Maximum Matching in $O(\log \log n)$ Passes in Dynamic Streams

Janani Sundaresan

UNIVERSITY OF WATERLOO

Joint Work with


Sepehr Assadi


Soheil Behnezhad


Christian Konrad

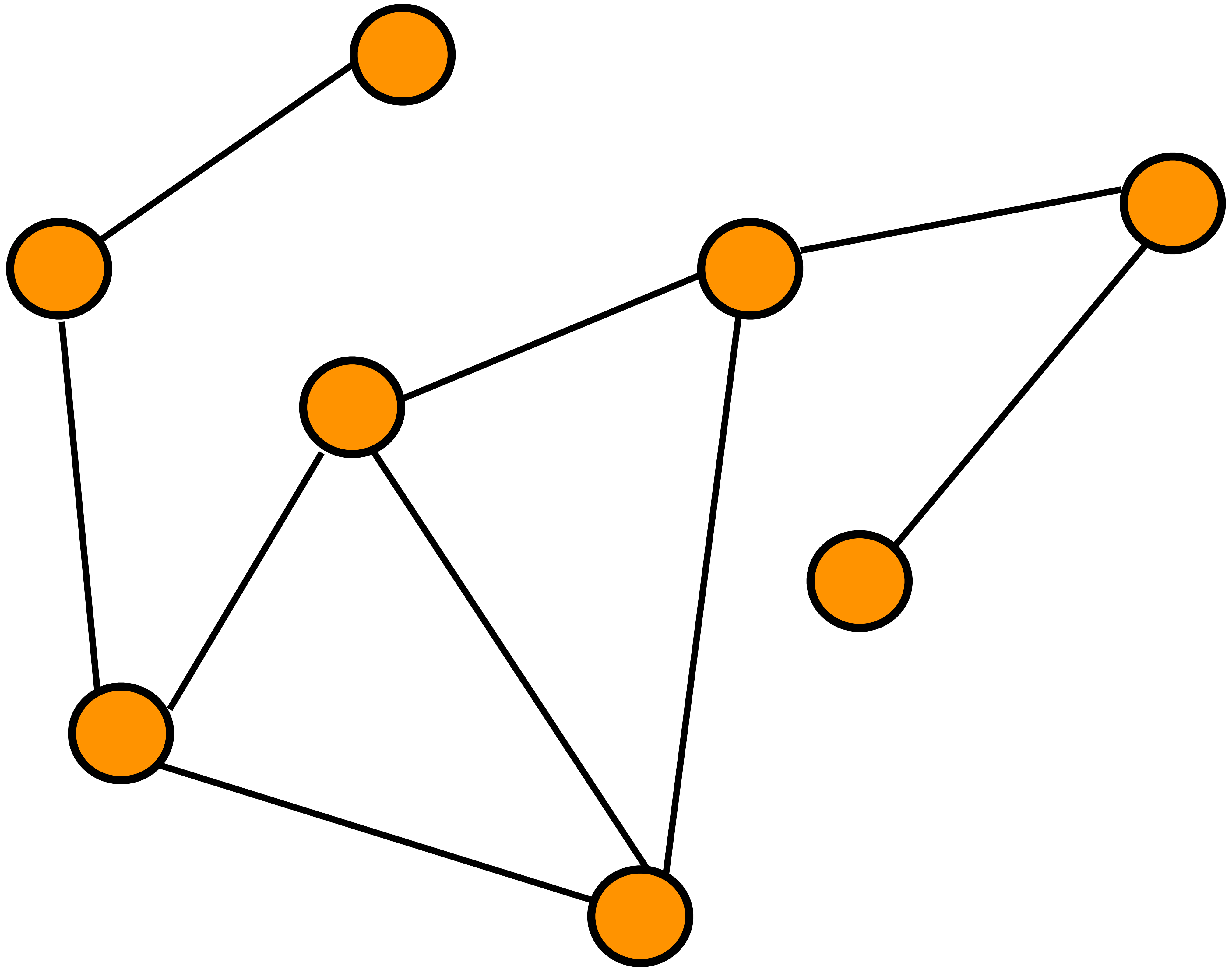

Kheeran K. Naidu

UNIVERSITY OF WATERLOO

NORTHEASTERN UNIVERSITY 18 98 LVX VERITAS VIRTVS

University of BRISTOL

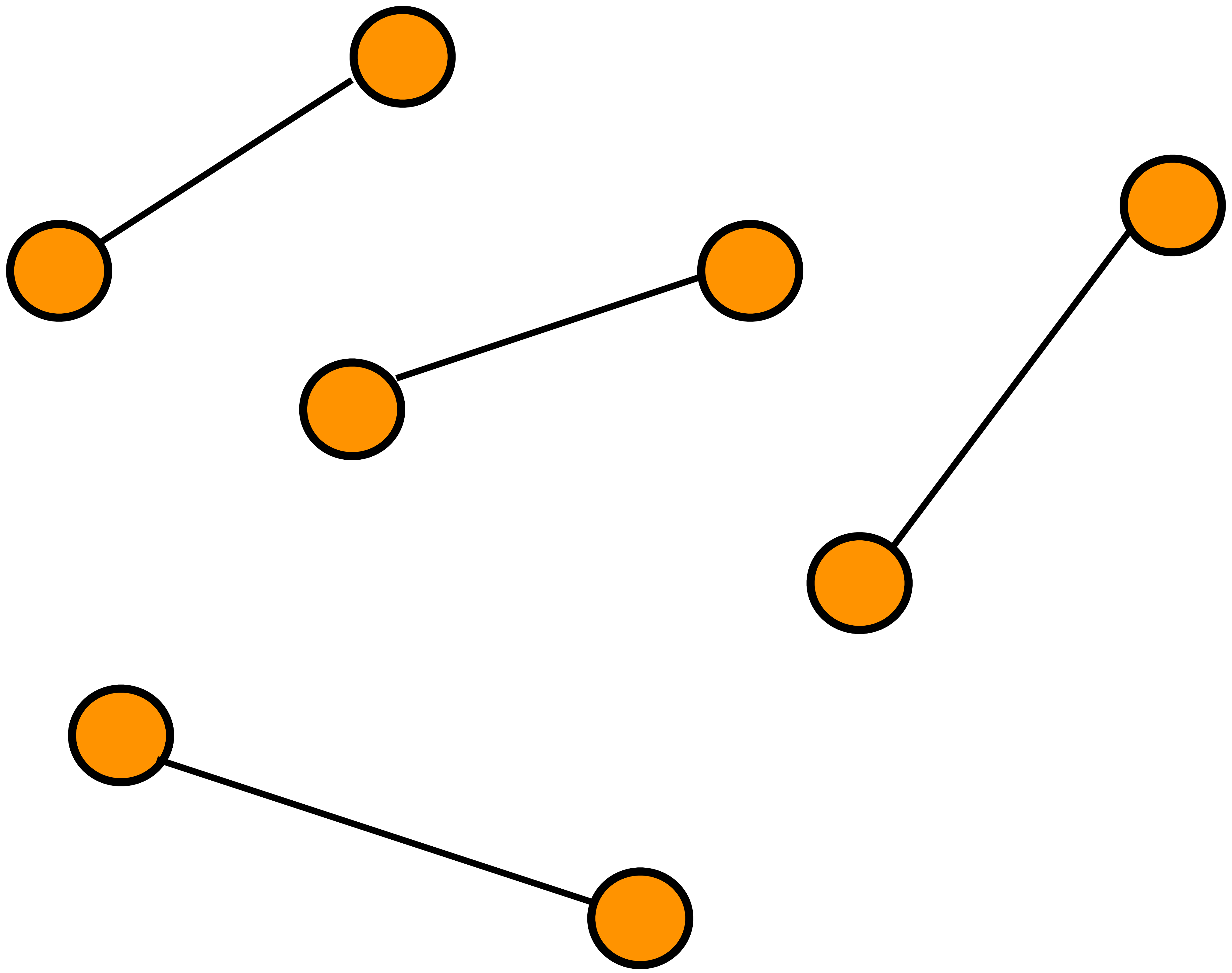
Matchings in Graphs



$$G = (V, E)$$

n vertices and m edges

Matchings in Graphs



$M \subseteq E$ of edges

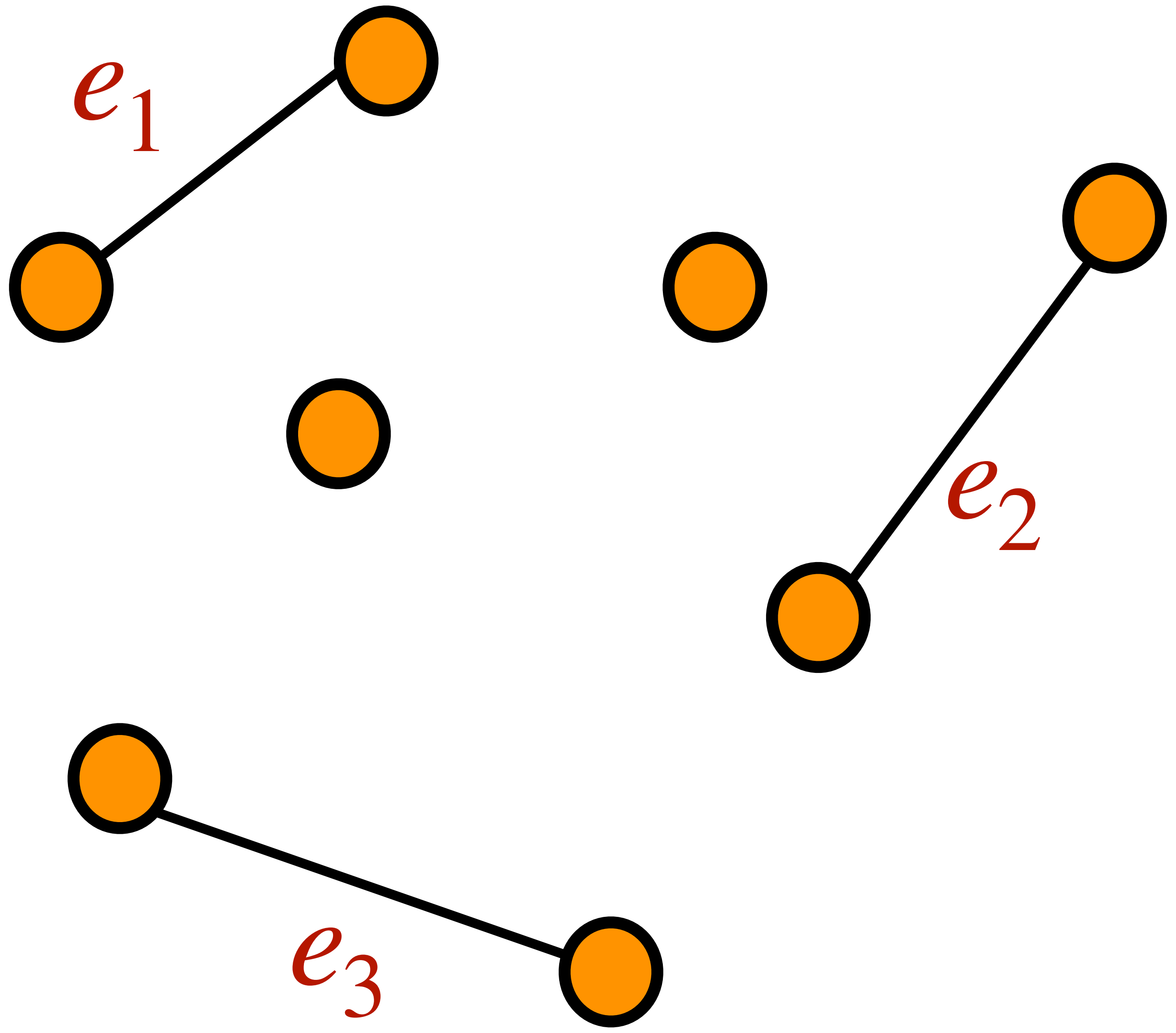
At most one edge
incident to every vertex

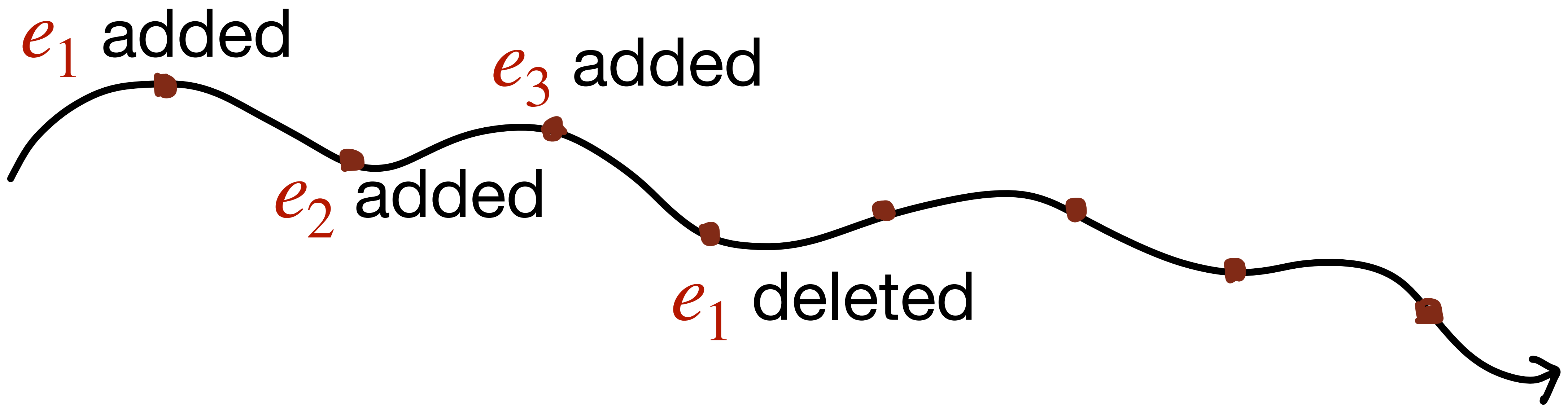
Dynamic Streams

[Ahn-Guha-McGregor '12]

Vertices are known.

$$G = (V, E)$$





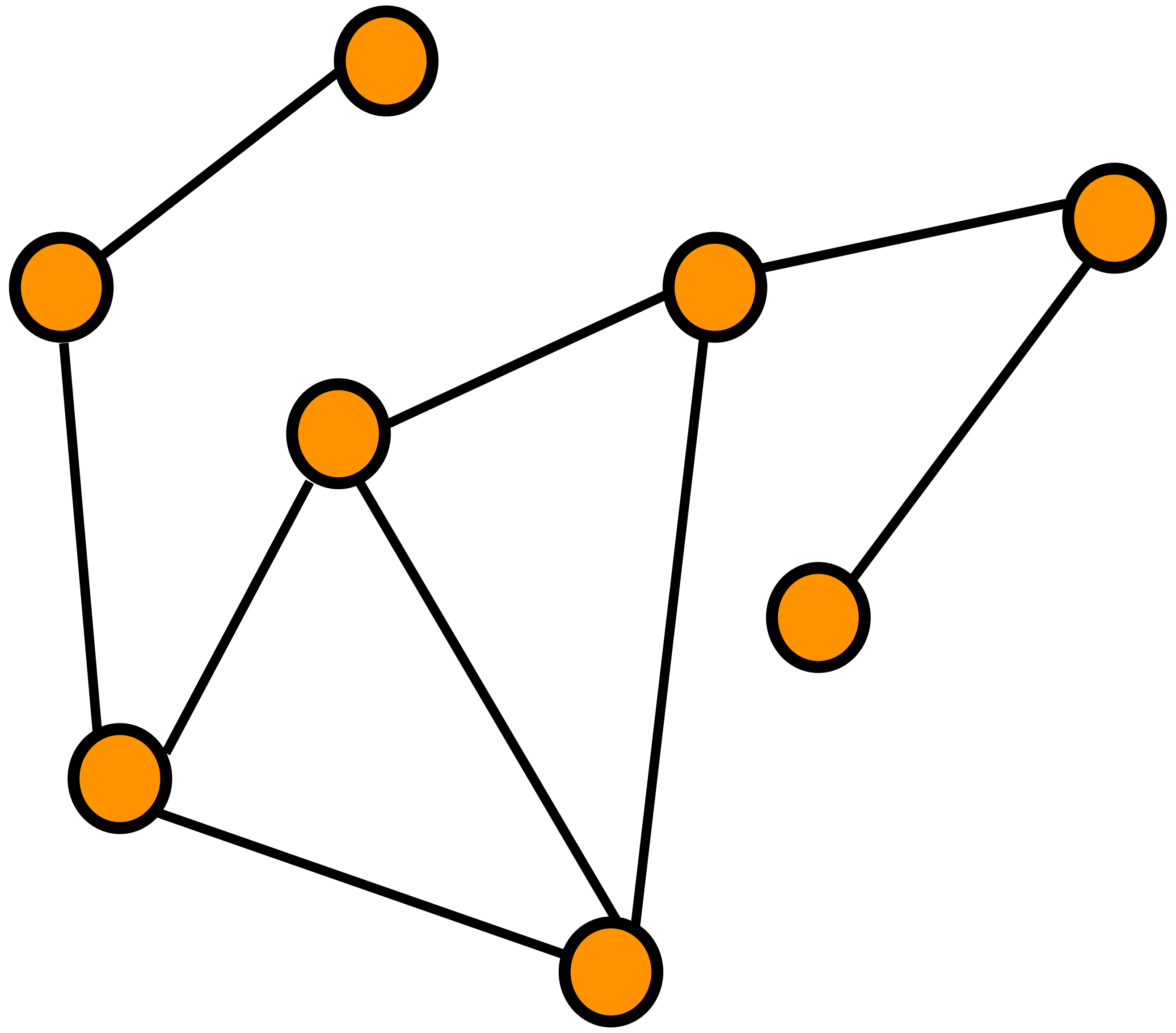
Edges are given as a stream of insertions and deletions

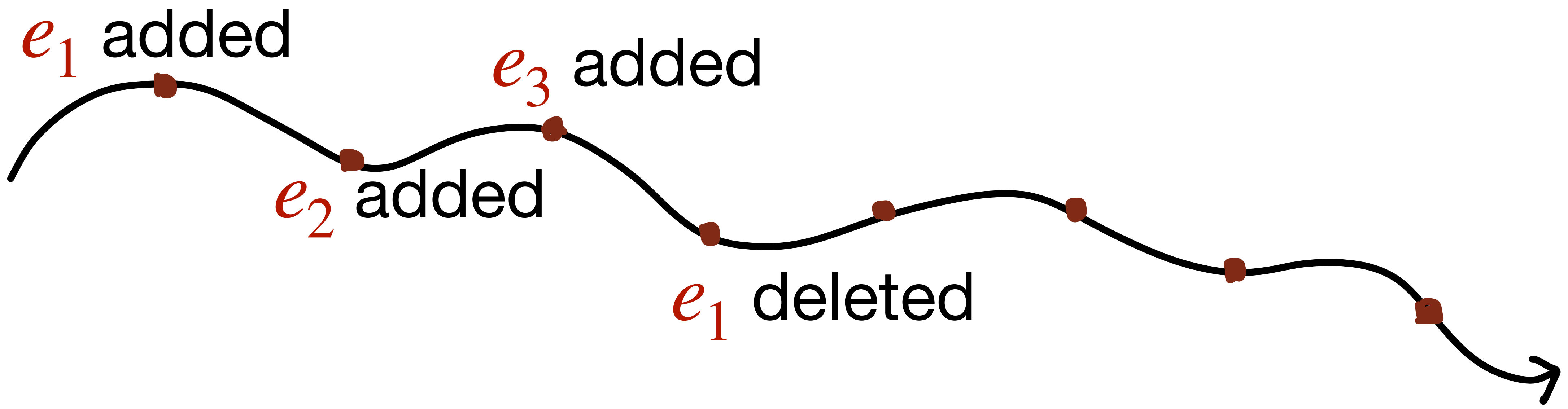
Dynamic Streams

[Ahn-Guha-McGregor '12]

Vertices are known.

$$G = (V, E)$$





Edges are given as a stream of insertions and deletions

Space: $O(n \text{ poly} \log n)$

May be repeated for multiple passes

Matching in Dynamic Streams

- Multi-Pass Algorithms

  [Ahn-Guha-McGregor ’12, Ahn-Guha ’15, Assadi ’24]

- Single Pass Results

  [Konrad ’15, Chitnis-Cormode-Hajiaghayi-Monemizadeh ’15, Assadi-Khanna-Li-Yaroslavtsev ’16, Chitnis-Cormode-Esfandiari-Hajiaghayi-McGregor-Monemizadeh-Vorotnikova ’16, Assadi-Khanna-Li ’17, Dark-Konrad ’20, Assadi-Shah ’22]

Focus on $O(1)$-approx

Boosting Approximation Ratio

Any algorithm for matching

$O(1)$-approximation
p passes
s space

Unweighted

[McGregor '05,
Gamlath-Kale-Mitrovic-Svensson '19]

$(1+\epsilon)$-approximation
$O_\epsilon(p)$ passes
$O_\epsilon(s)$ space

Weighted

Main Question

Number of passes?

Semi-streaming space: $O(n \text{ poly} \log n)$ $O(1)$-approximation ratio

Prior Work on $O(1)$-approx

[Lattanzi-Moseley-Suri-Vassilvitskii-11,
Ahn-Guha-McGregor '12, Ahn-Guha '15, Assadi '24]

Number of Passes

$O(\log n / \log \log n)$

This Work

$\Theta(\log \log n)$

> 1

[Assadi-Khanna-Li-Yaroslavtsev '16 , Dark-Konrad '20]

Our Results

In $O(n \text{ poly} \log n)$ space, there is an $O(\log \log n)$-pass algorithm for $O(1)$-approximation of maximum matching.

Reminder: We can boost to $(1+\epsilon)$ for any constant ϵ

In $O(n \text{ poly} \log n)$ space, for any constant $c > 1$, any c-approximation of maximum matching requires $\Omega(\log \log n)$ passes.

This Talk: Only Upper Bound

In $O(n \text{ poly} \log n)$ space, there is a $O(\log \log n)$-pass algorithm for $O(1)$-approximation of maximum matching.

This is a sketching algorithm.

Graph Sketching Technique

Sketch of the graph

$S \cdot \phi \in \mathbb{Z}^s$

$$\begin{bmatrix} \\ \text{Sketch Matrix} \\ S \in \mathbb{Z}^{s \times \binom{n}{2}} \\ \\ \end{bmatrix} \begin{bmatrix} (v_1, v_2) \\ (v_1, v_3) \\ \\ \\ \\ (v_{n-1}, v_n) \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

Incidence Vector of edges

$\phi \in \{0,1\}^{\binom{n}{2}}$

Graph Sketching Technique

Sketch of the graph

$S \cdot \phi \in \mathbb{Z}^s$

$$\begin{bmatrix} & \text{Not explicitly stored} & \\ & \text{Sketch Matrix} & \\ & S \in \mathbb{Z}^{s \times \binom{n}{2}} & \\ & & \end{bmatrix} \begin{bmatrix} (v_1, v_2) \\ (v_1, v_3) \\ \\ \\ \\ (v_{n-1}, v_n) \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

Stored in $\tilde{O}(s)$

Incidence Vector of edges

$\phi \in \{0,1\}^{\binom{n}{2}}$

For one Edge Insertion or Deletion

$$S \cdot \phi_1 + S \cdot \phi_2 = S \cdot (\phi_1 + \phi_2)$$

Update

Sketch of current graph

Sketch of updated graph

Just find $S \cdot \phi_{\text{update}}$ and add to current sketch

Multi-Round Adaptive Sketching

Decide on sketch matrix
S_1 for first round

Get sketch of graph $S_1 \cdot \phi$

Decide next sketch matrix S_2
for second round
based on $S_1 \cdot \phi$

Continue for next round

r-round adaptive sketching gives
r-pass dynamic streaming algorithm

Sketching Algorithm

In $O(n \text{ poly} \log n)$ space, there is an $O(\log \log n)$-pass algorithm for $O(1)$-approximation of maximum matching.

Adaptive sketching algorithm

$\tilde{O}(n)$-size sketches

$O(\log \log n)$-rounds

Implication for MPC model

[Czumaj-Lacki-Madry-Mitrovic-Onak-Sankowski ’18,
Ghaffari-Gouleakis-Konrad-Mitrovic-Rubinfeld ’18,
Assadi-Bateni-Bernstein-Mirrokni-Stein ’19,
Behnezhad-Hajiaghayi-Harris ’19]

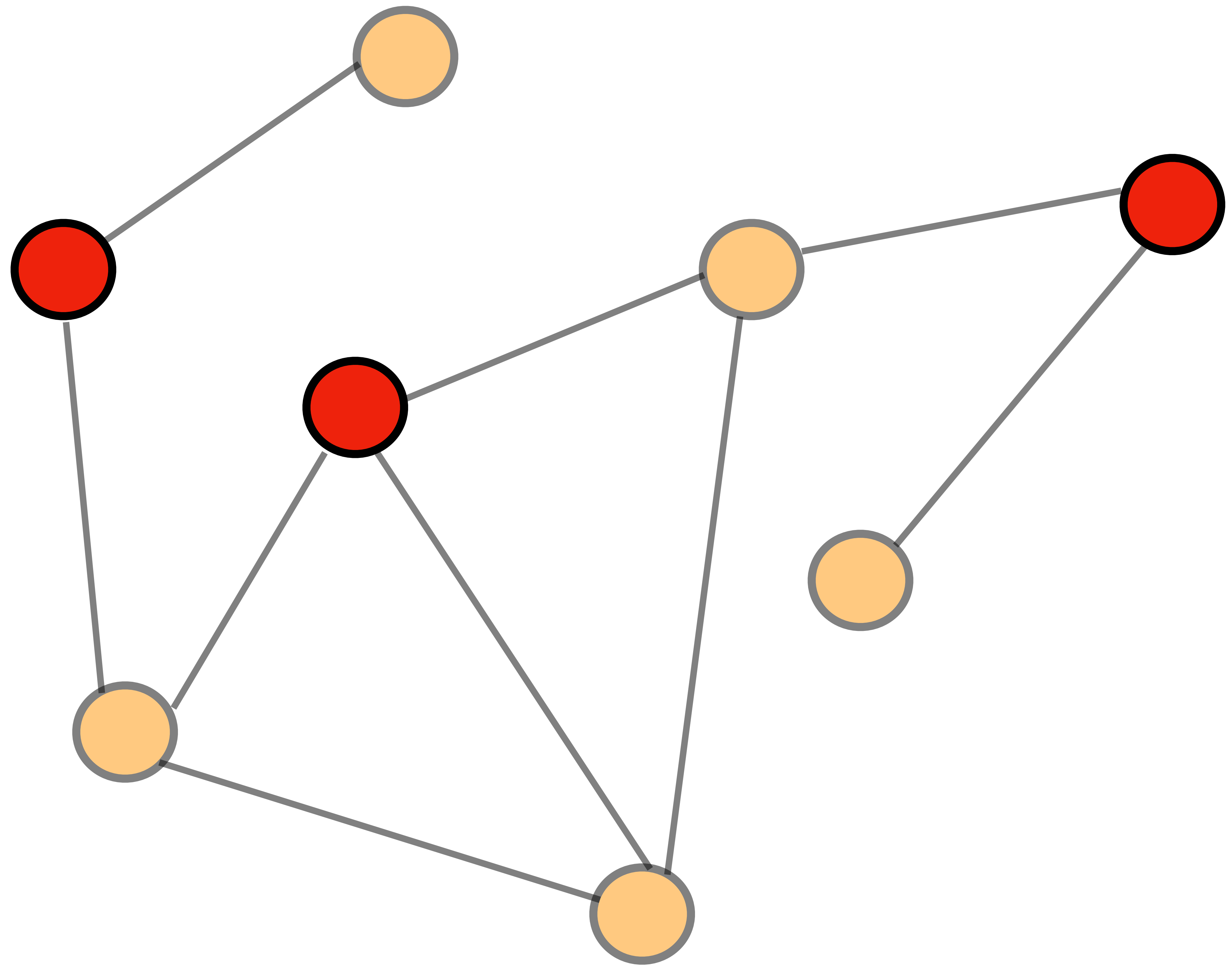
MPC algorithm with machines of $\tilde{O}(n)$ memory and $\tilde{O}(n)$ working memory in $O(\log \log n)$ rounds

Our Techniques

One Phrase Summary

Connection between Matching and Maximal Independent Sets

Maximal Independent Set (MIS)

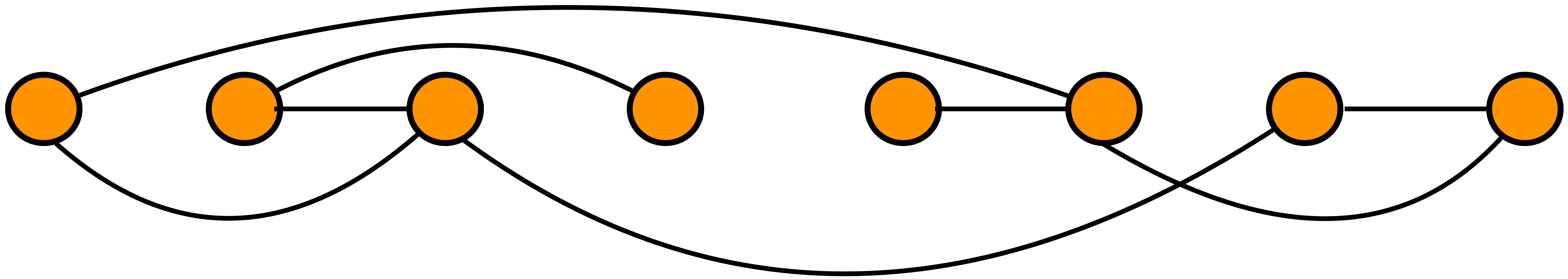


Any independent set
which is NOT a proper subset
of another independent set

Finding them is easy!

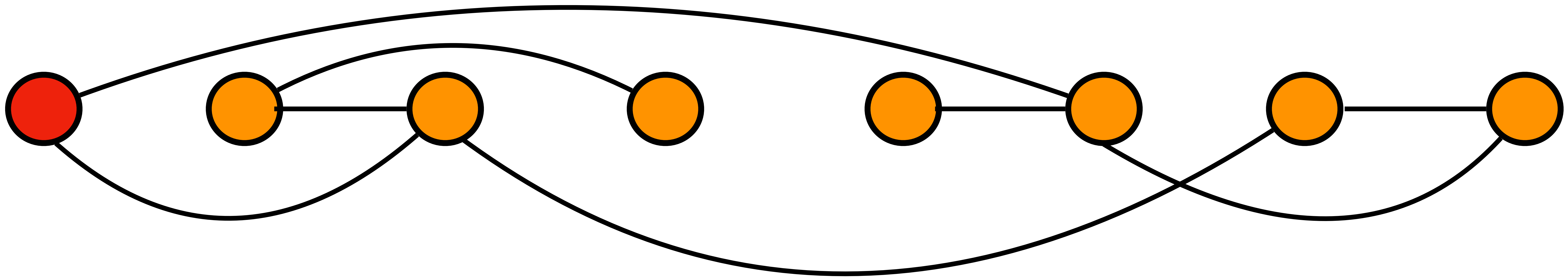
Greedy MIS

- Pick an arbitrary ordering of vertices



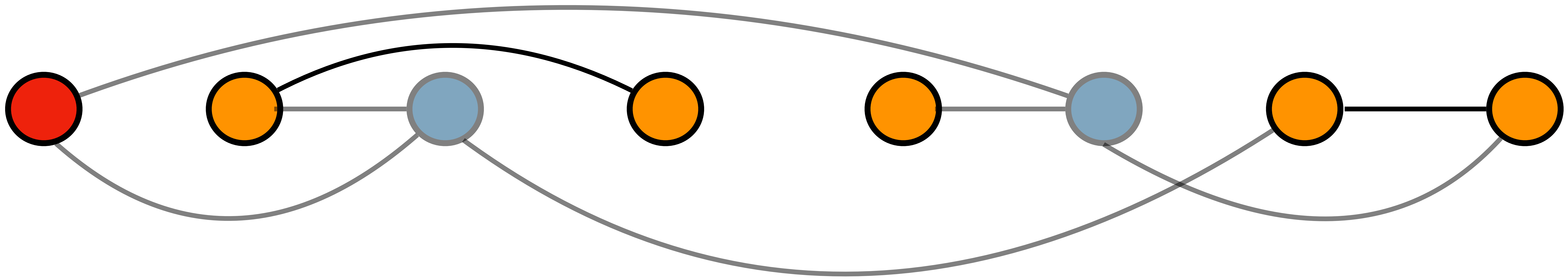
Greedy MIS

- Pick an arbitrary ordering of vertices
- Add the first existing vertex in the ordering to MIS



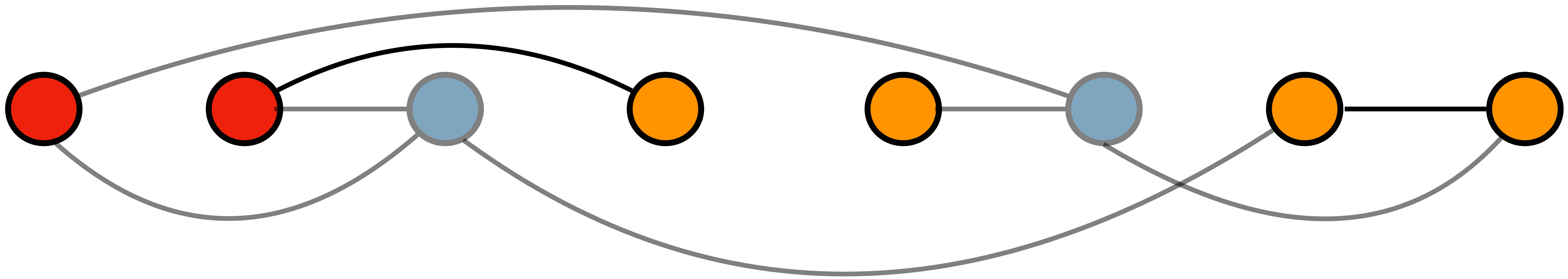
Greedy MIS

- Pick an arbitrary ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Remove its neighbors and their edges



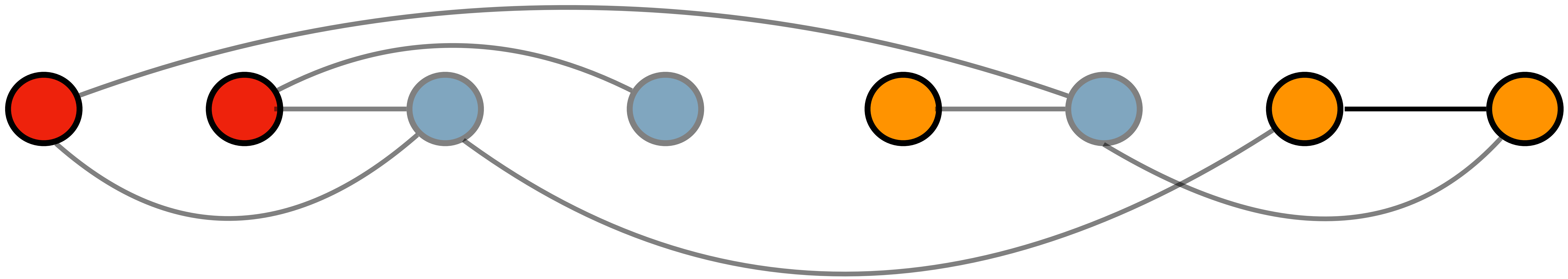
Greedy MIS

- Pick an arbitrary ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Remove its neighbors and their edges
- Repeat the process among remaining vertices



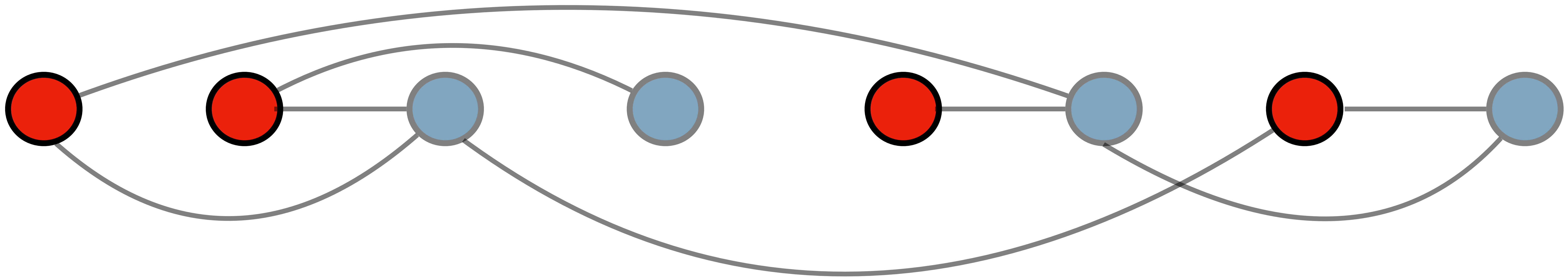
Greedy MIS

- Pick an arbitrary ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Remove its neighbors and their edges
- Repeat the process among remaining vertices



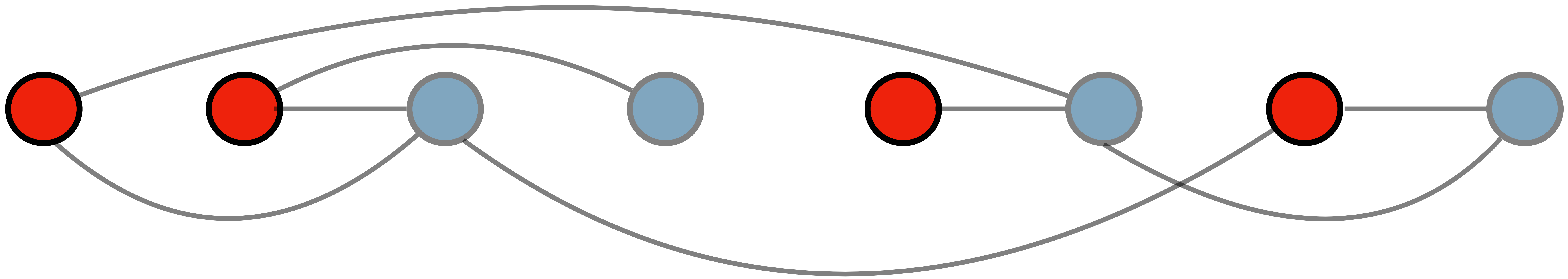
Greedy MIS

- Pick an ~~arbitrary~~ ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Remove its neighbors and their edges
- Repeat the process among remaining vertices



Random Greedy MIS (RGMIS)

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Remove its neighbors and their edges
- Repeat the process among remaining vertices



Random Greedy MIS (RGMIS)

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Remove its neighbors and their edges
- Repeat the process among remaining vertices

Can be implemented in $O(\log \log n)$-passes in $\tilde{O}(n)$ space

[Ahn-Cormode-Guha-McGregor-Wirth '15]

This is tight [Assadi-Konrad-Naidu-S '24]

Back to Our Results …

Our Results

MIS in $O(\log \log n)$ passes
[Ahn-Cormode-Guha-McGregor-Wirth '15]

$\longrightarrow$

$O(\log \log n)$ upper bound
for matchings
in dynamic streams

Machinery developed to
prove $\Omega(\log \log n)$ pass
lower bound for MIS
[Assadi-Konrad-Naidu-S '24]

$\longrightarrow$

$\Omega(\log \log n)$ lower bound
for matchings
in dynamic streams

Insertion Only Streams

Plan for the rest of the talk

- Fractional matching and Vertex Cover
- Connections to MIS
- Our reduction to MIS
- Challenges of implementation

Plan for the rest of the talk

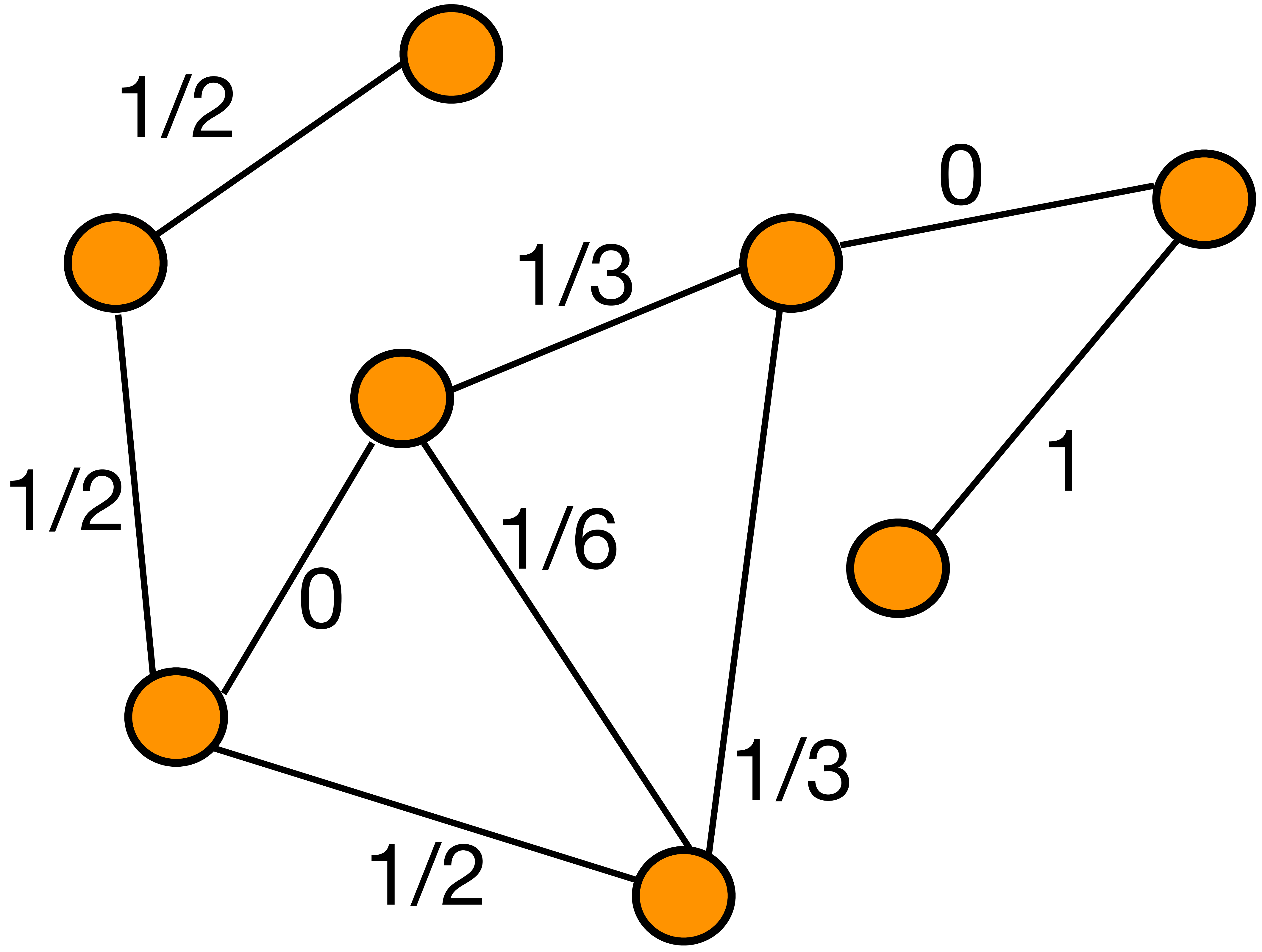
- **Fractional matching and Vertex Cover**
- Connections to MIS
- Our reduction to MIS
- Challenges of implementation

Fractional Matching

$$\max \sum_{e} x_e$$

For all vertices u, $\sum_{e \ni u} x_e \leq 1$

With $0 \leq x_e \leq 1$ for all edges



Fractional Matching

$$\max \sum_e x_e$$

For all vertices u, $\sum_{e \ni u} x_e \leq 1$

With $0 \leq x_e \leq 1$ for all edges

Sample each edge w.p. $x_e \cdot O(\log n)$ independently

Set of sampled edges contains $O(1)$-approx integral matching

Swallows integrality gap

Fractional Matching

$$\max \sum_e x_e$$

For all vertices u, $\sum_{e \ni u} x_e \leq 1$

With $0 \leq x_e \leq 1$ for all edges

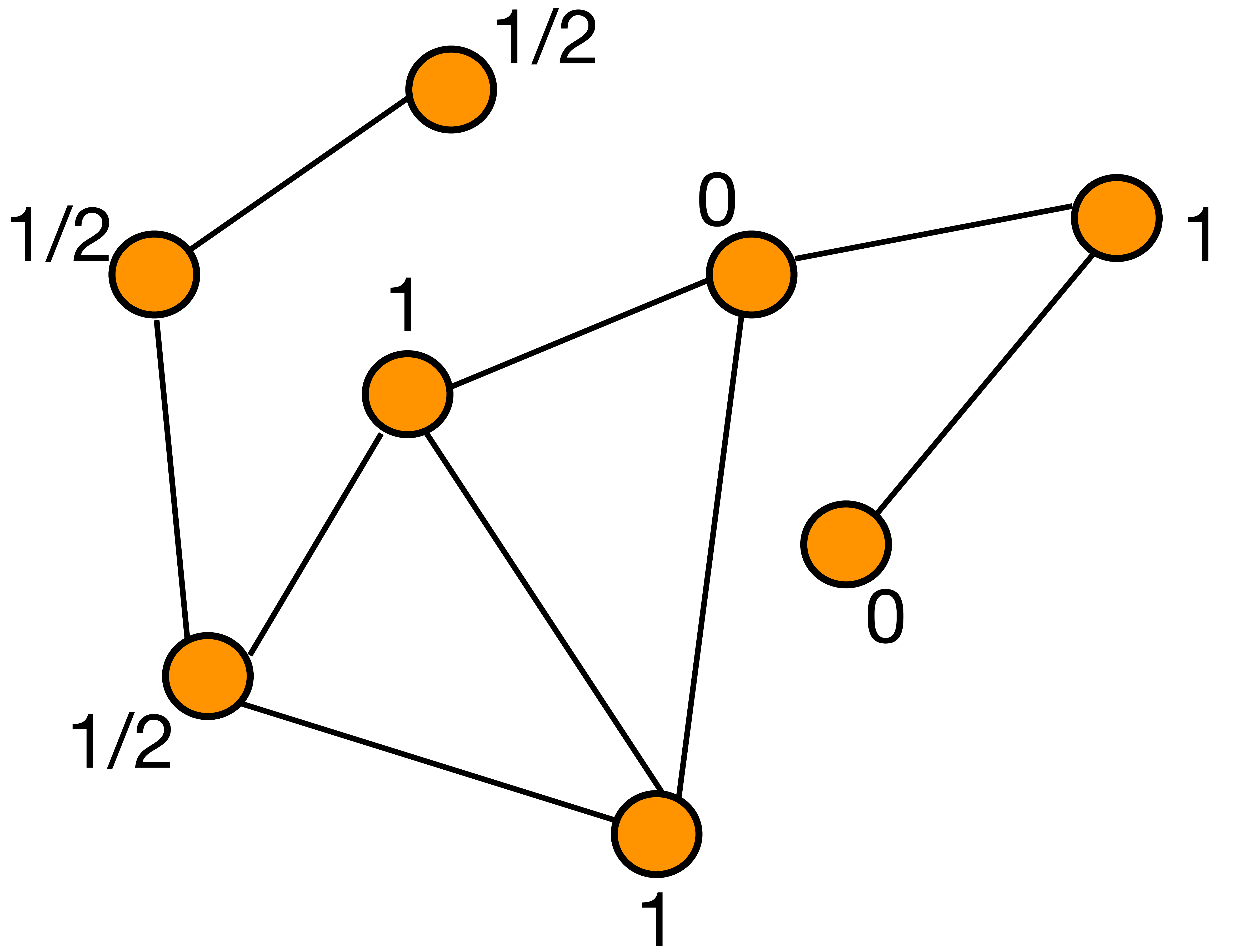
We try to find only
a fractional matching

Dual of Fractional Matching - Vertex Cover (VC)

$$\min \sum_u y_u$$

For all edges (u, v), $y_u + y_v \geq 1$

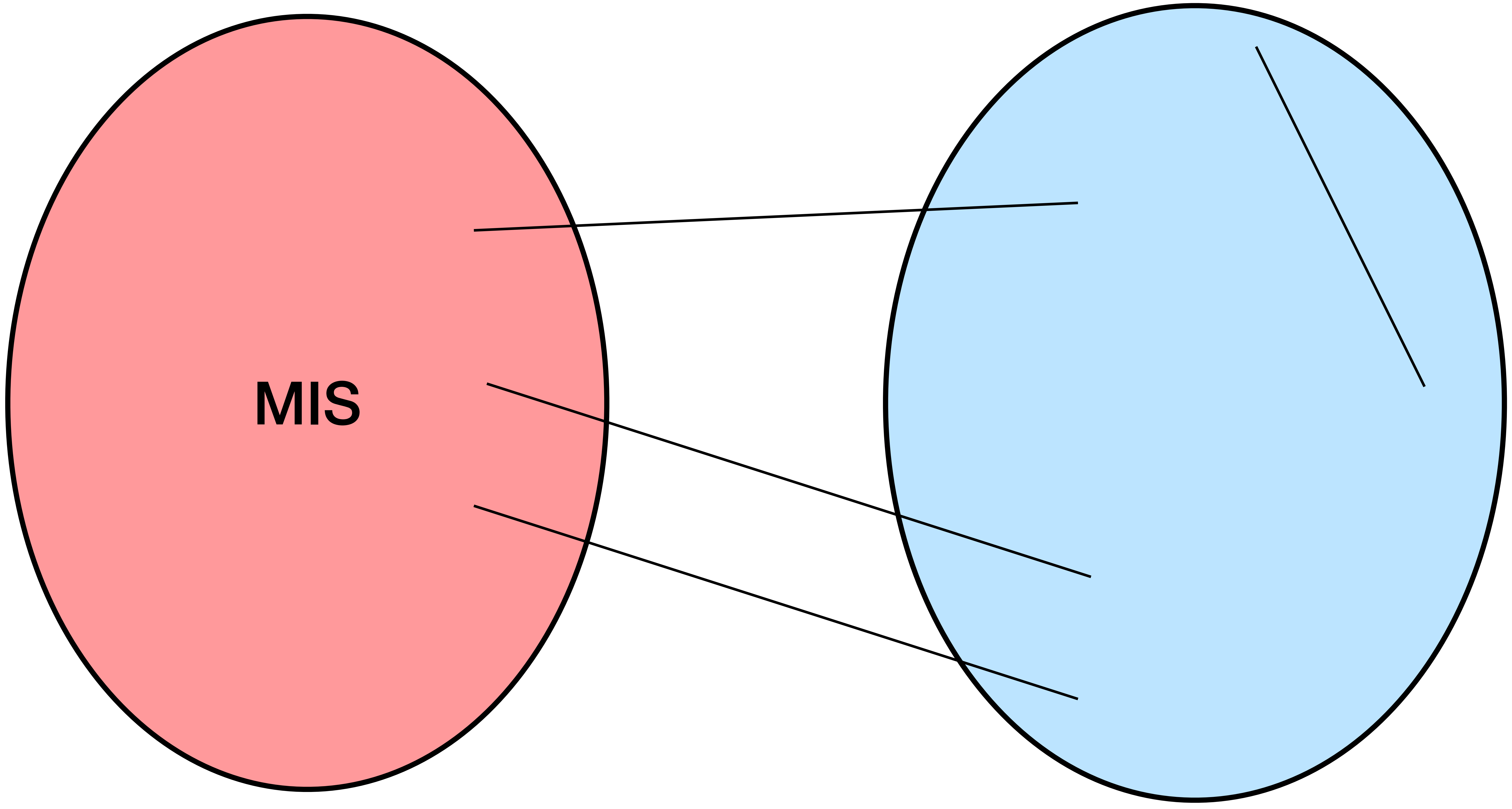
With $0 \leq y_u \leq 1$ for all vertices



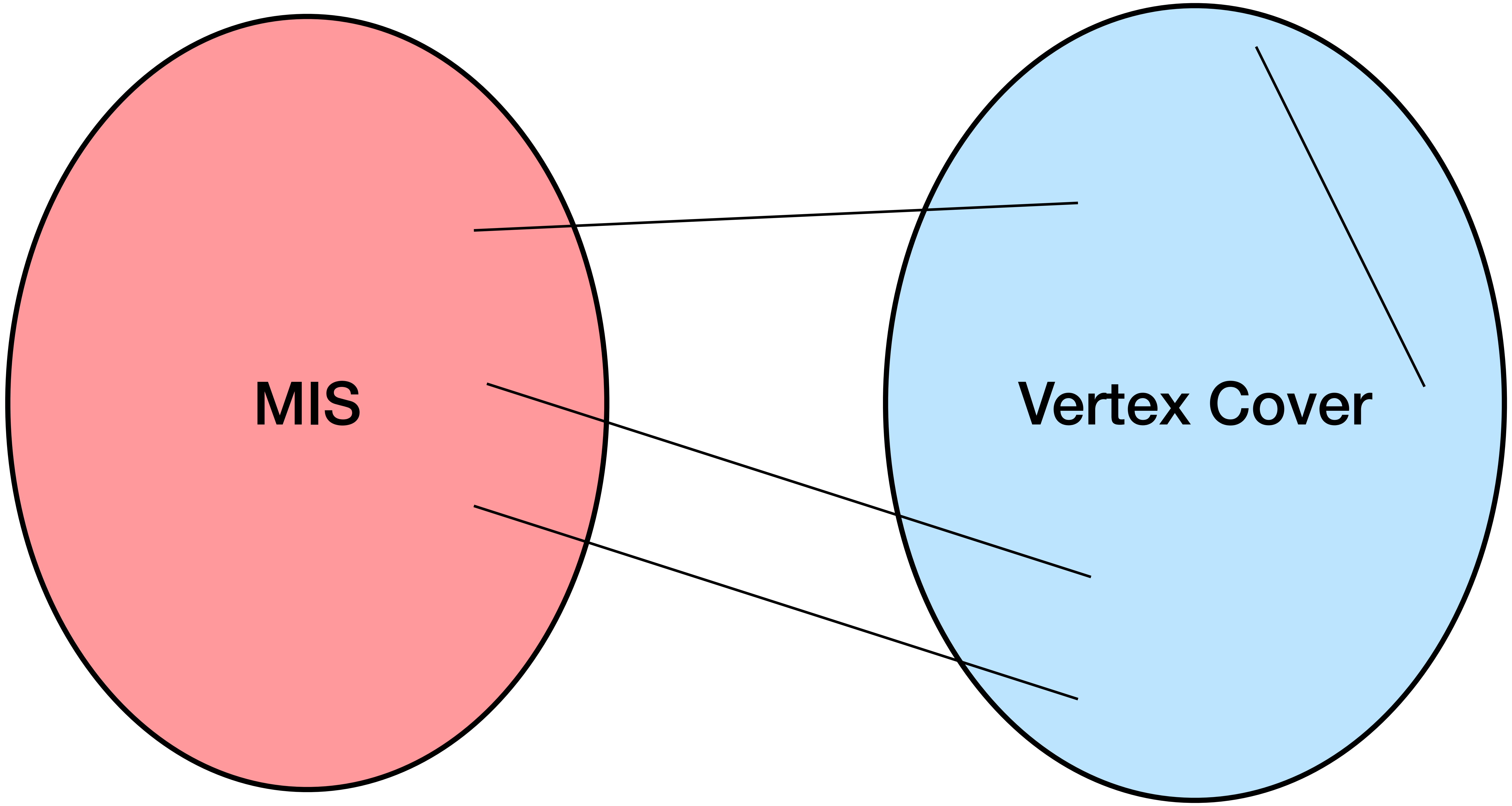
Plan for the rest of the talk

- Fractional matching and Vertex Cover
- **Connections to MIS**
- Our reduction to MIS
- Challenges of implementation

(Obvious) Connection to MIS?

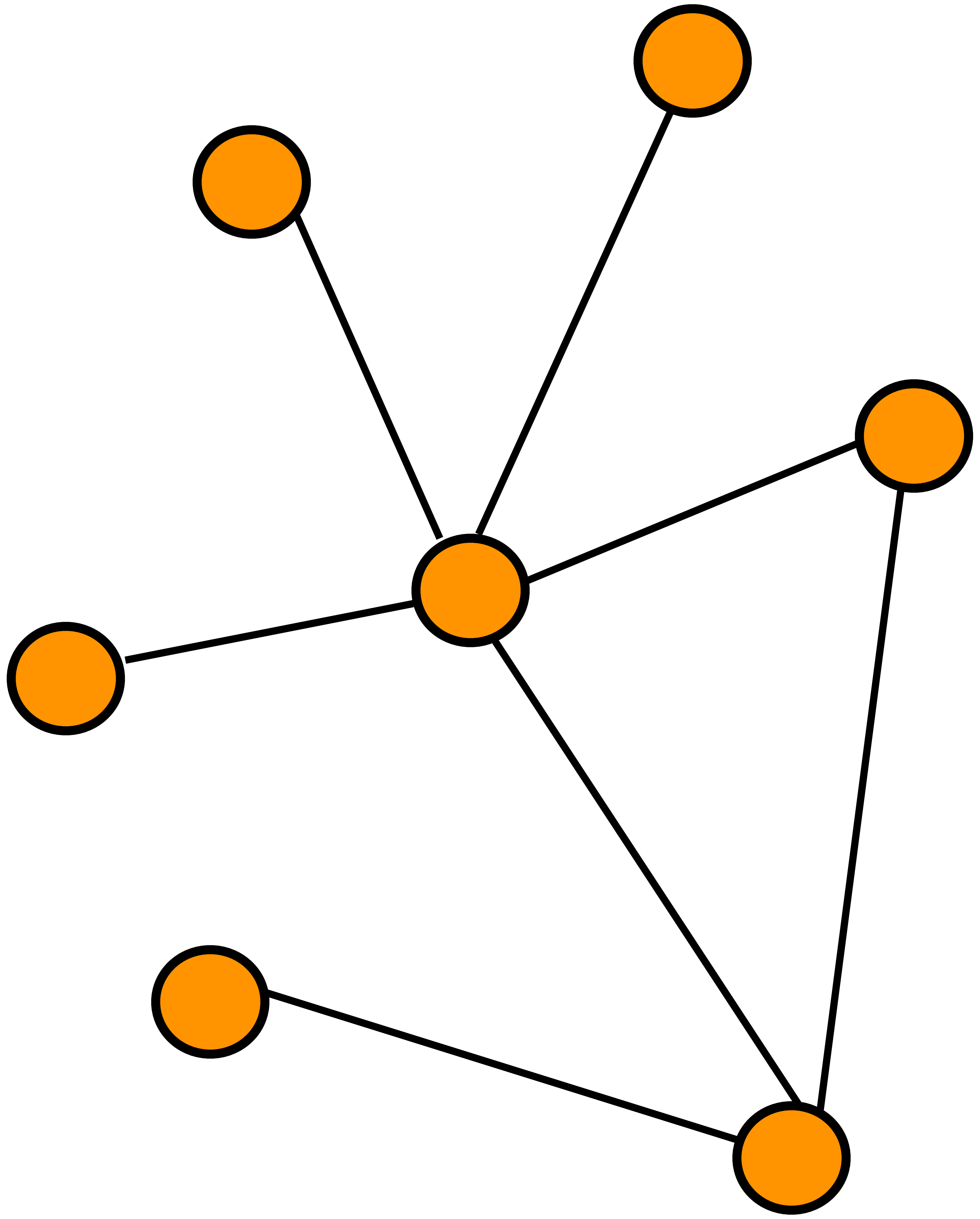


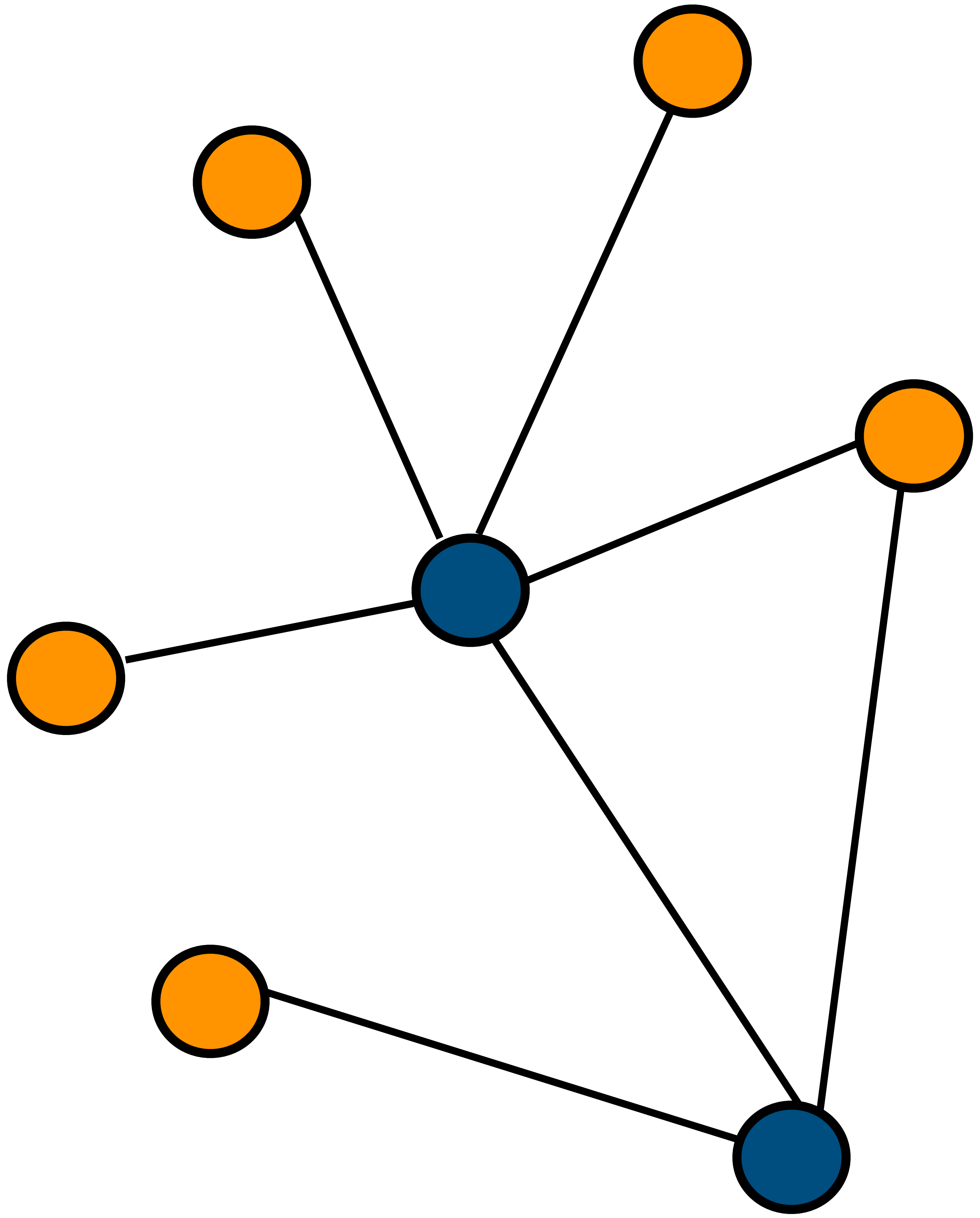
(Obvious) Connection to MIS?



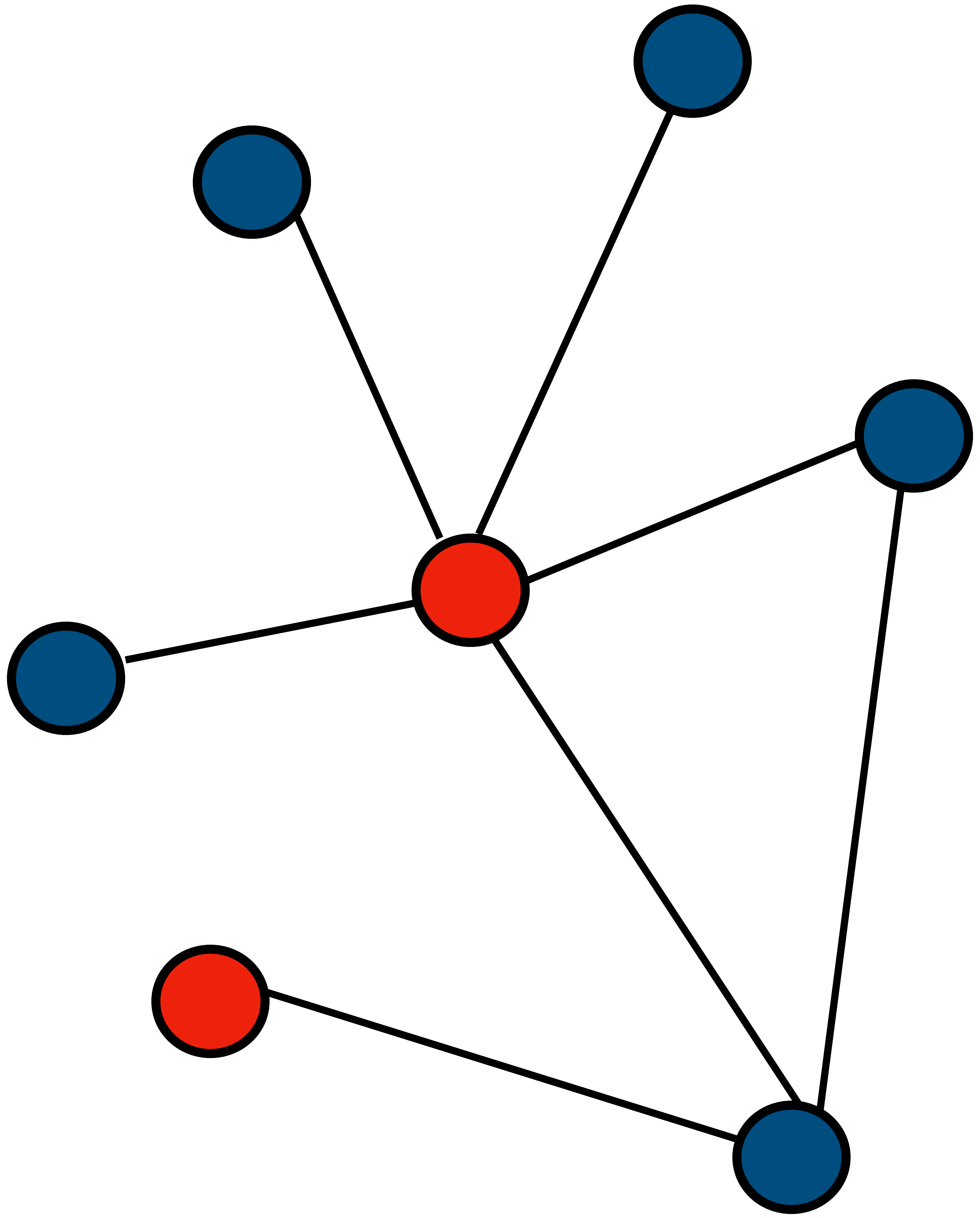
How good a vertex cover?

Generally not a good VC





Minimum VC

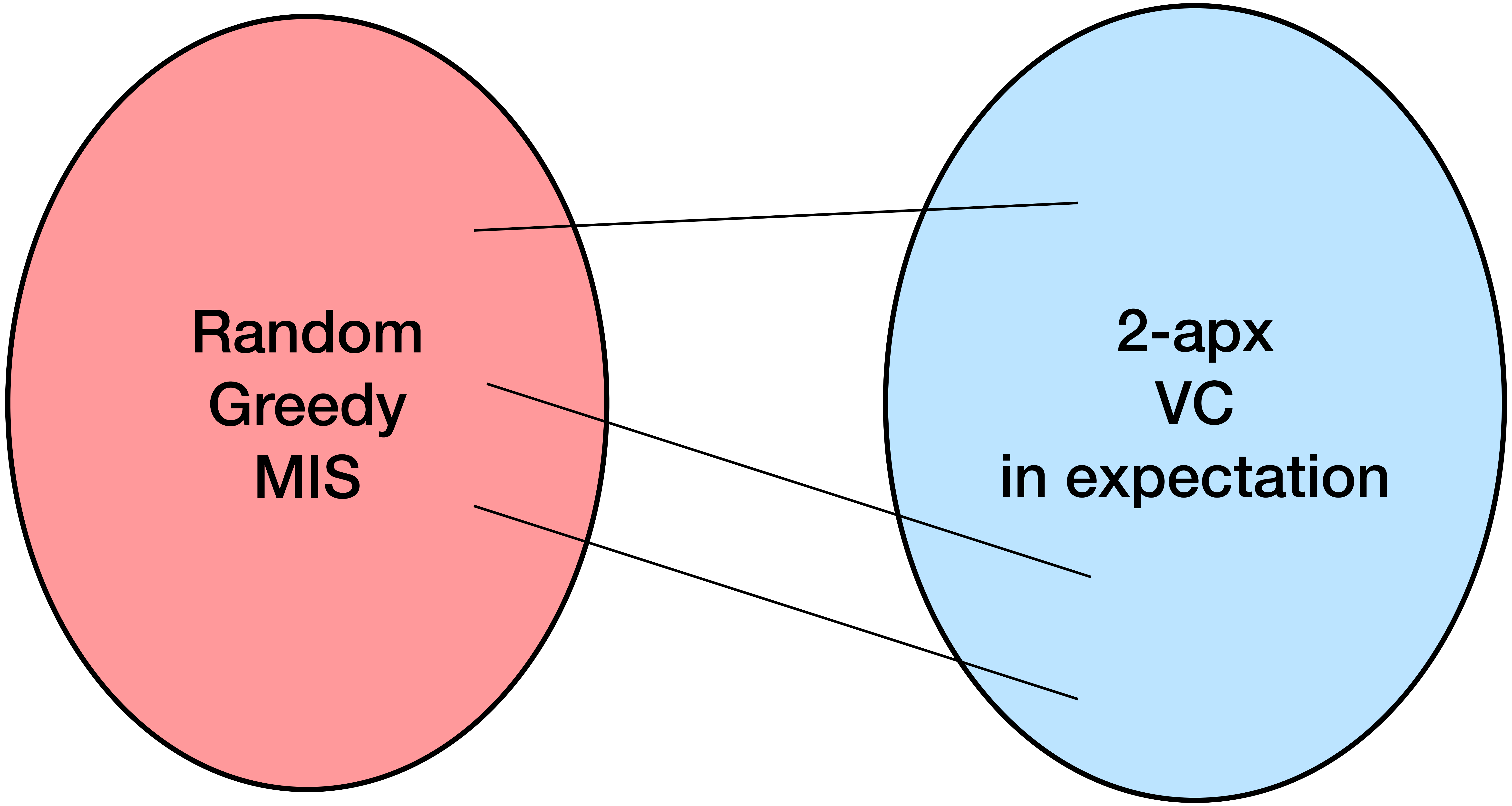


Complement of MIS

How about Complement of RGMIS?

Random Greedy MIS (RGMIS) - 2-apx VC

[Veldt '24]



Finding RGMIS

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Remove its neighbors and their edges
- Repeat the process among remaining vertices

Finding RGMIS

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices

Finding RGMIS

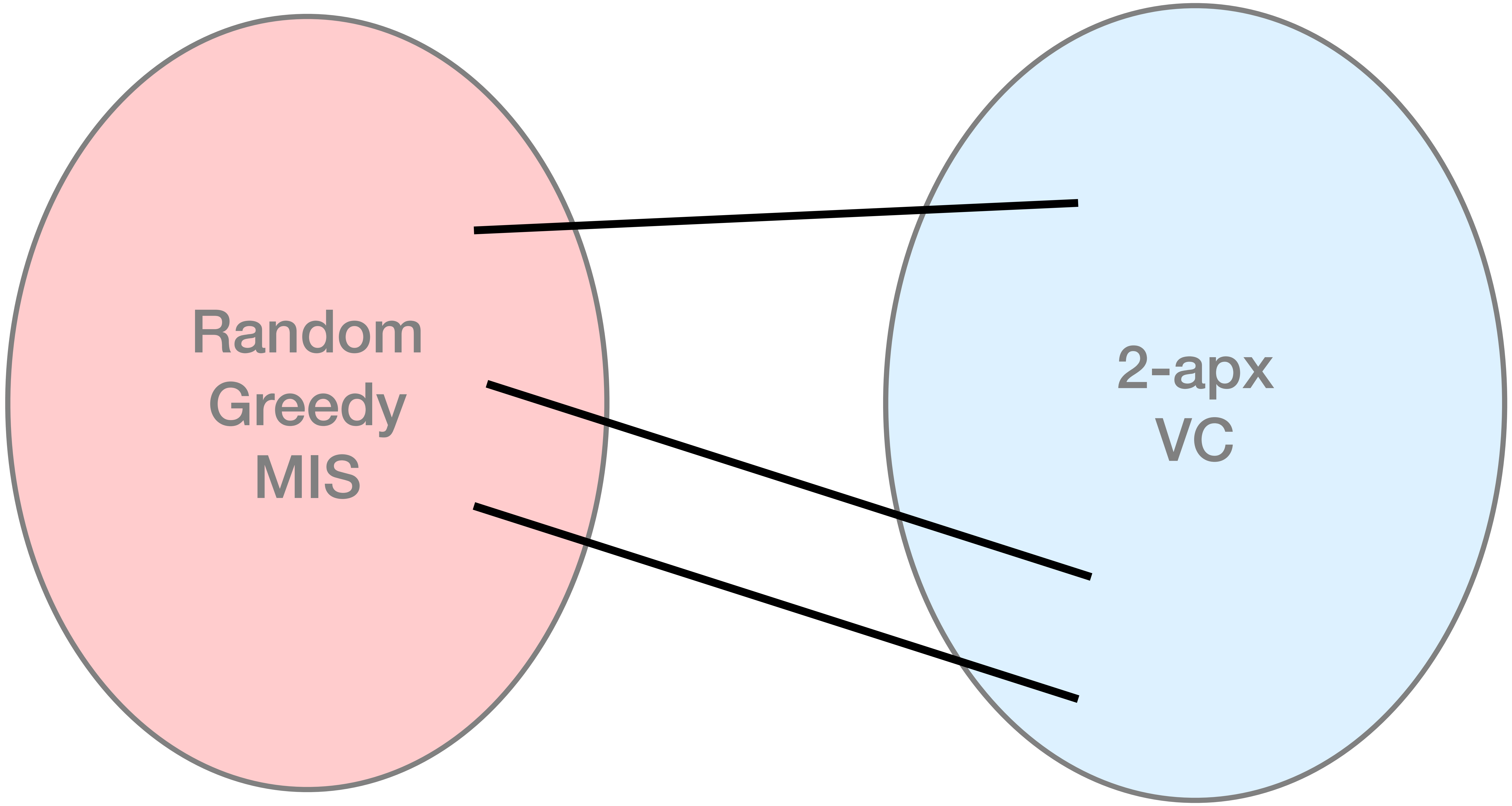
[Veldt ’24]

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices

This gives a 2-approx VC in expectation!

By an application of LP Duality

Proof of [Veldt '24]

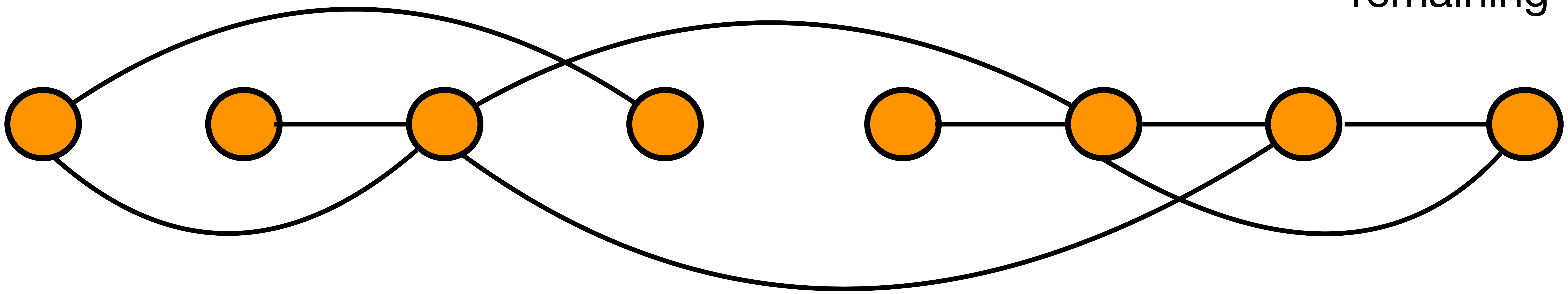


Focus on edges which "send" vertices to VC

Proof of [Veldt '24]

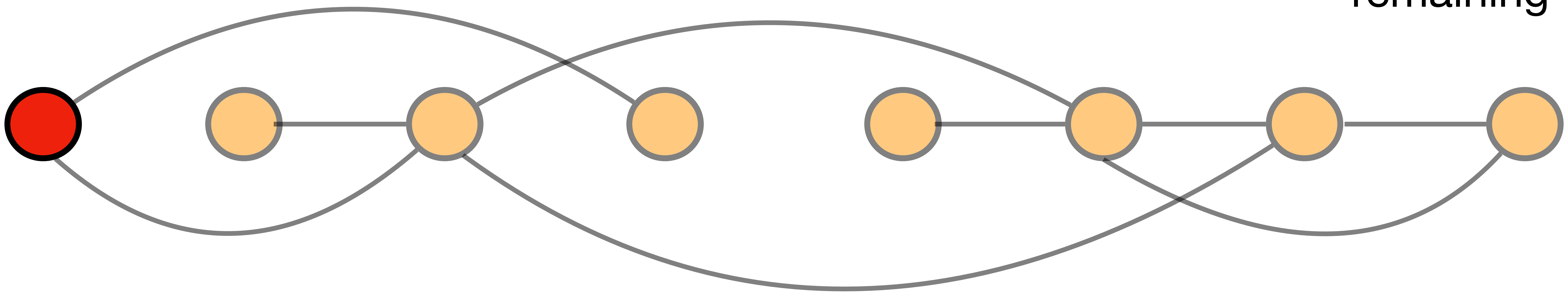
Process vertices in the order of the permutation

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices

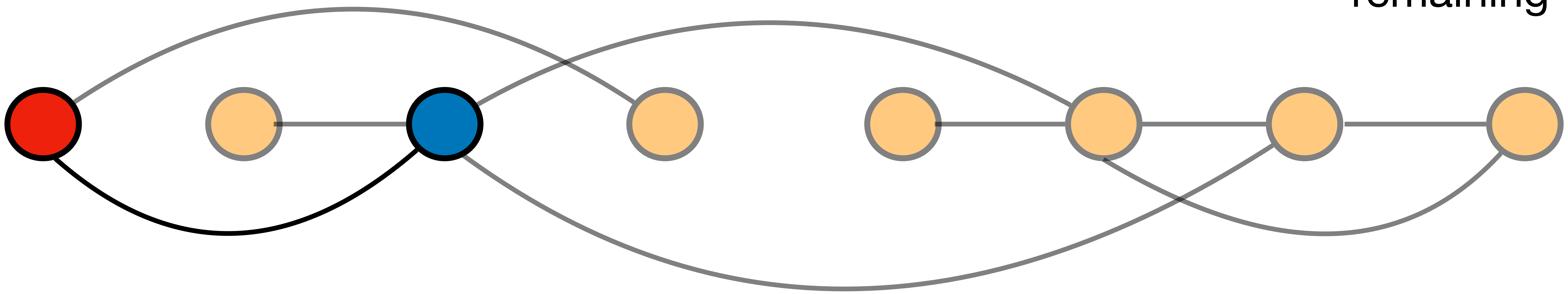


Proof of [Veldt '24]

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices



Proof of [Veldt '24]

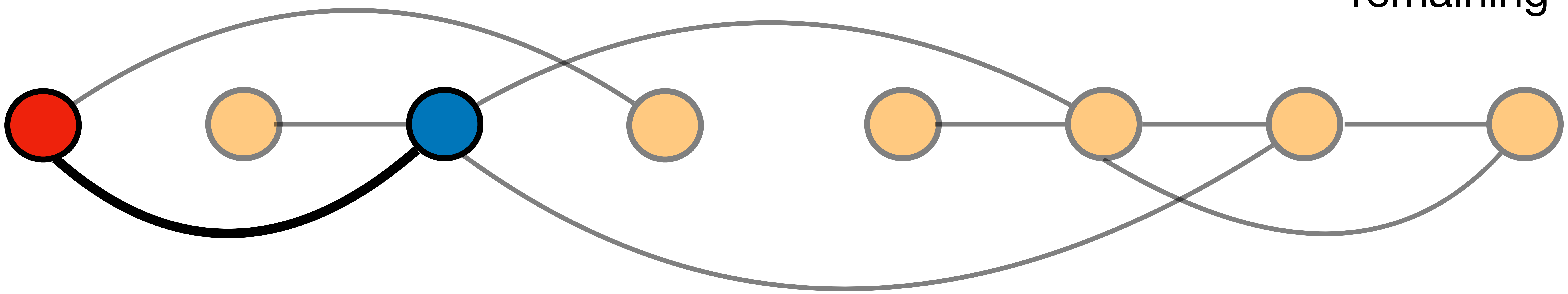


- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices

Proof of [Veldt '24]

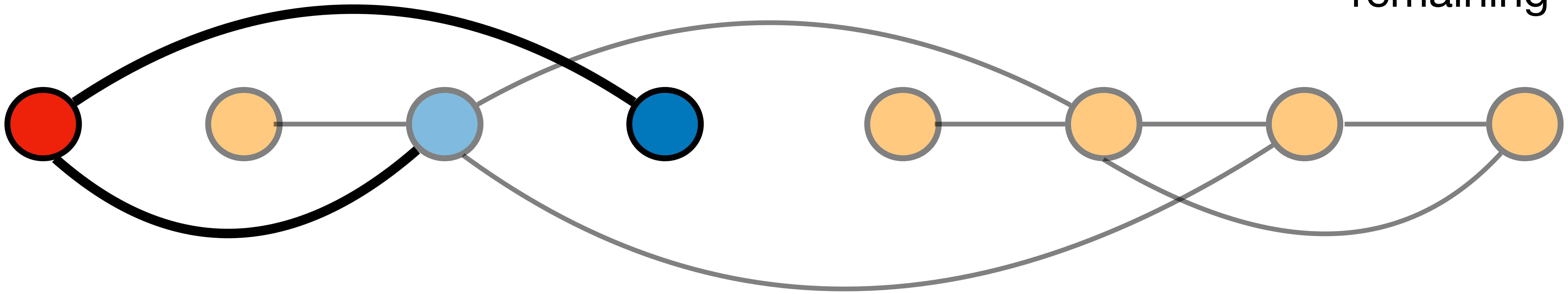
Edge which sends the blue vertex to VC

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices



Proof of [Veldt '24]

Edges which send blue vertices to VC

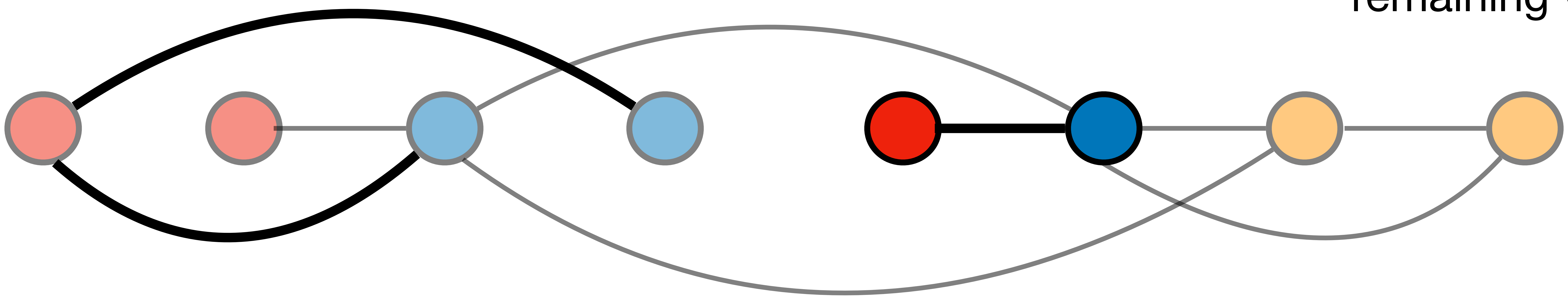


- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices

Proof of [Veldt '24]

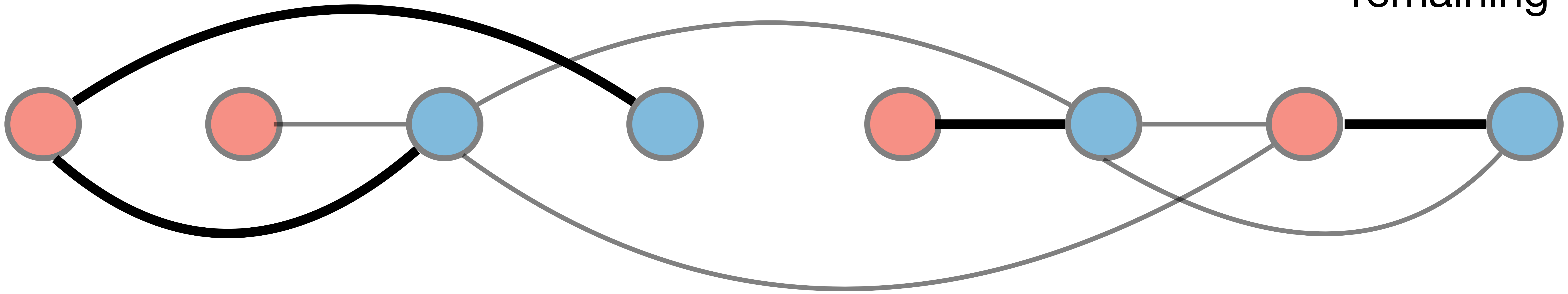
Edges which send blue vertices to VC

- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices



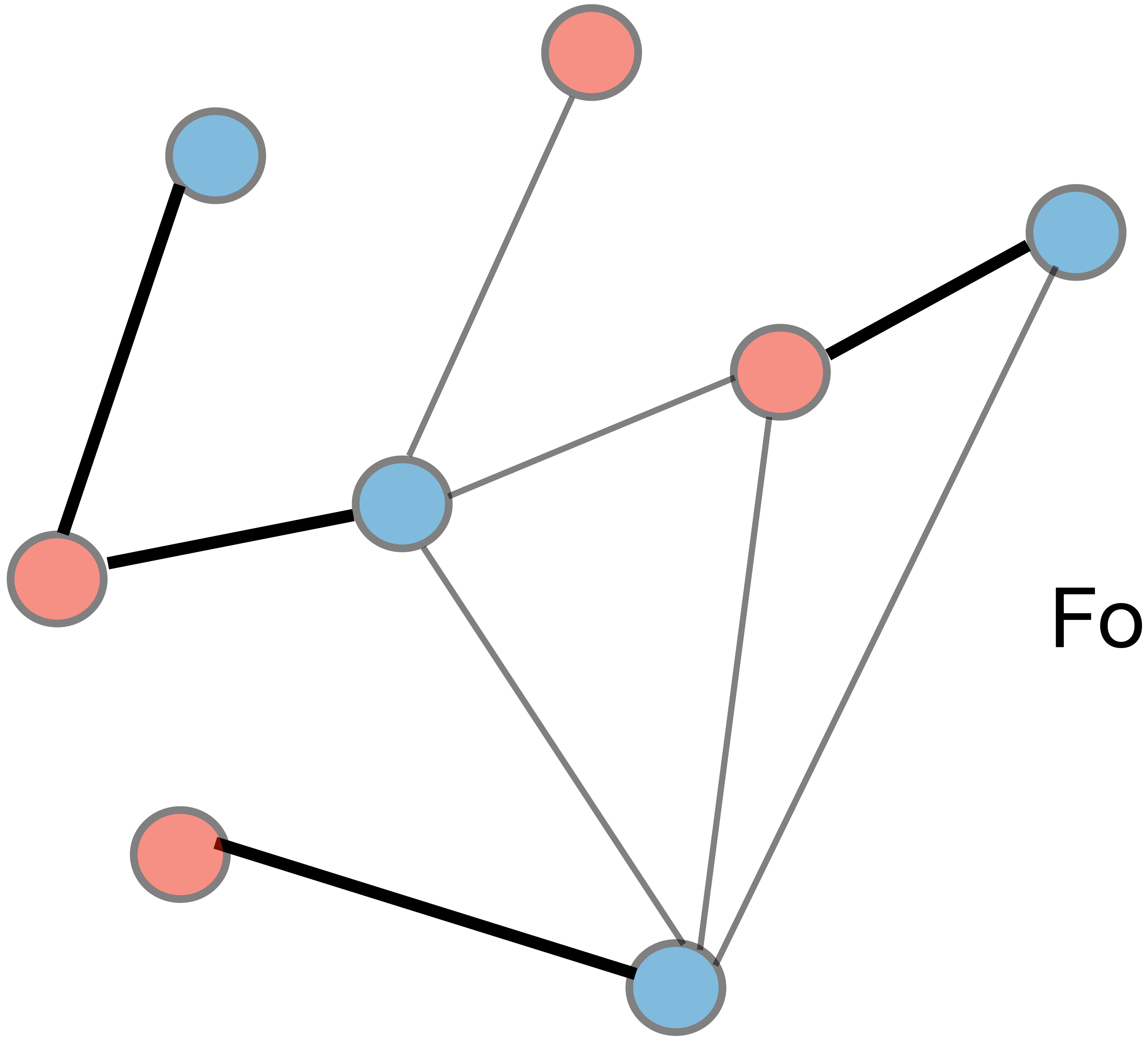
Proof of [Veldt '24]

After we process:



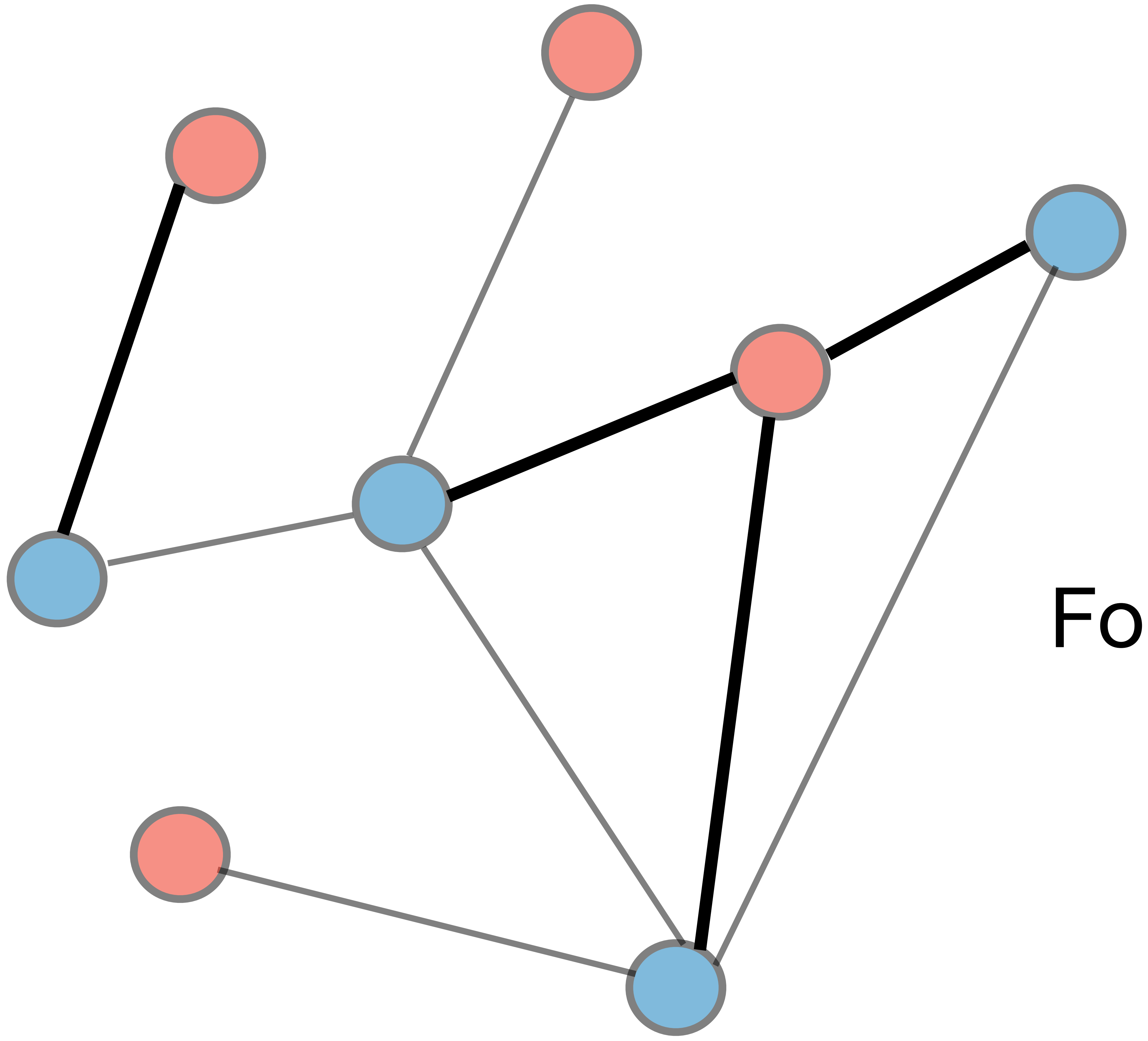
- Pick a random ordering of vertices
- Add the first existing vertex in the ordering to MIS
- Add its neighbors to VC and remove their edges
- Repeat the process among remaining vertices

In this random ordering …



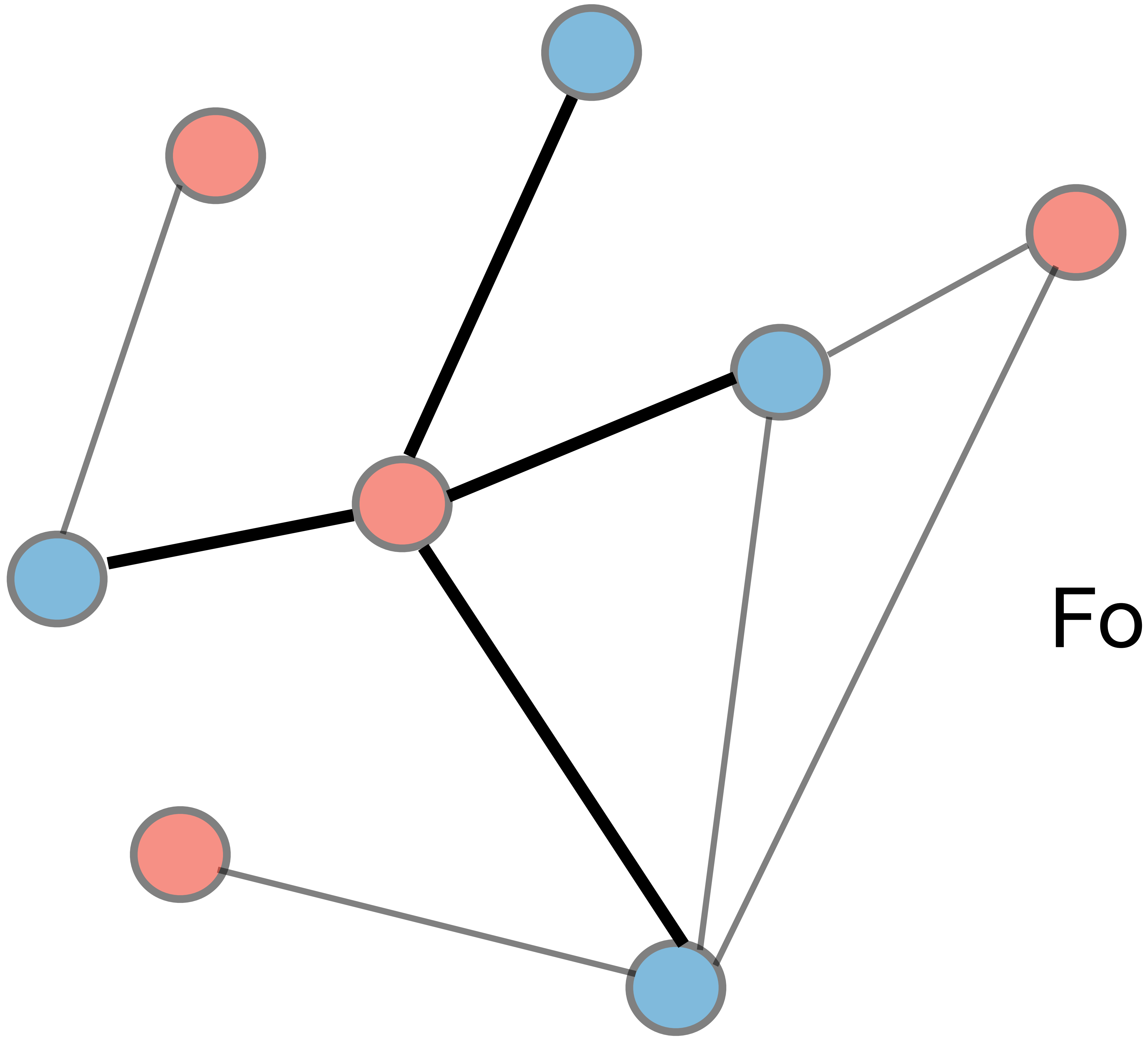
Focus on edges which “send” vertices to VC

In another random ordering …



Focus on edges which “send” vertices to VC

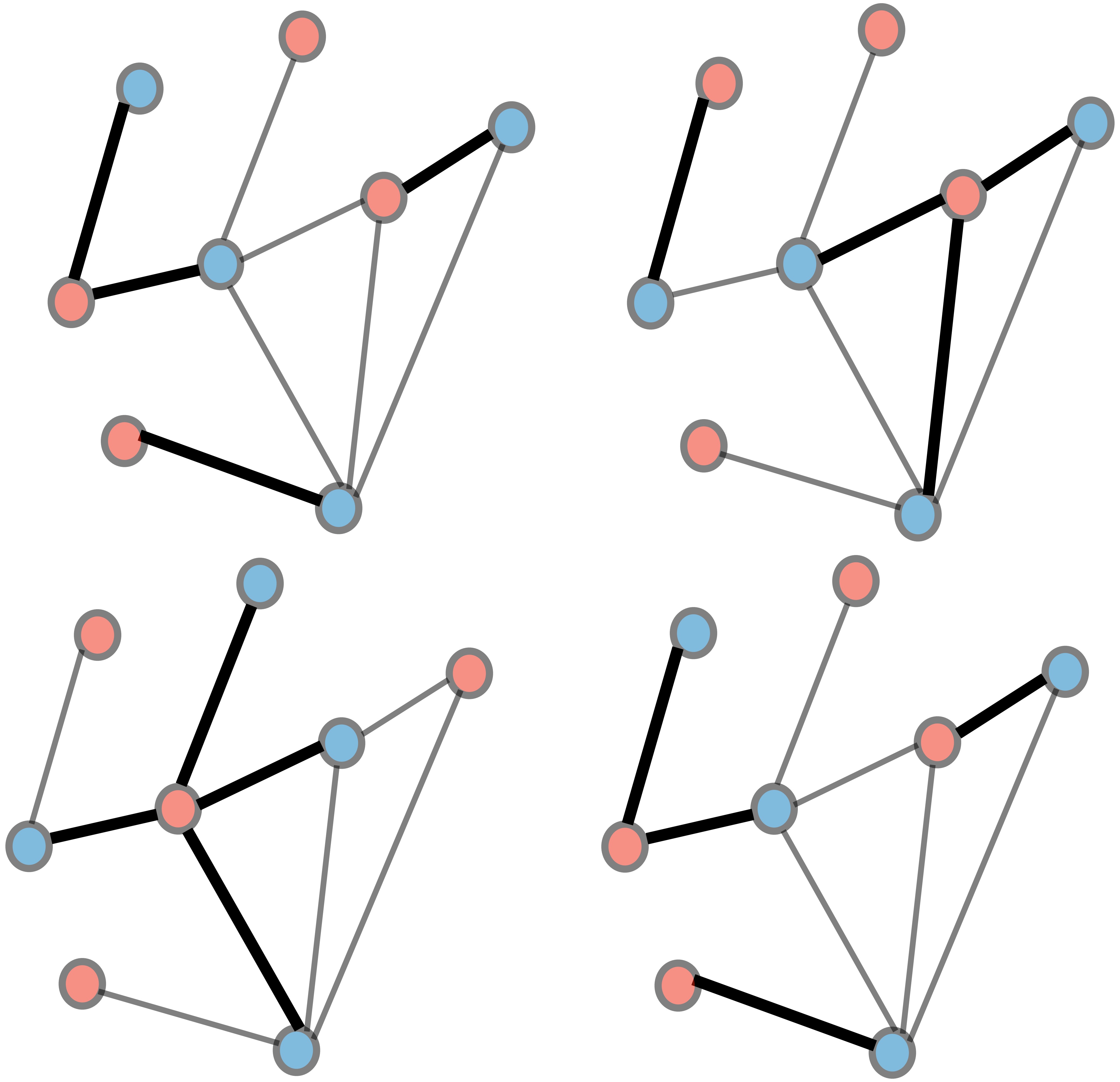
Yet another random ordering ...



Focus on edges which “send” vertices to VC

Proof of [Veldt '24]

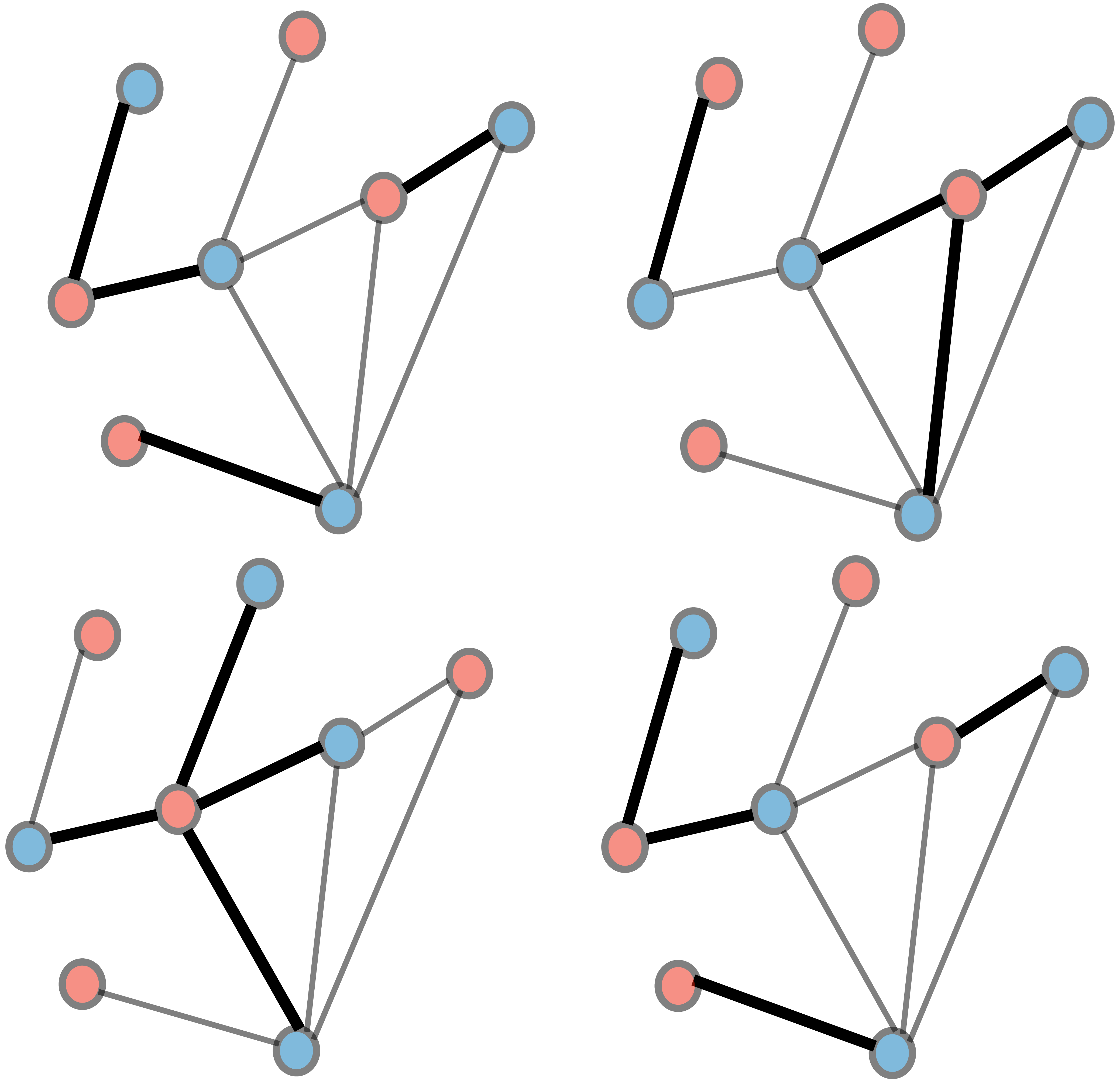
p_e: Probability of edge sending a vertex to VC (over the random ordering)



Proof of [Veldt '24]

p_e: Probability of edge sending a vertex to VC (over the random ordering)

Expected size of VC = $\sum_e p_e$



RGMIS to Fractional Matching

Expected size of VC = $\sum_e p_e$

$x_e = p_e/2$ is a fractional matching

[Veldt '24]

$$\max \sum_e x_e$$

For all vertices u, $\sum_{e \ni u} x_e \leq 1$

With $0 \leq x_e \leq 1$ for all edges

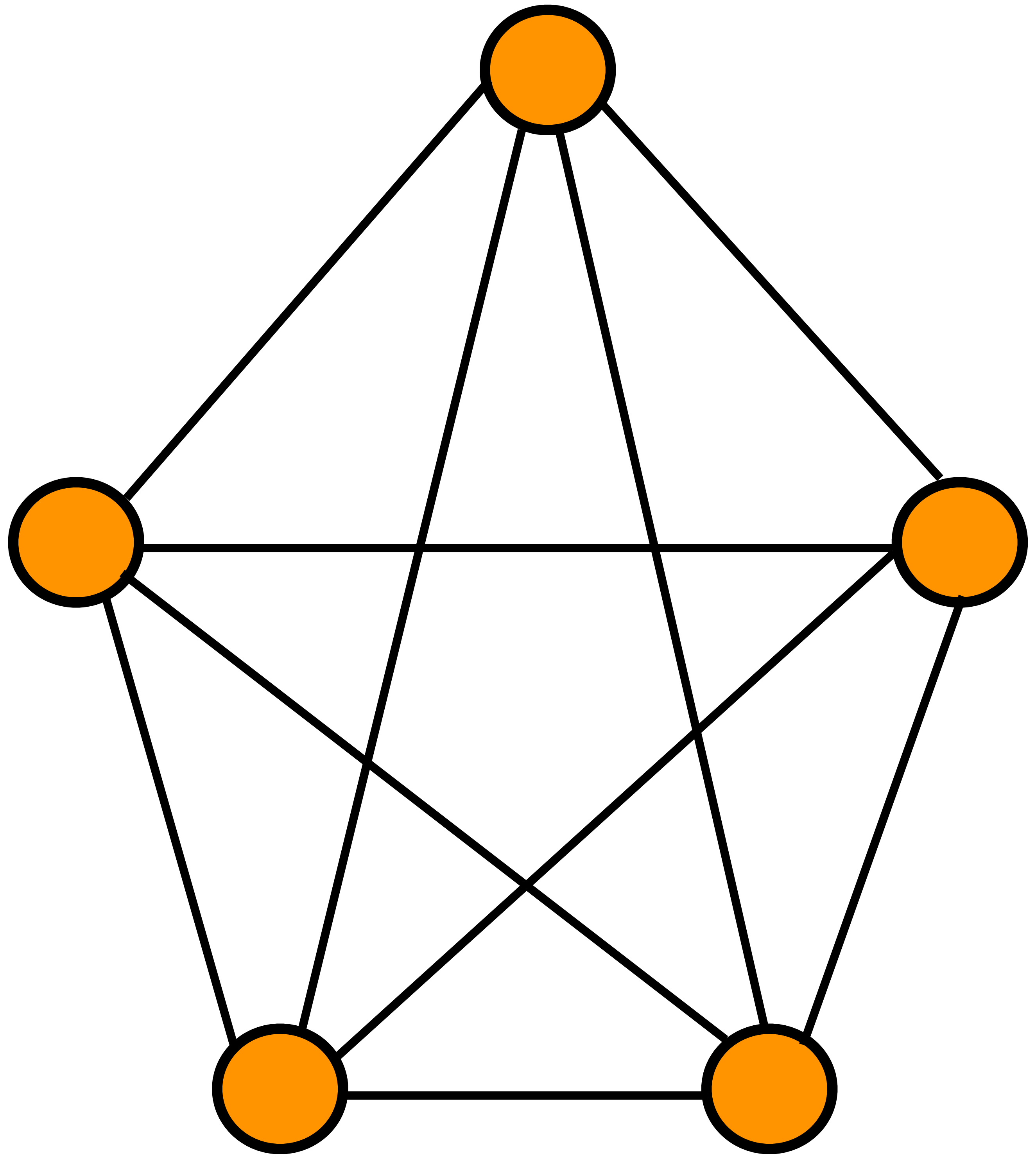
So are we done?

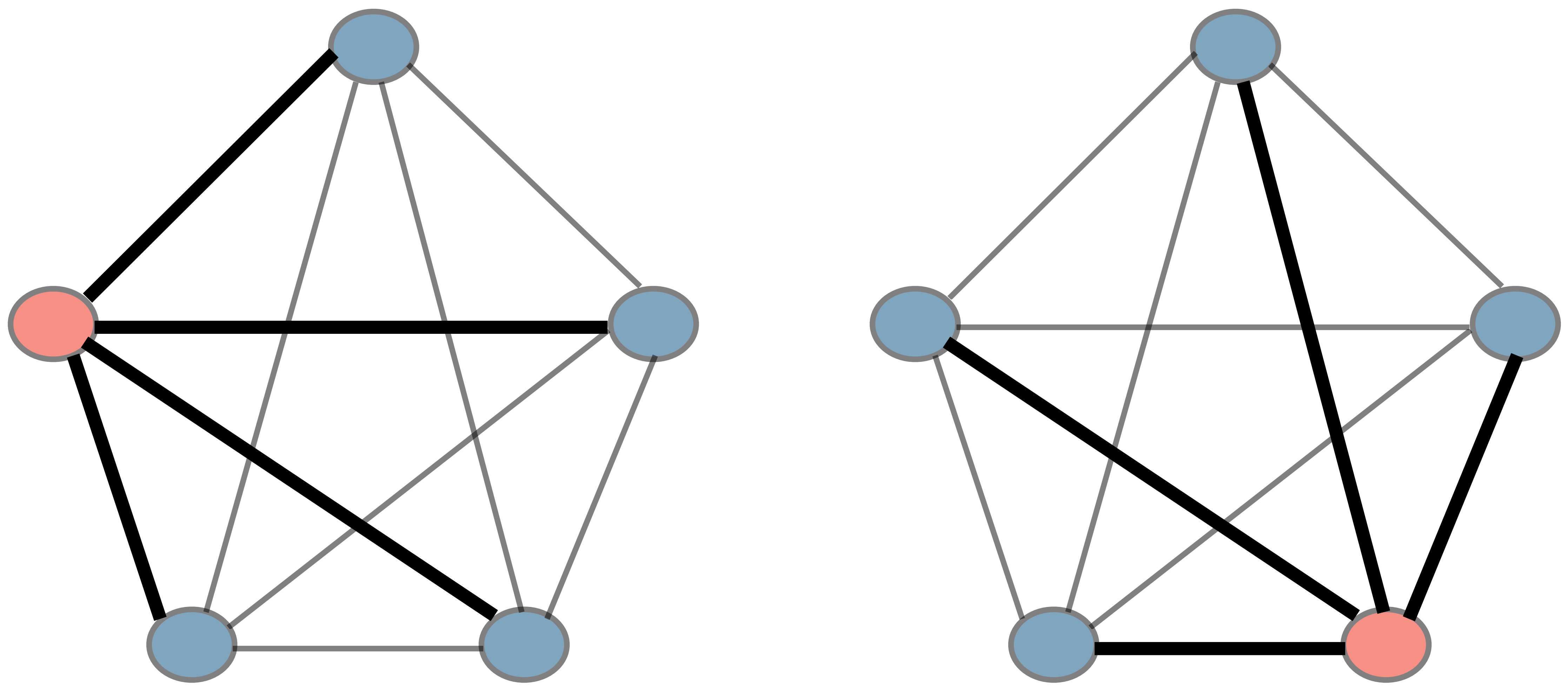
- Find MIS using [ACGMW '15] in $O(\log \log n)$ passes
- Find the edges which are sending vertices to VC - each edge is sampled here with probability $2 \cdot x_e$
- Try to find a large matching inside them

Sampling according to fractional matching is enough to find $O(1)$-approx integral matching

Sampling Process is not independent!

The Issue of the Star





Edges we get from each run of MIS are a bunch of stars

Plan for the rest of the talk

- Fractional matching and Vertex Cover
- Connections to MIS
- **Our reduction to MIS**
- Challenges of implementation

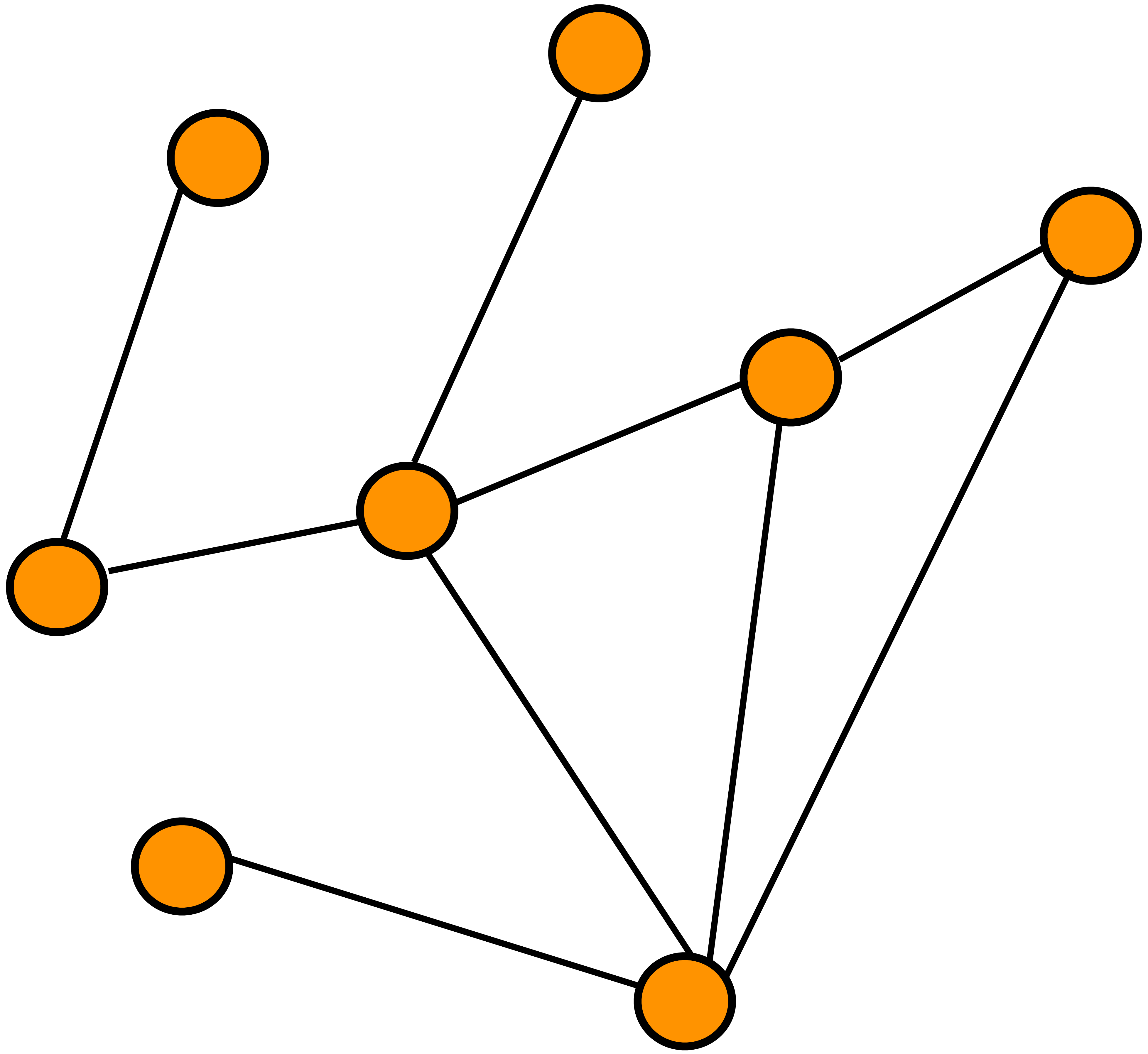
Plan for the rest of the talk

- Fractional matching and Vertex Cover
- Connections to MIS
- **Our reduction to MIS** **(from one run of RGMIS algorithm)**
- Challenges of implementation

Our Reduction

Whenever a vertex goes to VC,
focus on edges to the remaining graph.

Our Reduction



When vertex u goes to VC,

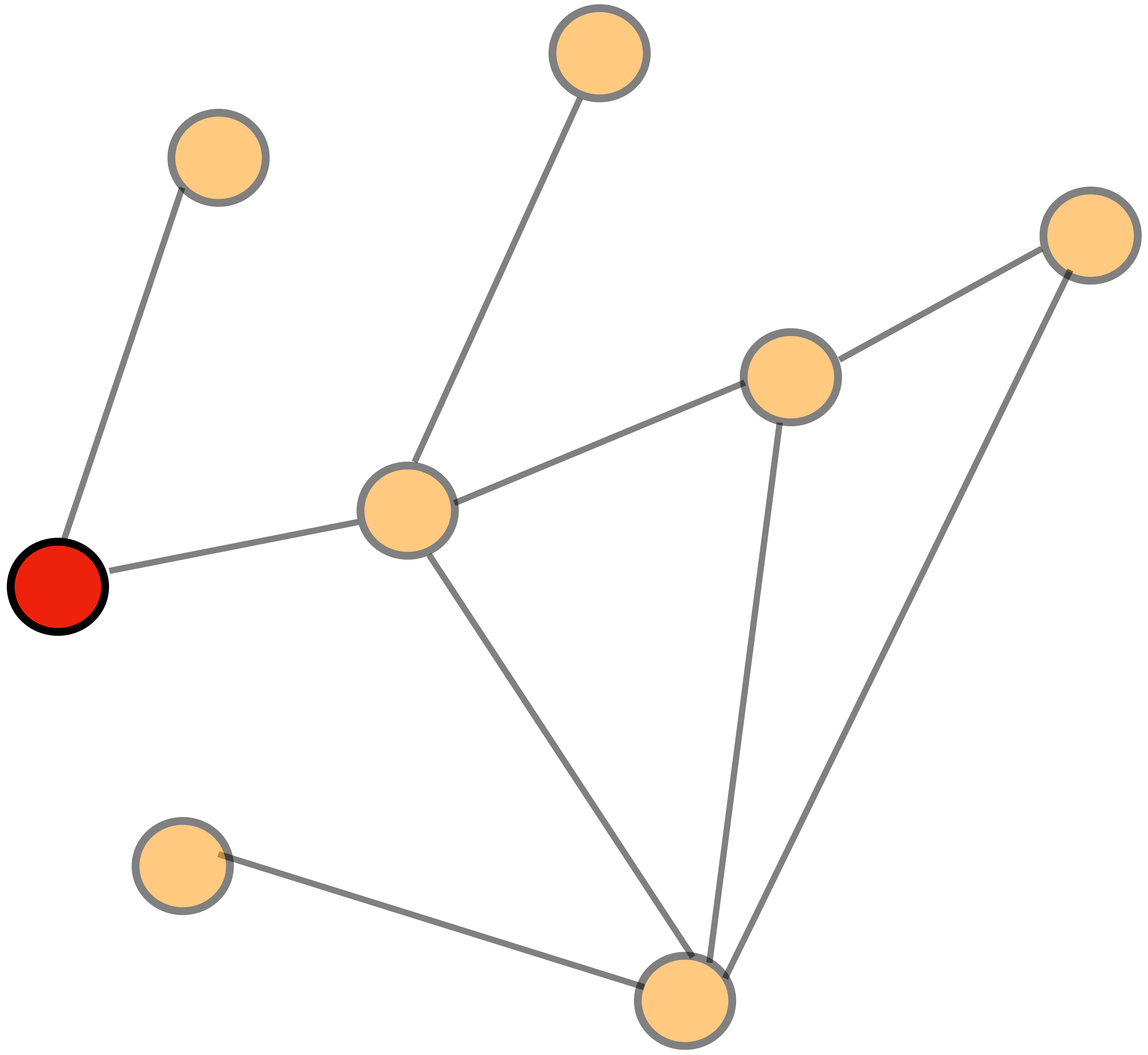
add a mass of $1/\deg(u)$

to the edges between u and the

remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

Our Reduction



When vertex u goes to VC,

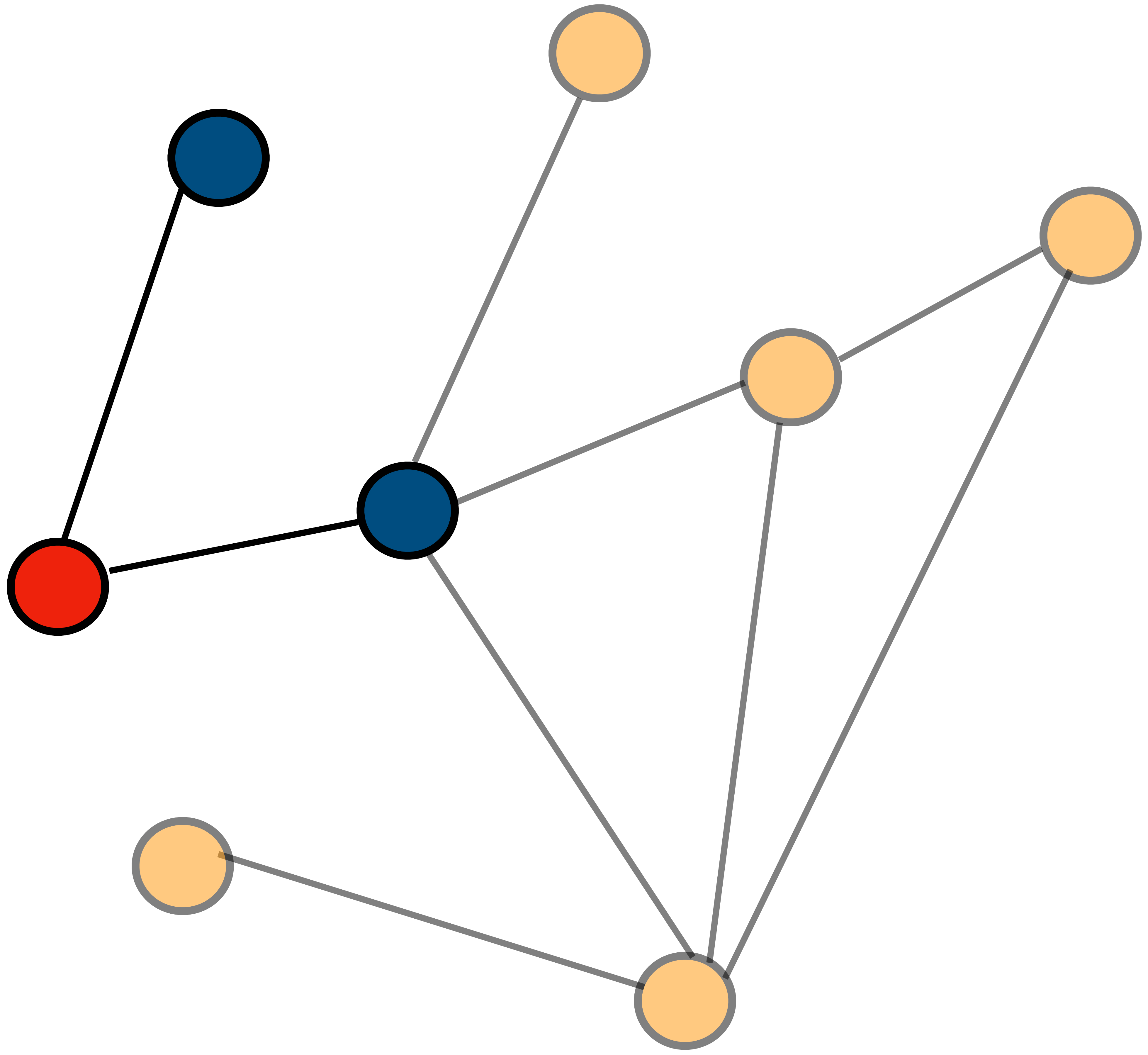
add a mass of $1/\deg(u)$

to the edges between u and the

remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

Our Reduction



When vertex u goes to VC,

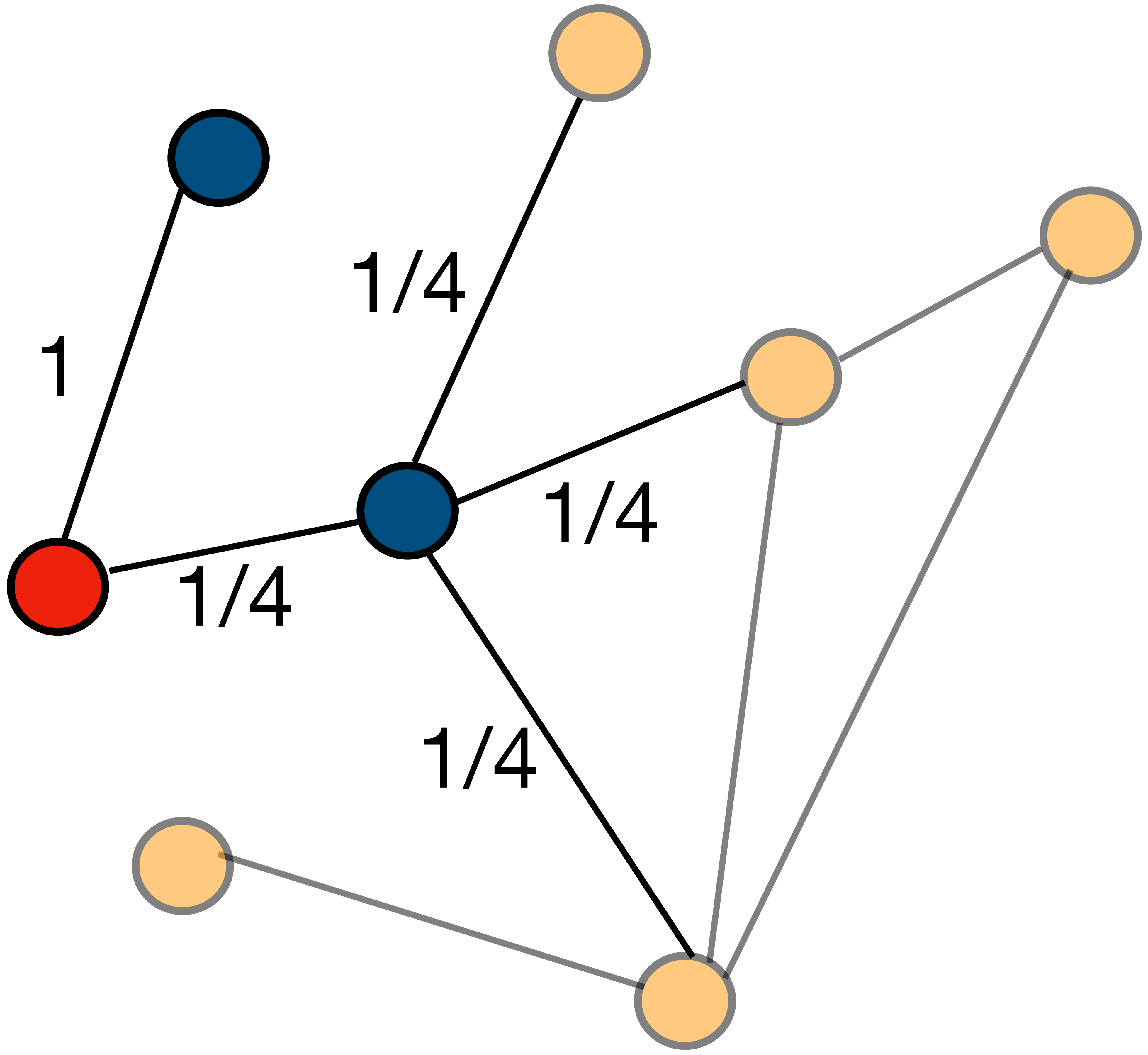
add a mass of $1/\deg(u)$

to the edges between u and the

remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

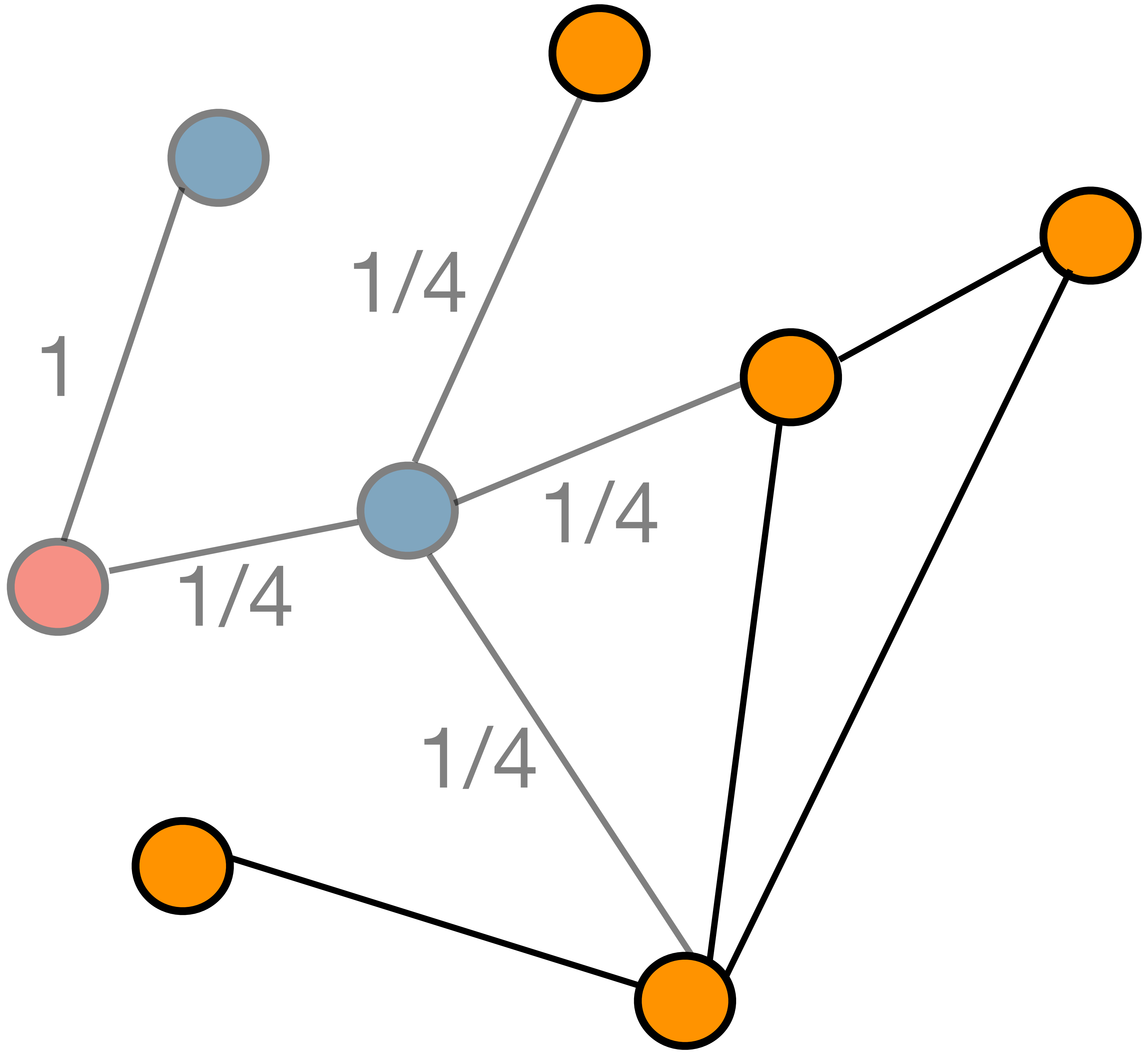
Our Reduction



When vertex u goes to VC,
add a mass of $1/\deg(u)$
to the edges between u and the
remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

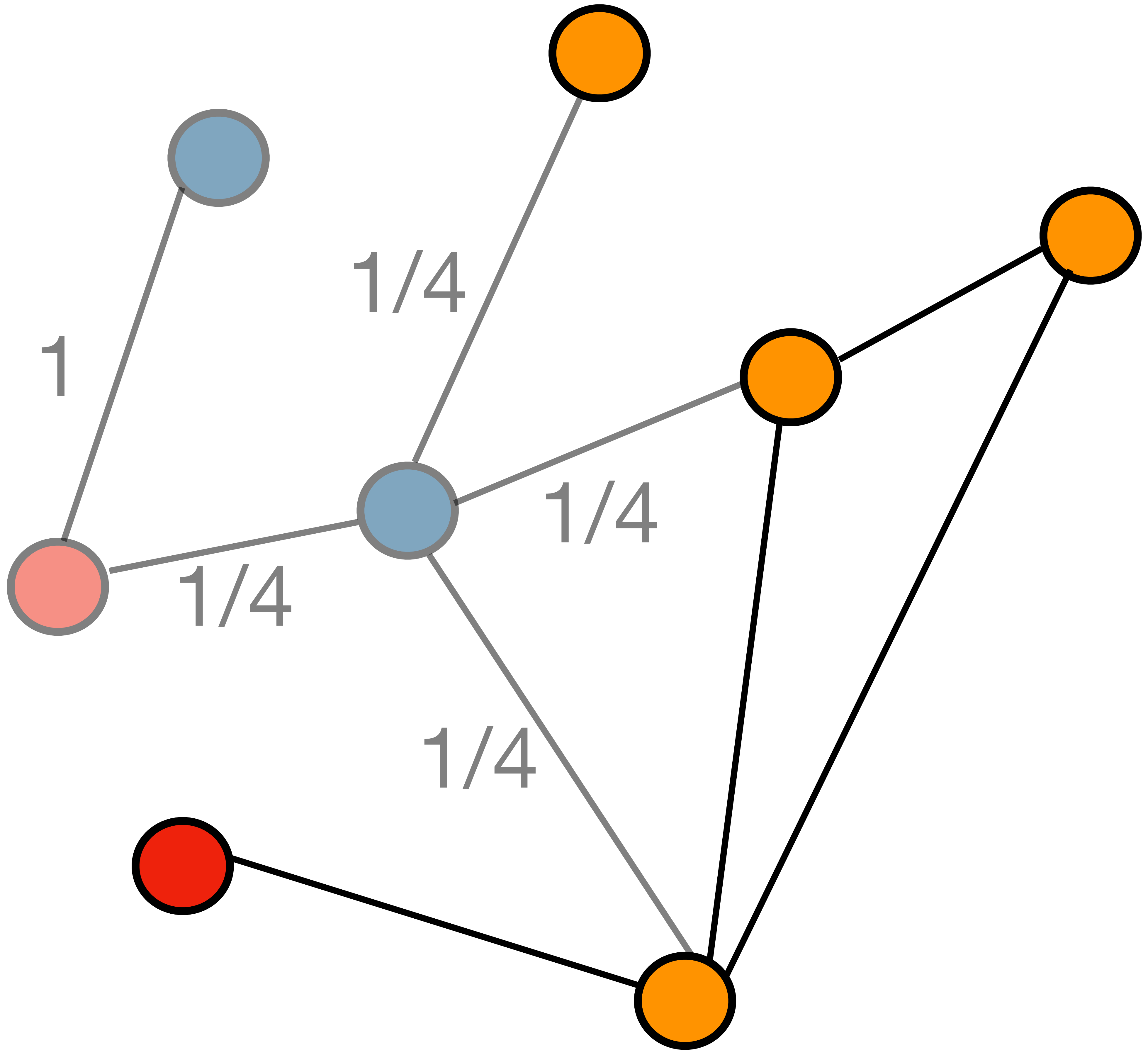
Our Reduction



When vertex u goes to VC,
add a mass of $1/\deg(u)$
to the edges between u and the
remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

Our Reduction



When vertex u goes to VC,

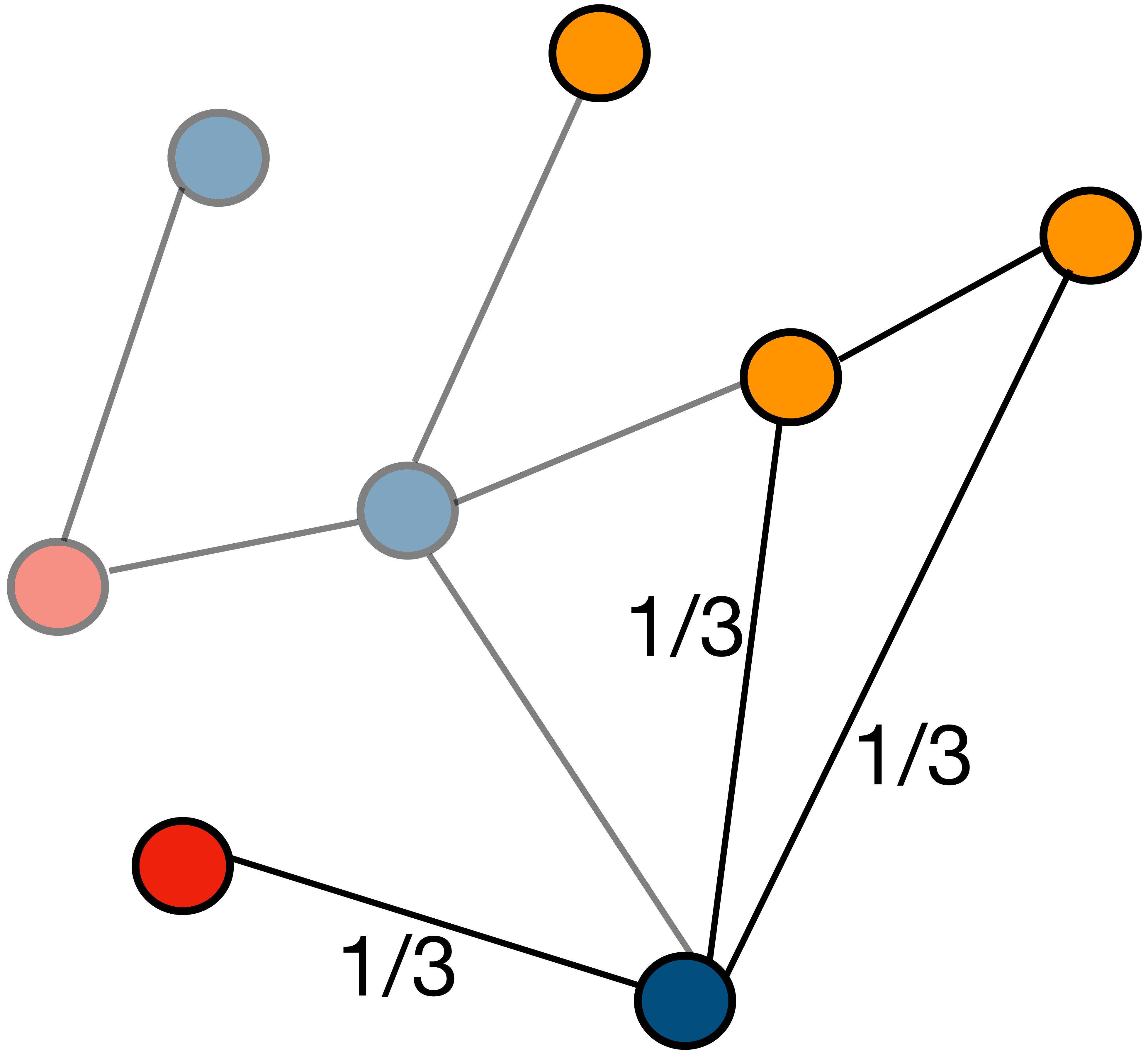
add a mass of $1/\deg(u)$

to the edges between u and the

remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

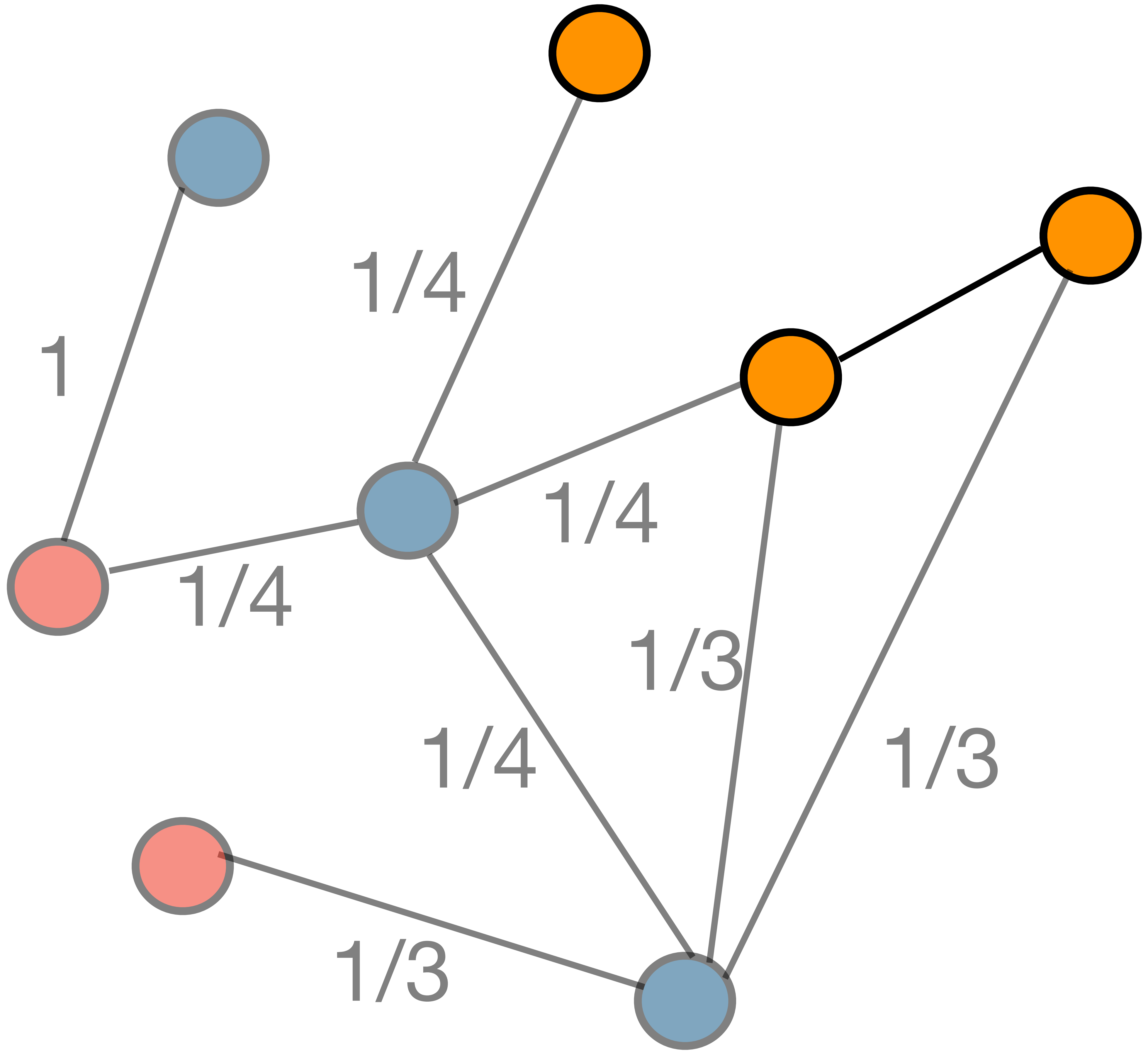
Our Reduction



When vertex u goes to VC,
add a mass of $1/\deg(u)$
to the edges between u and the
remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

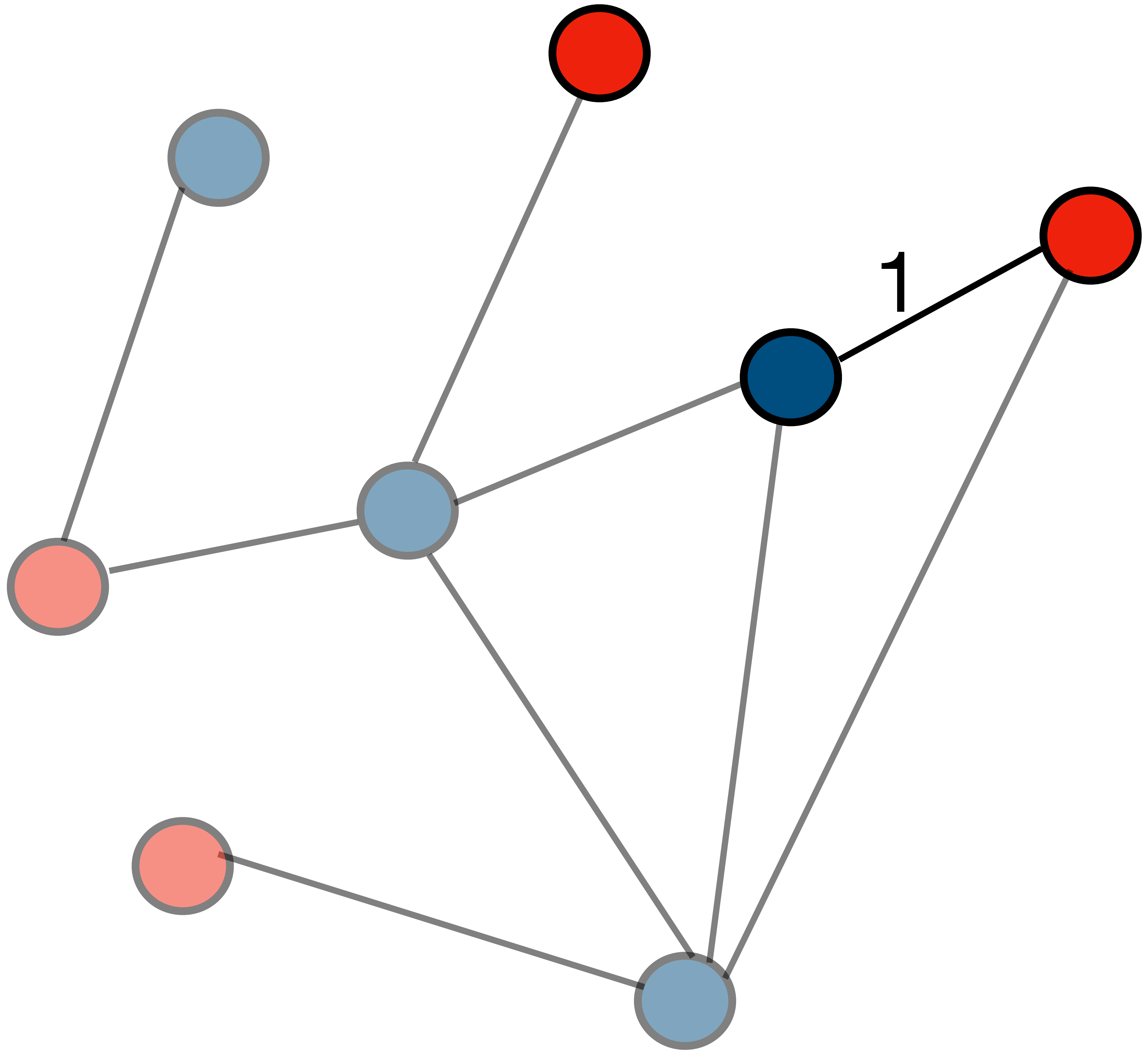
Our Reduction



When vertex u goes to VC,
add a mass of $1/\deg(u)$
to the edges between u and the
remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

Our Reduction



When vertex u goes to VC,

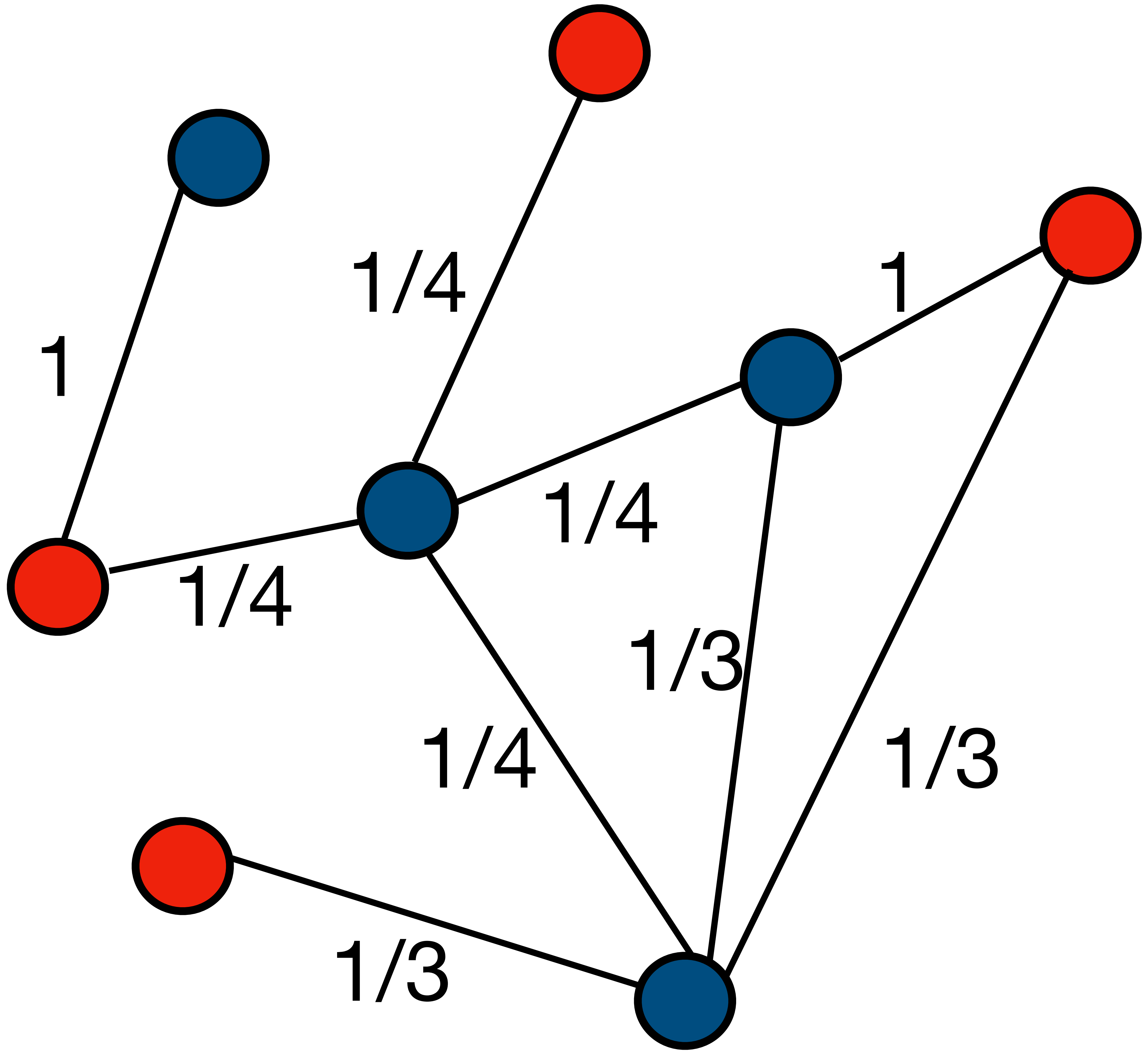
add a mass of $1/\deg(u)$

to the edges between u and the

remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

Our Reduction



When vertex u goes to VC,

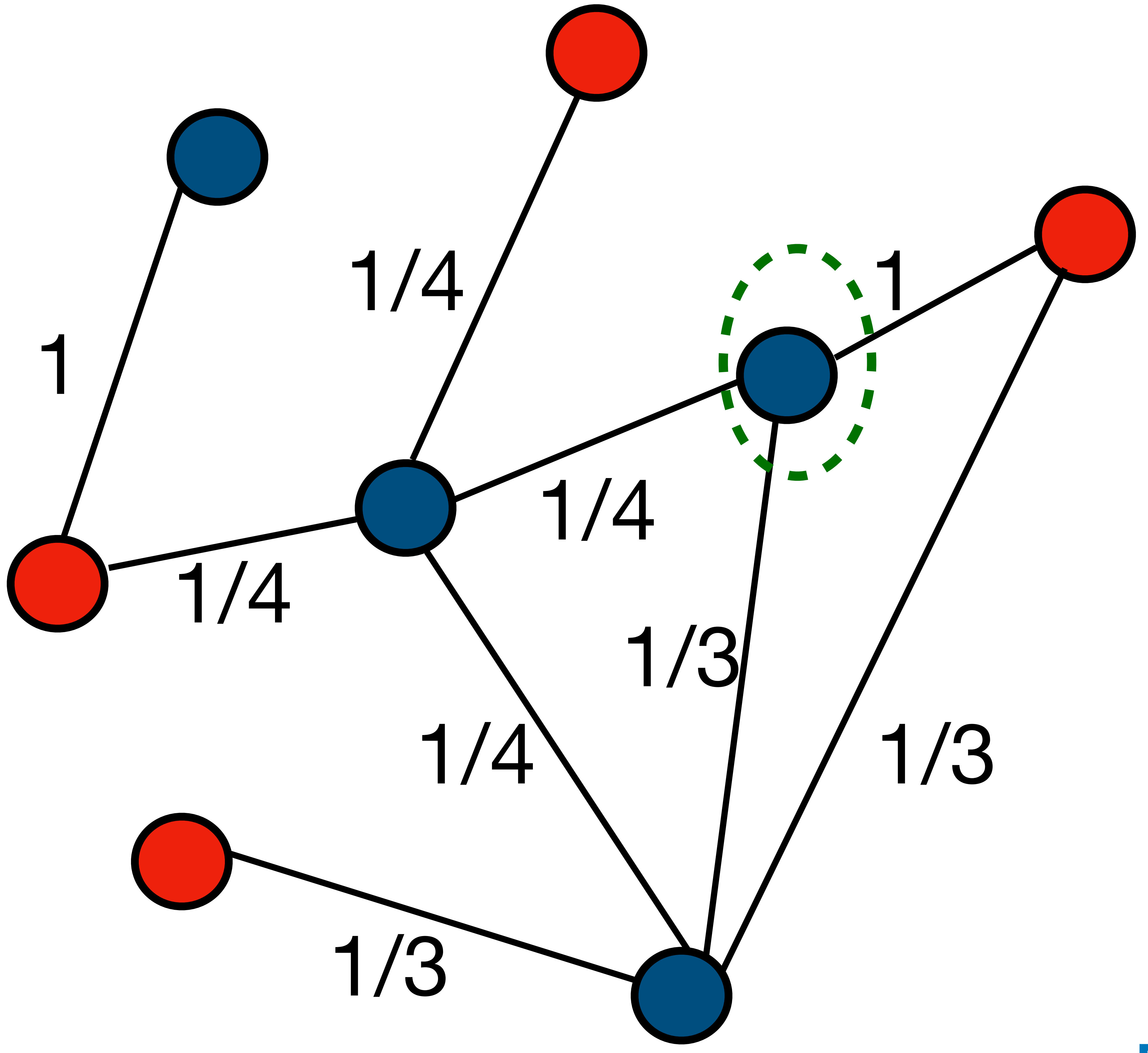
add a mass of $1/\deg(u)$

to the edges between u and the

remaining graph*.

*- we only add the mass to lower degree neighbors
for technical reasons we will see later

Back to fractional matching LP



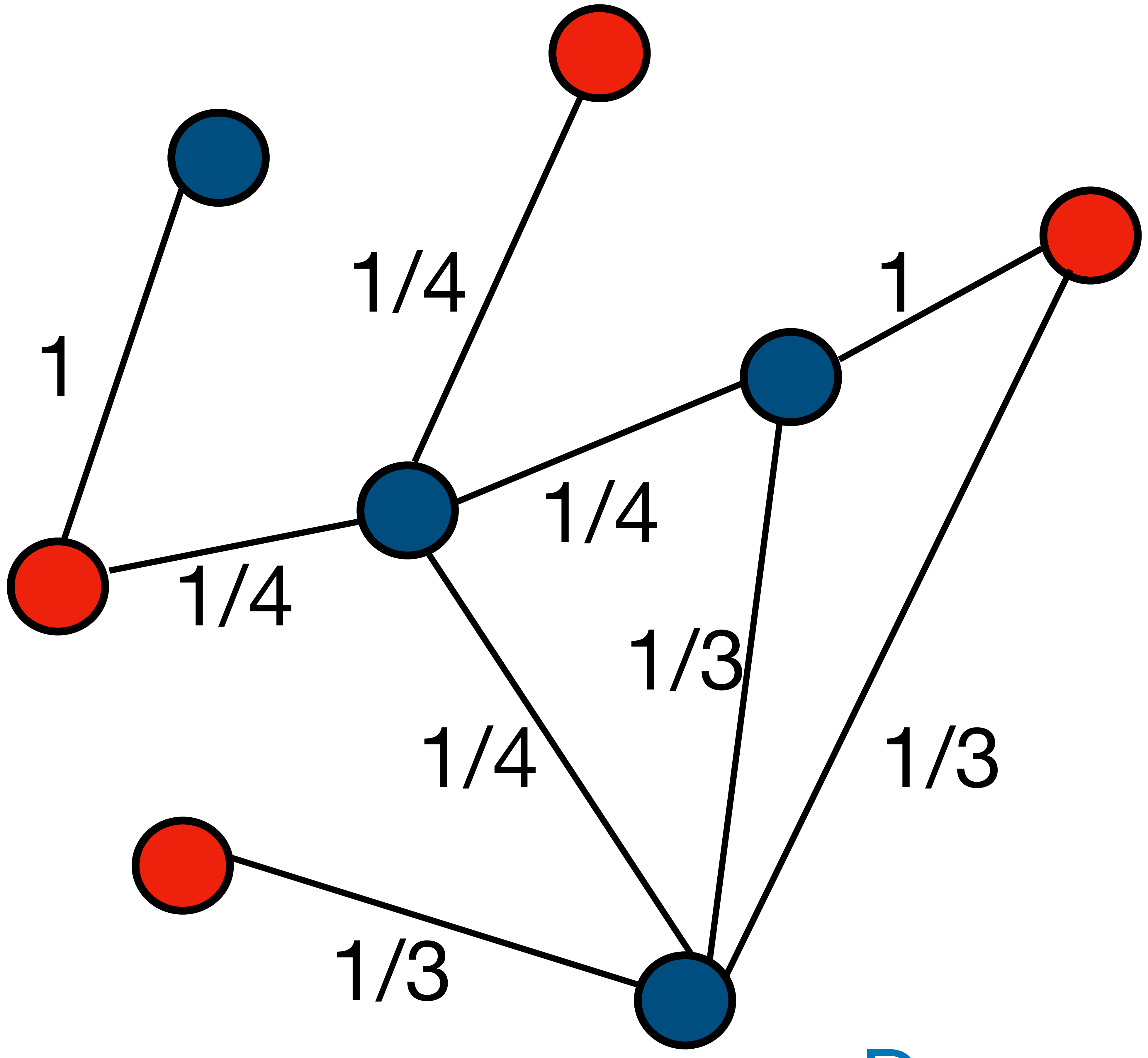
$$\max \sum_e x_e$$

For all vertices u, $\sum_{e \ni u} x_e \leq 1$

With $0 \leq x_e \leq 1$ for all edges

Do we satisfy LP with large value?

Back to fractional matching LP



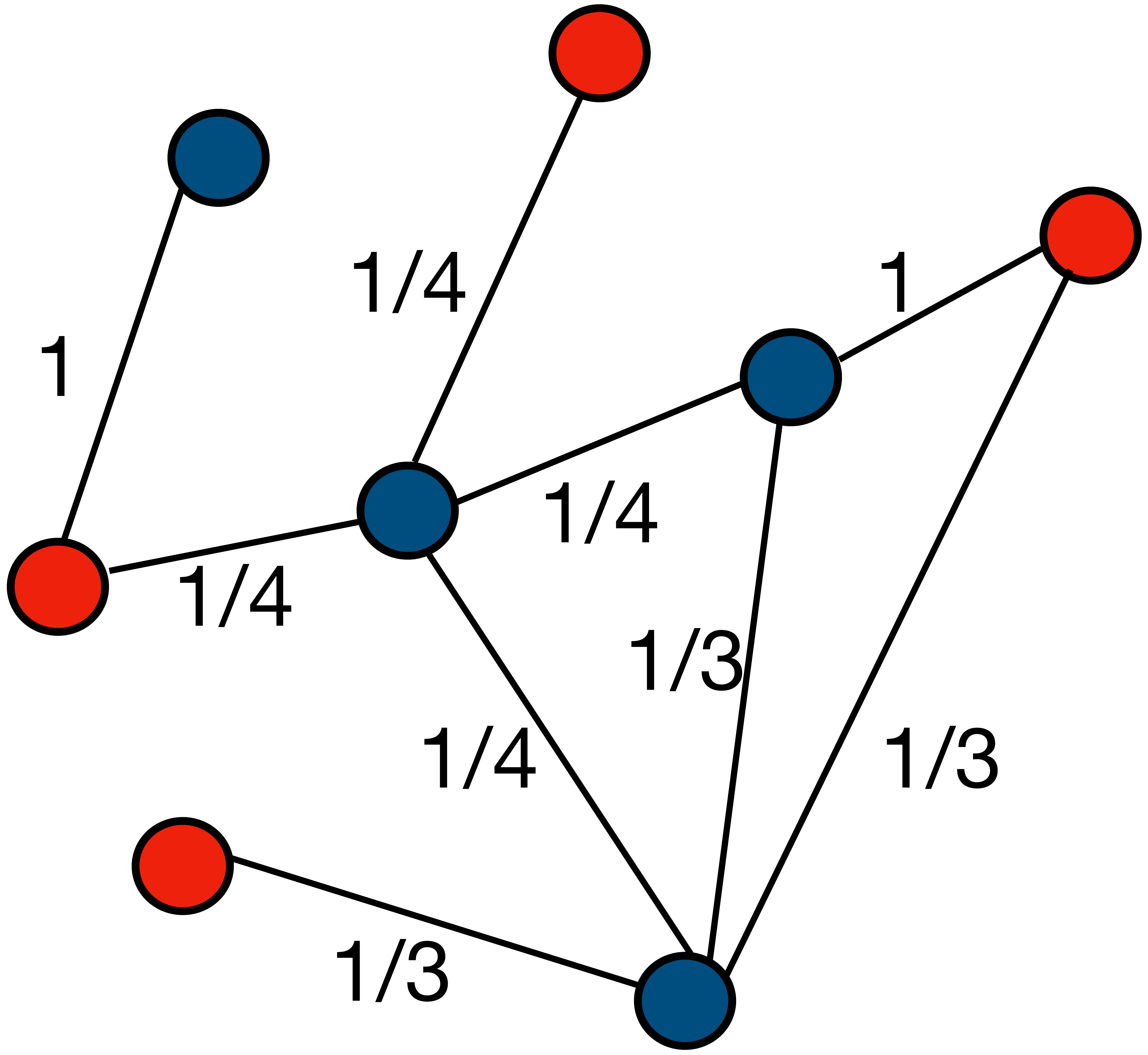
$$\max \sum_e x_e$$

For all vertices u, $\sum_{e \ni u} x_e \leq O(1)$

With $0 \leq x_e \leq 1$ for all edges

Do we approximately satisfy LP with large value?

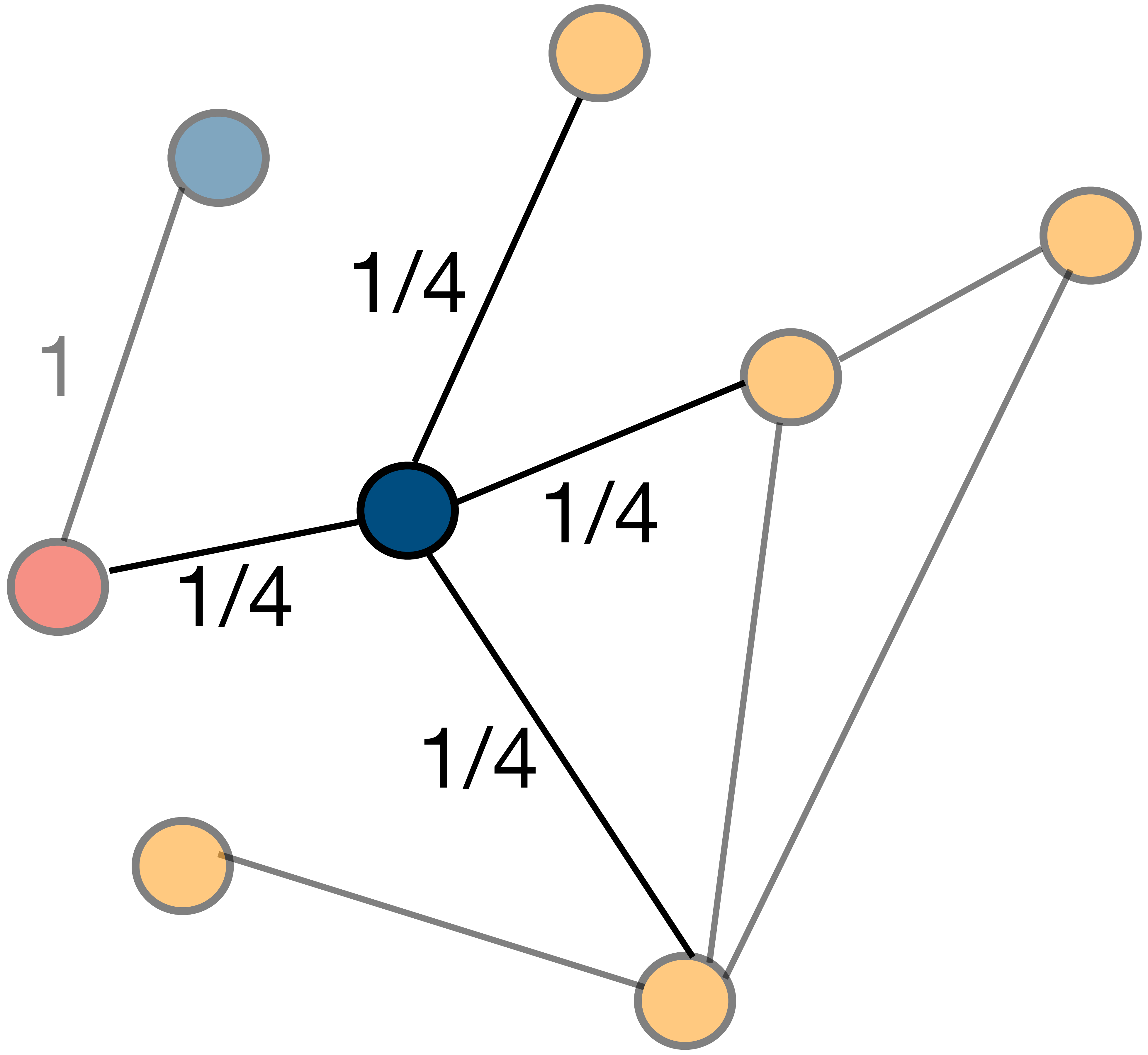
Value of Fractional Matching



Fractional Matching value: $\sum_e x_e$

Every time a vertex is added to VC, matching value increases by 1 (we add $1/\deg(u)$ value to $\deg(u)$ vertices)

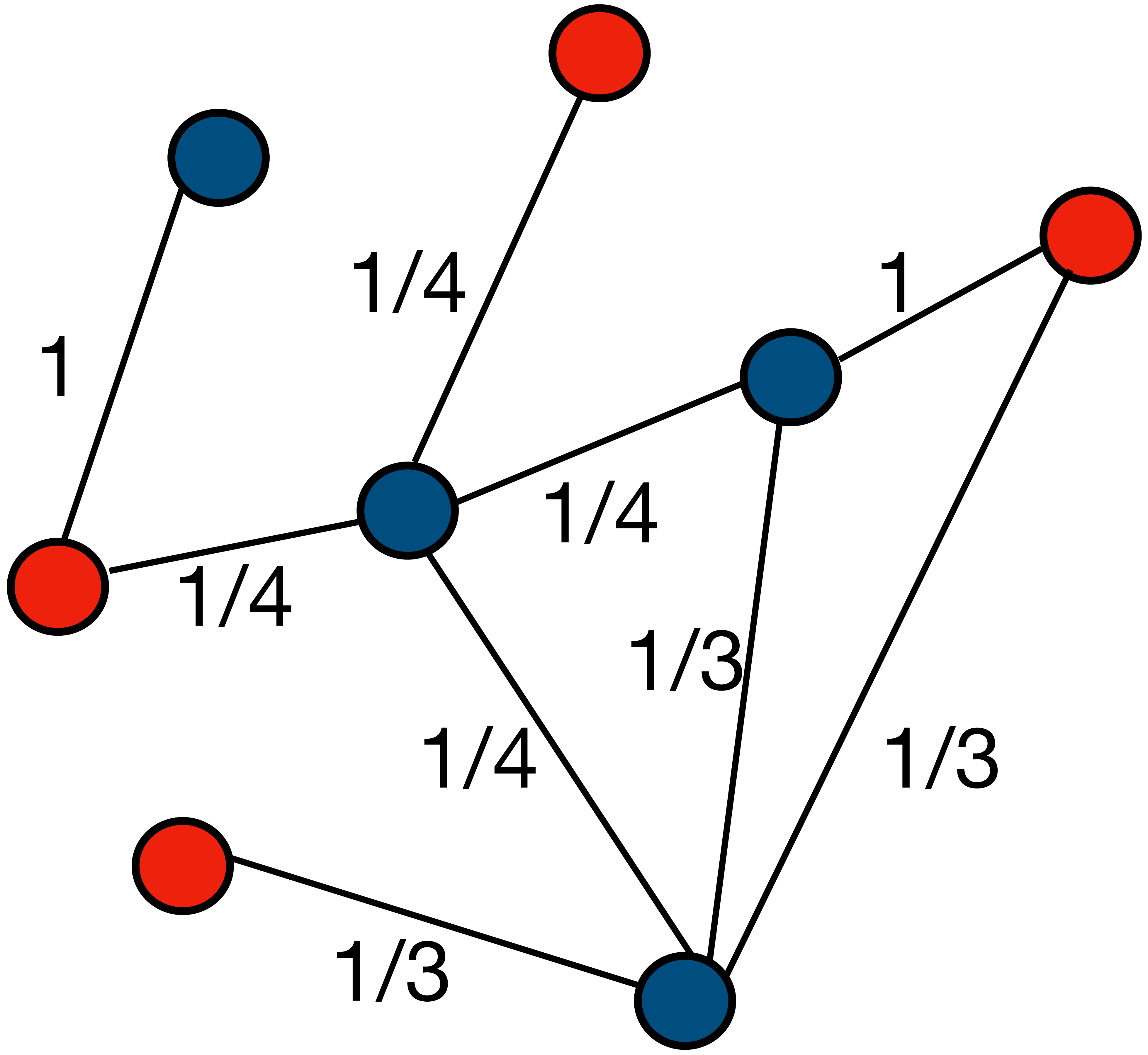
Value of Fractional Matching



Fractional Matching value: $\sum_e x_e$

Every time a vertex is added to VC, matching value increases by 1 (we add $1/\deg(u)$ value to $\deg(u)$ vertices)

Value of Fractional Matching

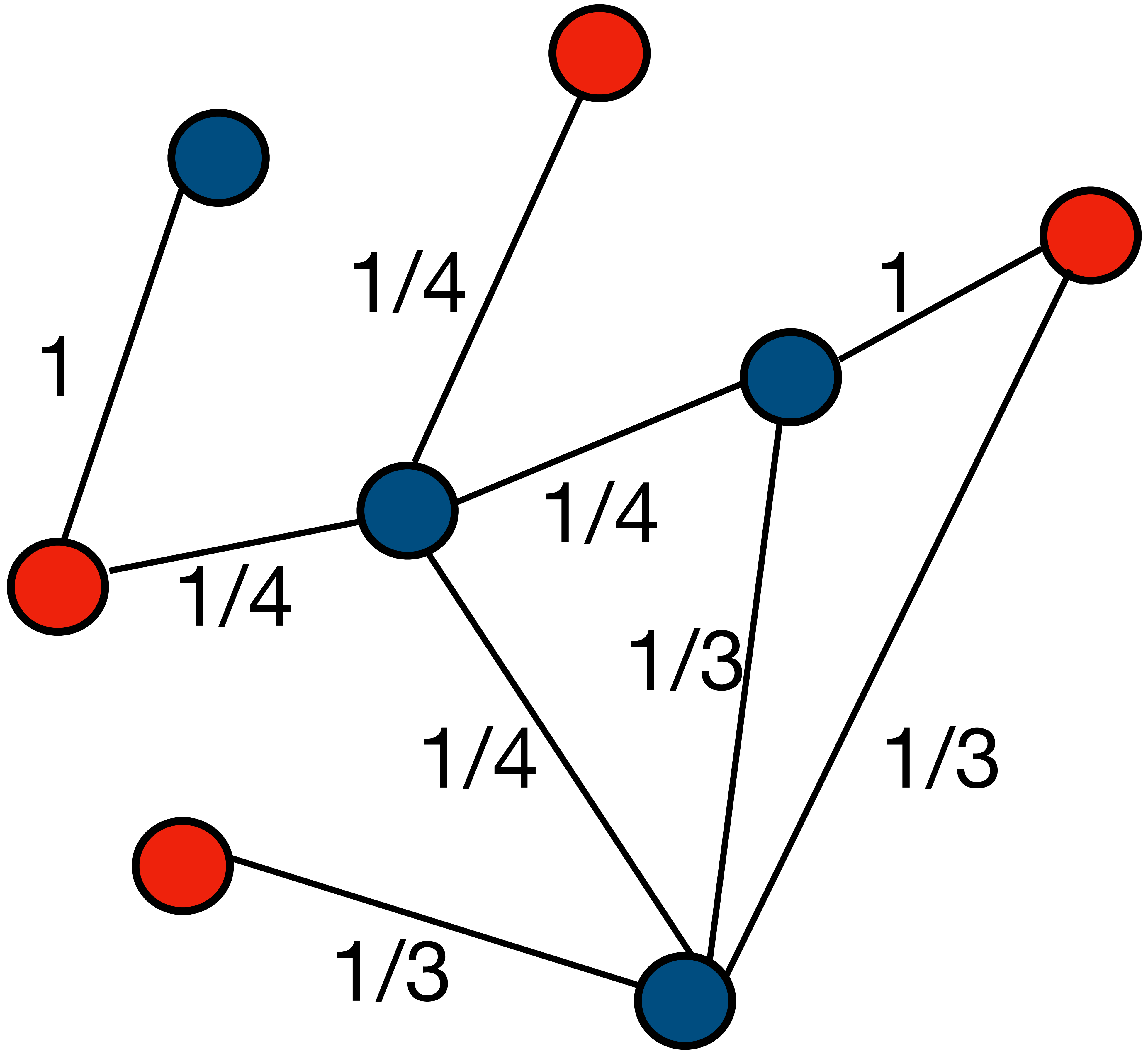


Fractional Matching value: $\sum_e x_e$

Same as the size of the Vertex Cover*
Larger than size of maximum matching

*-for now, this is not true in the actual reduction

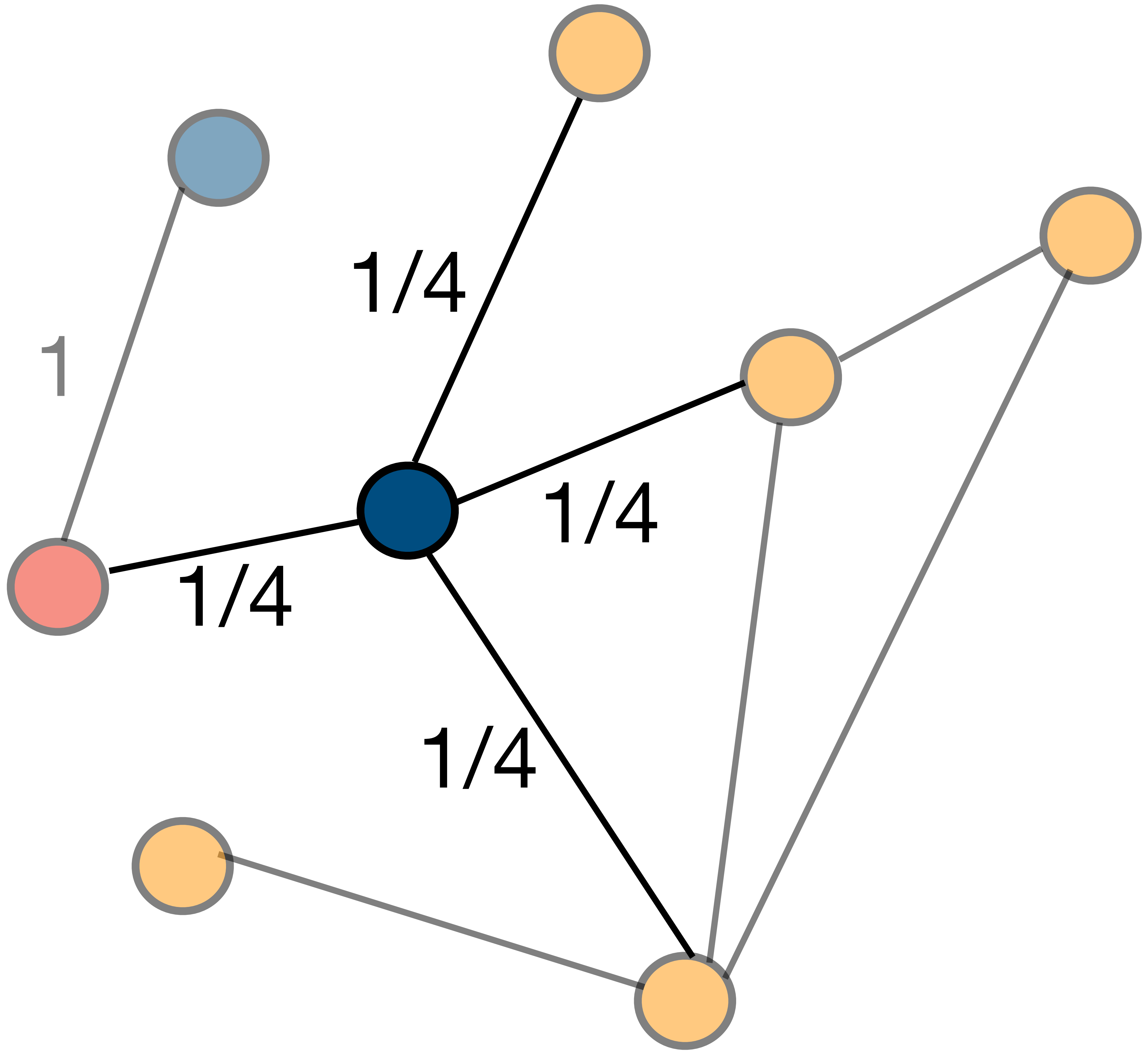
Vertex Constraints - Why is this a matching?



For all vertices u, $\sum_{e \ni u} x_e \leq O(1)$

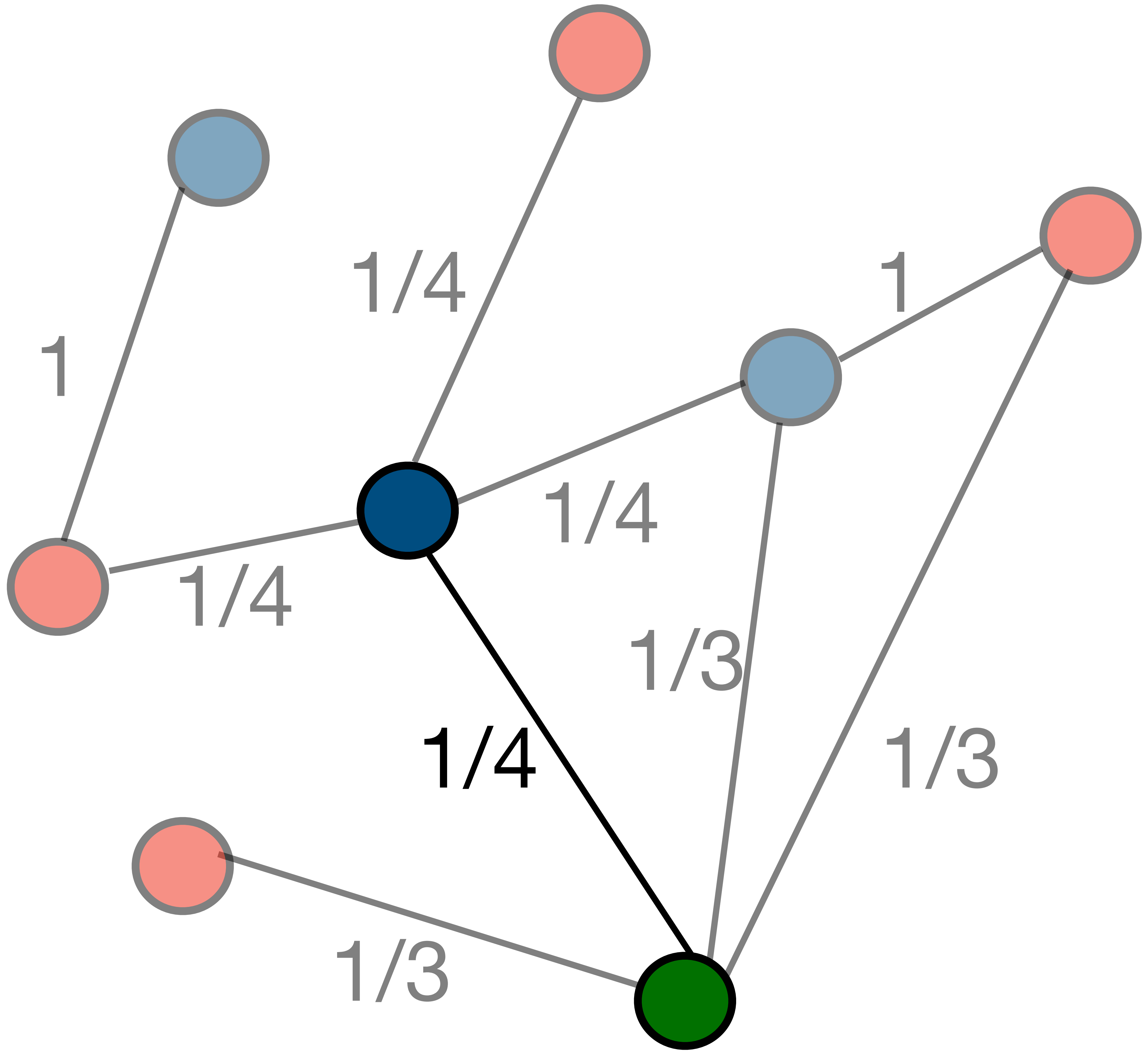
With $0 \leq x_e \leq 1$ for all edges

Source of Mass for a Vertex



- From itself - if it is added to VC (this value is exactly 1)

Source of Mass for a Vertex



- From itself - if it is added to VC (this value is exactly 1)
- From its neighbors?

Main Observation

In some iteration, if we expect a vertex to receive a lot of mass from neighbors,

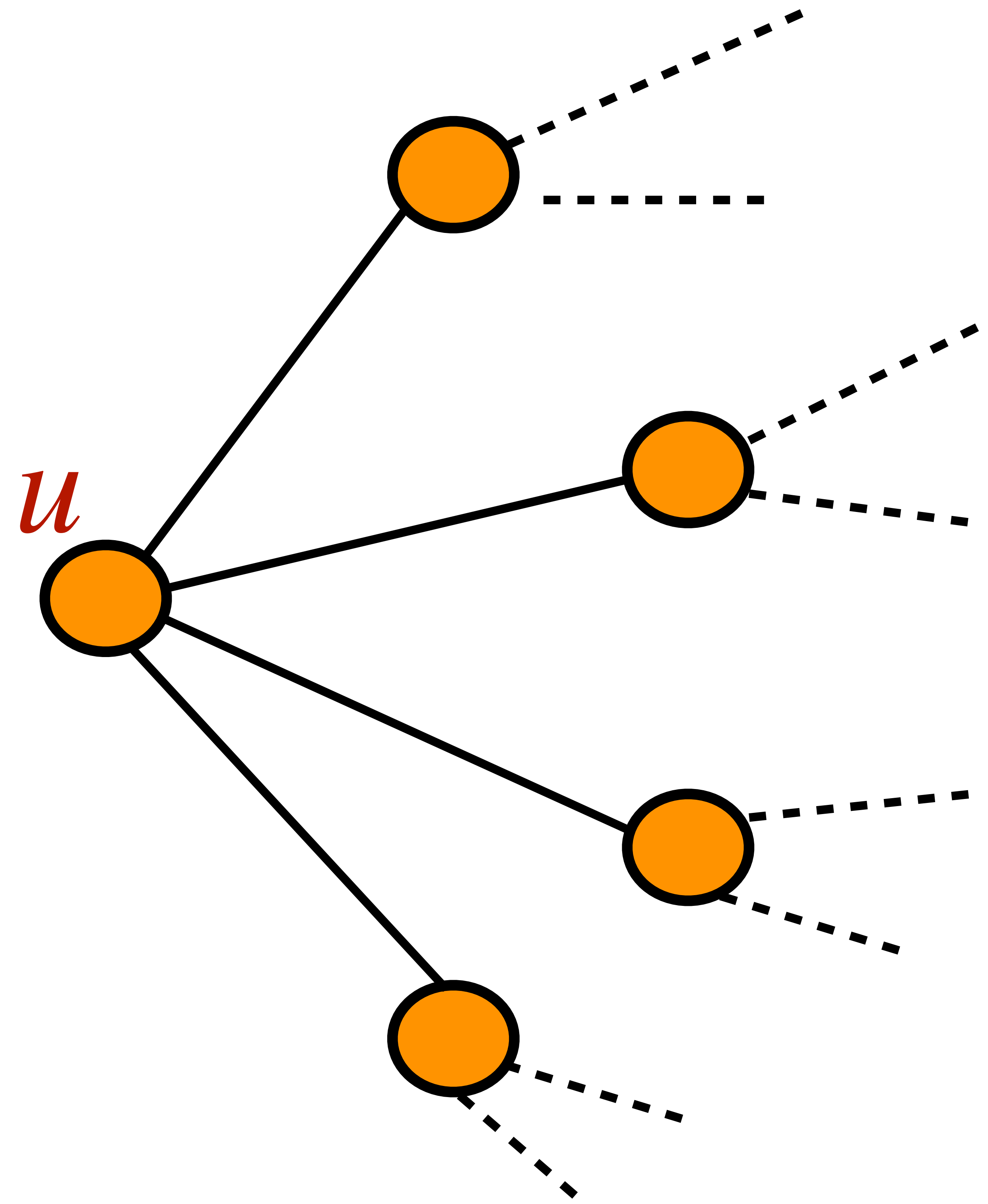
we can expect it to be removed altogether

Probability that vertex is removed

Chosen to be in MIS

$1/n$ probability

Neighbor chosen to be in MIS



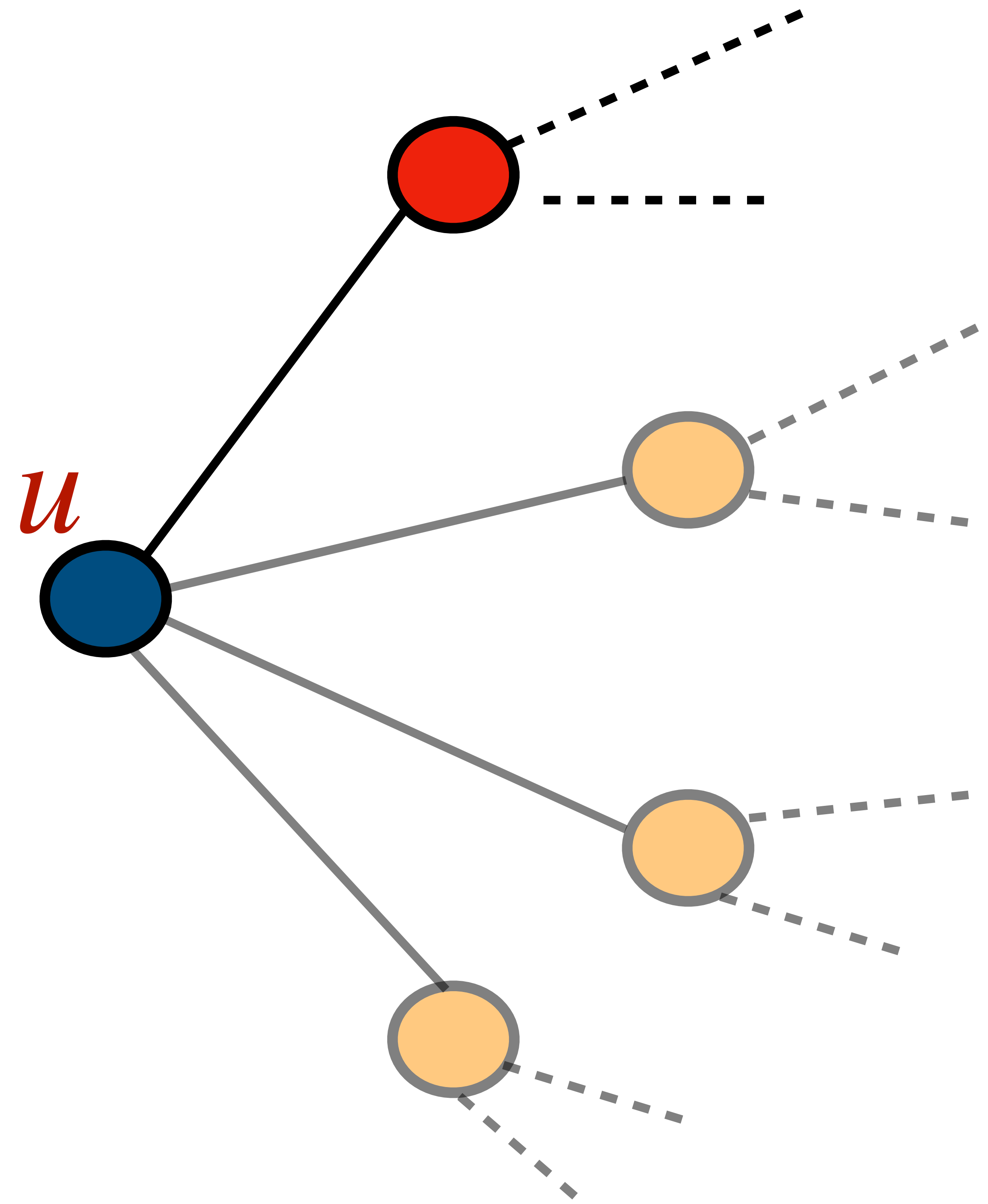
Probability that vertex is removed

Chosen to be in MIS

Neighbor chosen to be in MIS

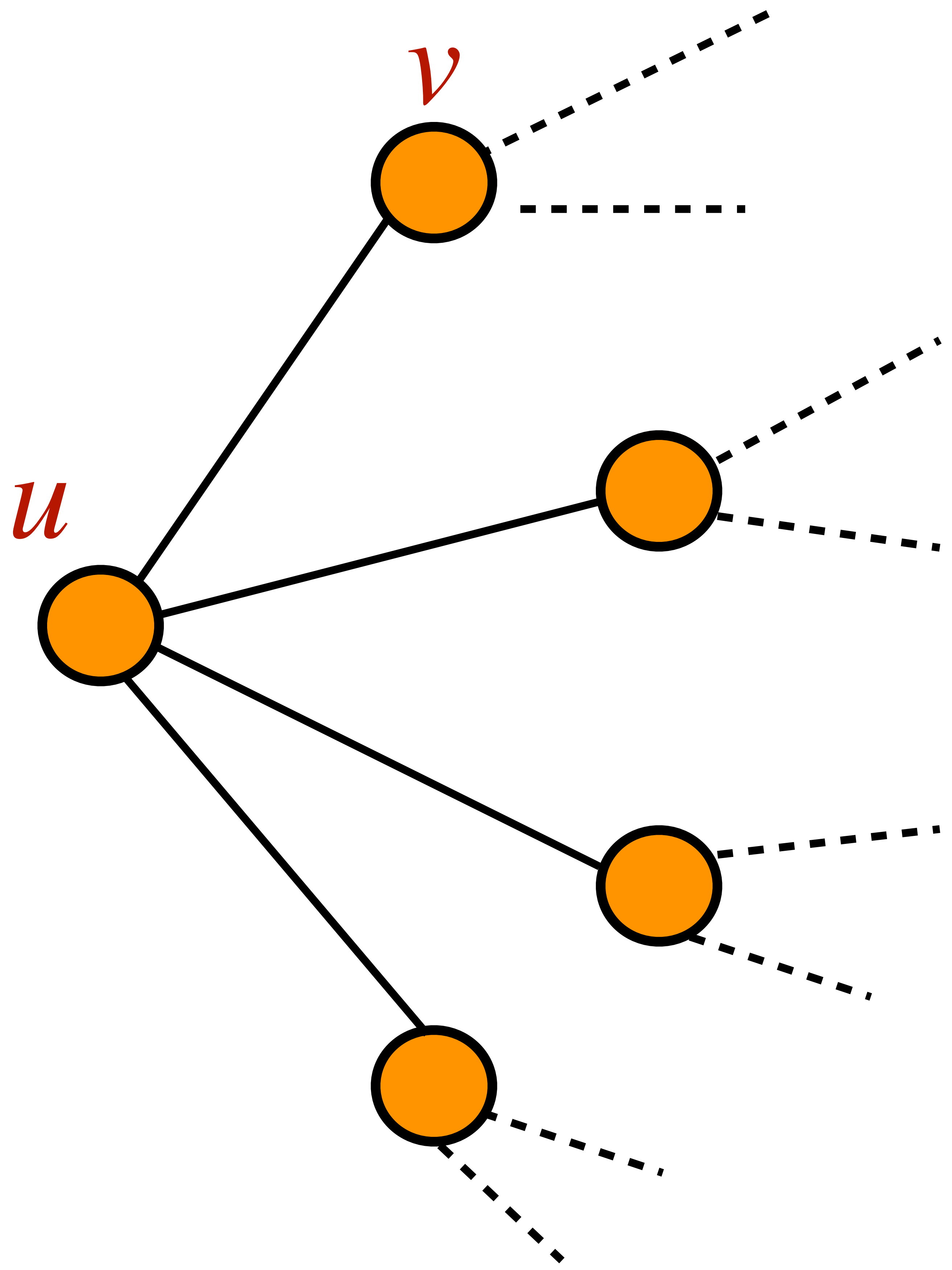
$1/n$ probability

$\deg(u)/n$ probability

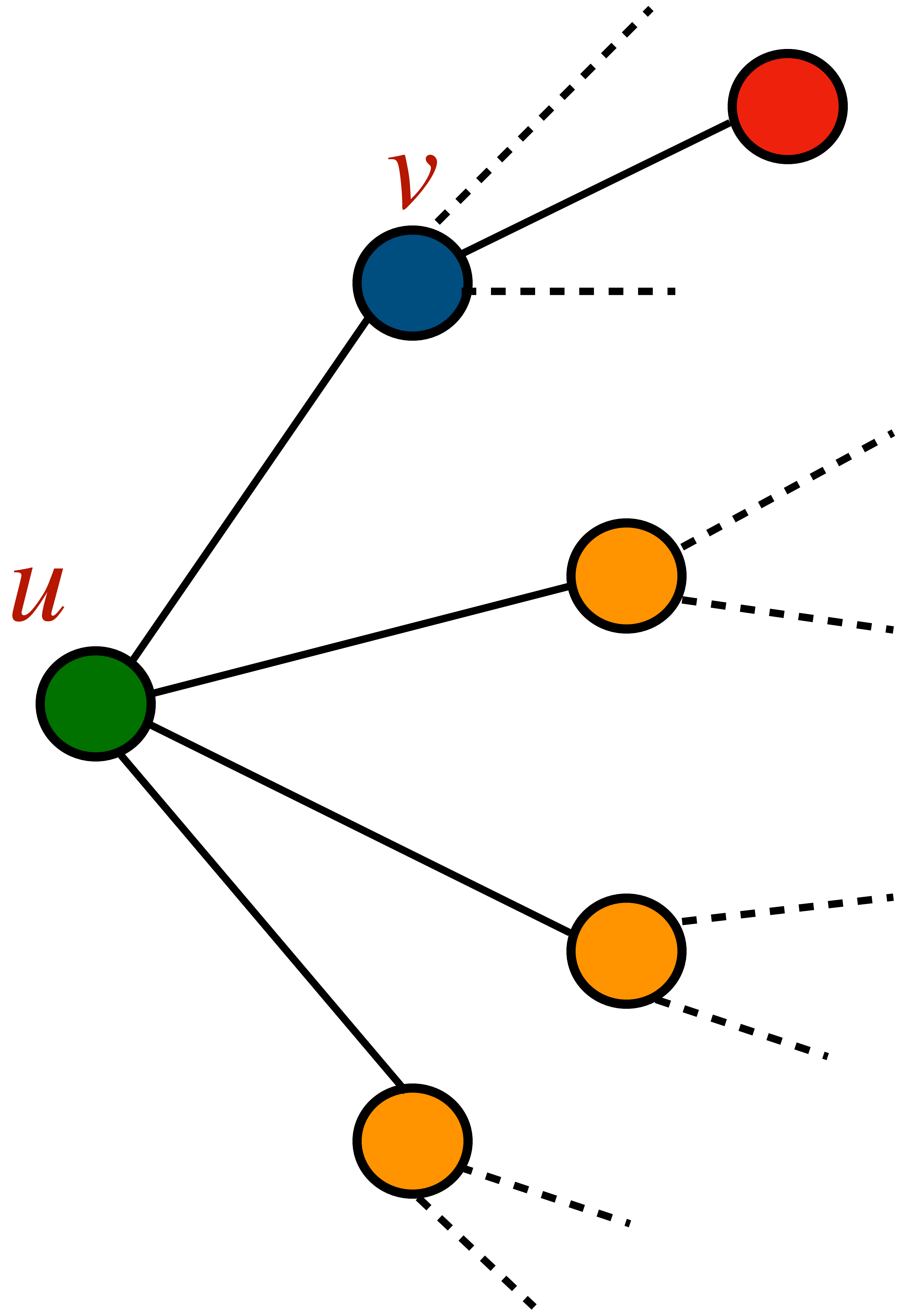


Probability is $(\deg(u) + 1)/n$ totally

Mass from Neighbors



Mass from Neighbors



- Probability of neighbor v sent to VC is $\deg(v)/n$
- Mass added across edge (u, v) is $1/\deg(v)$

In expectation, $1/n$ mass from each edge

Total expected mass is $\deg(u)/n$

If u gains a lot, it gets out

For any vertex u, in any iteration,

Expected mass from neighbors $\leq$ Probability that u is removed

Gain a lot, then Get out Game

X_i - mass gained by vertex u at iteration i

Random Variables $X_1, X_2, \ldots, X_n$ - mass gained at each iteration

Conditioned on any choice of $X_1, X_2, \ldots, X_{i-1}$,

Expected value of $X_i \leq$ Probability that all the later X_j with $j > i$ are zero.

Gain a lot, then Get out Game

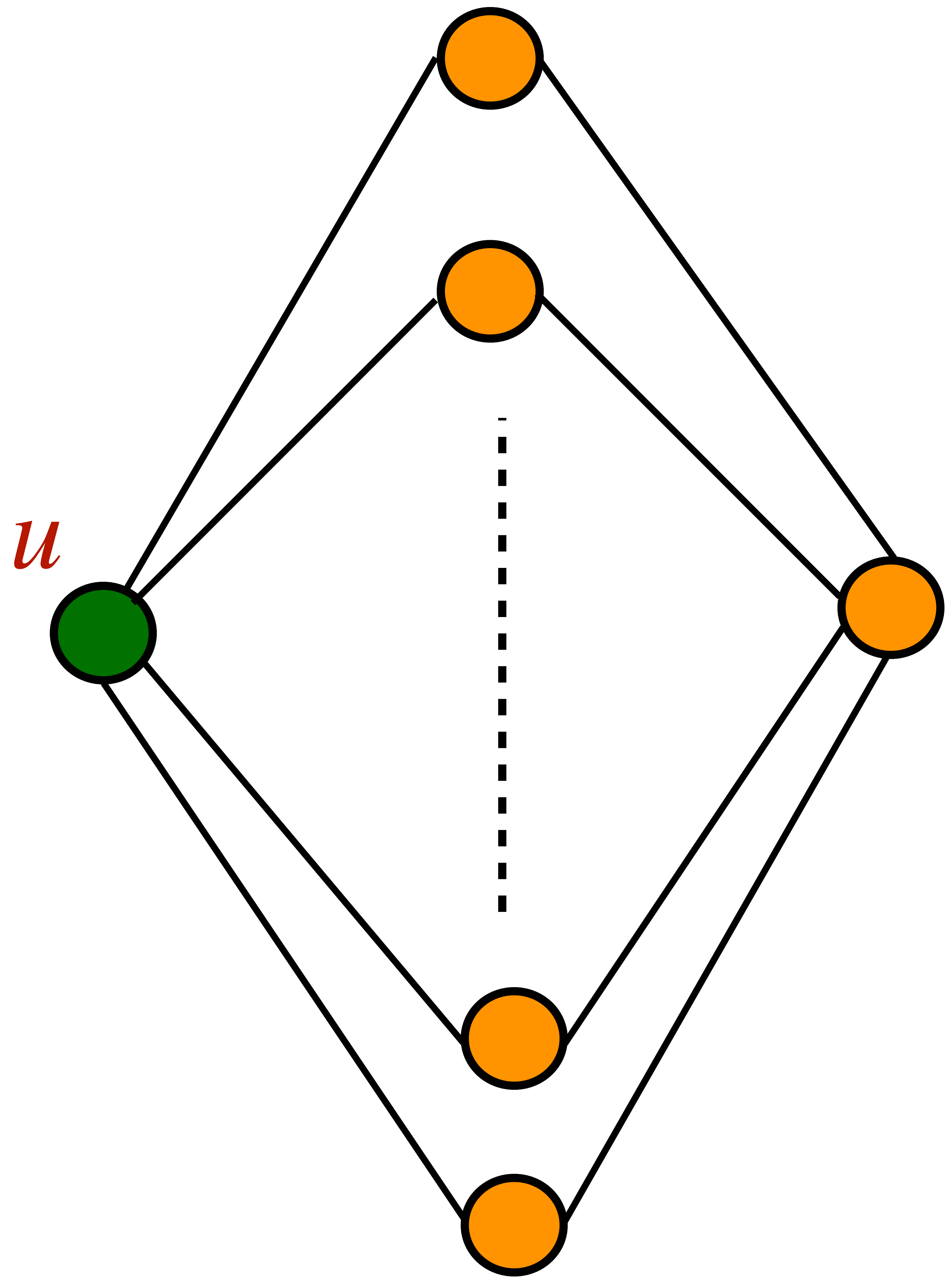
Random Variables $X_1, X_2, \ldots, X_n$ with:

Conditioned on any choice of $X_1, X_2, \ldots, X_{i-1}$,

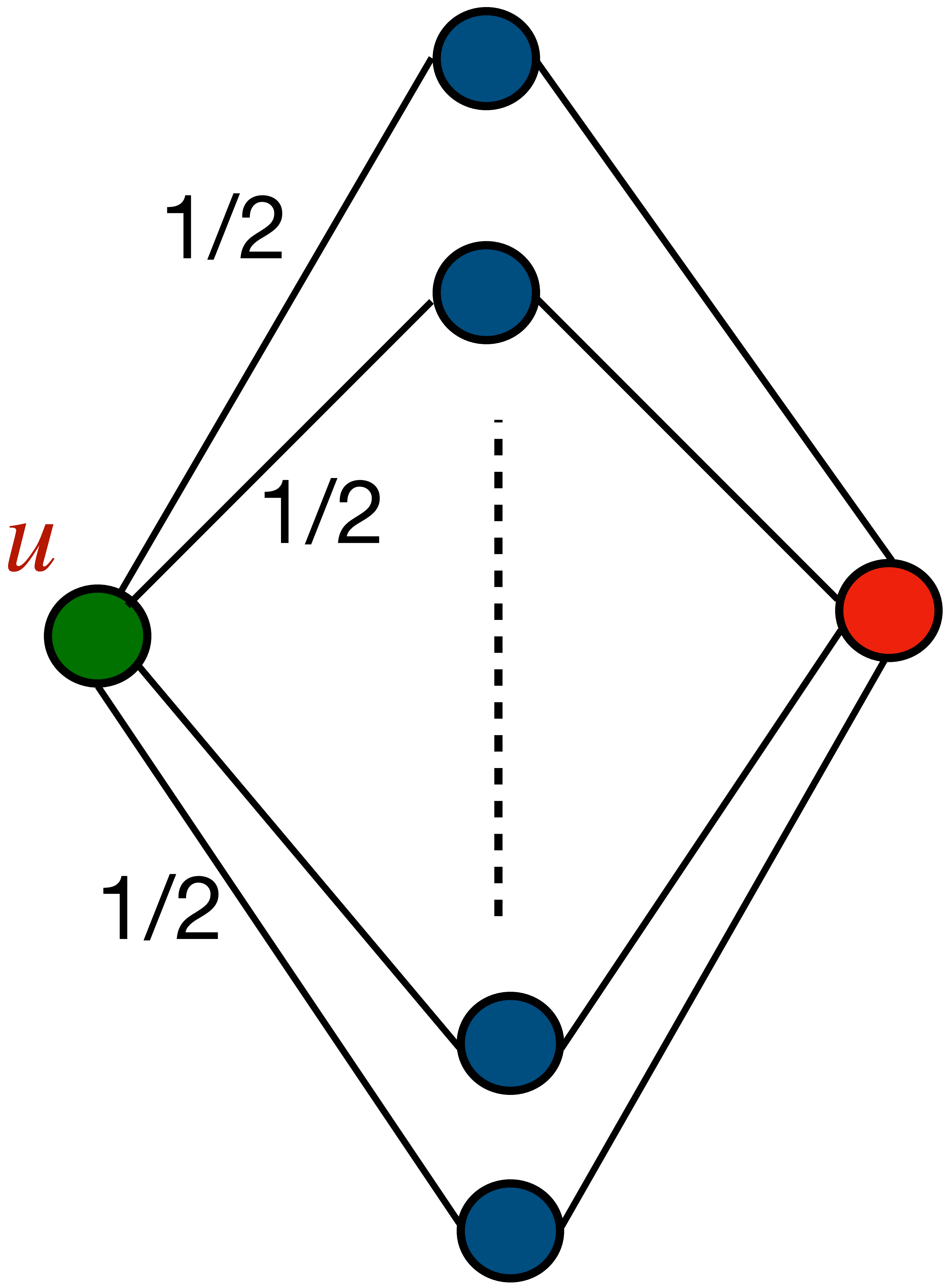
Expected value of $X_i \leq$ Probability that all the later X_j with $j > i$ are zero.

We want to bound expectation of sum to be $O(1)$ and variance of sum to be some $O(1) \cdot$ expectation.

This does not happen …



This does not happen …



W.p. $= 1/n$ mass received by u is $\approx n/2$

Expectation of mass is still $O(1)$.

Variance of mass is $\approx n$

Gain a lot, then Get out Game

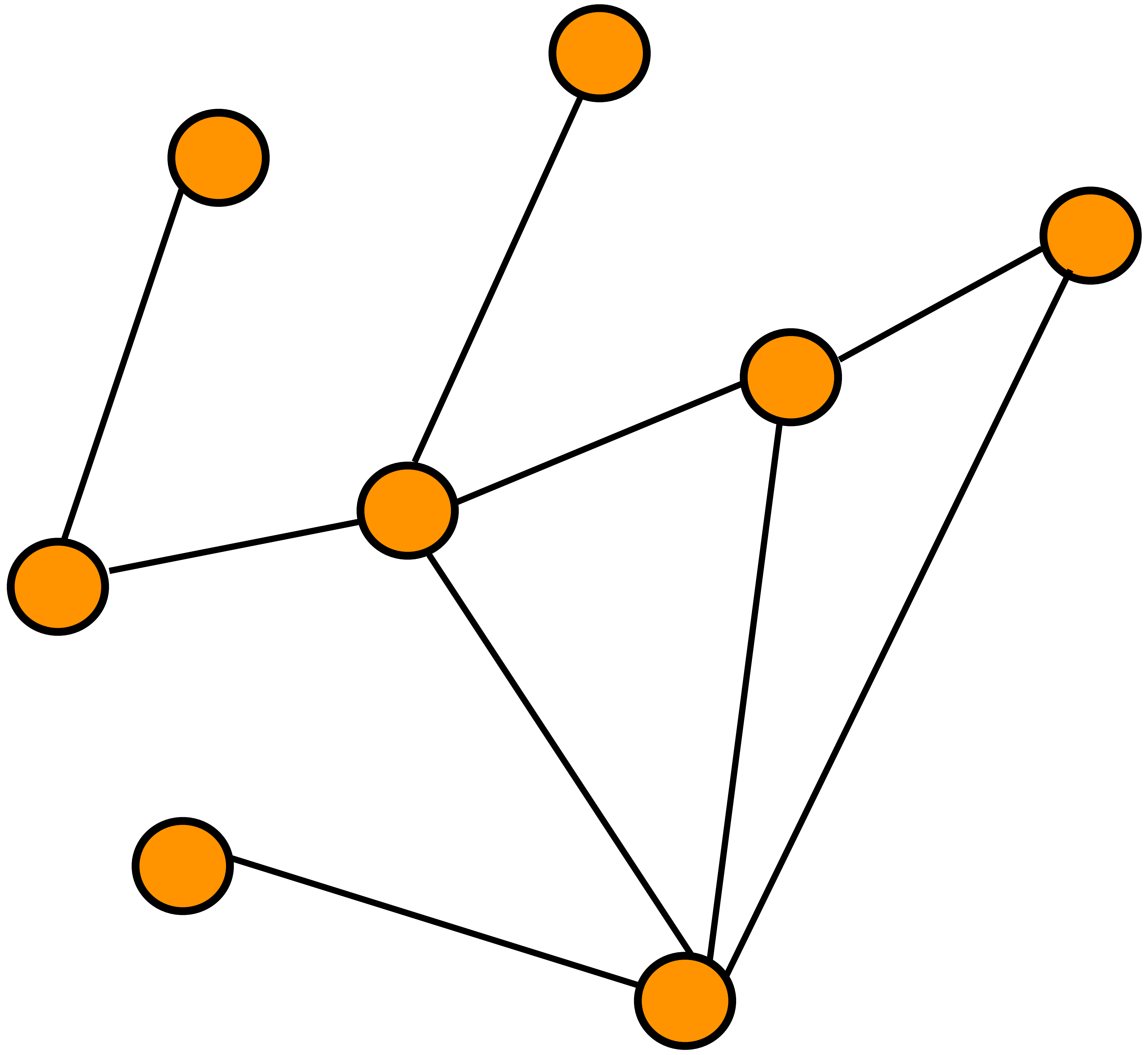
Random Variables $X_1, X_2, \ldots, X_n$ with:

Conditioned on any choice of $X_1, X_2, \ldots, X_{i-1}$,

Expected value of $X_i \leq$ Probability that all the later X_j with $j > i$ are zero.

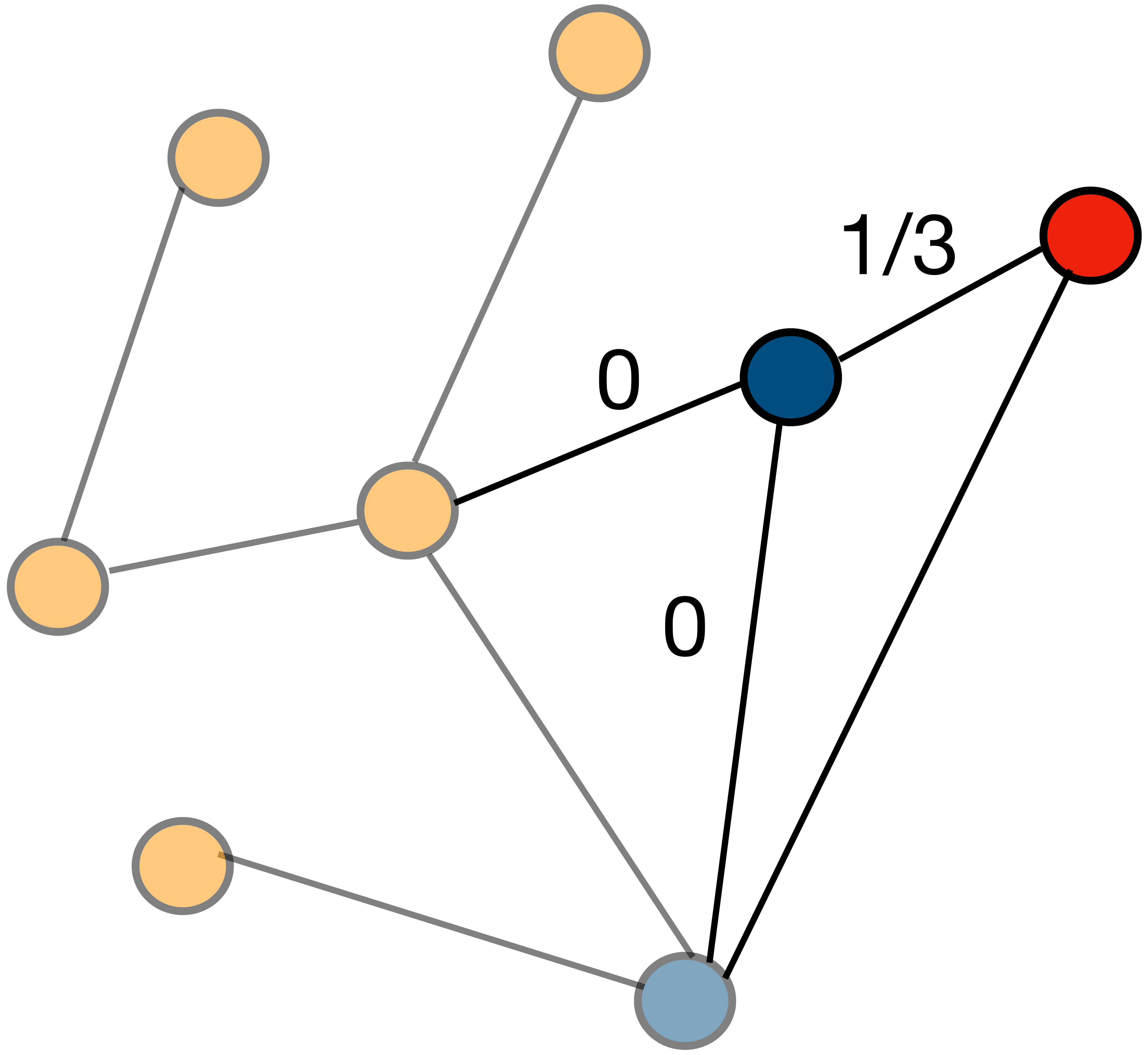
Need a good absolute bound on each X_i first

Final Reduction



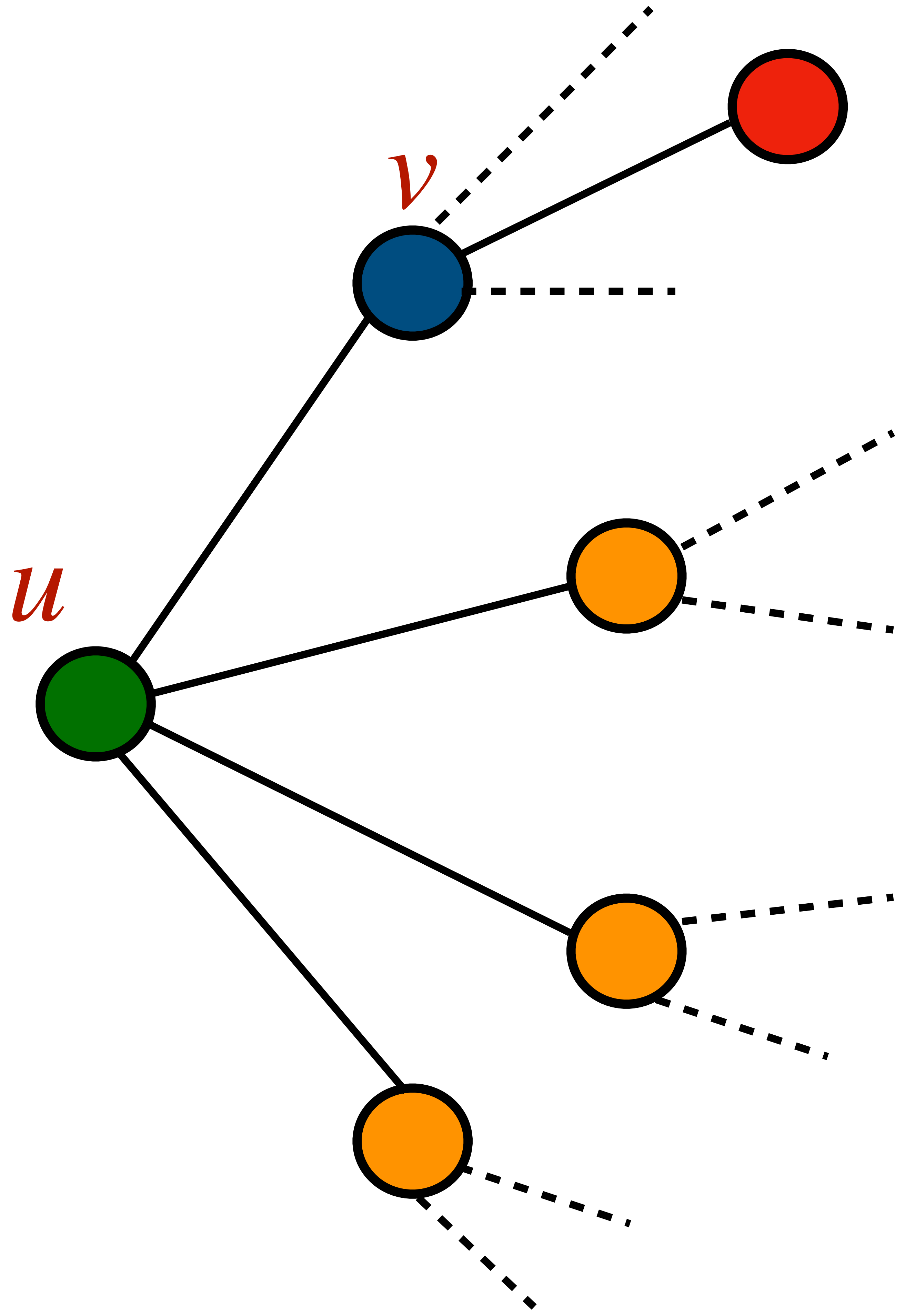
When vertex u goes to VC,
add a mass of $1/\deg(u)$
to the edges from u to
lower-degree neighbors of u
in the remaining graph.

Final Reduction



When vertex u goes to VC,
add a mass of $1/\deg(u)$
to the edges from u to
lower-degree neighbors of u
in the remaining graph.

Bounding Mass at each Iteration



Mass added across edge (u, v) is $1/\deg(v)$ only if $\deg(u)$ is at most $\deg(v)$

$$1/\deg(v) \leq 1/\deg(u)$$

Total mass is at most 1 at each iteration

Gain a lot, then Get out Game

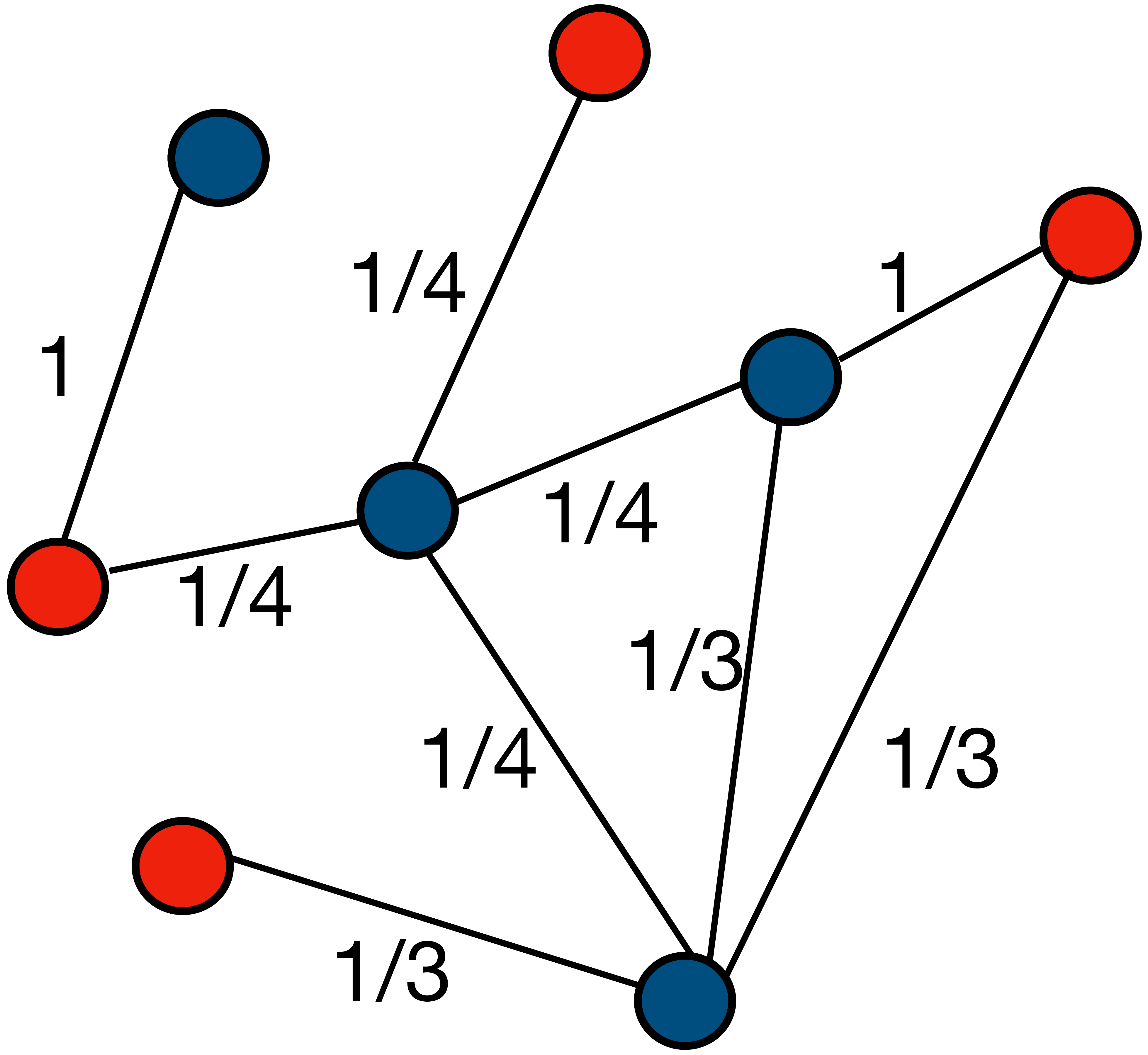
X_i - mass gained by vertex u at iteration i with each $X_i \leq 1$

Random Variables $X_1, X_2, \ldots, X_n$:

Expected value of $X_i \leq$ Probability that all the later X_j with $j > i$ are zero.

We can bound expectation of sum to be $O(1)$ and
variance of sum to be some $O(1) \cdot$ expectation.

Value of Fractional Matching?



Fractional Matching value: $\sum_{e} x_e$

~~Same as the size of the Vertex Cover~~

Half of size of Vertex Cover
in expectation

Back to Fractional Matching LP

$$\max \sum_e x_e$$

For all vertices u, $\sum_{e \ni u} x_e \leq 1$

With $0 \leq x_e \leq 1$ for all edges

$\sum_e x_e \geq$ half of expected size of VC

For all vertices u, $\sum_{e \ni u} x_e \leq O(1)$

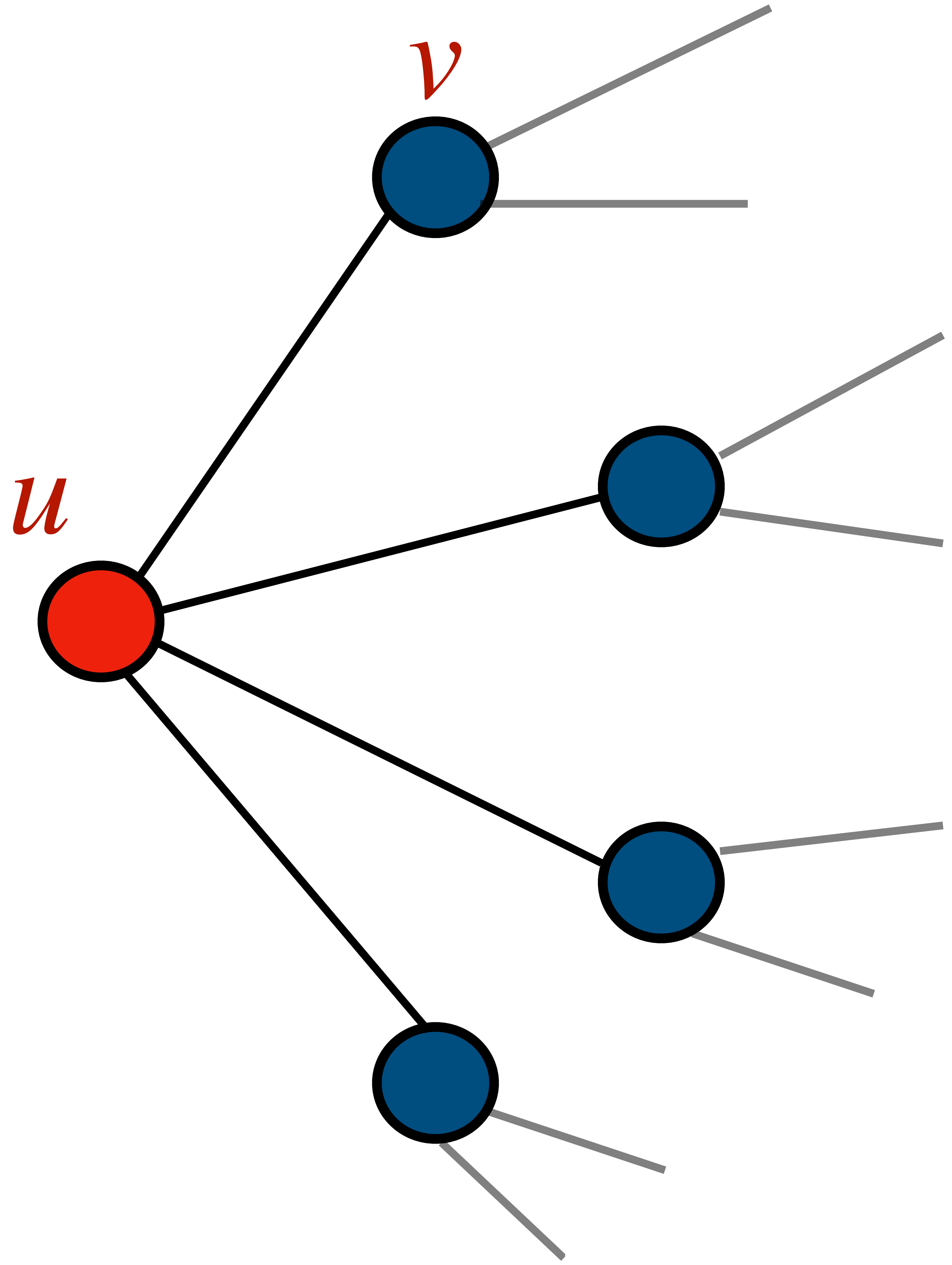
With $0 \leq x_e \leq 1$ for all edges

We satisfy LP with large value in expectation.

Plan for the rest of the talk

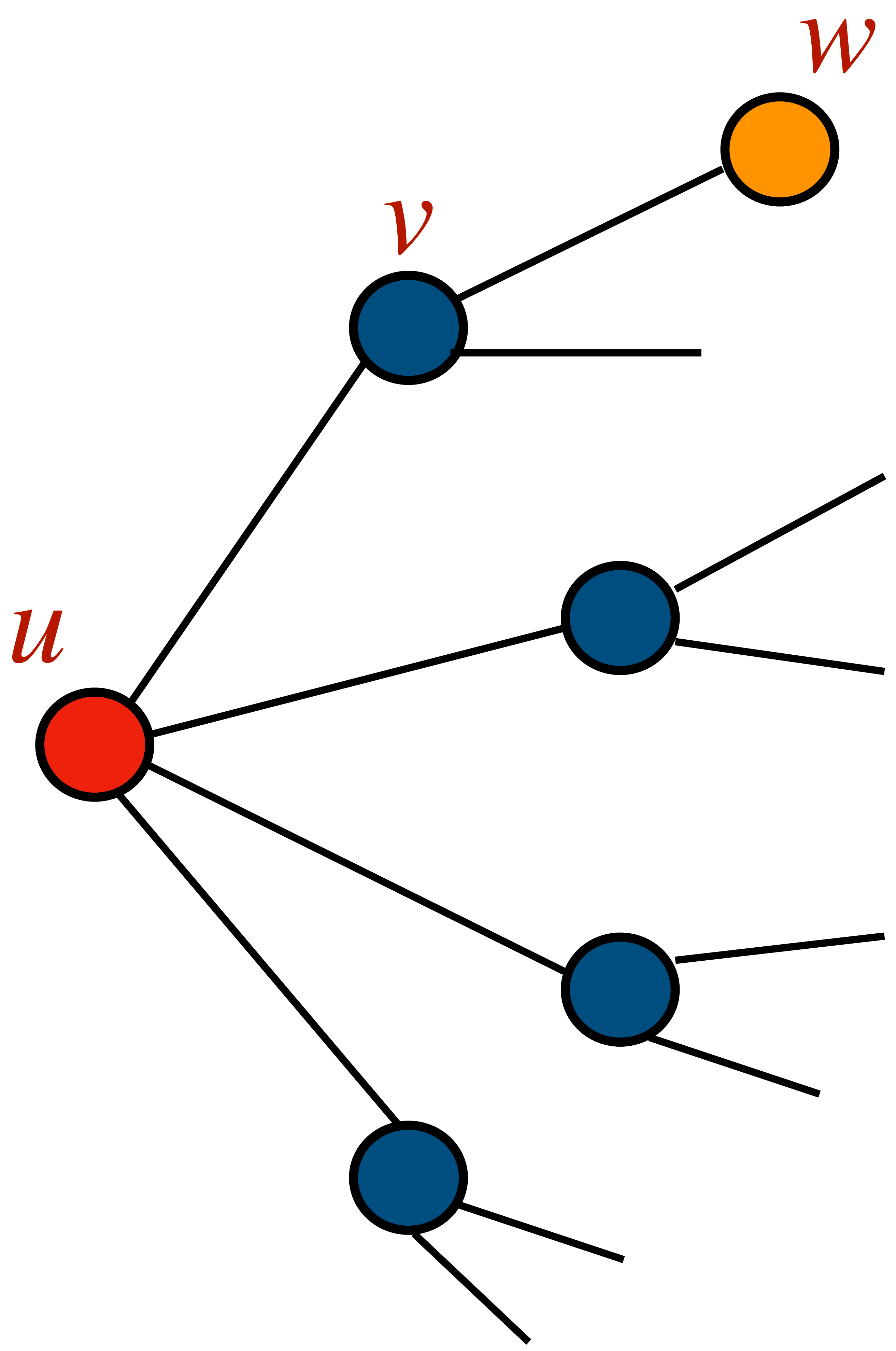
- Fractional matching and Vertex Cover
- Connections to MIS
- Our reduction to MIS
- **Challenges of implementation**

RGMIS Algorithm



[ACGMW '15] only needs to look at edges in the neighborhood of u

What do we want?

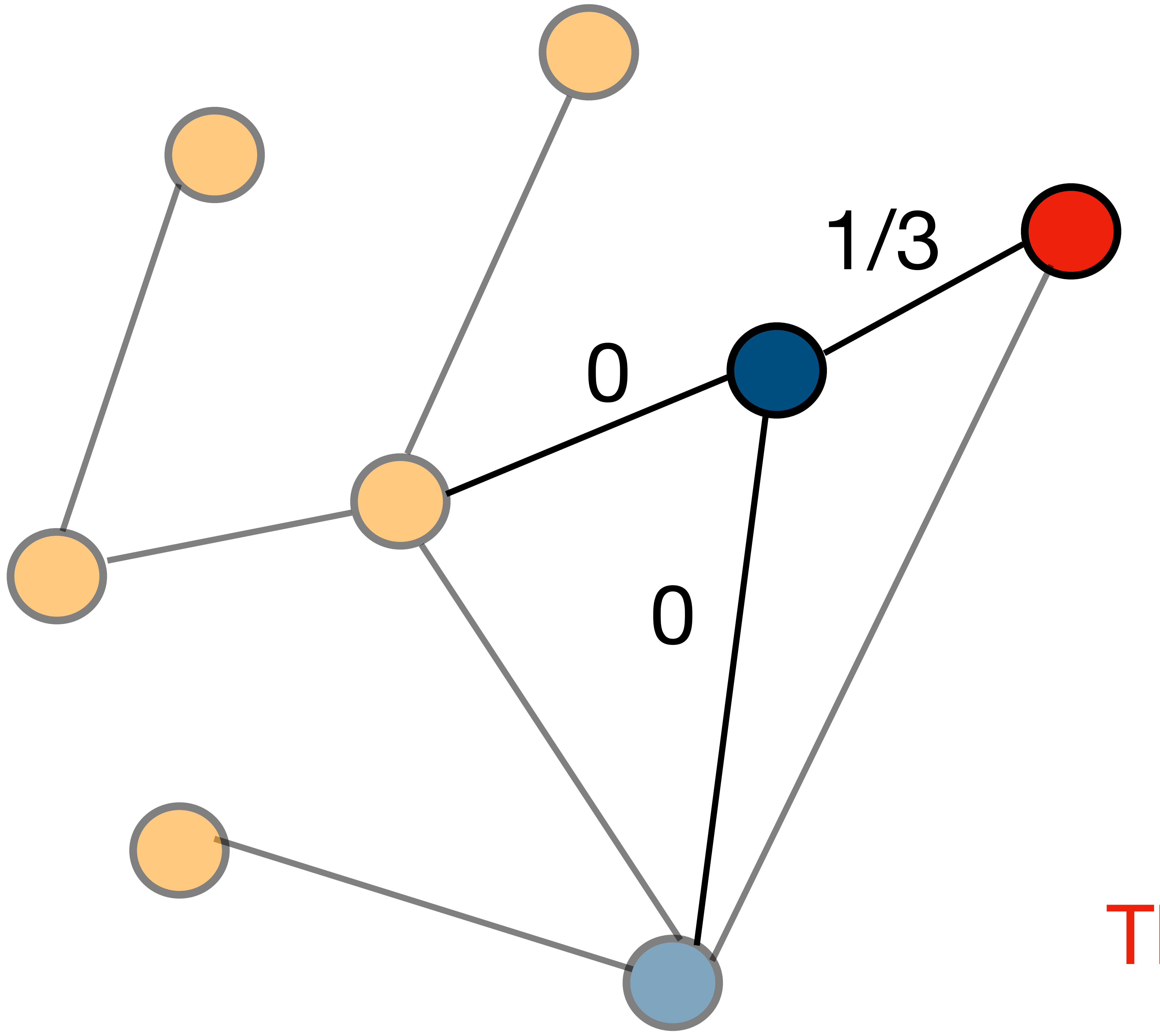


[ACGMW '15] only needs to look at edges in the neighborhood of u

We add mass to edges in 2-neighborhood of u

We can get this (somehow)

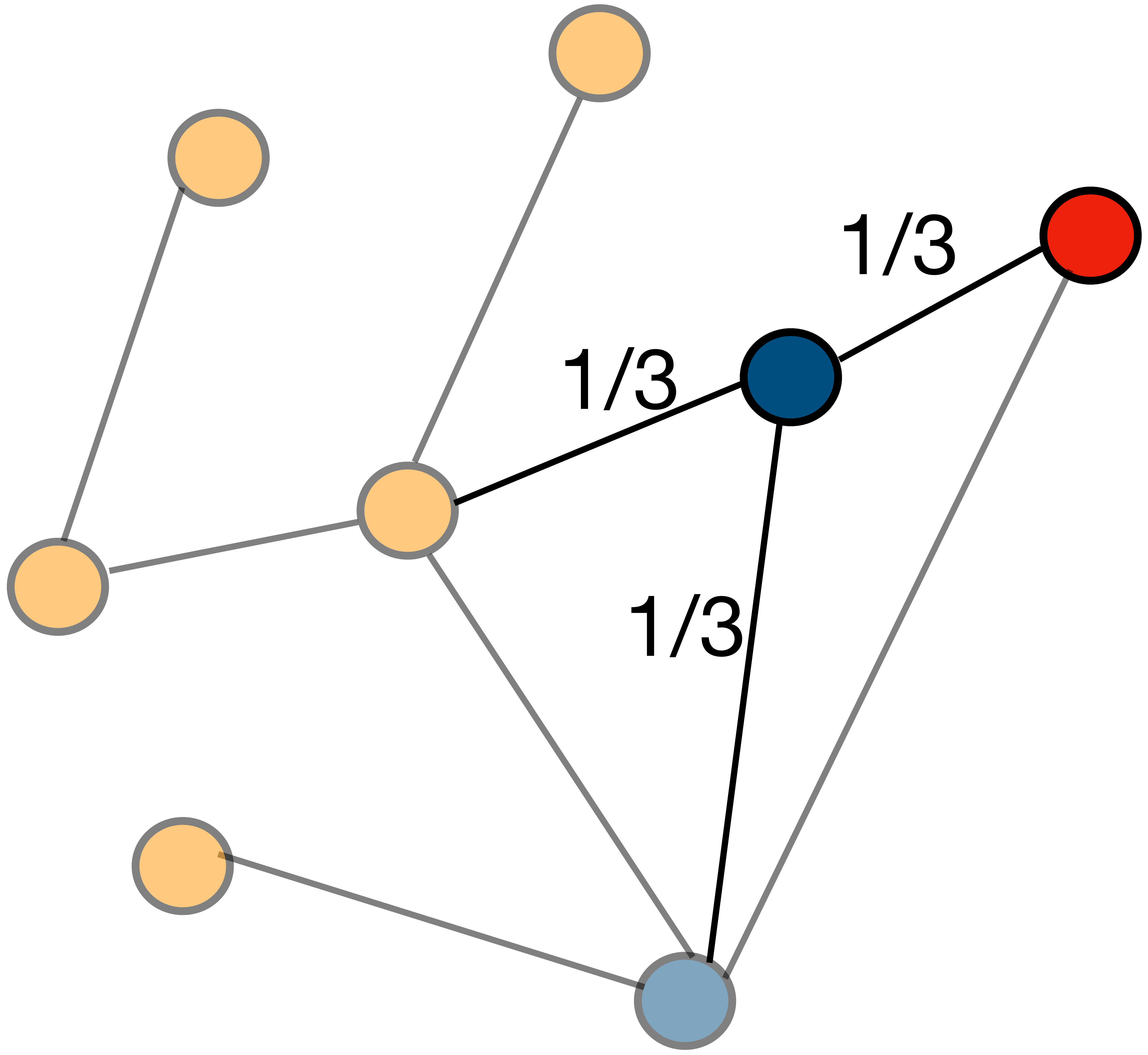
What more do we want?



When vertex u goes to VC,
add a mass of $1/\deg(u)$
to the edges from u to
lower-degree neighbors of u
in the remaining graph.

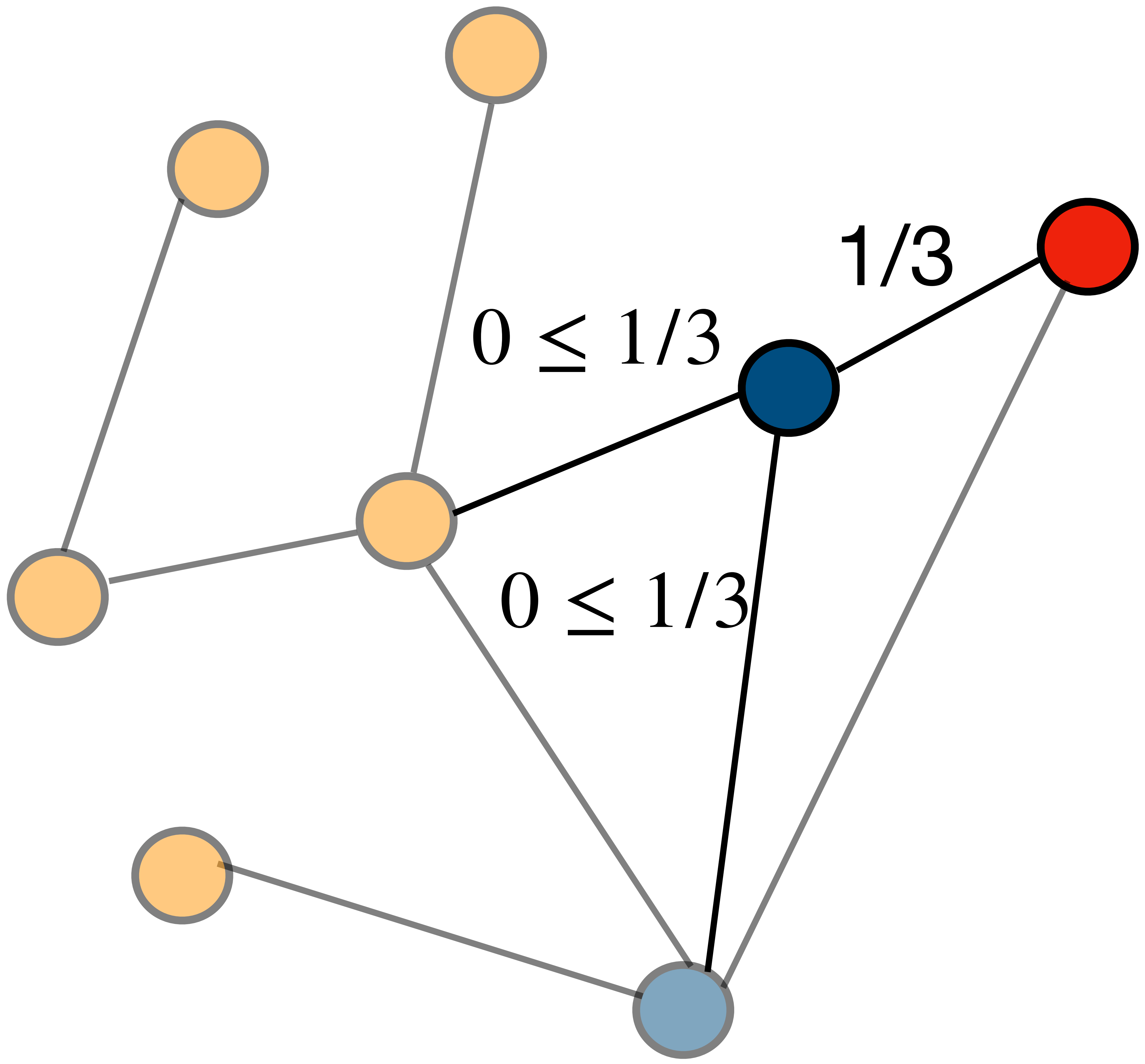
This we don't get - we do not implement this reduction exactly

What we actually implement (briefly)



Sample ALL THE EDGES
in the 2-neighborhood
with the same value

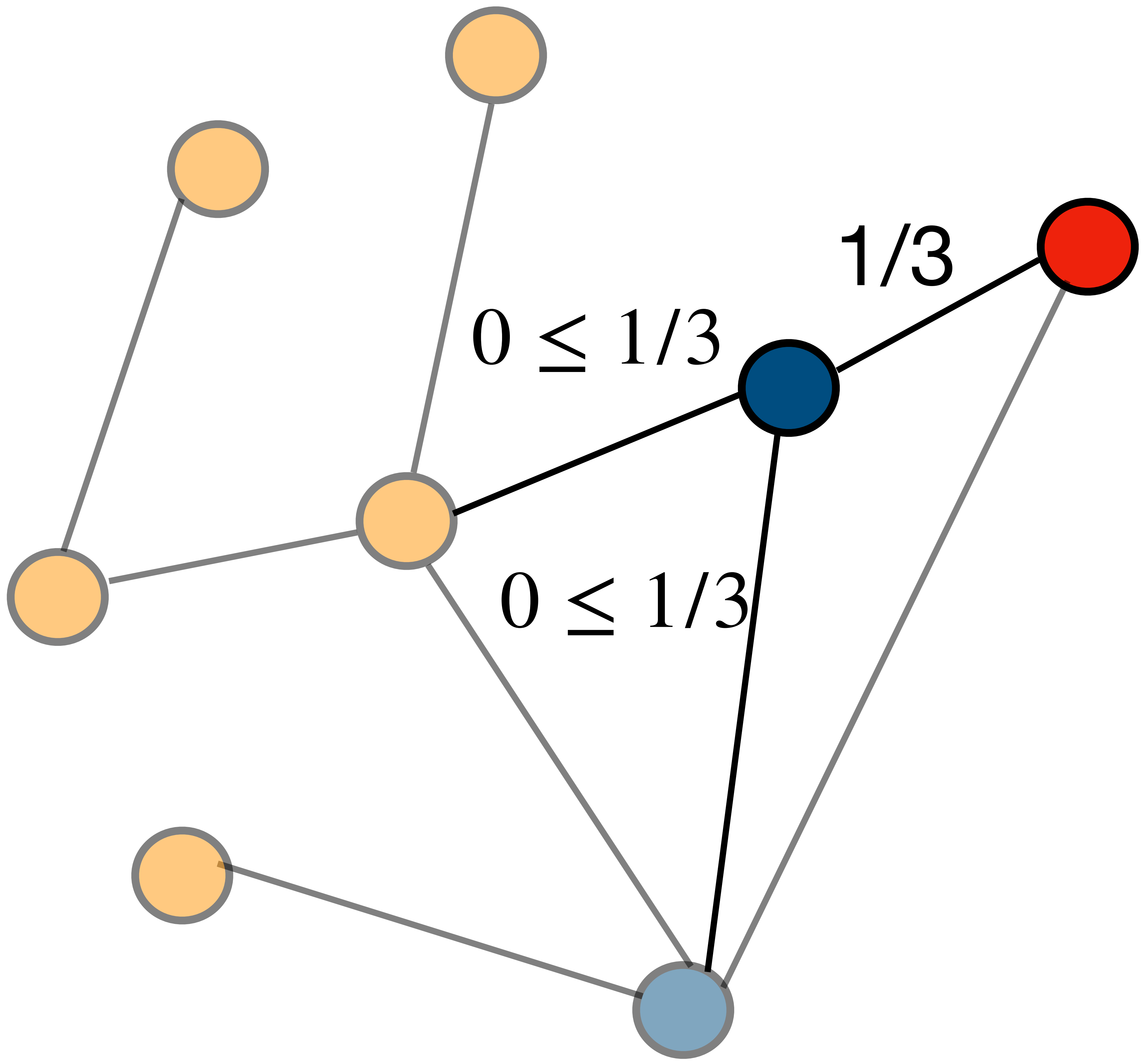
What we actually implement (briefly)



Sample ALL THE EDGES
in the 2-neighborhood
with the same value

We sample with a higher probability
than in our fractional matching

What we actually implement (briefly)

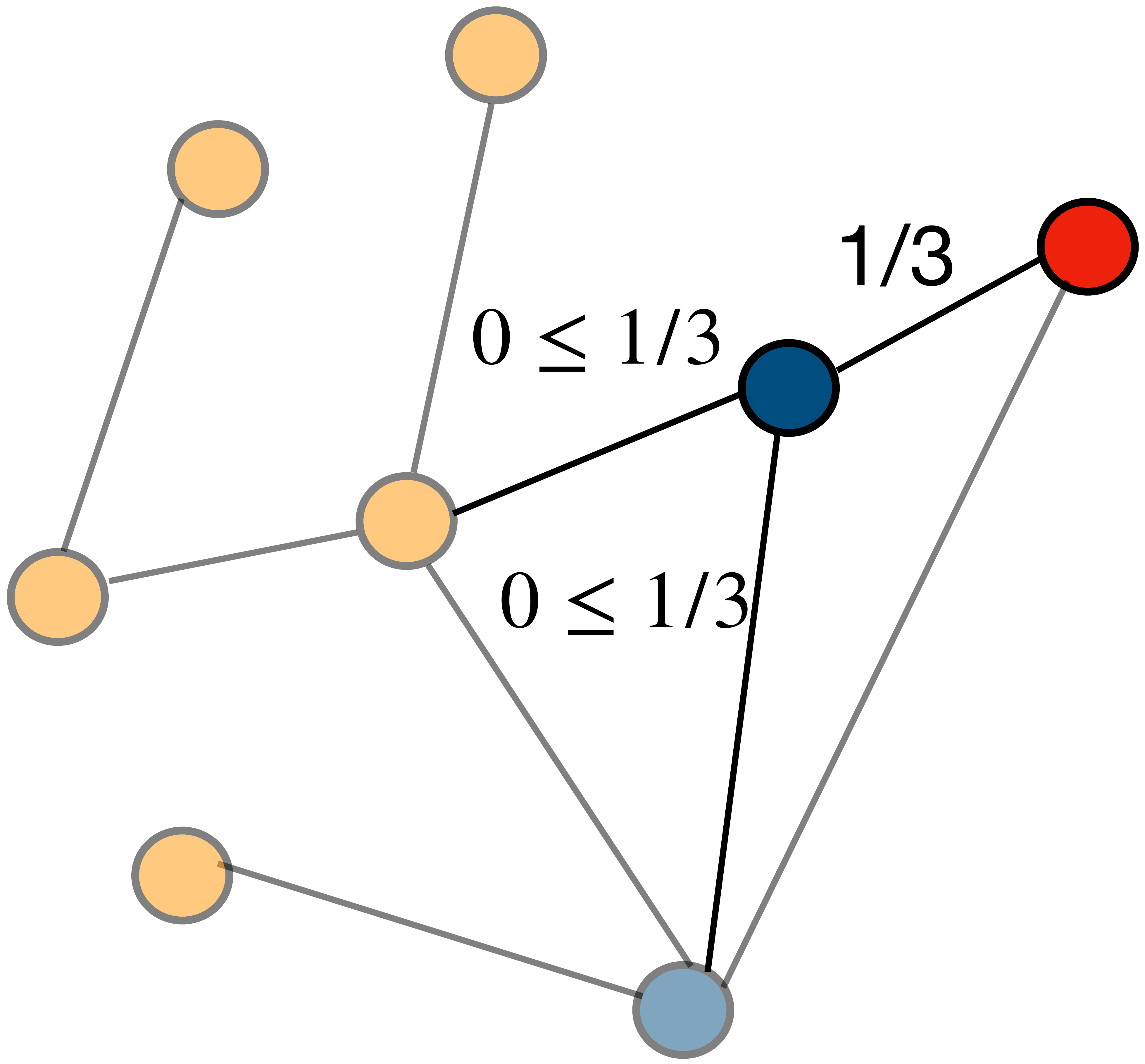


Sample ALL THE EDGES
in the 2-neighborhood
with the same value

We sample with a higher probability
than in our fractional matching

Support will contain a large matching!

What we actually implement (briefly)



Sample ALL THE EDGES
in the 2-neighborhood
with the same value

Do we sample too many edges?

Well, no.

Concluding Remarks

Takeaways

We gave a reduction from random greedy MIS to $O(1)$-approximate maximum matching

- Inspired by [Veldt ’24] reduction to vertex cover, generalized for matchings
- Using this reduction we gave a sketching algorithm for $O(1)$-apx matching in $O(\log \log n)$ rounds with $\tilde{O}(n)$ size sketches

Conclusion

Pass Complexity of $(1+\epsilon)$-approx maximum matching for any constant ϵ in dynamic streaming in $O(n \text{ poly} \log n)$ space is $\Theta(\log \log n)$

Exploiting connections to maximal independent sets
[Veldt ’24, ACGMW ’15, AKNS ’24]

Conclusion

Pass Complexity of $(1+\epsilon)$-approx maximum matching for any constant ϵ in dynamic streaming in $O(n \text{ poly} \log n)$ space is $\Theta(\log \log n)$

Open Questions:

- Dependence on ϵ for $(1+\epsilon)$-approx? Currently poly in $1/\epsilon$.
  [Ahn-Guha '18, Assadi-Liu-Tarjan '21, Chen-Kol-Paramonov-Saxena-Song-Yu '21, Assadi-S '23, Shang-En Huang Hsin-Hao Su '23]

- Better upper bounds or conditional lower bounds for MPC model?

Thank You!