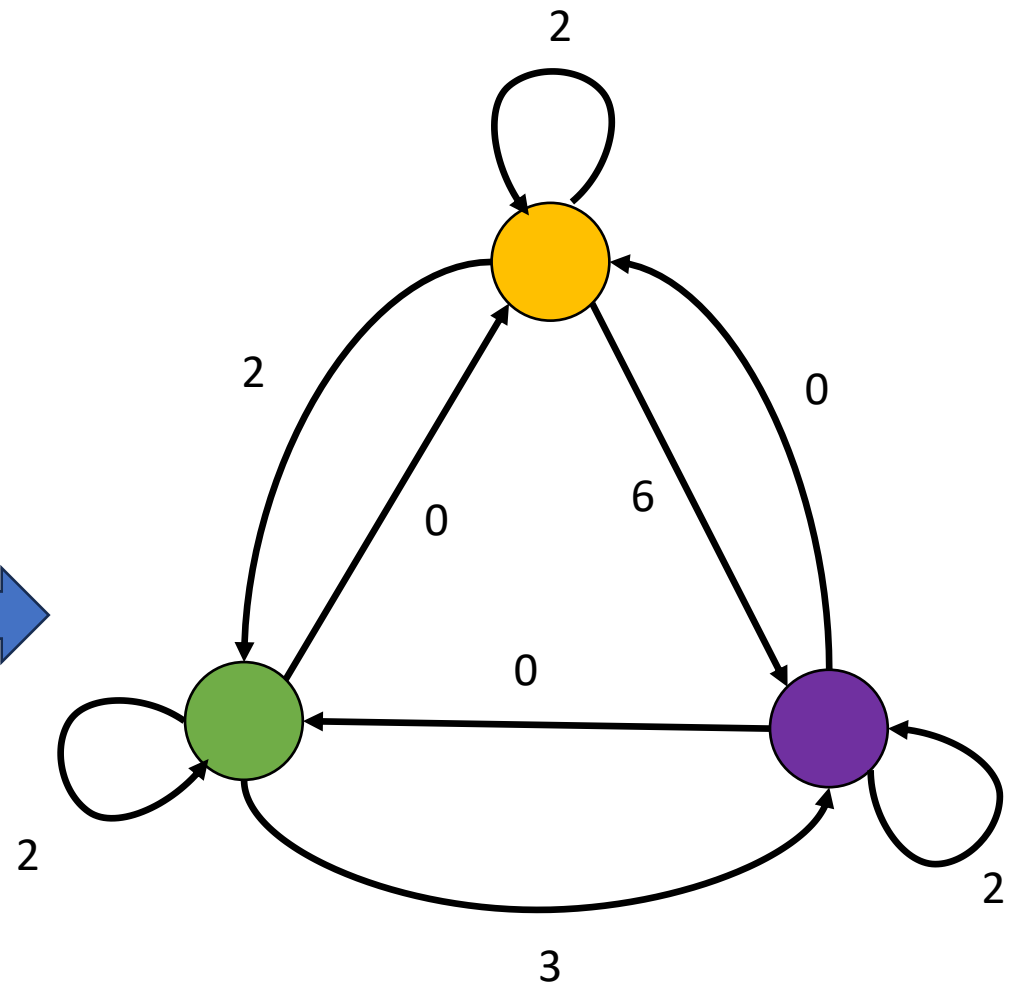
polynomial space



Snapshot



Snapshot



Snapshot

$$b_v = \frac{d^{out} - d^{in}}{d}$$

Snapshot

$$b_v = \frac{d^{out} - d^{in}}{d}$$

$$b_v \in \overset{-1}{} \overset{1}{}$$


Snapshot

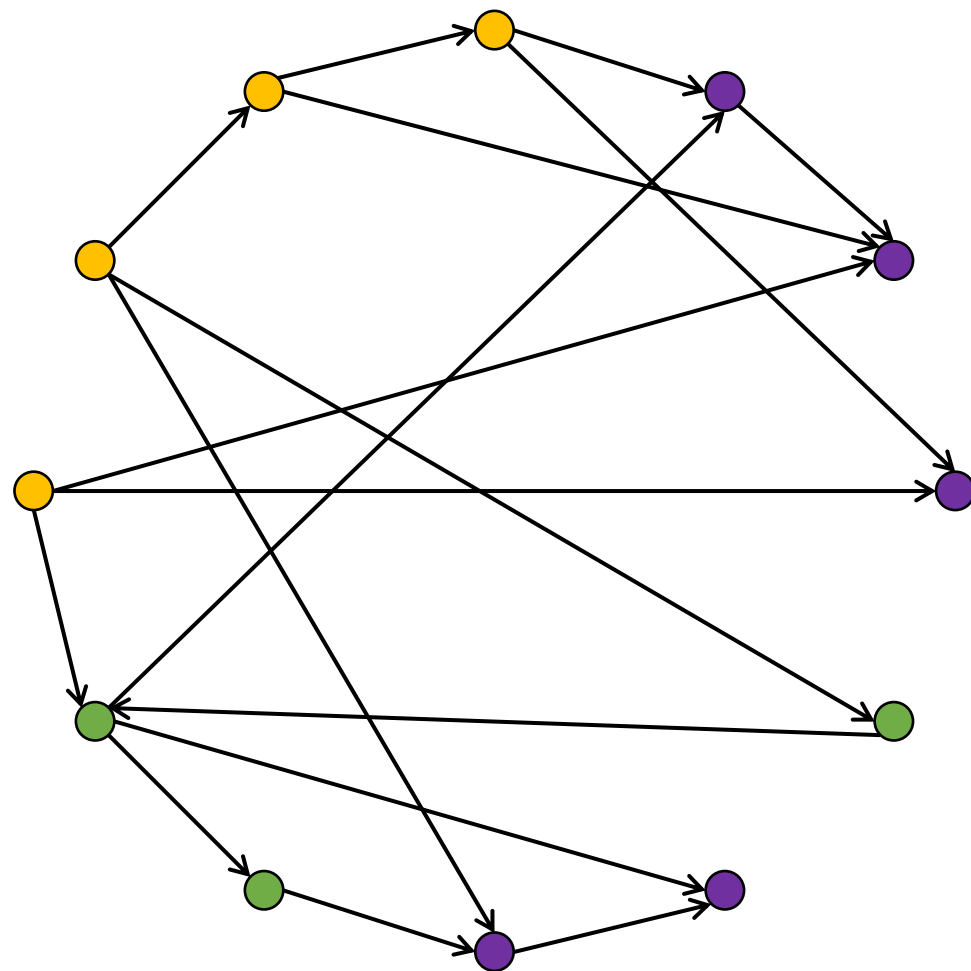
$$b_v = \frac{d^{out} - d^{in}}{d}$$

$$b_v \in \begin{array}{c} -1 \qquad \qquad \qquad 1 \\ \text{purple} \quad \text{green} \quad \text{yellow} \end{array}$$

Snapshot

$$b_v = \frac{d^{out} - d^{in}}{d}$$

$b_v \in$ 

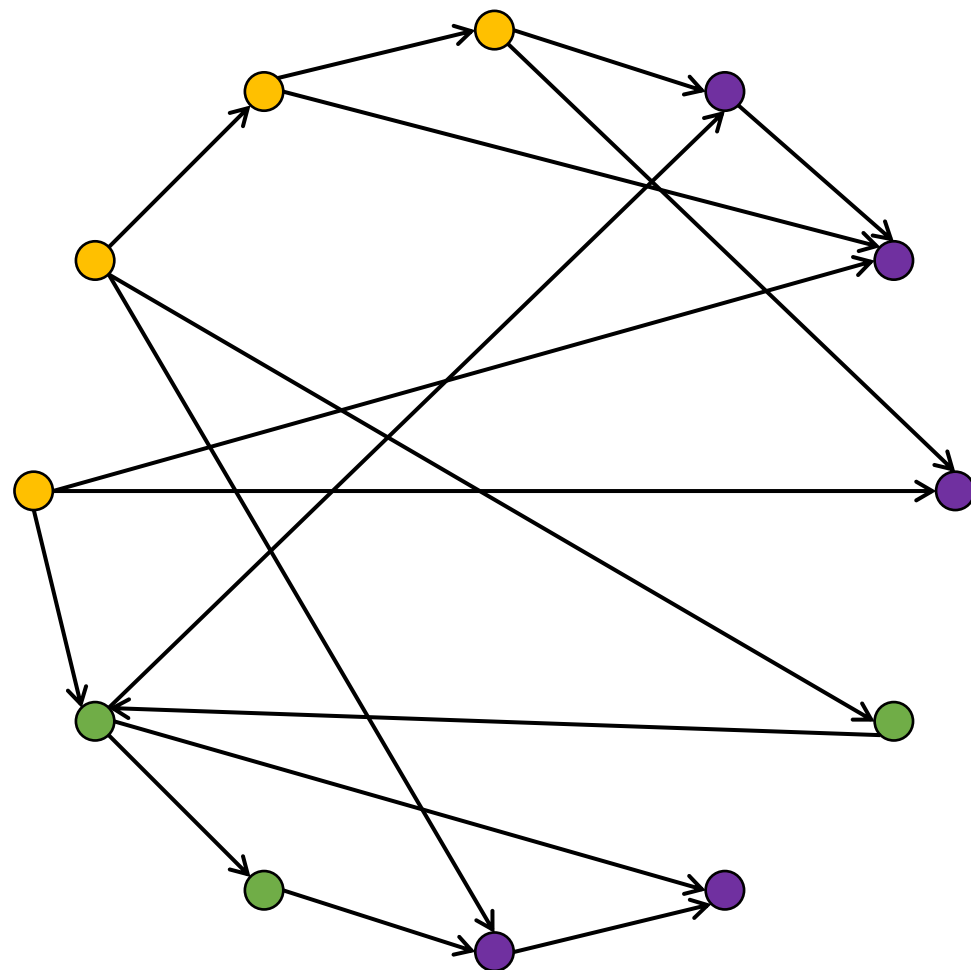


Snapshot

$$b_v = \frac{d^{out} - d^{in}}{d}$$



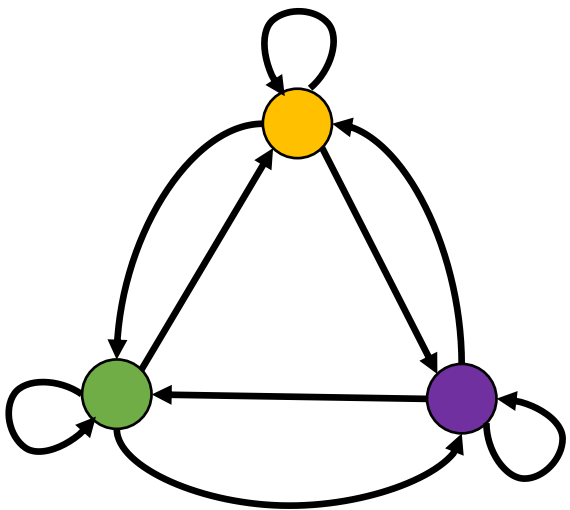
	2	3	6
	0	2	2
	0	0	2



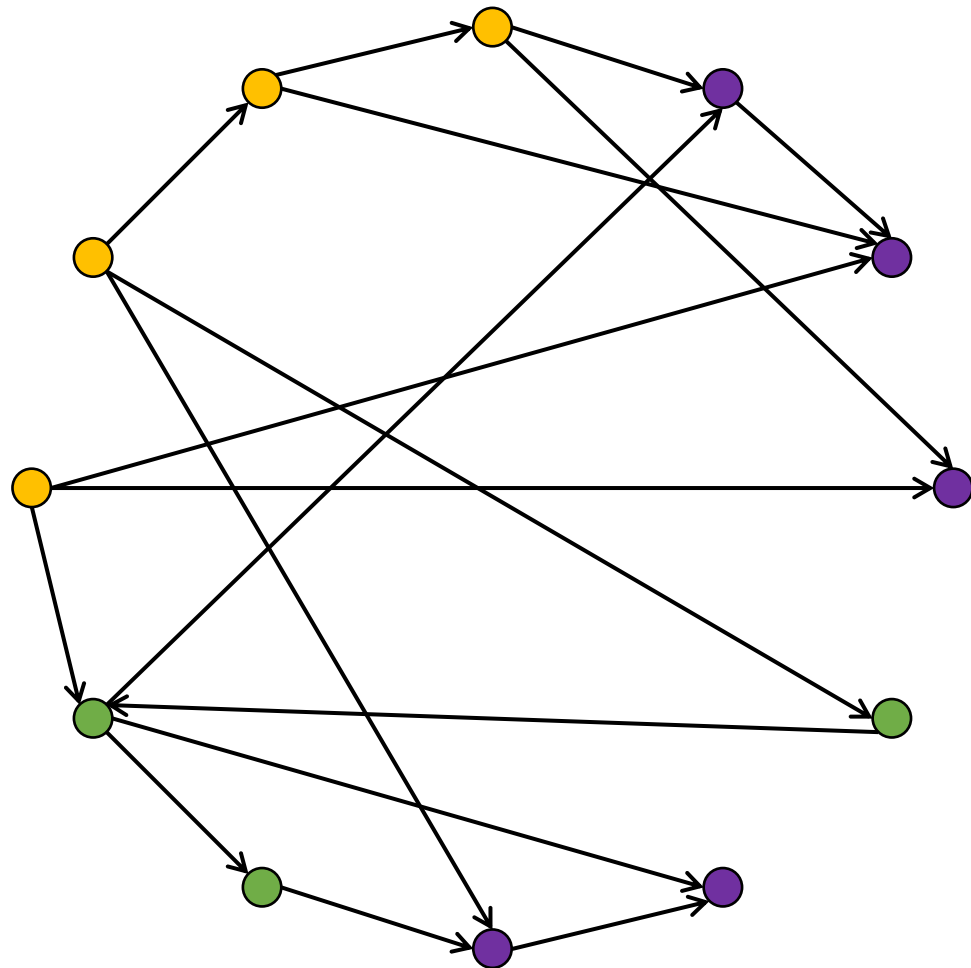
Snapshot

$$b_v = \frac{d^{out} - d^{in}}{d}$$

$b_v \in$ 



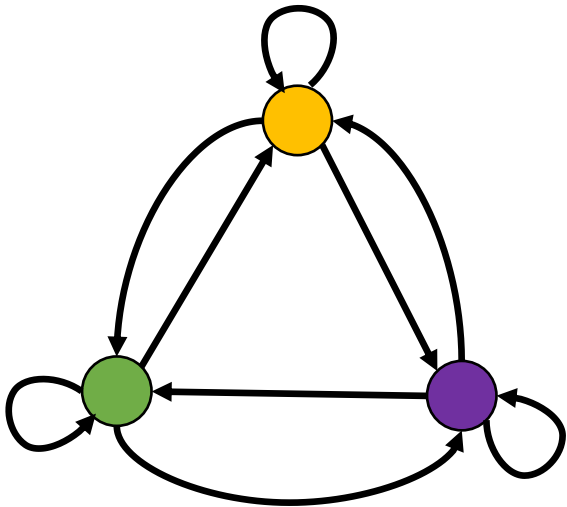
			
	2	3	6
	0	2	2
	0	0	2



Snapshot

$$b_v = \frac{d^{out} - d^{in}}{d}$$

$$b_v \in \begin{array}{c} -1 \qquad \qquad \qquad 1 \\ \text{[Purple] [Green] [Yellow]} \end{array}$$

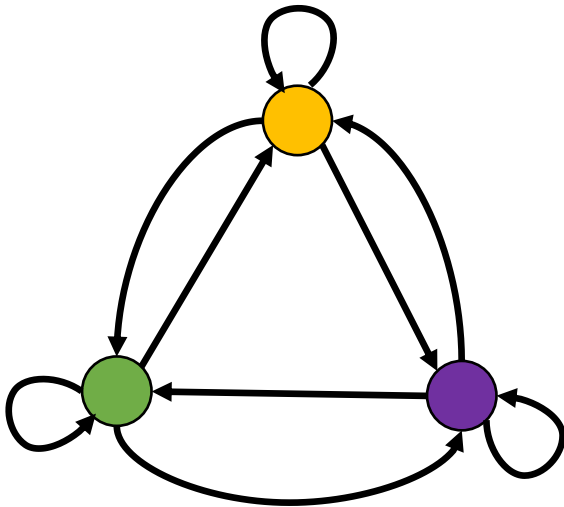


- [FJ'15], [S'23]
Snapshot with constant number of bias classes is sufficient for an 0.484-approximation of Maximum Directed Cut

Snapshot

$$b_v = \frac{d^{out} - d^{in}}{d}$$

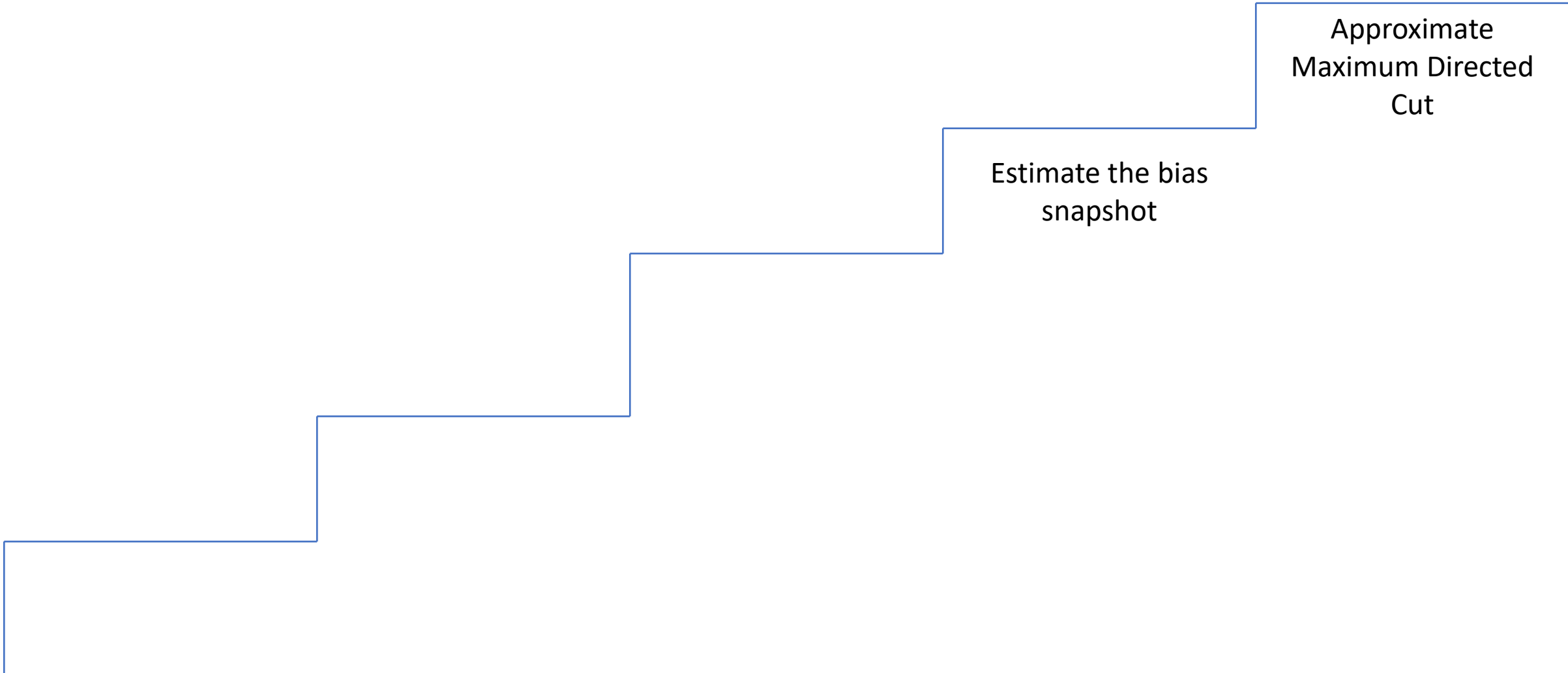
$$b_v \in \begin{array}{c} -1 \qquad \qquad \qquad 1 \\ \text{[Purple] [Green] [Yellow]} \end{array}$$



	<div style="background-color: purple; width: 20px; height: 10px;"></div>	<div style="background-color: green; width: 20px; height: 10px;"></div>	<div style="background-color: yellow; width: 20px; height: 10px;"></div>
<div style="background-color: purple; width: 20px; height: 10px;"></div>	2	3	6
<div style="background-color: green; width: 20px; height: 10px;"></div>	0	2	2
<div style="background-color: yellow; width: 20px; height: 10px;"></div>	0	0	2

- [FJ'15], [S'23]
Snapshot with constant number of bias classes is sufficient for an 0.484-approximation of Maximum Directed Cut
- [SSSV'23]
Snapshot approximation
 - Computable with $\tilde{O}(\sqrt{n})$ bits
 - Sufficient for 0.484-approximation of Maximum Directed Cut

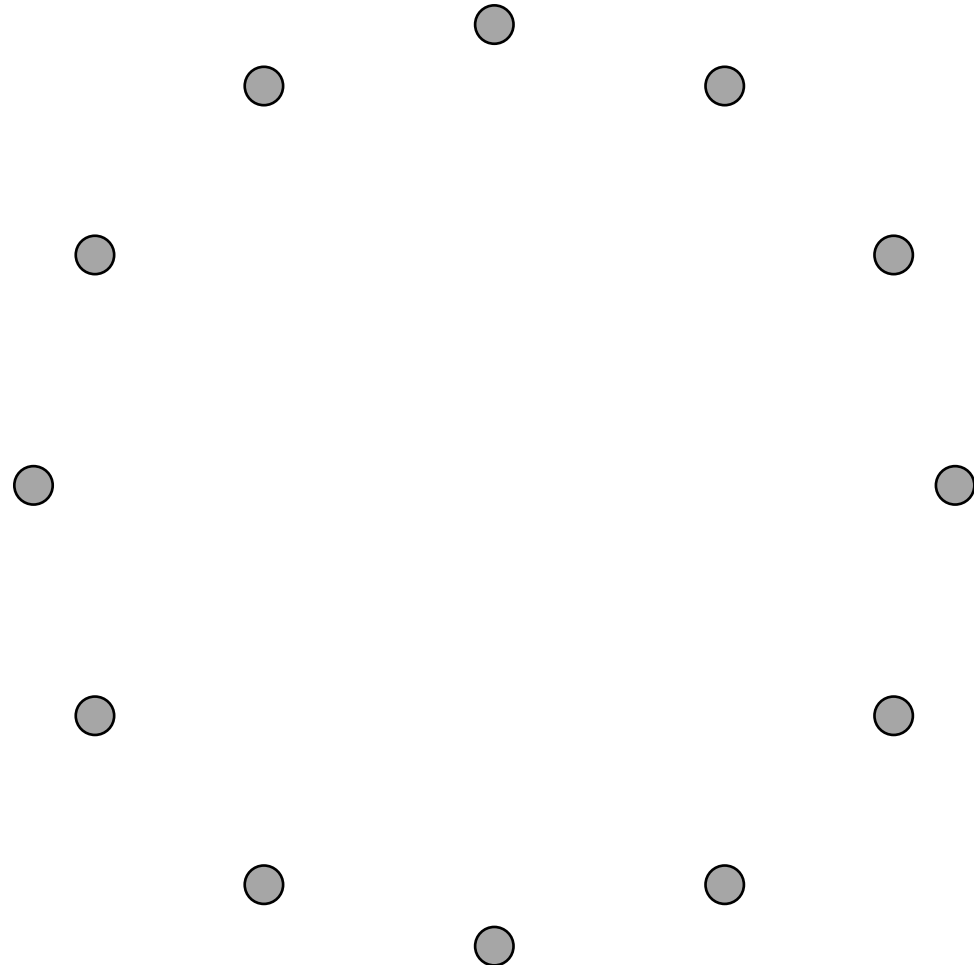
Progress map



Estimate the bias
snapshot

Approximate
Maximum Directed
Cut

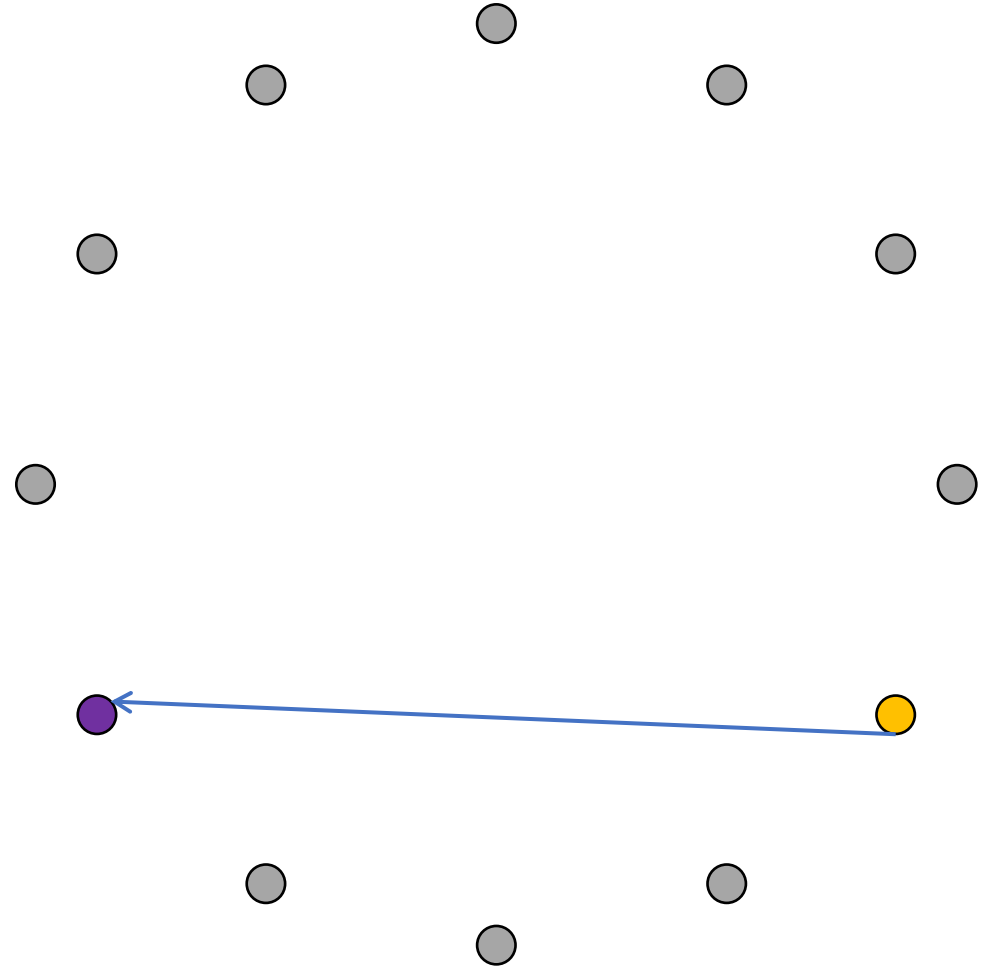
Snapshot in the stream



Snapshot in the stream

Every edge

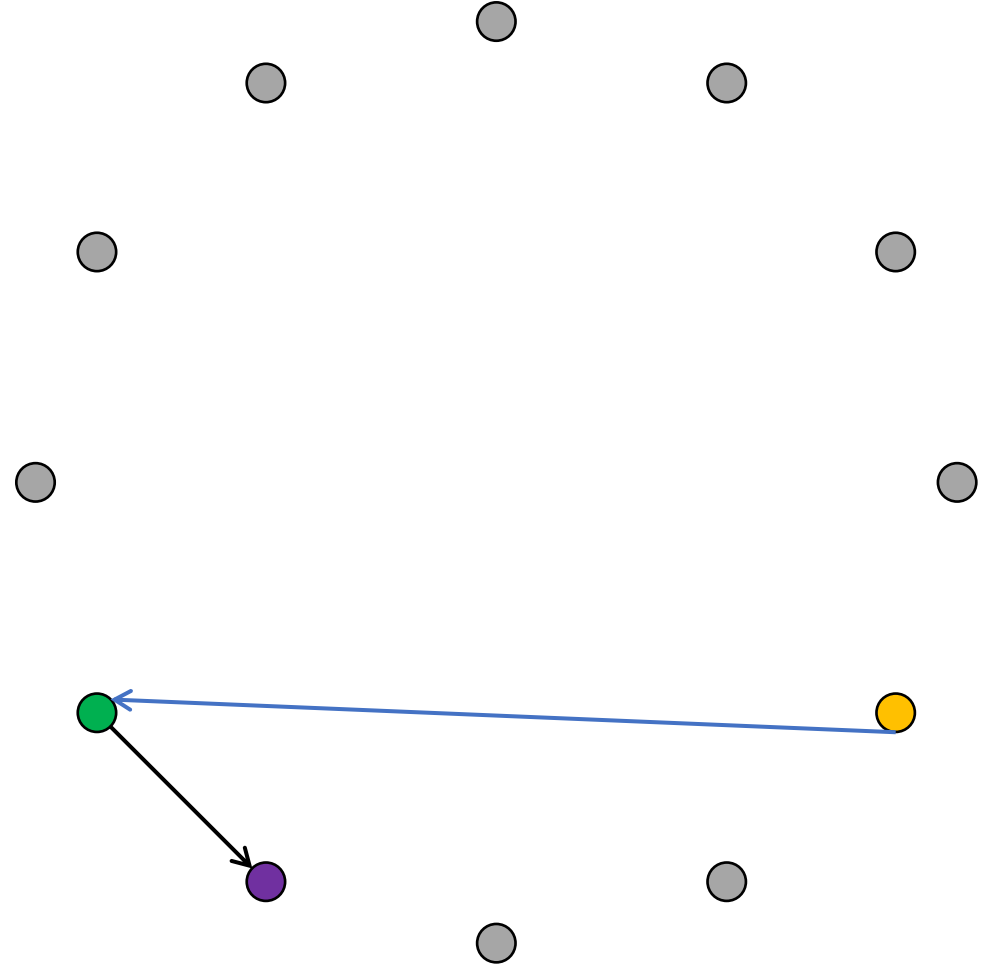
- Increases the number of edges in the snapshot, and



Snapshot in the stream

Every edge

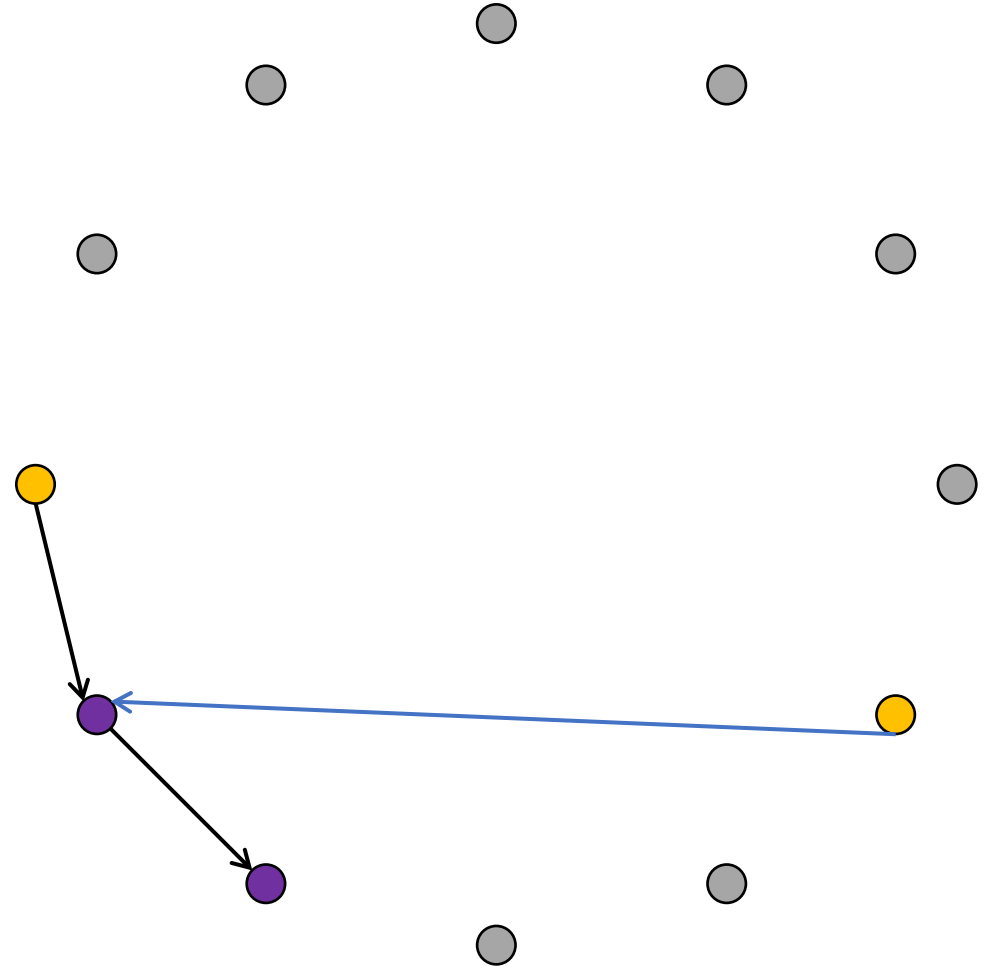
- Increases the number of edges in the snapshot, and
- Changes biases of endpoints



Snapshot in the stream

Every edge

- Increases the number of edges in the snapshot, and
- Changes biases of endpoints

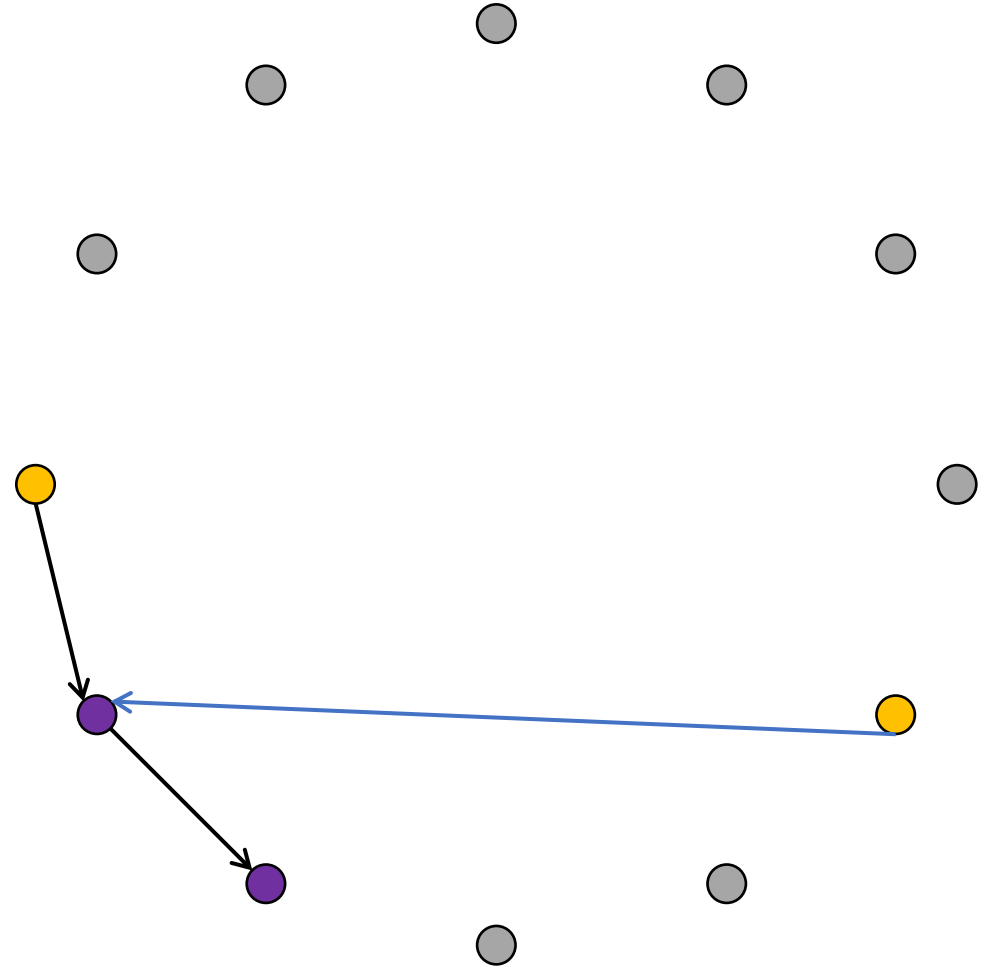


Snapshot in the stream

Every edge

- Increases the number of edges in the snapshot, and
- Changes biases of endpoints

We need to take care of both simultaneously!



Snapshot in the stream

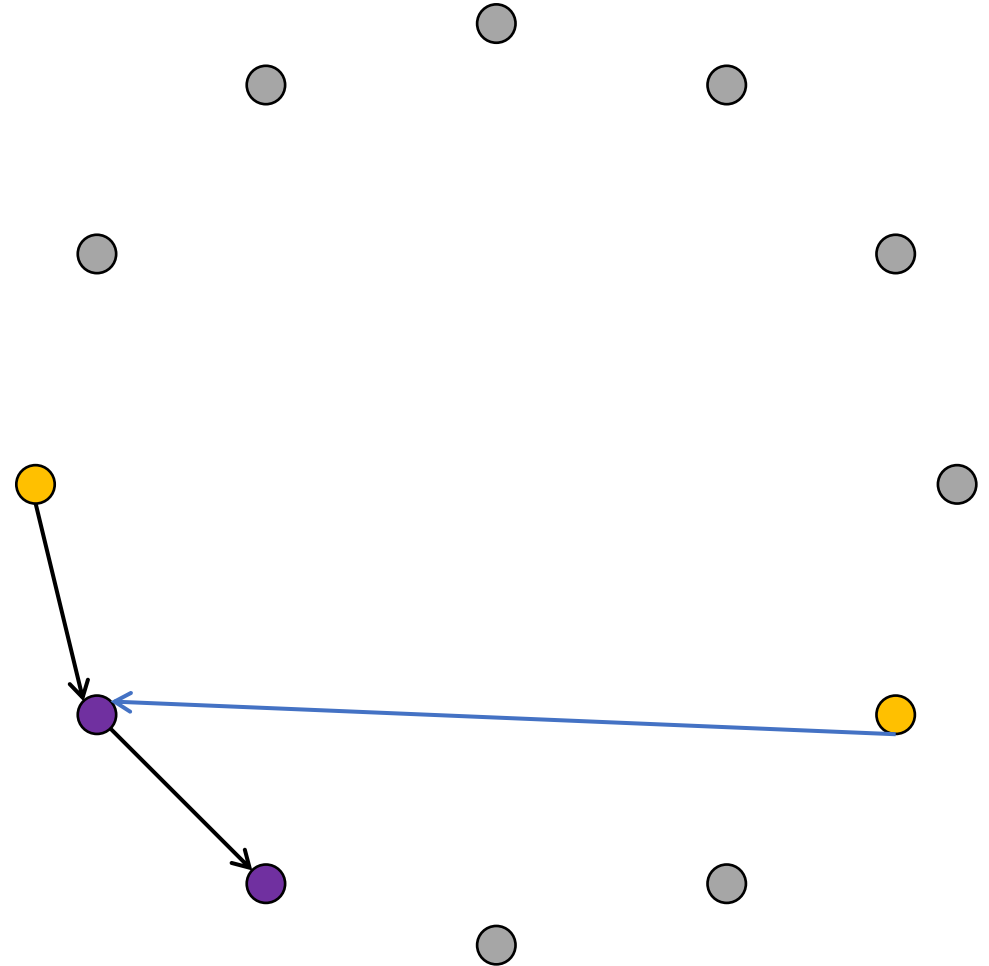
Every edge

- Increases the number of edges in the snapshot, and
- Changes biases of endpoints

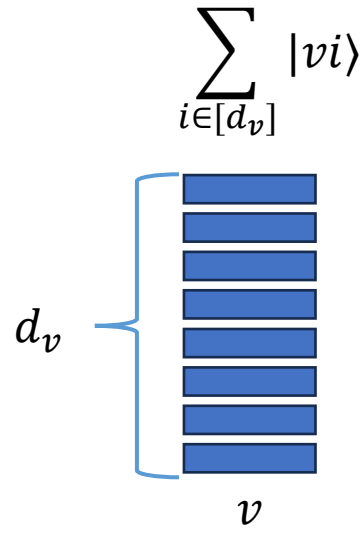
We need to take care of both simultaneously!

Simpler problem that still captures these conditions:

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

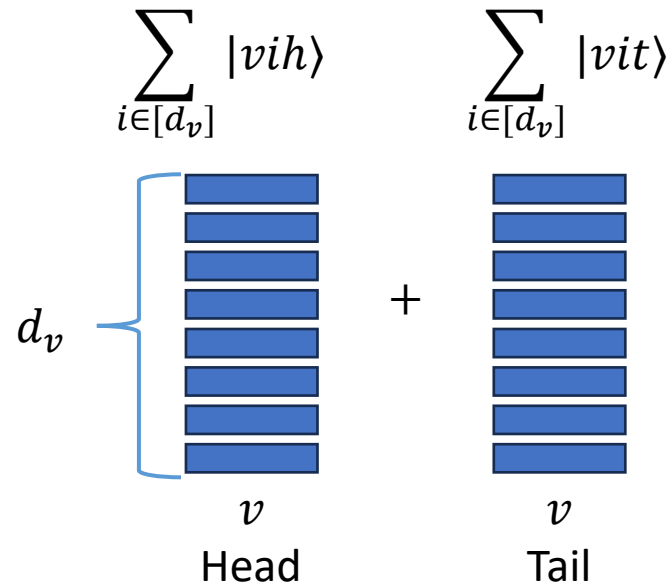


Sketch: edges with high endpoints' degree

**Problem:**

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

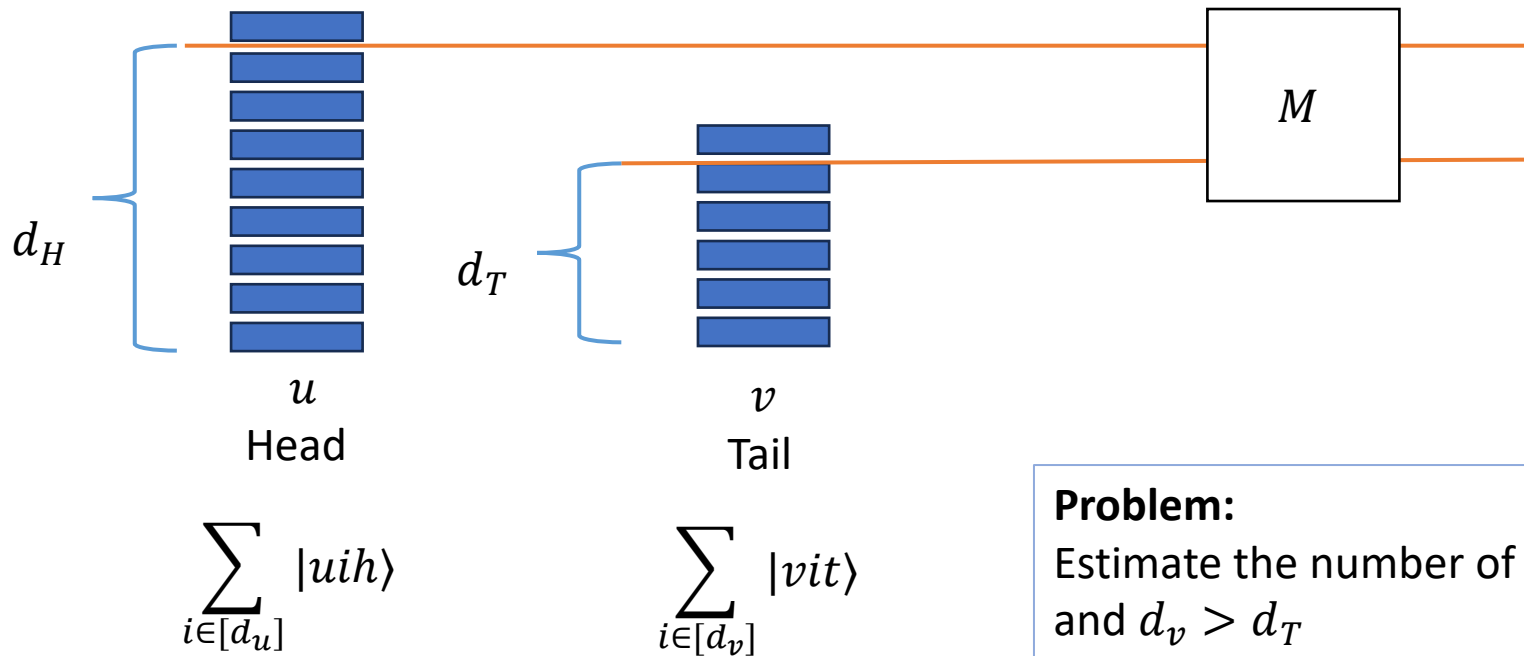
Sketch: edges with high endpoints' degree

**Problem:**

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

Checking the degrees of endpoints

When $\overrightarrow{uv}$ arrives:



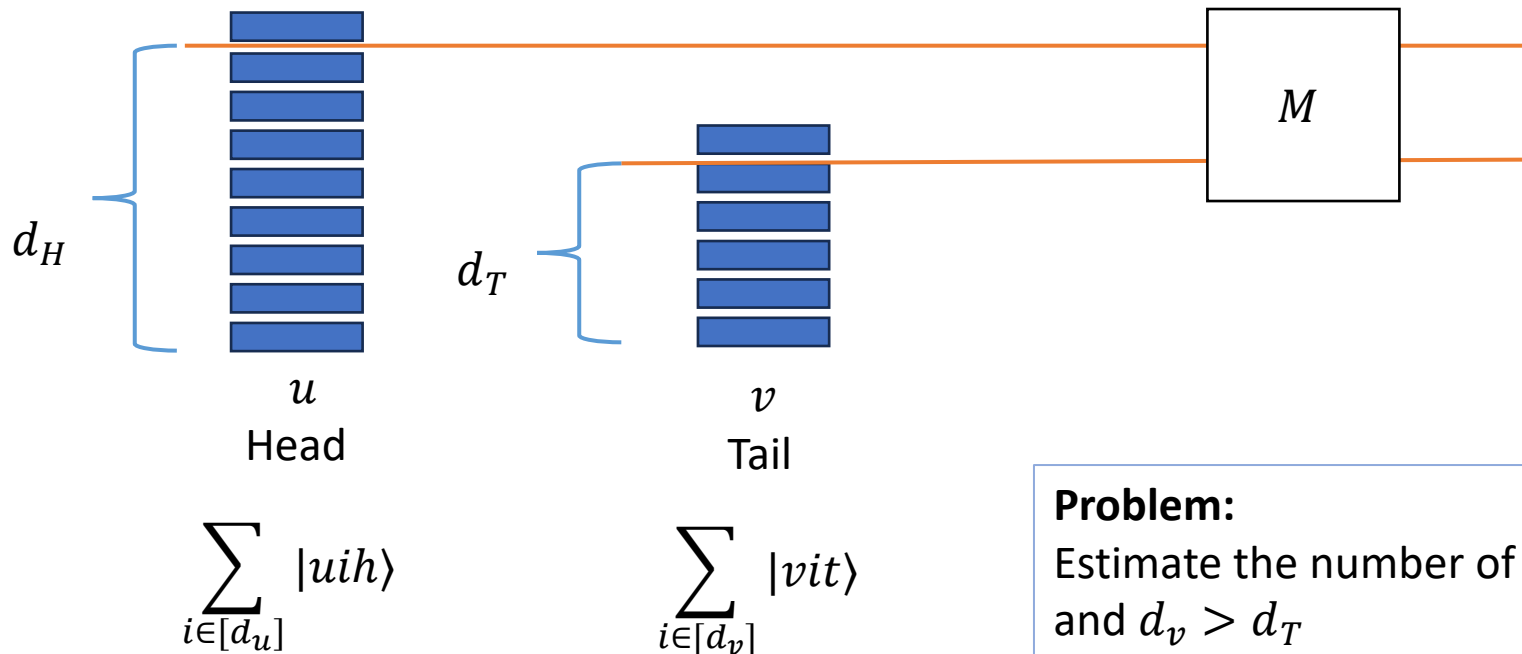
Problem:

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

Checking the degrees of endpoints

When $\overrightarrow{uv}$ arrives:

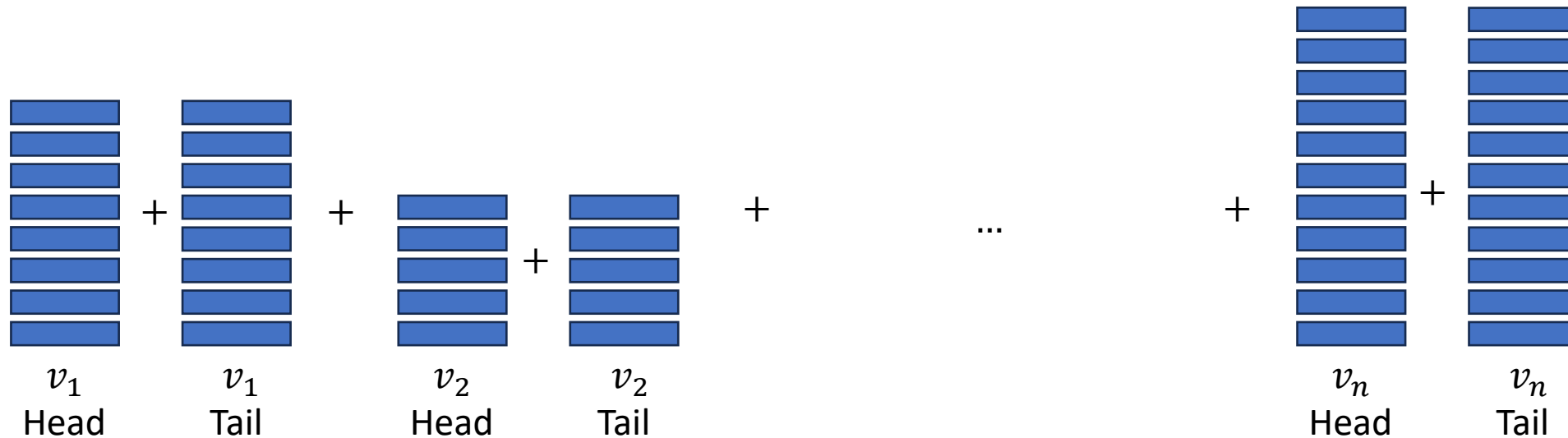
- $|ud_Hh\rangle + |vd_Tt\rangle$
- $|ud_Hh\rangle - |vd_Tt\rangle$
- Complementary projector



Problem:

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

Sketch: edges with high endpoints' degree



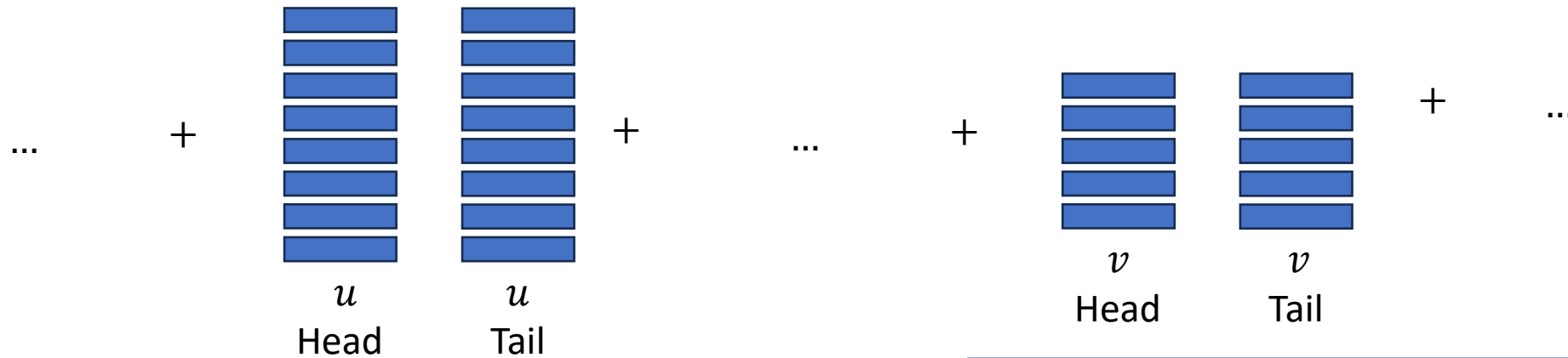
Takes $O(\log n)$ qubits

Problem:

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

Maintaining the sketch

When $\overrightarrow{uv}$ arrives:



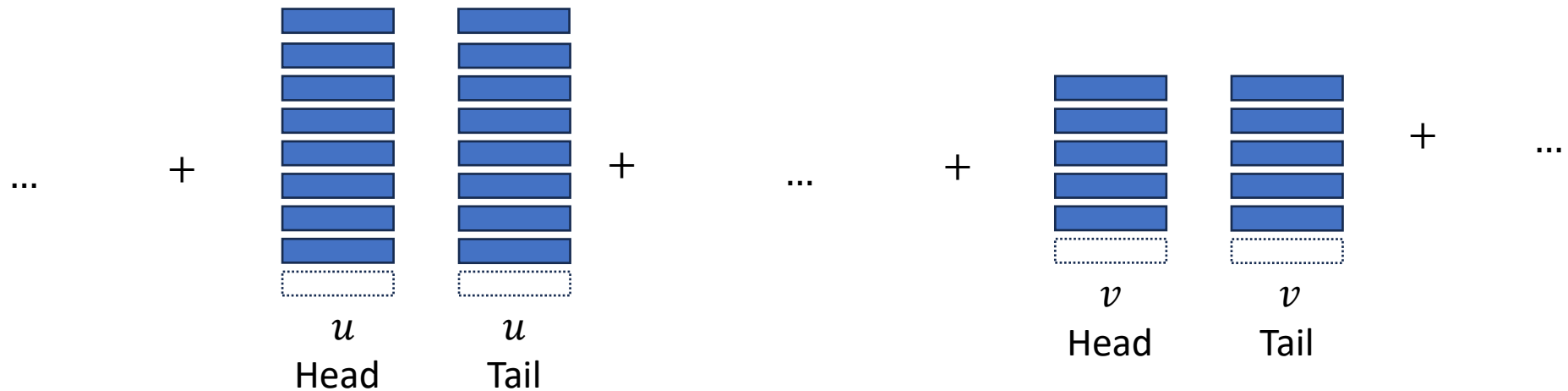
Problem:

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

Maintaining the sketch

When $\overrightarrow{uv}$ arrives:

$$|uih\rangle \rightarrow |u(i+1)h\rangle$$

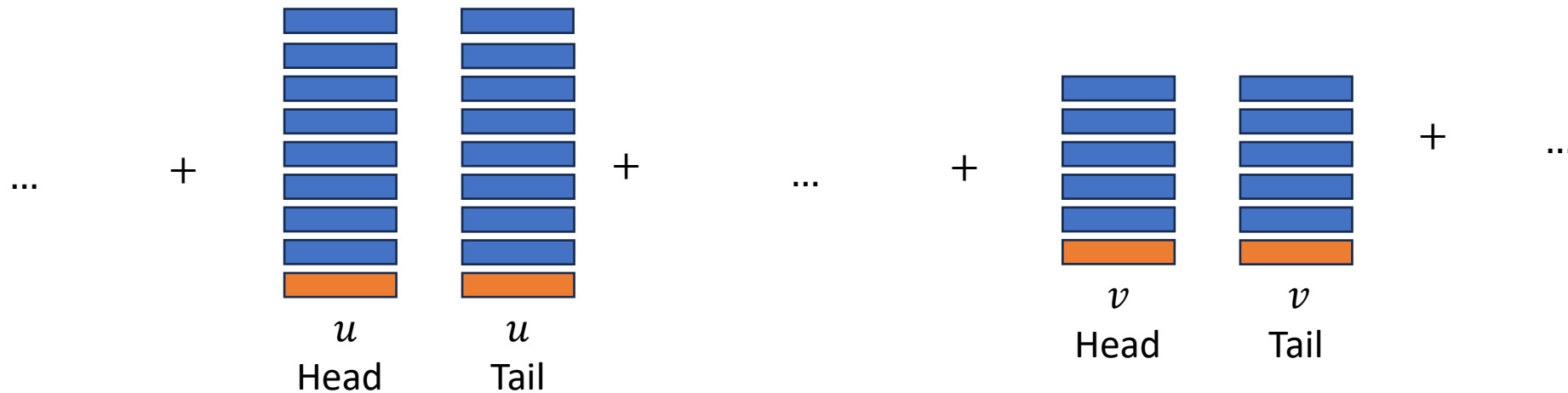


Problem:

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

Maintaining the sketch

When $\overrightarrow{uv}$ arrives:

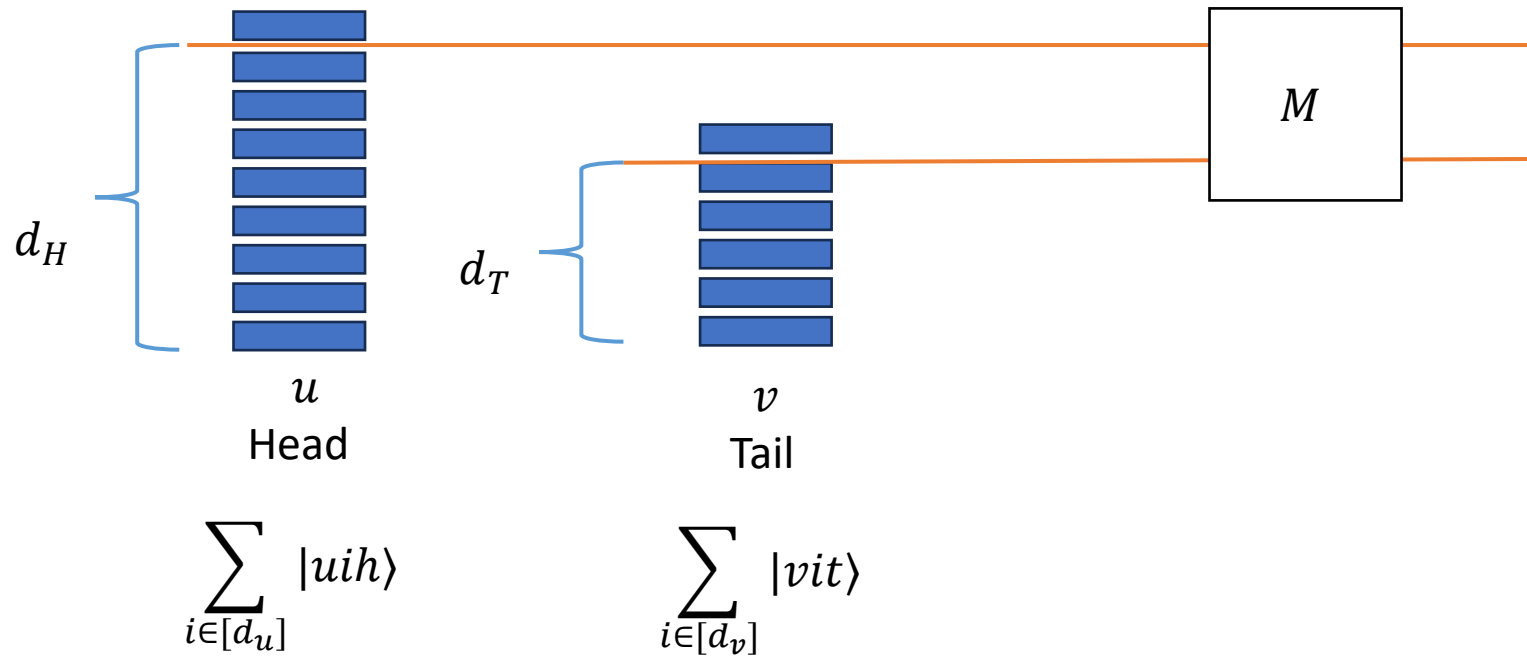


Problem:

Estimate the number of edges $\overrightarrow{uv}$ with $d_u > d_H$ and $d_v > d_T$

Checking the degrees of endpoints

One big state for all edges

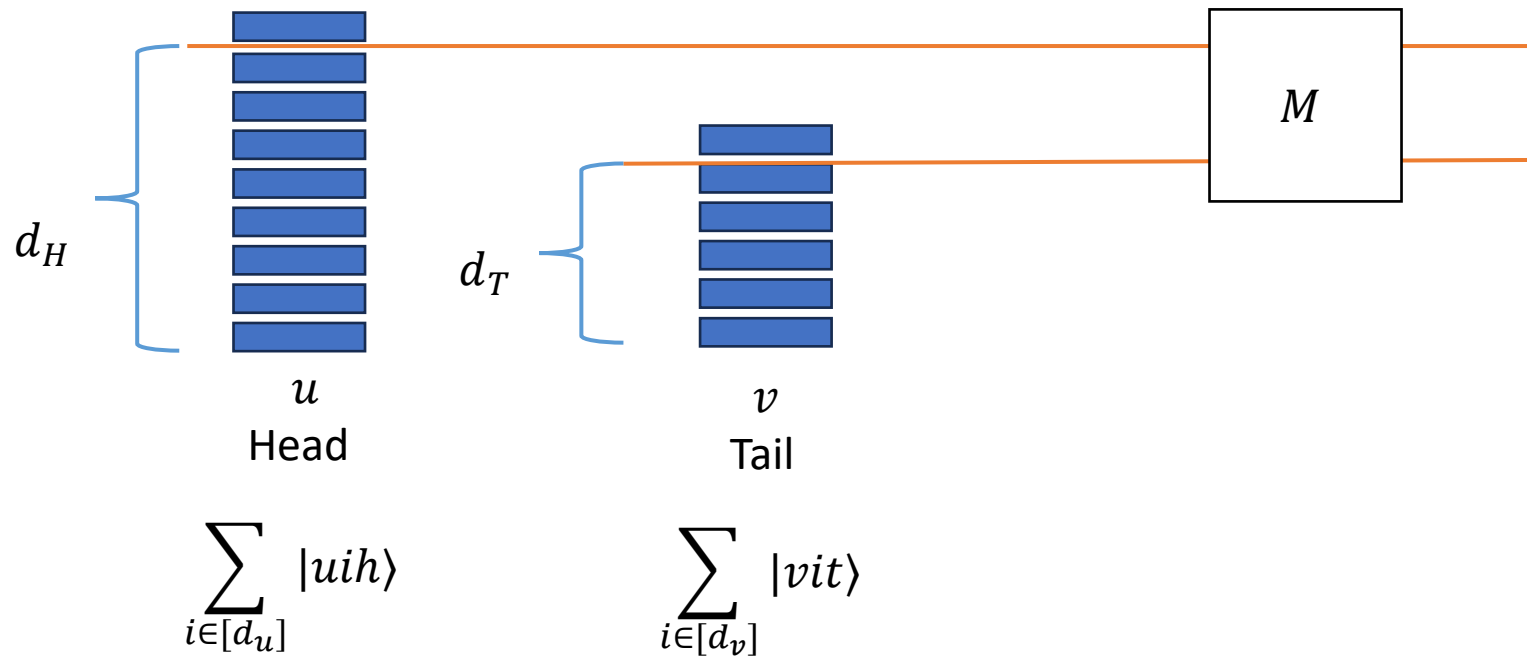


Measure with

- $|ud_Hh\rangle + |vd_Tt\rangle$
- $|ud_Hh\rangle - |vd_Tt\rangle$
- Complementary projector

Checking the degrees of endpoints

One big state for all edges



Measure with

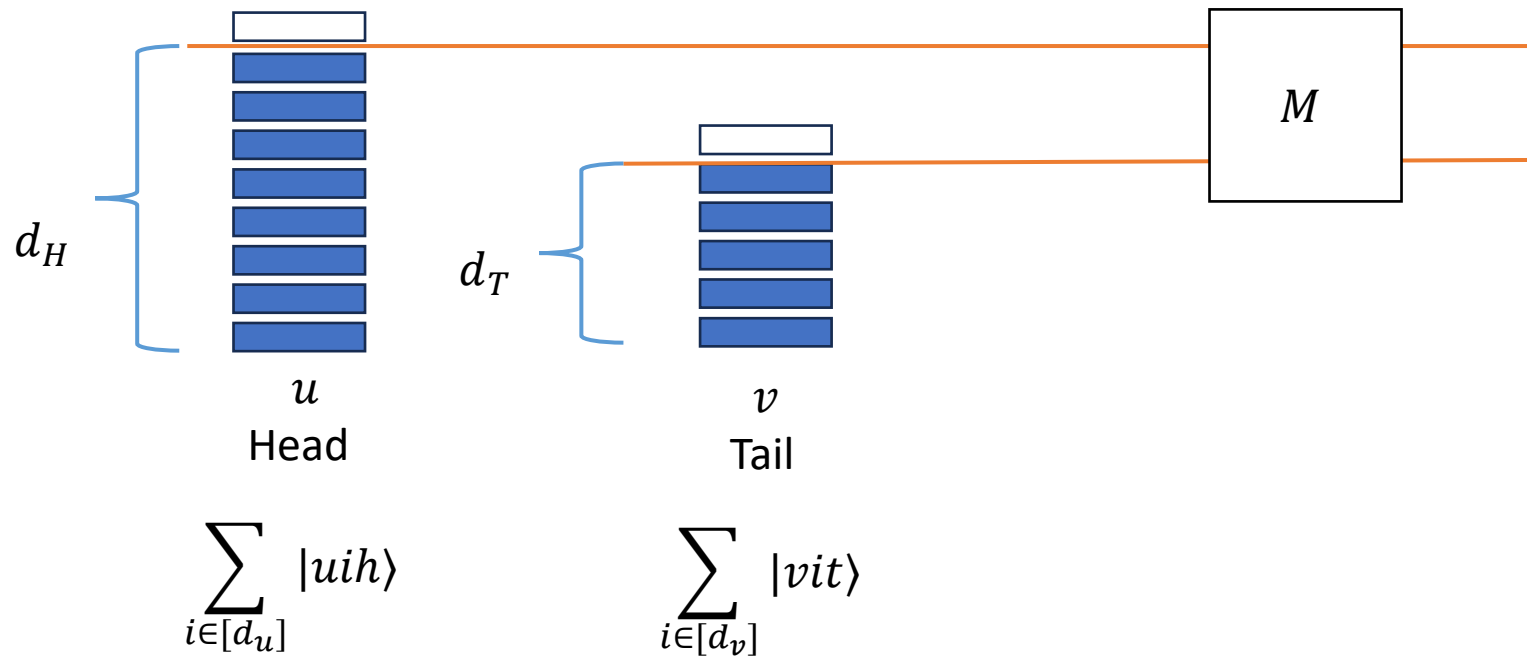
- $|ud_Hh\rangle + |vd_Tt\rangle$
- $|ud_Hh\rangle - |vd_Tt\rangle$
- Complementary projector

If $|ud_Hh\rangle \pm |vd_Tt\rangle$

- Destroys the state
- Gives a sample $\overrightarrow{uv}$ for classical postprocessing

Checking the degrees of endpoints

One big state for all edges



Measure with

- $|ud_Hh\rangle + |vd_Tt\rangle$
- $|ud_Hh\rangle - |vd_Tt\rangle$
- Complementary projector

If $|ud_Hh\rangle \pm |vd_Tt\rangle$

- Destroys the state
- Gives a sample $\overrightarrow{uv}$ for classical postprocessing

If complementary projector

- Erases $|ud_Hh\rangle, |vd_Tt\rangle$
- Allows to continue with the same state

Progress map

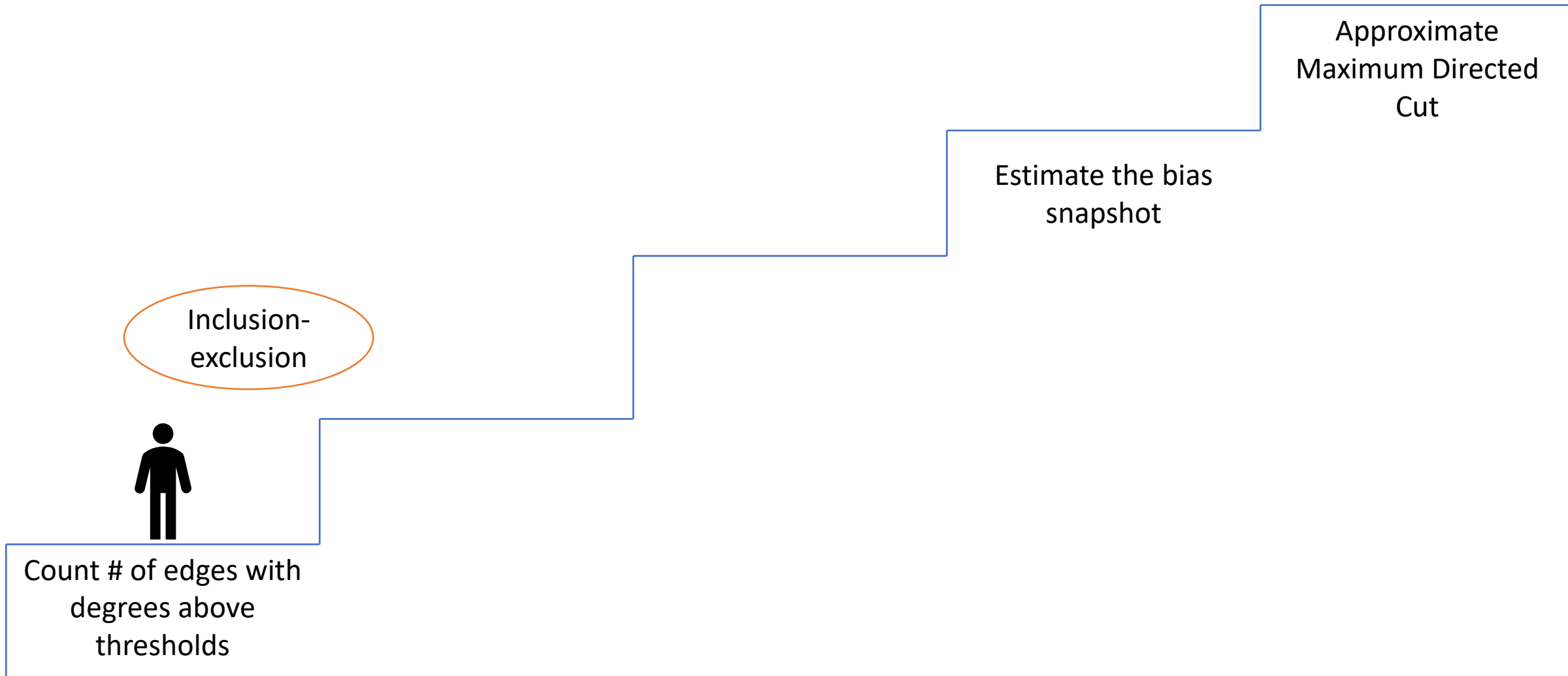


Count # of edges with
degrees above
thresholds

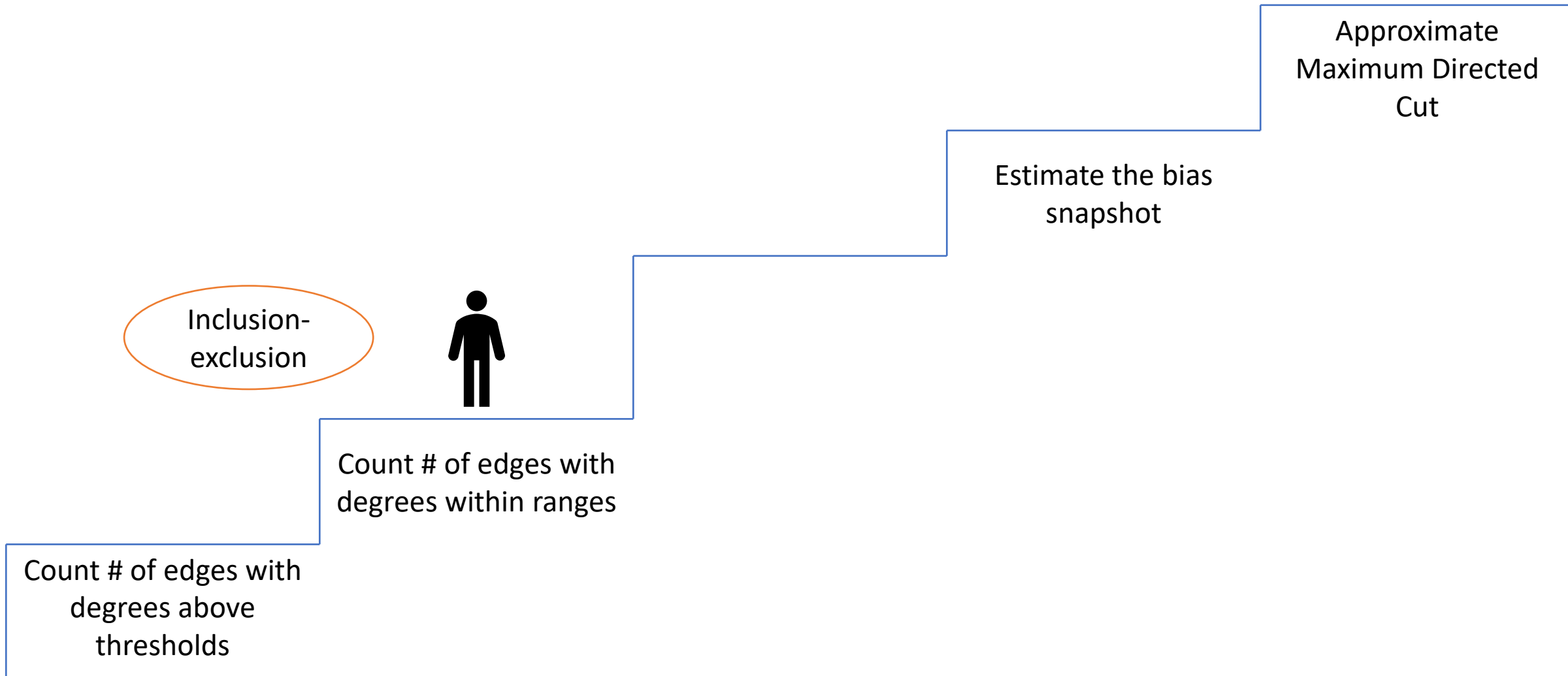
Estimate the bias
snapshot

Approximate
Maximum Directed
Cut

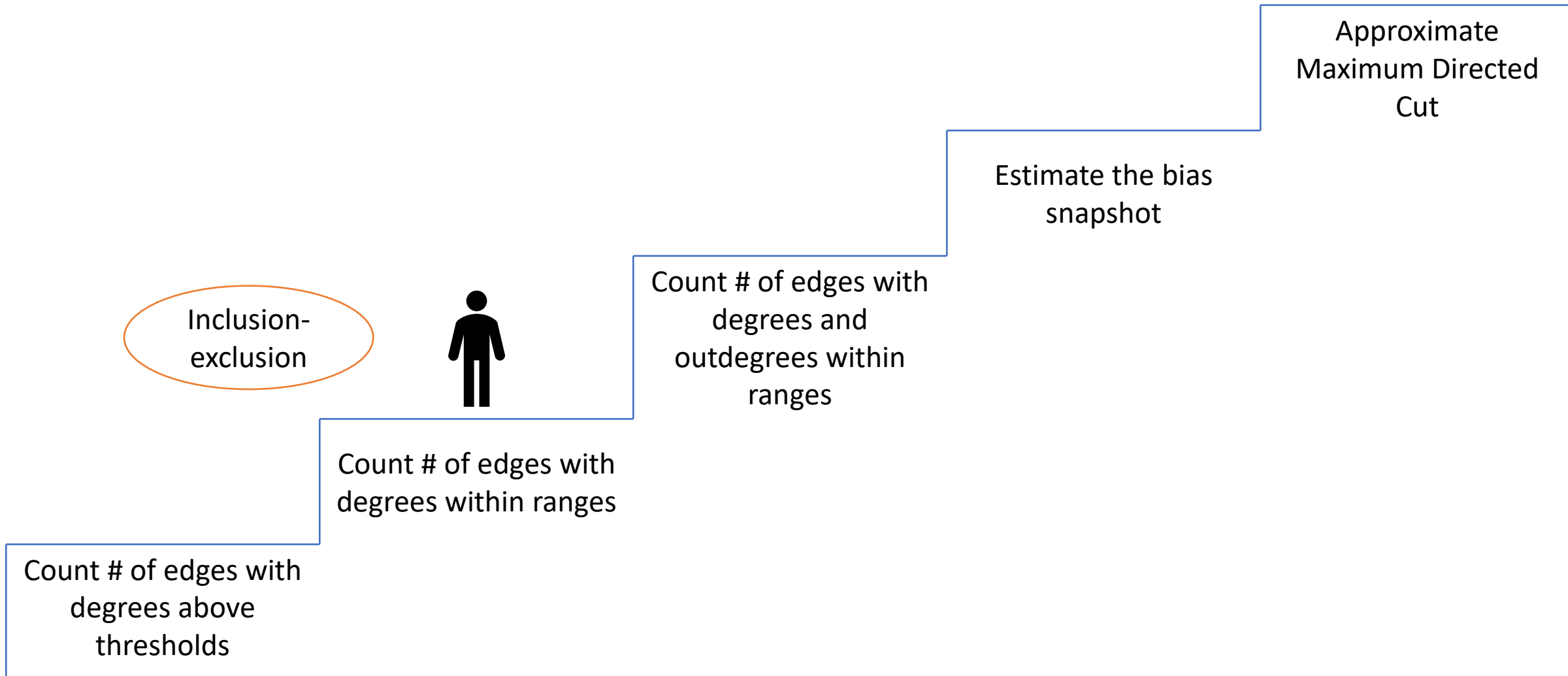
Progress map



Progress map



Progress map



Outdegree processing

- Can't process the outdegrees the same way in *polylog* space

Outdegree processing

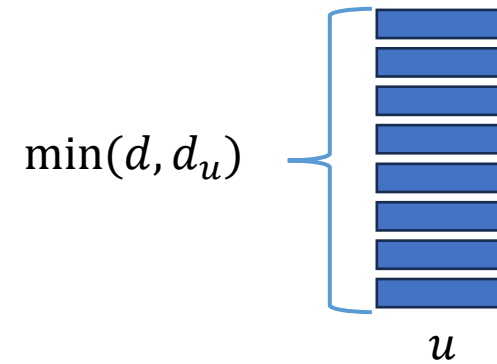
- Can't process the outdegrees the same way in *polylog* space
- Instead, we store the outdegrees in “higher order bits” of the stack height

Outdegree processing

- Can't process the outdegrees the same way in *polylog* space
- Instead, we store the outdegrees in “higher order bits” of the stack height

Degree range $[d, d')$

When $\overrightarrow{uv}$ arrives:



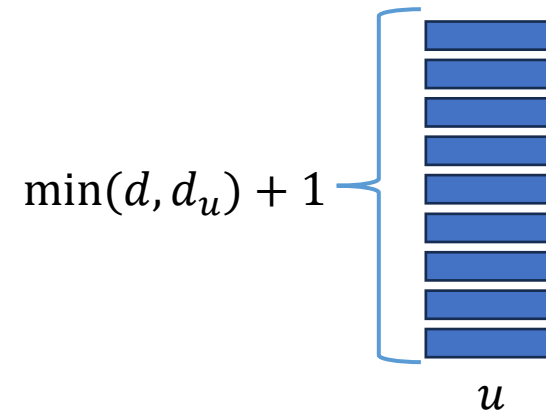
Outdegree processing

- Can't process the outdegrees the same way in *polylog* space
- Instead, we store the outdegrees in “higher order bits” of the stack height

Degree range $[d, d')$

When $\overrightarrow{uv}$ arrives:

1. Update the degree



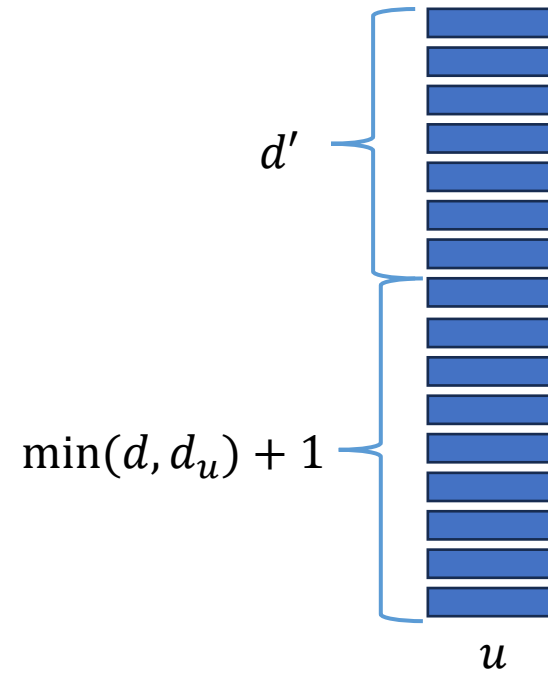
Outdegree processing

- Can't process the outdegrees the same way in *polylog* space
- Instead, we store the outdegrees in “higher order bits” of the stack height

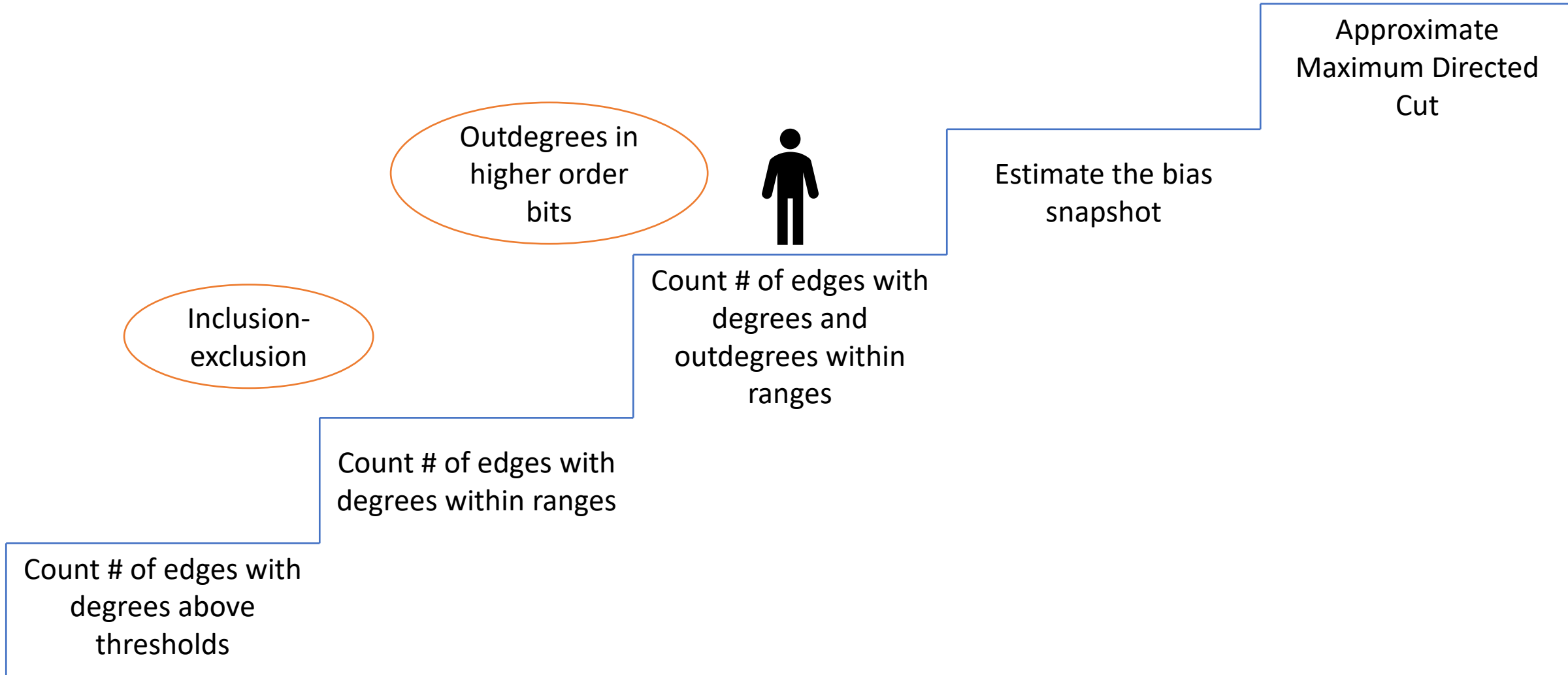
Degree range $[d, d')$

When $\overrightarrow{uv}$ arrives:

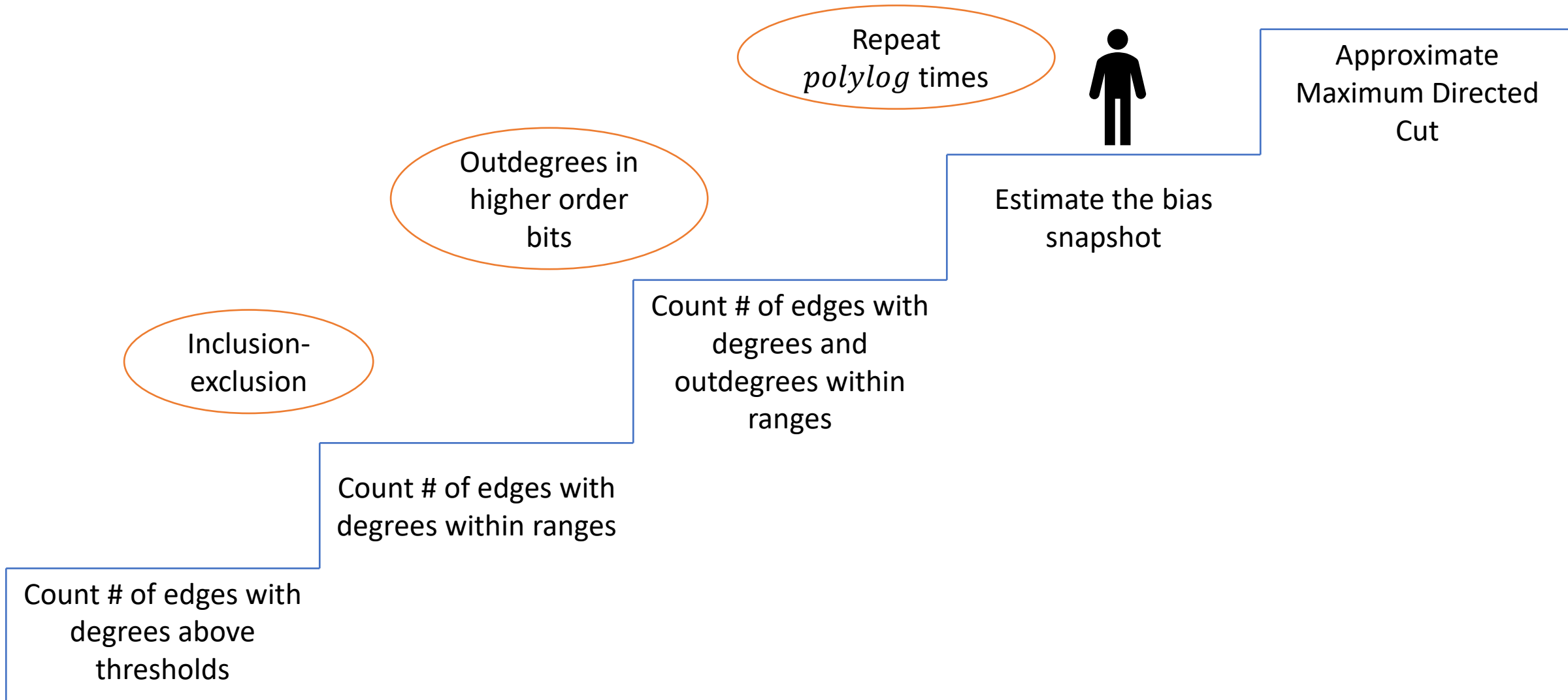
1. Update the degree
2. With probability $\sim 1/d$ update outdegree



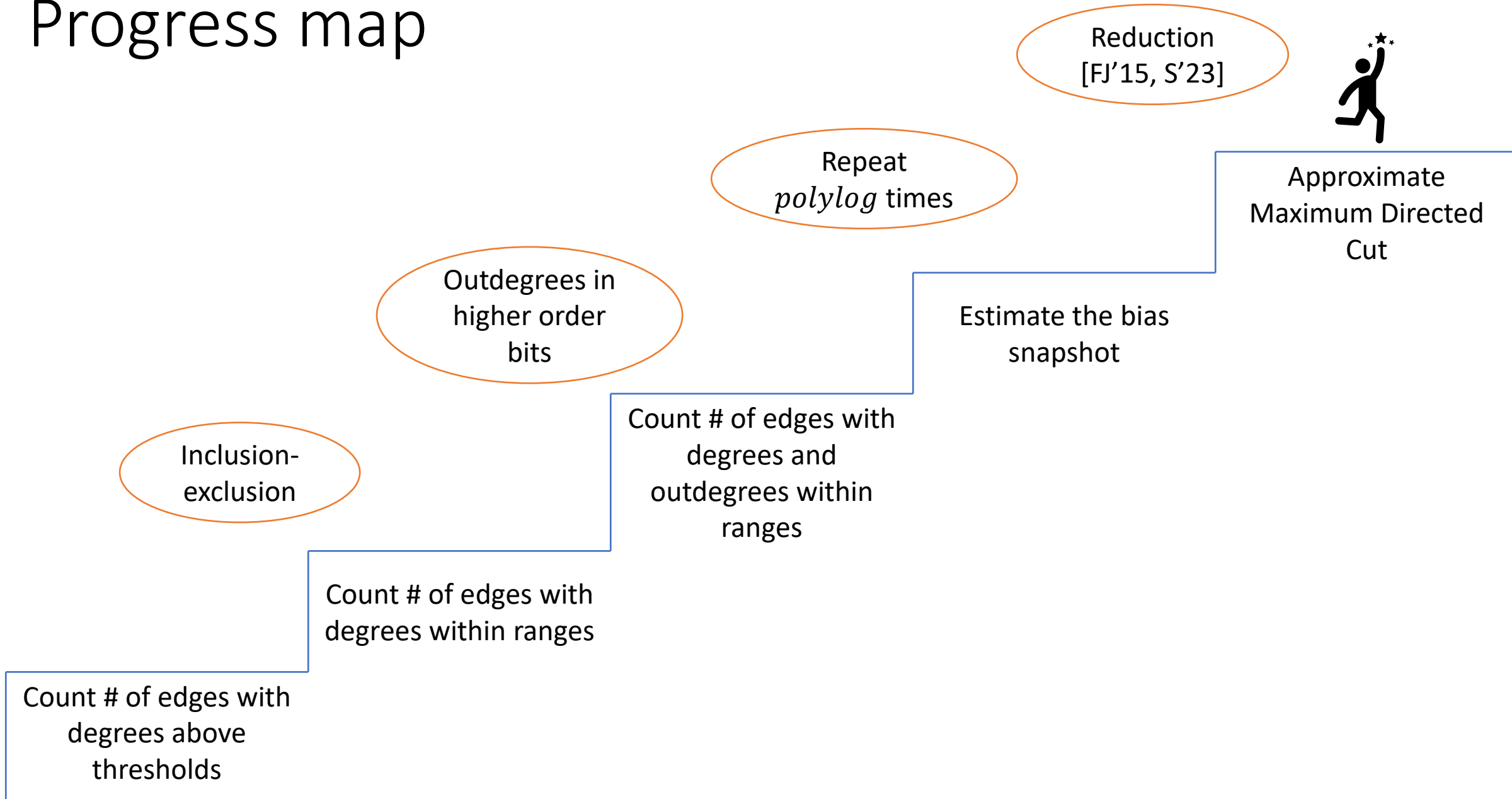
Progress map



Progress map



Progress map



Conclusion

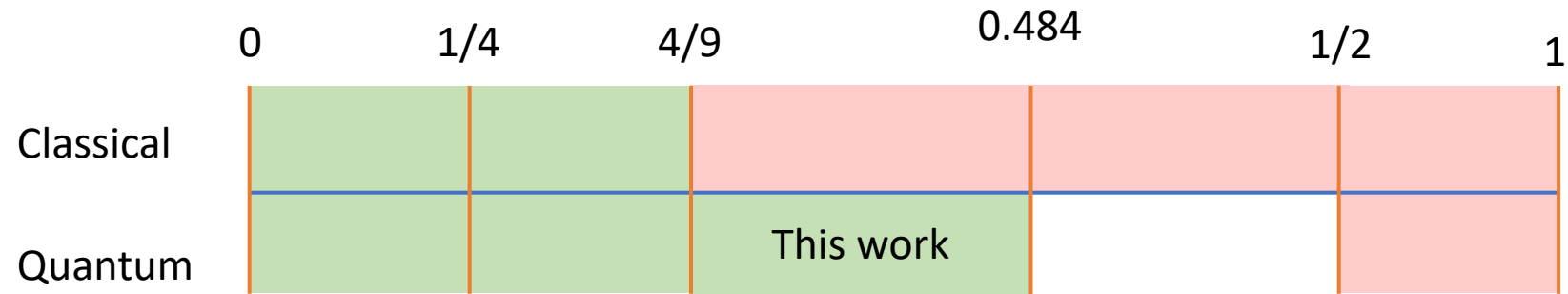
[Kallaugher, Parekh, V]

There exists a quantum streaming algorithm that 0.484-approximates Maximum Directed Cut in $\text{polylog}(n)$ space.

Conclusion

[Kallaugher, Parekh, V]

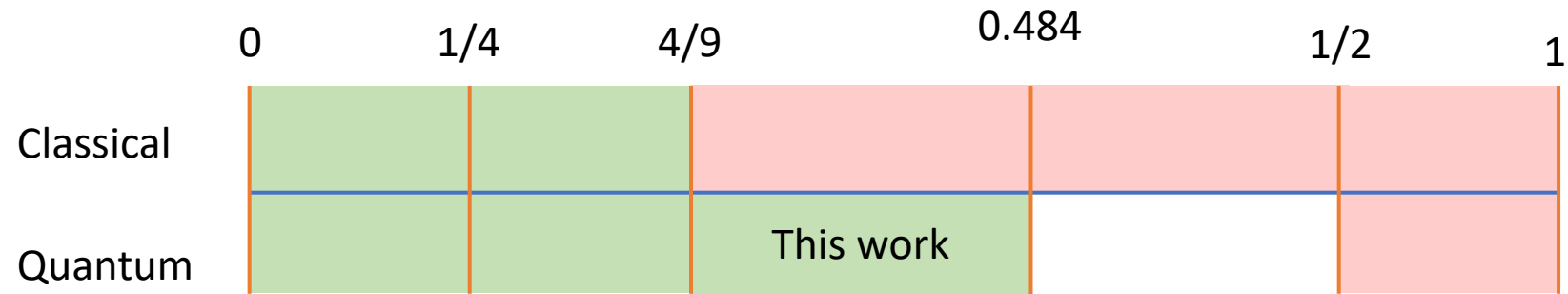
There exists a quantum streaming algorithm that 0.484-approximates Maximum Directed Cut in $\text{polylog}(n)$ space.



Conclusion

[Kallaugher, Parekh, V]

There exists a quantum streaming algorithm that 0.484-approximates Maximum Directed Cut in $\text{polylog}(n)$ space.

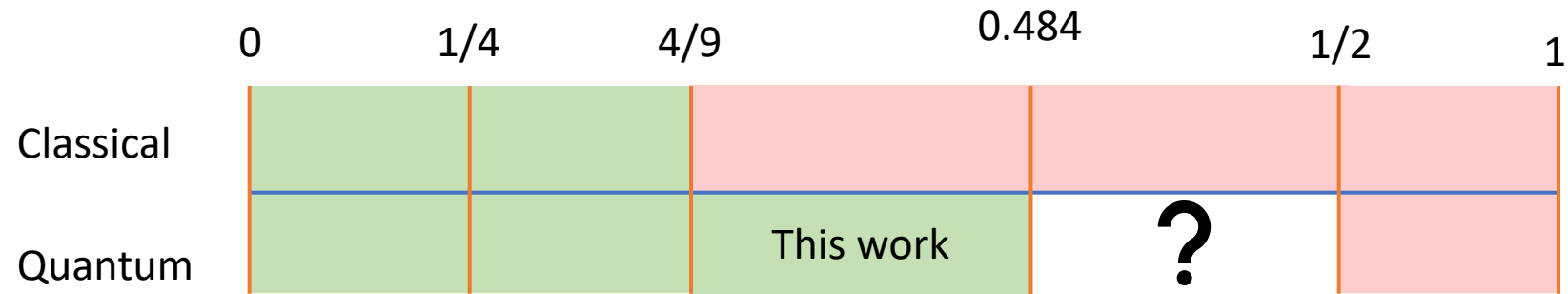


Open questions:

Conclusion

[Kallaugher, Parekh, V]

There exists a quantum streaming algorithm that 0.484-approximates Maximum Directed Cut in $\text{polylog}(n)$ space.



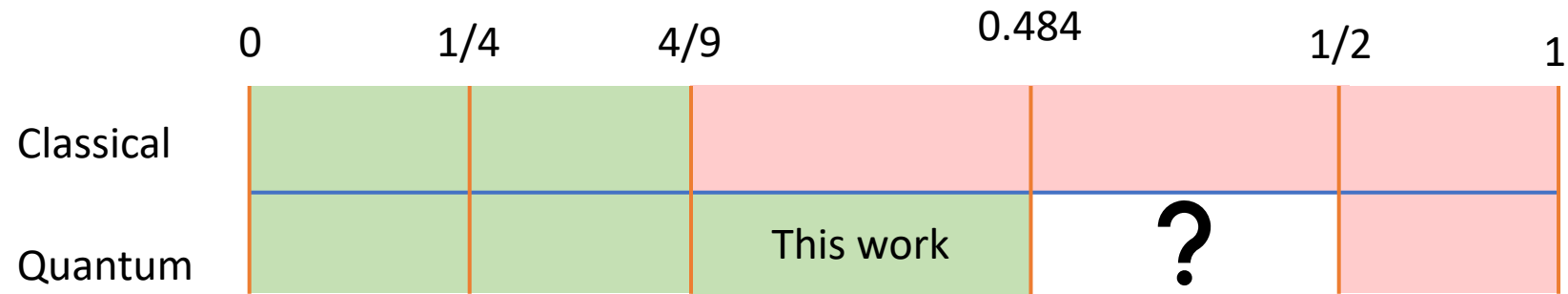
Open questions:

- Quantum streaming complexity of MaxDicut

Conclusion

[Kallaugher, Parekh, V]

There exists a quantum streaming algorithm that 0.484-approximates Maximum Directed Cut in $\text{polylog}(n)$ space.



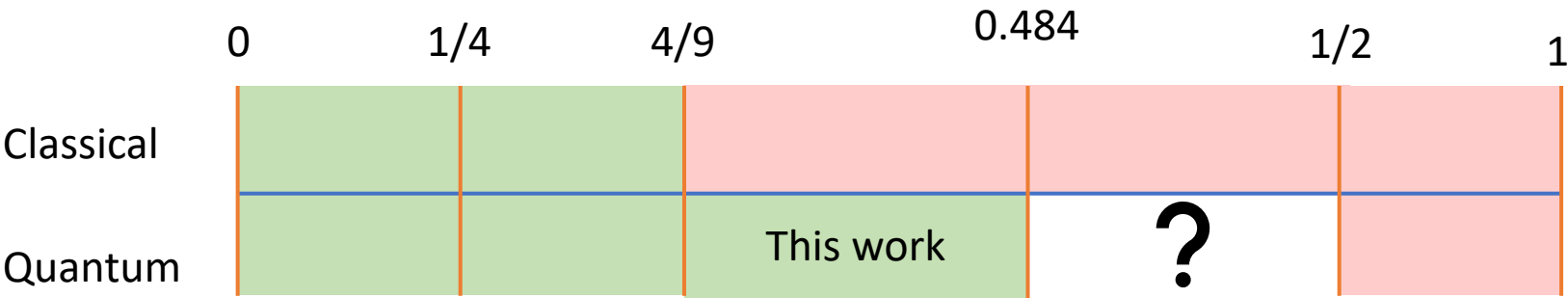
Open questions:

- Quantum streaming complexity of MaxDicut
- Characterizing which CSPs allow quantum advantage

Conclusion

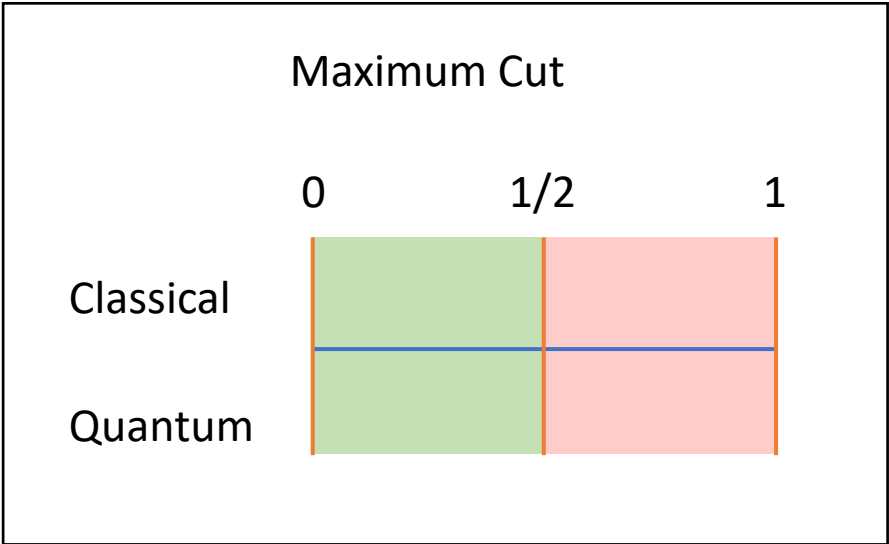
[Kallaugher, Parekh, V]

There exists a quantum streaming algorithm that 0.484-approximates Maximum Directed Cut in $\text{polylog}(n)$ space.



Open questions:

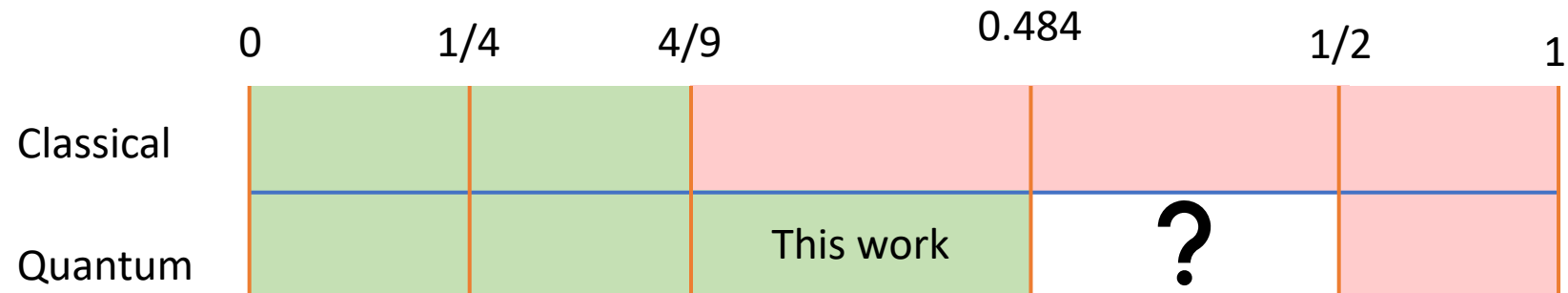
- Quantum streaming complexity of MaxDicut
- Characterizing which CSPs allow quantum advantage



Conclusion

[Kallaugher, Parekh, V]

There exists a quantum streaming algorithm that 0.484-approximates Maximum Directed Cut in $\text{polylog}(n)$ space.



Open questions:

- Quantum streaming complexity of MaxDicut
- Characterizing which CSPs allow quantum advantage

Thank you!

