Building spanning graphs in the semirandom tree process

Dominik Schmid

(Joint work with Michael Anastos, Maurício Collares, Joshua Erde, Mihyun Kang, and Gregory Sorkin)

TU Graz

DIMEA Combinatorial Potluck 2024
November 16, 2024

Overview

- Power of two choices

Overview

- Power of two choices
- Random graph processes

Overview

- Power of two choices
- Random graph processes
- Semirandom **star** process

Overview

- Power of two choices
- Random graph processes
- Semirandom **star** process
- Semirandom **tree** process

Overview

- Power of two choices
- Random graph processes
- Semirandom **star** process
- Semirandom **tree** process
  - Main result

Overview

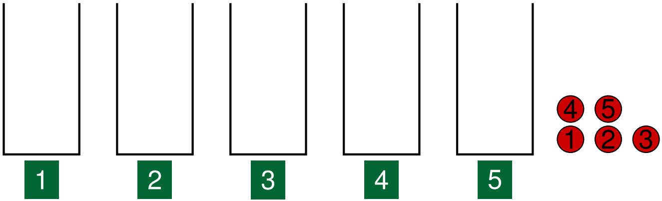
- Power of two choices
- Random graph processes
- Semirandom **star** process
- Semirandom **tree** process
  - Main result
  - Proof outline

Power of two choices

- Place n balls into n bins

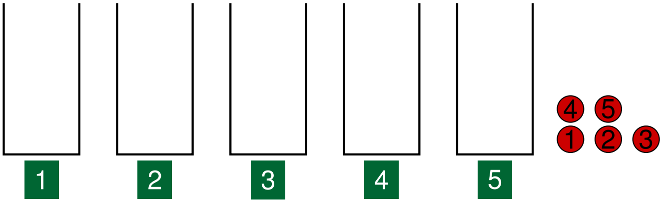
Power of two choices

- Place n balls into n bins



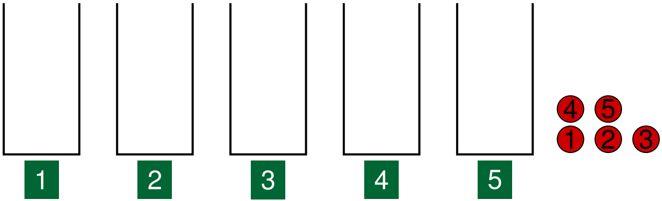
Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort



Power of two choices

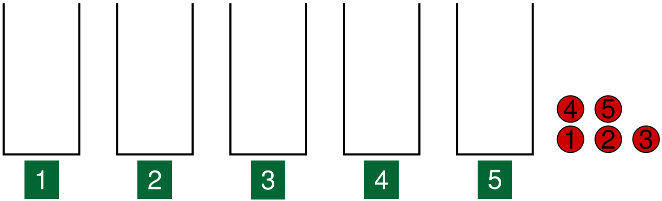
- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science



Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

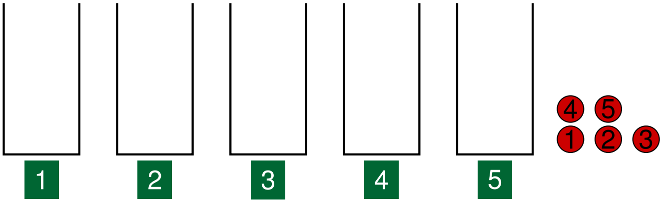


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose bin **uniformly at random**

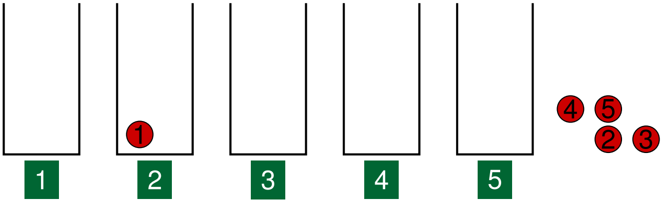


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose bin **uniformly at random**

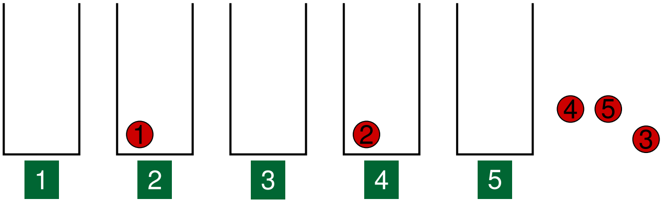


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose bin **uniformly at random**

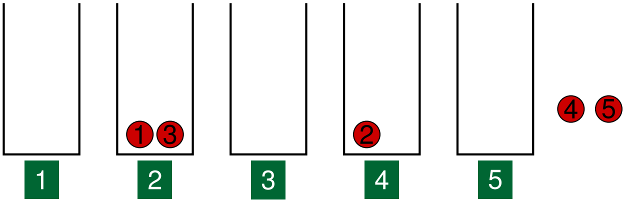


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose bin **uniformly at random**

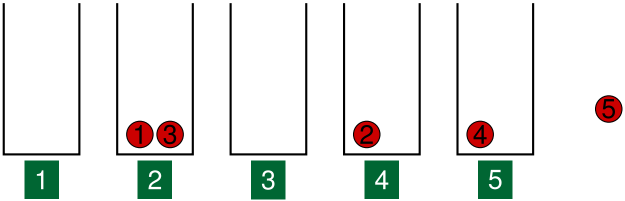


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose bin **uniformly at random**

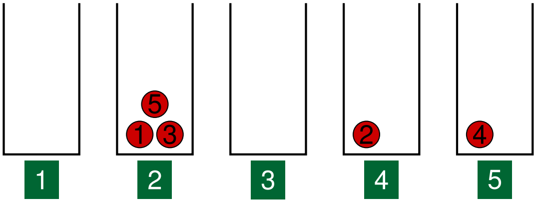


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose bin **uniformly at random**

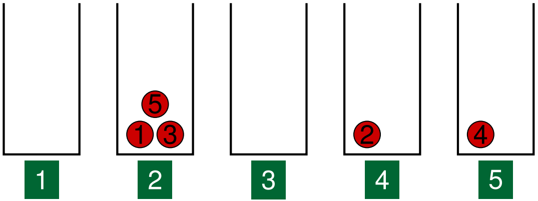


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

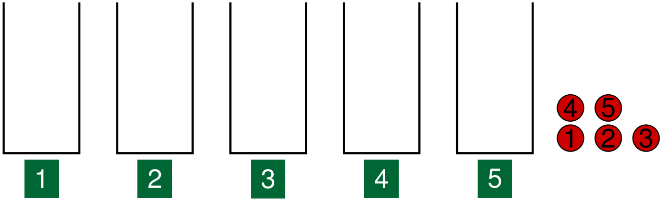
- Choose bin **uniformly at random**
- Maximum load $M(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$



Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

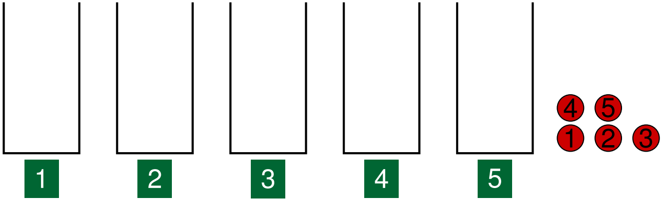


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**

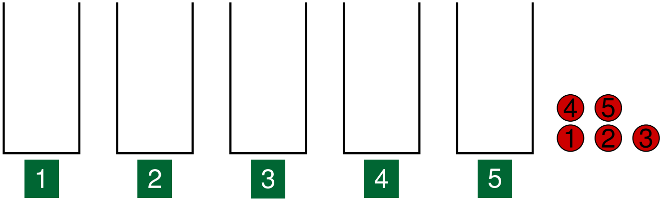


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

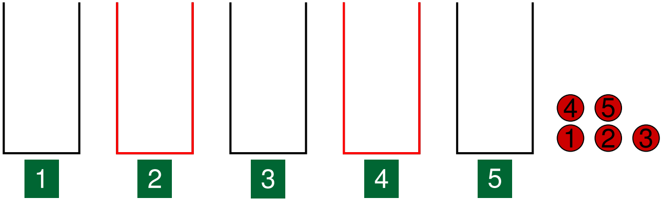


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

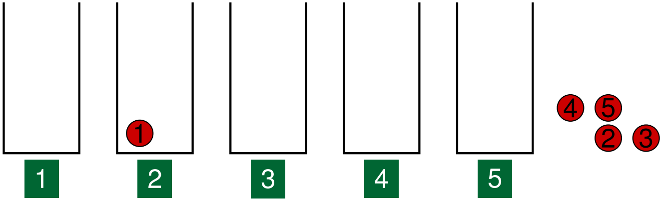


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

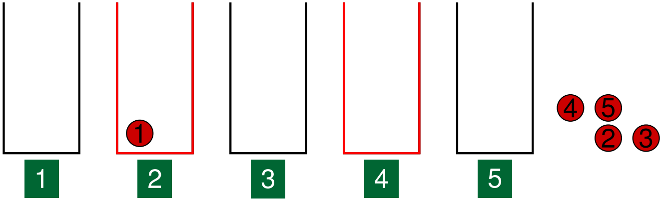


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

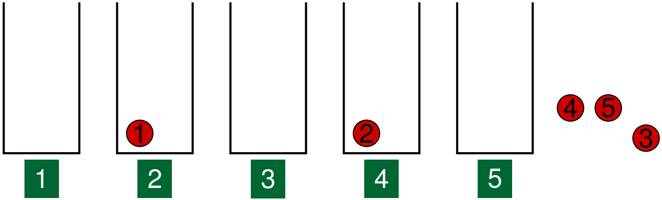


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

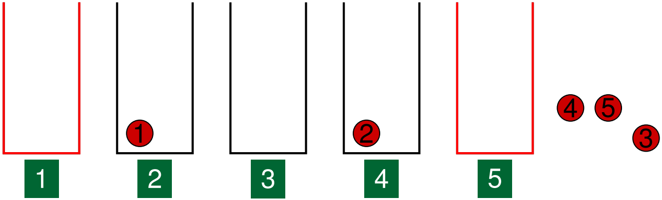


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

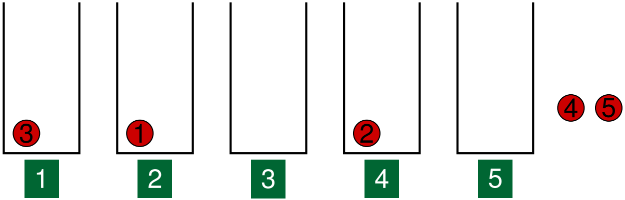


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

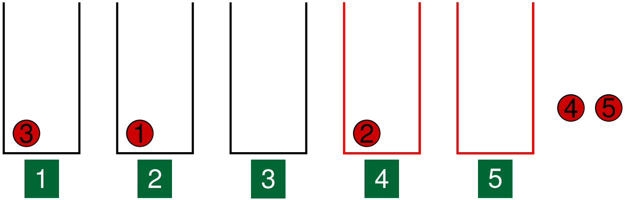


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

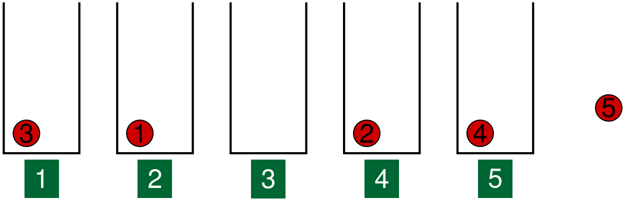


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

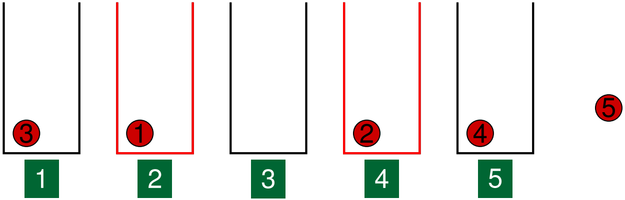


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

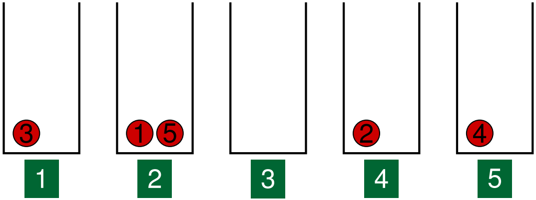


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls

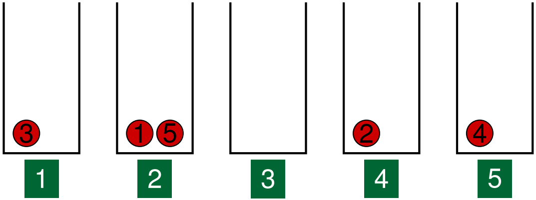


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls
- Maximum load $M(n) = \Theta(\log \log n)$ Azar, Broder, Karlin, and Upfal (1999)

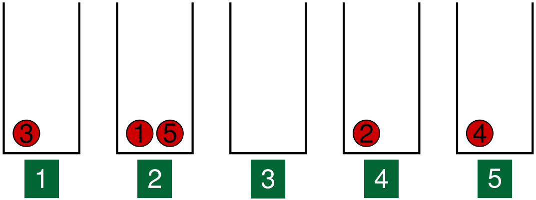


Power of two choices

- Place n balls into n bins
- Goal: Balance loads using minimal effort
  - Applications in computer science

Which bin to choose for which ball?

- Choose **two** bins **uniformly at random**
- **Deterministically** place ball into bin with fewer balls
- Maximum load $M(n) = \Theta(\log \log n)$ Azar, Broder, Karlin, and Upfal (1999)
- Exponential decrease of maximum load



Erdős-Rényi random graph process

- Empty graph $G_0 = ([n], \emptyset)$

Erdős-Rényi random graph process

- Empty graph $G_0 = ([n], \emptyset)$
- Sequentially add edges chosen uniformly at random

Erdős-Rényi random graph process

- Empty graph $G_0 = ([n], \emptyset)$
- Sequentially add edges chosen uniformly at random
  - e_m is chosen uniformly at random from $\binom{[n]}{2} \setminus \{e_1, \ldots, e_{m-1}\}$

Erdős-Rényi random graph process

- Empty graph $G_0 = ([n], \emptyset)$
- Sequentially add edges chosen uniformly at random
  - e_m is chosen uniformly at random from $\binom{[n]}{2} \setminus \{e_1, \ldots, e_{m-1}\}$
  - $G_m = ([n], \{e_1, \ldots, e_m\})$

Erdős-Rényi random graph process

- Empty graph $G_0 = ([n], \emptyset)$
- Sequentially add edges chosen uniformly at random
  - e_m is chosen uniformly at random from $\binom{[n]}{2} \setminus \{e_1, \ldots, e_{m-1}\}$
  - $G_m = ([n], \{e_1, \ldots, e_m\}) \sim G(n, m) =$ random graph on n vertices with m edges

Phase transition in G_m

Emergence of the Giant component

Erdős and Rényi, 1960

Phase transition in G_m

Emergence of the Giant component

Erdős and Rényi, 1960

- $m = \frac{(1-\epsilon)n}{2}$: Trees of order $O(\log n)$

Phase transition in G_m

Emergence of the Giant component

Erdős and Rényi, 1960

- $m = \frac{(1-\epsilon)n}{2}$: Trees of order $O(\log n)$
- $m = \frac{(1+\epsilon)n}{2}$: Unique component of order $\Theta(n)$

Phase transition in G_m

Emergence of the Giant component

Erdős and Rényi, 1960

- $m = \frac{(1-\epsilon)n}{2}$: Trees of order $O(\log n)$
- $m = \frac{(1+\epsilon)n}{2}$: Unique component of order $\Theta(n)$

Delaying/accelerating the phase transition: The achlioptas process

Phase transition in G_m

Emergence of the Giant component

Erdős and Rényi, 1960

- $m = \frac{(1-\epsilon)n}{2}$: Trees of order $O(\log n)$
- $m = \frac{(1+\epsilon)n}{2}$: Unique component of order $\Theta(n)$

Delaying/accelerating the phase transition: The achlioptas process

- Each step, sample **two** edges uniformly at random

Phase transition in G_m

Emergence of the Giant component

Erdős and Rényi, 1960

- $m = \frac{(1-\epsilon)n}{2}$: Trees of order $O(\log n)$
- $m = \frac{(1+\epsilon)n}{2}$: Unique component of order $\Theta(n)$

Delaying/accelerating the phase transition: The achlioptas process

- Each step, sample **two** edges uniformly at random
  - Pick one of those edges according to some rule

Phase transition in G_m

Emergence of the Giant component — Erdős and Rényi, 1960

- $m = \frac{(1-\epsilon)n}{2}$: Trees of order $O(\log n)$
- $m = \frac{(1+\epsilon)n}{2}$: Unique component of order $\Theta(n)$

Delaying/accelerating the phase transition: The achlioptas process

- Each step, sample **two** edges uniformly at random
  - Pick one of those edges according to some rule
- There exists a rule, s.t. whp — Bohman and Frieze, 2001
  G_m^A has no Giant for $m = 0.535n$

Phase transition in G_m

Emergence of the Giant component

Erdős and Rényi, 1960

- $m = \frac{(1-\epsilon)n}{2}$: Trees of order $O(\log n)$
- $m = \frac{(1+\epsilon)n}{2}$: Unique component of order $\Theta(n)$

Delaying/accelerating the phase transition: The achlioptas process

- Each step, sample **two** edges uniformly at random
  - Pick one of those edges according to some rule
- There exists a rule, s.t. whp
  G_m^A has no Giant for $m = 0.535n$

  Bohman and Frieze, 2001
- There exists a rule, s.t. whp
  G_m^A has a Giant when $m = 0.385n$

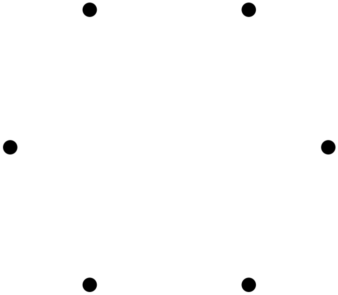
  Bohman and Kravitz, 2006

Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.

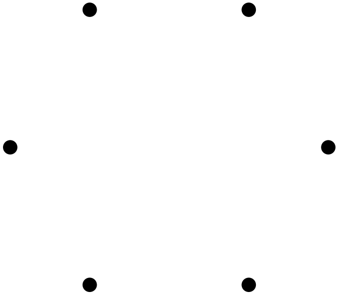
Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.



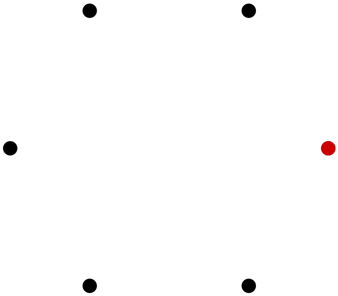
Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random



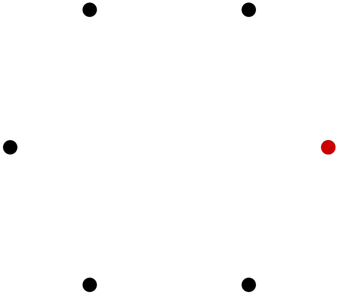
Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random



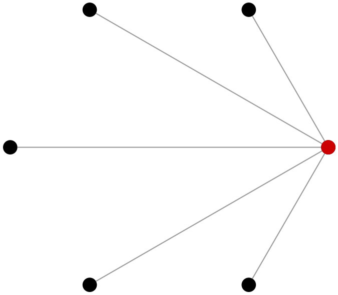
Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v



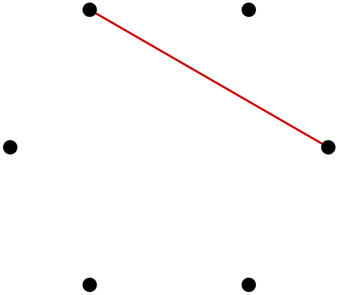
Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v



Semirandom **star** process

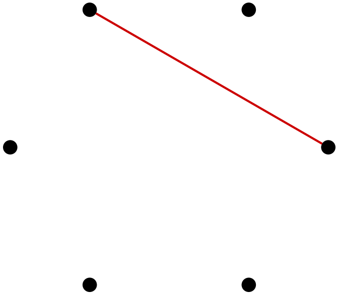
- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v



Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v

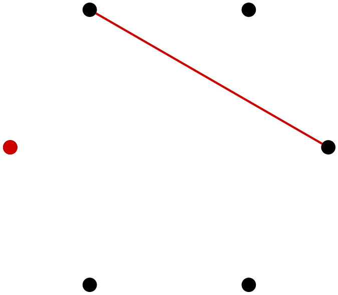
► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$



Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v

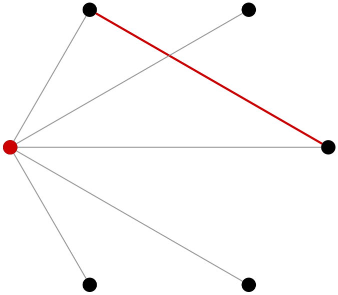
► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$



Semirandom **star** process

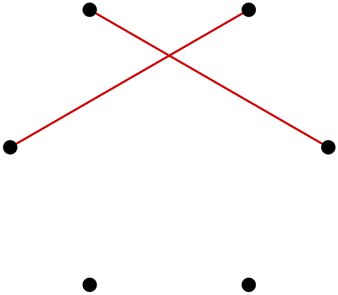
- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v

► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$



Semirandom **star** process

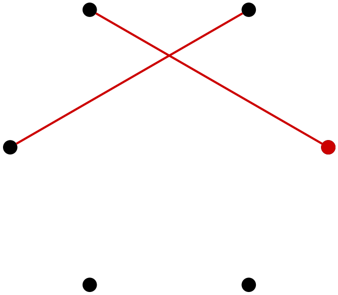
- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v
- ► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$



Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v

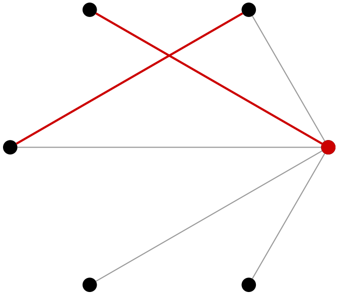
► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$



Semirandom **star** process

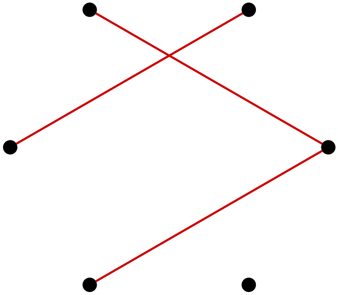
- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v

► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$



Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v
- $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$



Semirandom **star** process

Question: How fast can Builder construct graphs with certain properties?

Semirandom **star** process

Question: How fast can Builder construct graphs with certain properties?

Given graph property $\mathcal{P}$ and strategy σ

- $\tau(\mathcal{P}, \sigma) := \min_t \{G_t \in \mathcal{P}\}$

Semirandom **star** process

Question: How fast can Builder construct graphs with certain properties?

Given graph property $\mathcal{P}$ and strategy σ

- $\tau(\mathcal{P}, \sigma) := \min_t \{G_t \in \mathcal{P}\}$

There exist strategies $\sigma_1, \sigma_2, \sigma_3$ such that

Semirandom **star** process

Question: How fast can Builder construct graphs with certain properties?

Given graph property $\mathcal{P}$ and strategy σ

- $\tau(\mathcal{P}, \sigma) := \min_t \{G_t \in \mathcal{P}\}$

There exist strategies $\sigma_1, \sigma_2, \sigma_3$ such that

- $\tau(\mathcal{P}_C, \sigma_1) = n - 1$

Semirandom **star** process

Question: How fast can Builder construct graphs with certain properties?

Given graph property $\mathcal{P}$ and strategy σ

- $\tau(\mathcal{P}, \sigma) := \min_t \{G_t \in \mathcal{P}\}$

There exist strategies $\sigma_1, \sigma_2, \sigma_3$ such that

- $\tau(\mathcal{P}_C, \sigma_1) = n - 1$
- Whp. $\tau(\mathcal{P}_{PM}, \sigma_2) \leq 1.206n$ Gao, MacRury, and Prałat (2022)

Semirandom **star** process

Question: How fast can Builder construct graphs with certain properties?

Given graph property $\mathcal{P}$ and strategy σ

- $\tau(\mathcal{P}, \sigma) := \min_t \{G_t \in \mathcal{P}\}$

There exist strategies $\sigma_1, \sigma_2, \sigma_3$ such that

- $\tau(\mathcal{P}_C, \sigma_1) = n - 1$
- Whp. $\tau(\mathcal{P}_{PM}, \sigma_2) \leq 1.206n$ Gao, MacRury, and Prałat (2022)
- Whp. $\tau(\mathcal{P}_{HC}, \sigma_3) \leq 1.817n$ Frieze, Gao, MacRury, Prałat, and Sorkin (2023)

Semirandom **star** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- In round t, a vertex $v \in [n]$ is chosen uniformly at random
- Builder picks any edge e_t incident to v

► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$

Semirandom **star** process

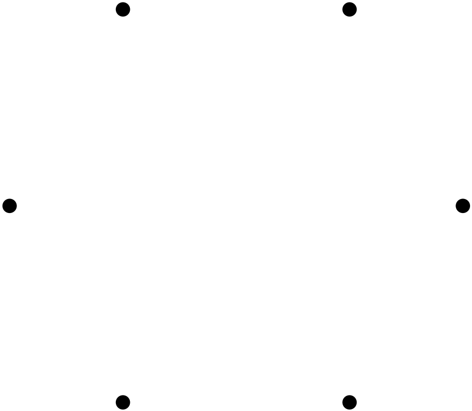
- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- Sample random spanning **star** S_t uniformly at random
- Builder picks any edge $e_t \in E(S_t)$

► $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$

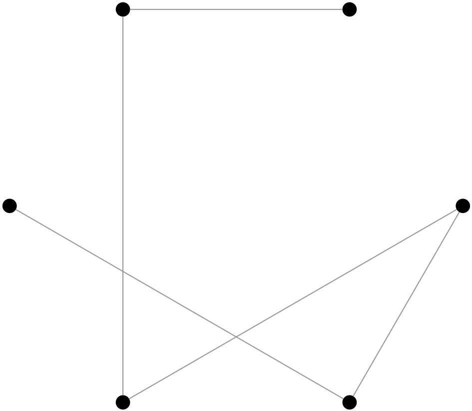
Semirandom **tree** process

- Set $G_0 = ([n], E_0)$ with $E_0 = \emptyset$.
- Sample random spanning **tree** T_t uniformly at random
- Builder picks any edge $e_t \in E(T_t)$
  - $G_t = ([n], E_t)$ with $E_t = E_{t-1} \cup e_t$

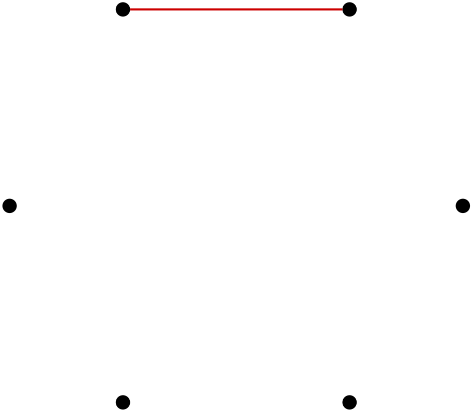
Connectedness in semirandom tree process



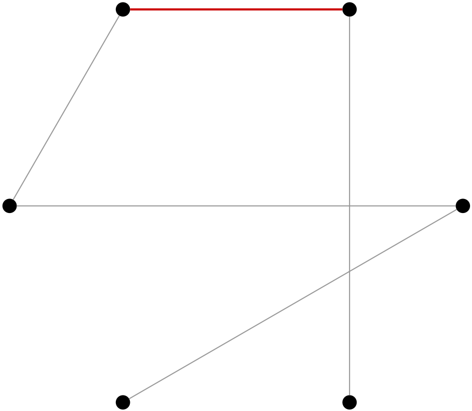
Connectedness in semirandom tree process



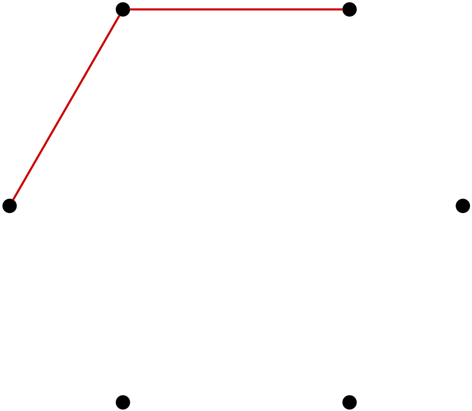
Connectedness in semirandom tree process



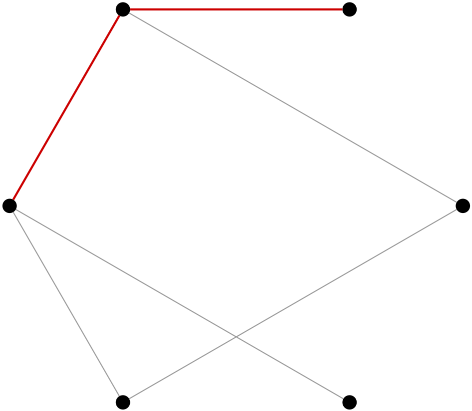
Connectedness in semirandom tree process



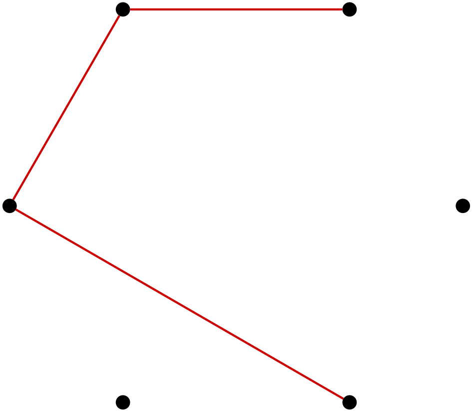
Connectedness in semirandom tree process



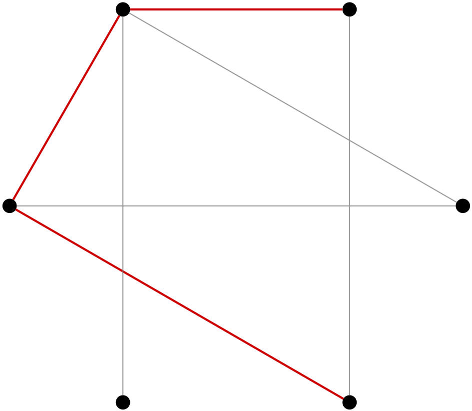
Connectedness in semirandom tree process



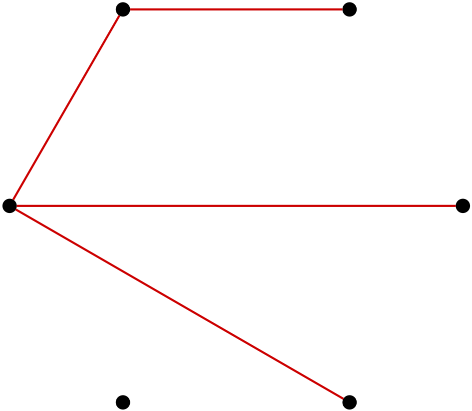
Connectedness in semirandom tree process



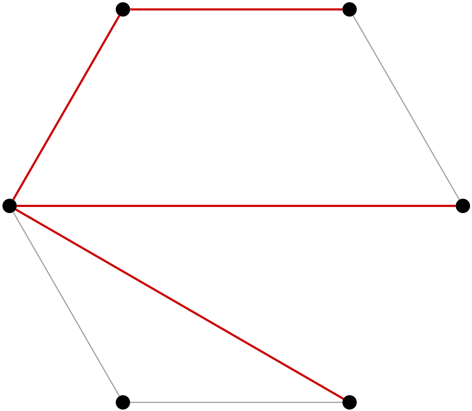
Connectedness in semirandom tree process



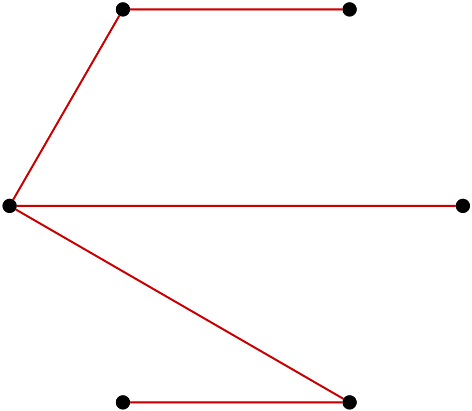
Connectedness in semirandom tree process



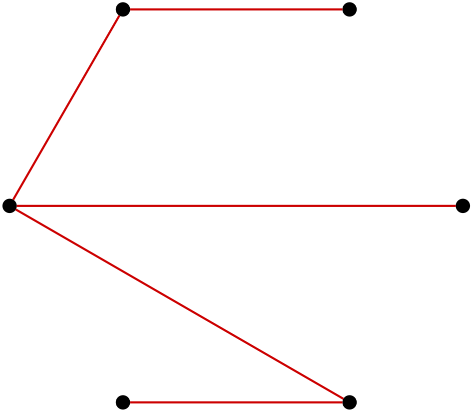
Connectedness in semirandom tree process



Connectedness in semirandom tree process

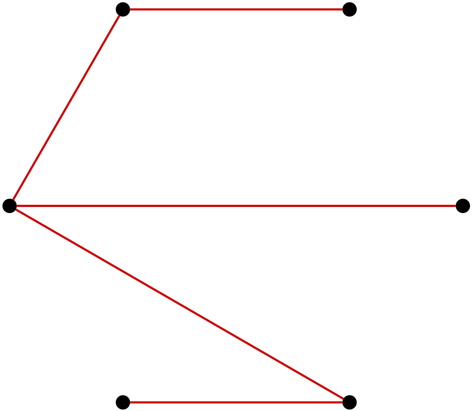


Connectedness in semirandom tree process



- There exists σ, such that $\tau(\mathcal{P}_C, \sigma) = n - 1$

Connectedness in semirandom tree process



- There exists σ, such that $\tau(\mathcal{P}_C, \sigma) = n - 1$
- Connectedness can be achieved deterministically in optimal time

Previous results

There exist strategies $\sigma_1, \sigma_2, \sigma_3$, such that whp

Previous results

There exist strategies $\sigma_1, \sigma_2, \sigma_3$, such that whp

- $\tau(\mathcal{P}_{PM}, \sigma_1) \leq \frac{n}{2} + o\left(n^{4/5}\right)$ Burova and Lichev (2022)

Previous results

There exist strategies $\sigma_1, \sigma_2, \sigma_3$, such that whp

- $\tau(\mathcal{P}_{PM}, \sigma_1) \leq \frac{n}{2} + o\left(n^{4/5}\right)$ Burova and Lichev (2022)
- $\tau(\mathcal{P}_{HC}, \sigma_2) \leq n + o\left(n^{4/5}\right)$

Previous results

There exist strategies $\sigma_1, \sigma_2, \sigma_3$, such that whp

- $\tau(\mathcal{P}_{PM}, \sigma_1) \leq \frac{n}{2} + o\left(n^{4/5}\right)$ Burova and Lichev (2022)
- $\tau(\mathcal{P}_{HC}, \sigma_2) \leq n + o\left(n^{4/5}\right)$
- $\tau(\mathcal{P}_{min,k}, \sigma_3) \leq \frac{kn}{2} + o\left(n^{1/2}\right)$

Previous results

There exist strategies $\sigma_1, \sigma_2, \sigma_3$, such that whp

- $\tau(\mathcal{P}_{PM}, \sigma_1) \leq \frac{n}{2} + o\left(n^{4/5}\right)$ Burova and Lichev (2022)
- $\tau(\mathcal{P}_{HC}, \sigma_2) \leq n + o\left(n^{4/5}\right)$
- $\tau(\mathcal{P}_{min,k}, \sigma_3) \leq \frac{kn}{2} + o\left(n^{1/2}\right)$

Results are asymptotically optimal

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$
- $\mathcal{P}_H$: Containment of H

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$
- $\mathcal{P}_H$: Containment of H

Erdős-Rényi graph process G_m :

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$
- $\mathcal{P}_H$: Containment of H

Erdős-Rényi graph process G_m :

- $\tau(\mathcal{P}_H) \sim n^{2-\frac{1}{\Delta}}$

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$
- $\mathcal{P}_H$: Containment of H

Erdős-Rényi graph process G_m :

- $\tau(\mathcal{P}_H) \sim n^{2-\frac{1}{\Delta}}$

Semirandom **star** process:

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$
- $\mathcal{P}_H$: Containment of H

Erdős-Rényi graph process G_m :

- $\tau(\mathcal{P}_H) \sim n^{2-\frac{1}{\Delta}}$

Semirandom **star** process:

- $\exists \sigma : \tau(\mathcal{P}_H, \sigma) \leq \frac{3\Delta n}{2}(1 + o_\Delta(1))$

Ben-Eliezer, Gishboliner, Hefetz and Krivelevich (2020)

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$
- $\mathcal{P}_H$: Containment of H

Erdős-Rényi graph process G_m :

- $\tau(\mathcal{P}_H) \sim n^{2-\frac{1}{\Delta}}$

Semirandom **star** process:

- $\exists \sigma : \tau(\mathcal{P}_H, \sigma) \leq \frac{3\Delta n}{2}(1 + o_\Delta(1))$

  Ben-Eliezer, Gishboliner, Hefetz and Krivelevich (2020)

  - H can have as many as $\frac{\Delta n}{2}$ edges

Building spanning graphs

- H : (vertex-)spanning graph with maximum degree $\Delta \geq 3$
- $\mathcal{P}_H$: Containment of H

Erdős-Rényi graph process G_m :

- $\tau(\mathcal{P}_H) \sim n^{2-\frac{1}{\Delta}}$

Semirandom **star** process:

- $\exists \sigma : \tau(\mathcal{P}_H, \sigma) \leq \frac{3\Delta n}{2}(1 + o_\Delta(1))$

  Ben-Eliezer, Gishboliner, Hefetz and Krivelevich (2020)

  - H can have as many as $\frac{\Delta n}{2}$ edges
  - Need at least $\frac{\Delta n}{2}$ rounds

Main result

Theorem

[Anastos, Collares, Erde, Kang, S., Sorkin, 2024^{+}]

*Let H be graph with maximum degree Δ. Then, in the semirandom **tree** process, there is a strategy σ such that whp*

Main result

Theorem

[Anastos, Collares, Erde, Kang, S., Sorkin, 2024^+]

Let H be graph with maximum degree Δ. Then, in the semirandom ***tree*** *process, there is a strategy σ such that whp*

$$\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$$

Main result

Theorem

[Anastos, Collares, Erde, Kang, S., Sorkin, 2024^{+}]

*Let H be graph with maximum degree Δ. Then, in the semirandom **tree** process, there is a strategy σ such that whp*

$$\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$$

- $o_\Delta(1) \iff$ tending to 0 as $\Delta \to \infty$

Main result

Theorem

[Anastos, Collares, Erde, Kang, S., Sorkin, 2024^+]

Let H be graph with maximum degree Δ. Then, in the semirandom ***tree*** *process, there is a strategy σ such that whp*

$$\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$$

- $o_\Delta(1) \iff$ tending to 0 as $\Delta \to \infty$
- If Δ is a small constant, $\tau(\mathcal{P}_H, \sigma) = \Theta(n)$

Main result

Theorem

[Anastos, Collares, Erde, Kang, S., Sorkin, 2024^+]

*Let H be graph with maximum degree Δ. Then, in the semirandom **tree** process, there is a strategy σ such that whp*

$$\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}(1 + o_\Delta(1))$$

- $o_\Delta(1) \iff$ tending to 0 as $\Delta \to \infty$
- If Δ is a small constant, $\tau(\mathcal{P}_H, \sigma) = \Theta(n)$
- Asymptotically optimal for $\Delta = \Delta(n) \to \infty$

Notation

H: target graph with maximum degree Δ

Notation

H: target graph with maximum degree Δ

$G_t = ([n], E_t)$: Builder's graph at time t

Notation

H: target graph with maximum degree Δ
$G_t = ([n], E_t)$: Builder's graph at time t

- Find copy of H in G_t

Notation

$H\colon$ target graph with maximum degree Δ

$G_t = ([n], E_t)\colon$ Builder's graph at time t

- Find copy of H in G_t
- Embedding $\Phi\colon V(H) \to [n]$

Notation

H: target graph with maximum degree Δ

$G_t = ([n], E_t)$: Builder's graph at time t

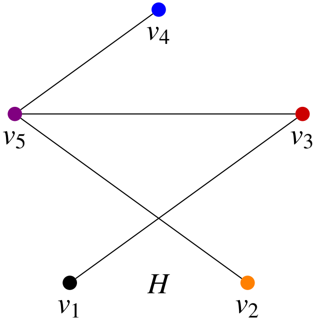
- Find copy of H in G_t
- Embedding $\Phi\colon V(H) \to [n]$
  - $\forall\, vw \in E(H)\colon \Phi(v)\Phi(w) \in E_t$

Notation

H: target graph with maximum degree Δ

$G_t = ([n], E_t)$: Builder's graph at time t

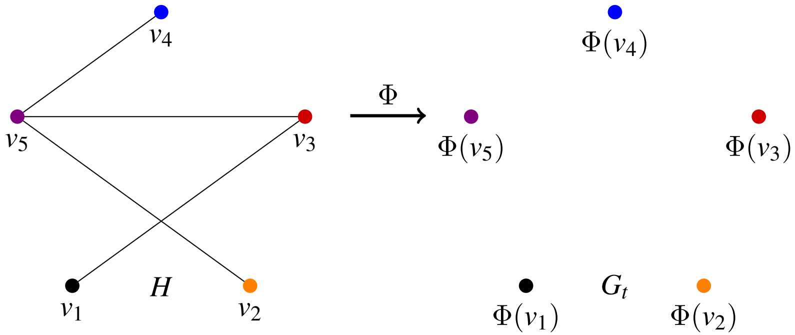
- Find copy of H in G_t
- Embedding $\Phi\colon V(H) \to [n]$
  - $\forall\, vw \in E(H)\colon \Phi(v)\Phi(w) \in E_t$



Notation

H: target graph with maximum degree Δ
$G_t = ([n], E_t)$: Builder's graph at time t

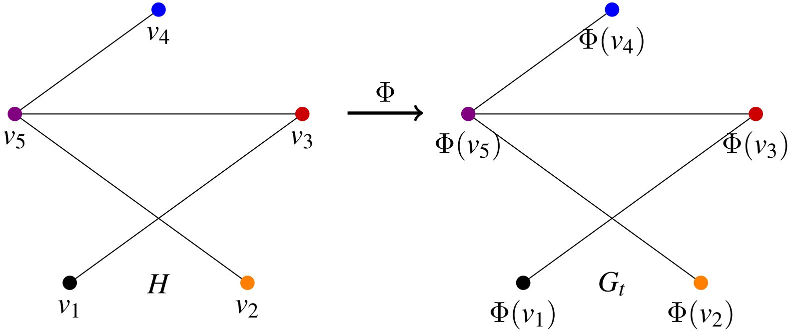
- Find copy of H in G_t
- Embedding $\Phi : V(H) \to [n]$
  - $\forall\, vw \in E(H) : \Phi(v)\Phi(w) \in E_t$



Notation

$H:$ target graph with maximum degree Δ
$G_t = ([n], E_t):$ Builder's graph at time t

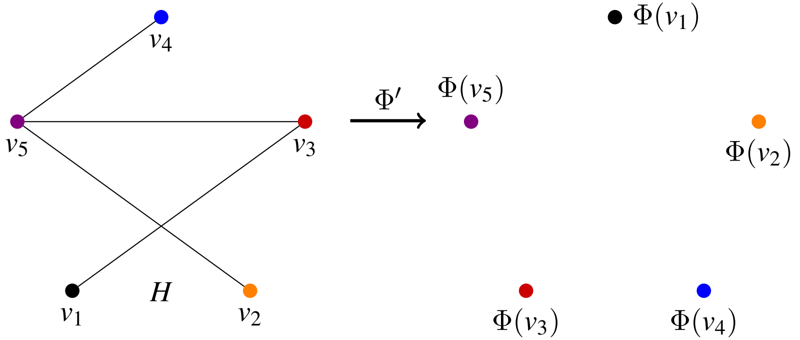
- Find copy of H in G_t
- Embedding $\Phi\colon V(H) \to [n]$
  - $\forall\, vw \in E(H)\colon \Phi(v)\Phi(w) \in E_t$



Notation

H: target graph with maximum degree Δ
$G_t = ([n], E_t)$: Builder's graph at time t

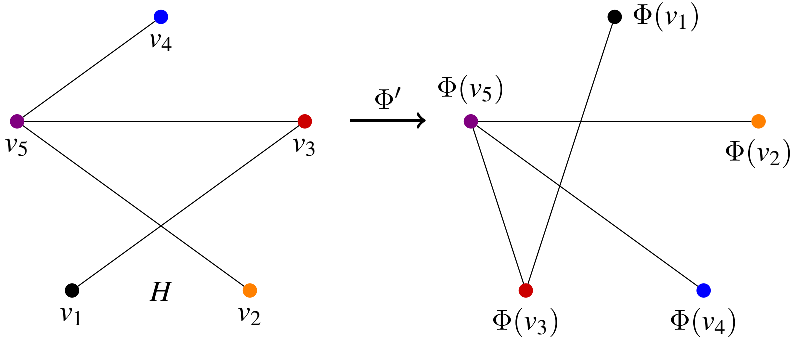
- Find copy of H in G_t
- Embedding $\Phi\colon V(H) \to [n]$
  - $\forall\, vw \in E(H)\colon \Phi(v)\Phi(w) \in E_t$



Notation

H: target graph with maximum degree Δ
$G_t = ([n], E_t)$: Builder's graph at time t

- Find copy of H in G_t
- Embedding $\Phi: V(H) \to [n]$
  - $\forall\, vw \in E(H): \Phi(v)\Phi(w) \in E_t$



Outline of strategy

- $\frac{\Delta n}{2}(1 + o_\Delta(1))$ rounds to ‘spend’

Outline of strategy

- $\frac{\Delta n}{2}(1 + o_\Delta(1))$ rounds to 'spend'
- Phase I: Greedy strategy

Outline of strategy

- $\frac{\Delta n}{2}(1 + o_\Delta(1))$ rounds to 'spend'
- Phase I: Greedy strategy
  - $\frac{\Delta n}{2}$ rounds

Outline of strategy

- $\frac{\Delta n}{2}(1 + o_\Delta(1))$ rounds to 'spend'
- Phase I: Greedy strategy
  - $\frac{\Delta n}{2}$ rounds
  - Build 'almost complete' copy of H

Outline of strategy

- $\frac{\Delta n}{2}(1 + o_\Delta(1))$ rounds to 'spend'
- Phase I: Greedy strategy
  - $\frac{\Delta n}{2}$ rounds
  - Build 'almost complete' copy of H
- Phase II: Replacement strategy

Outline of strategy

- $\frac{\Delta n}{2}(1 + o_\Delta(1))$ rounds to 'spend'
- Phase I: Greedy strategy
  - $\frac{\Delta n}{2}$ rounds
  - Build 'almost complete' copy of H
- Phase II: Replacement strategy
  - $\frac{\Delta n}{2} \cdot o_\Delta(1)$ rounds

Outline of strategy

- $\frac{\Delta n}{2}(1 + o_\Delta(1))$ rounds to 'spend'
- Phase I: Greedy strategy
  - $\frac{\Delta n}{2}$ rounds
  - Build 'almost complete' copy of H
- Phase II: Replacement strategy
  - $\frac{\Delta n}{2} \cdot o_\Delta(1)$ rounds
  - Extend to a complete copy

Phase I: Greedy strategy

- Fix arbitrary embedding Φ

Phase I: Greedy strategy

- Fix arbitrary embedding Φ
- Claim edges corresponding to this embedding, whenever offered

Phase I: Greedy strategy

- Fix arbitrary embedding Φ
- Claim edges corresponding to this embedding, whenever offered
  - $M(t) := \{\Phi(v)\Phi(w) \mid vw \in E(H)\} \setminus E_t$

Phase I: Greedy strategy

- Fix arbitrary embedding Φ
- Claim edges corresponding to this embedding, whenever offered
  - $M(t) := \{\Phi(v)\Phi(w) \mid vw \in E(H)\} \setminus E_t$

How efficient is this strategy?

Phase I: Greedy strategy

- Fix arbitrary embedding Φ
- Claim edges corresponding to this embedding, whenever offered
  - $M(t) := \{\Phi(v)\Phi(w) \mid vw \in E(H)\} \setminus E_t$

How efficient is this strategy?

- $\mathbb{P}\left[E(T_t) \cap M(t) \neq \emptyset\right]$

Phase I: Greedy strategy

- Fix arbitrary embedding Φ
- Claim edges corresponding to this embedding, whenever offered
  - $M(t) := \{\Phi(v)\Phi(w) \mid vw \in E(H)\} \setminus E_t$

How efficient is this strategy?

- $\mathbb{P}\left[E(T_t) \cap M(t) \neq \emptyset\right]$
- $|M(t)| \gg n$: Claim edge almost every round

Phase I: Greedy strategy

- Fix arbitrary embedding Φ
- Claim edges corresponding to this embedding, whenever offered
  - $M(t) := \{\Phi(v)\Phi(w) \mid vw \in E(H)\} \setminus E_t$

How efficient is this strategy?

- $\mathbb{P}\left[E(T_t) \cap M(t) \neq \emptyset\right]$
- $|M(t)| \gg n:$ Claim edge almost every round
- $|M(t)| \approx n:$ Claim edge with constant probability

Phase I: Greedy strategy

- Fix arbitrary embedding Φ
- Claim edges corresponding to this embedding, whenever offered
  - $M(t) := \{\Phi(v)\Phi(w) \mid vw \in E(H)\} \setminus E_t$

How efficient is this strategy?

- $\mathbb{P}\left[E(T_t) \cap M(t) \neq \emptyset\right]$
- $|M(t)| \gg n$: Claim edge almost every round
- $|M(t)| \approx n$: Claim edge with constant probability
- $|M(t)| \ll n$: Almost never claim edge

Limitations of the greedy strategy

- Fix edge e

Limitations of the greedy strategy

- Fix edge e
- $\mathbb{P}\left[e \cap E(T_t)\right] = \frac{n-1}{\binom{n}{2}} = \frac{2}{n}$

Limitations of the greedy strategy

- Fix edge e
- $\mathbb{P}\left[e \cap E(T_t)\right] = \frac{n-1}{\binom{n}{2}} = \frac{2}{n}$
- Coupon collector problem

Limitations of the greedy strategy

- Fix edge e
- $\mathbb{P}\left[e \cap E(T_t)\right] = \frac{n-1}{\binom{n}{2}} = \frac{2}{n}$
- Coupon collector problem
  - Need $\Theta(n \log n)$ rounds to claim 'last' n edges

Limitations of the greedy strategy

- Fix edge e
- $\mathbb{P}\left[e \cap E(T_t)\right] = \frac{n-1}{\binom{n}{2}} = \frac{2}{n}$
- Coupon collector problem
  - Need $\Theta(n \log n)$ rounds to claim 'last' n edges

- $F(t, \Phi) :=$ vertices with missing edges (**failed** vertices)

Limitations of the greedy strategy

- Fix edge e
- $\mathbb{P}\left[e \cap E(T_t)\right] = \frac{n-1}{\binom{n}{2}} = \frac{2}{n}$
- Coupon collector problem
  - Need $\Theta(n \log n)$ rounds to claim 'last' n edges

- $F(t, \Phi) :=$ vertices with missing edges (**failed** vertices)

Set $t_0 := \frac{\Delta n}{2}$

Limitations of the greedy strategy

- Fix edge e
- $\mathbb{P}\left[e \cap E(T_t)\right] = \frac{n-1}{\binom{n}{2}} = \frac{2}{n}$
- Coupon collector problem
  - Need $\Theta(n \log n)$ rounds to claim 'last' n edges

- $F(t, \Phi) :=$ vertices with missing edges (**failed** vertices)

Set $t_0 := \frac{\Delta n}{2}$

Lemma

Whp $|F(t_0, \Phi)| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

- Swap v with w

$$\Phi' : \ \Phi'(v) = \Phi(w) \text{ and } \Phi'(w) = \Phi(v)$$

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

- Swap v with w

$$\Phi' : \ \Phi'(v) = \Phi(w) \text{ and } \Phi'(w) = \Phi(v)$$

- Swap only if $v, w \notin F(t, \Phi')$

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

▶ Swap v with w

$$\Phi' : \; \Phi'(v) = \Phi(w) \text{ and } \Phi'(w) = \Phi(v)$$

- Swap only if $v, w \notin F(t, \Phi')$
- $\Gamma(v, w)$: Set of edges required for successful swap

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

► Swap v with w

$$\Phi' : \ \Phi'(v) = \Phi(w) \text{ and } \Phi'(w) = \Phi(v)$$

- Swap only if $v, w \notin F(t, \Phi')$
- $\Gamma(v, w)$: Set of edges required for successful swap
  - ► Swap if $\Gamma(v, w) \in E_t$

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

  - Swap v with w

$$\Phi' : \ \Phi'(v) = \Phi(w) \text{ and } \Phi'(w) = \Phi(v)$$

- Swap only if $v, w \notin F(t, \Phi')$
- $\Gamma(v, w)$: Set of edges required for successful swap
  - Swap if $\Gamma(v, w) \in E_t$

However:

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

▶ Swap v with w

$$\Phi' : \ \Phi'(v) = \Phi(w) \text{ and } \Phi'(w) = \Phi(v)$$

- Swap only if $v, w \notin F(t, \Phi')$
- $\Gamma(v, w)$: Set of edges required for successful swap
  - ▶ Swap if $\Gamma(v, w) \in E_t$

However:

- Coupon collector argument

Phase II: Adapting the embedding Φ

- $v \in F(t, \Phi)$

- Swap v with w

$$\Phi' : \ \Phi'(v) = \Phi(w) \text{ and } \Phi'(w) = \Phi(v)$$

- Swap only if $v, w \notin F(t, \Phi')$
- $\Gamma(v, w)$: Set of edges required for successful swap
  - Swap if $\Gamma(v, w) \in E_t$

However:

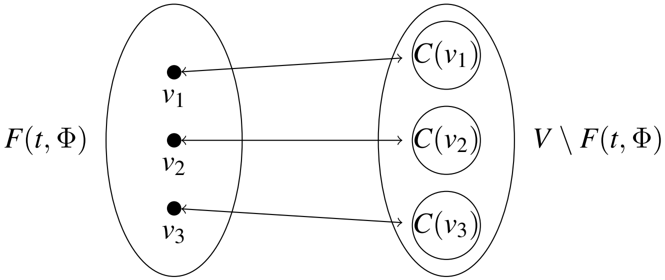
- Coupon collector argument
  - $\mathbb{P}\left[\Gamma(v, w) \subseteq E_t\right] = o(1)$

Phase II: Swapping simultaneously

- For each $v \in F(t, \Phi)$ define large set $C(v)$ of candidate vertices

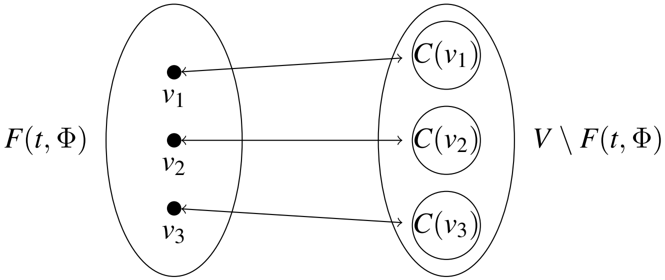
Phase II: Swapping simultaneously

- For each $v \in F(t, \Phi)$ define large set $C(v)$ of candidate vertices



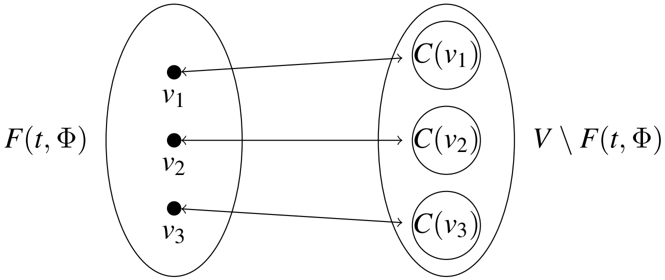
Phase II: Swapping simultaneously

- For each $v \in F(t, \Phi)$ define large set $C(v)$ of candidate vertices
- **Simultaneously** try swapping failed vertices with candidate vertices



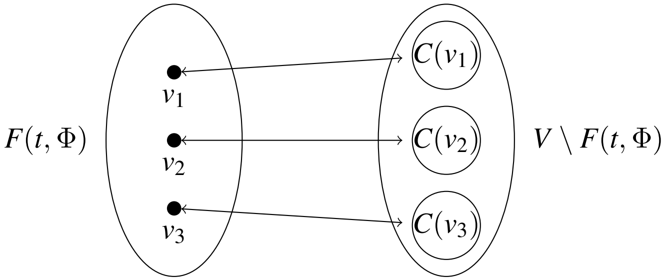
Phase II: Swapping simultaneously

- For each $v \in F(t, \Phi)$ define large set $C(v)$ of candidate vertices
- **Simultaneously** try swapping failed vertices with candidate vertices
- $\mathbb{P}\left[\exists w \in C(v) \colon \Gamma(v, w) \subseteq E_t\right] = \epsilon > 0$



Phase II: Swapping simultaneously

- For each $v \in F(t, \Phi)$ define large set $C(v)$ of candidate vertices
- **Simultaneously** try swapping failed vertices with candidate vertices
- $\mathbb{P}\left[\exists w \in C(v) \colon \Gamma(v, w) \subseteq E_t\right] = \epsilon > 0$



- Update embedding by swapping all possible vertices

Phase II: Swapping in batches

- Proceed in batches of rounds

Phase II: Swapping in batches

- Proceed in batches of rounds
- Swap constant fraction of failed vertices in each batch

Phase II: Swapping in batches

- Proceed in batches of rounds
- Swap constant fraction of failed vertices in each batch
- Number of required rounds is proportional to number of failed vertices

Phase II: Swapping in batches

- Proceed in batches of rounds
- Swap constant fraction of failed vertices in each batch
- Number of required rounds is proportional to number of failed vertices
  - Total number of rounds is dominated by first batch

Phase II: Swapping in batches

- Proceed in batches of rounds
- Swap constant fraction of failed vertices in each batch
- Number of required rounds is proportional to number of failed vertices
  - Total number of rounds is dominated by first batch
  - Run strategy until $F = \emptyset$

Phase II: Swapping in batches

- Proceed in batches of rounds
- Swap constant fraction of failed vertices in each batch
- Number of required rounds is proportional to number of failed vertices
  - Total number of rounds is dominated by first batch
  - Run strategy until $F = \emptyset$

Set $t_1 := \frac{\Delta n}{2} \left(1 + o_\Delta(1)\right)$

Phase II: Swapping in batches

- Proceed in batches of rounds
- Swap constant fraction of failed vertices in each batch
- Number of required rounds is proportional to number of failed vertices
  - Total number of rounds is dominated by first batch
  - Run strategy until $F = \emptyset$

Set $t_1 := \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$

Lemma

There exists an embedding Φ, s.t whp $\quad F(t_1, \Phi) = \emptyset$

Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$

Recap

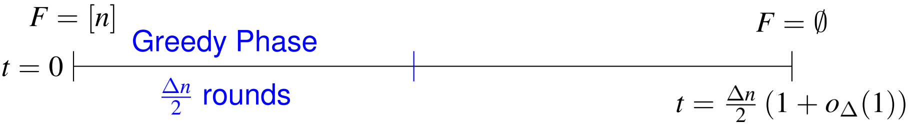
- $\tau(\mathcal{P}_H, \sigma) \le \frac{\Delta n}{2}(1 + o_\Delta(1))$

$F = [n]$ $\qquad\qquad\qquad\qquad\qquad$ $F = \emptyset$

$t = 0$ $\qquad\qquad\qquad\qquad\qquad$ $t = \frac{\Delta n}{2}(1 + o_\Delta(1))$

Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}(1 + o_\Delta(1))$
- Use ‘almost all’ rounds in greedy phase



Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}(1 + o_\Delta(1))$
- Use ‘almost all’ rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$

$F = [n]$

$|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$

$F = \emptyset$

Greedy Phase

$t = 0$

$\frac{\Delta n}{2}$ rounds

$t = \frac{\Delta n}{2}(1 + o_\Delta(1))$

Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2} (1 + o_\Delta(1))$
- Use 'almost all' rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$
- Swap remaining vertices in batches

$F = [n]$

$|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$

$F = \emptyset$

Greedy Phase

Phase II

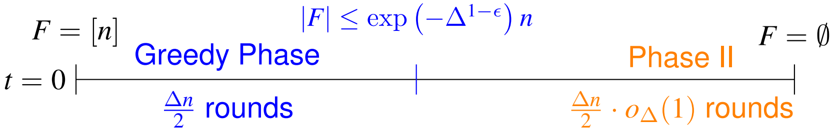
$t = 0$

$\frac{\Delta n}{2}$ rounds

$\frac{\Delta n}{2} \cdot o_\Delta(1)$ rounds

Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$
- Use ‘almost all’ rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$
- Swap remaining vertices in batches
  - Required time t^* dominated by first batch



Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$
- Use ‘almost all’ rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$
- Swap remaining vertices in batches
  - Required time t^* dominated by first batch
  - $t^* \leq \frac{\Delta n}{2} \cdot o_\Delta(1)$

$F = [n]$

$|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$

$F = \emptyset$

Greedy Phase

Phase II

$t = 0$

$\frac{\Delta n}{2}$ rounds

$\frac{\Delta n}{2} \cdot o_\Delta(1)$ rounds

Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}(1 + o_\Delta(1))$
- Use ‘almost all’ rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$
- Swap remaining vertices in batches
  - Required time t^* dominated by first batch
  - $t^* \leq \frac{\Delta n}{2} \cdot o_\Delta(1) \iff |F| \leq \exp\left(-\Delta^{1+\epsilon}\right) n$

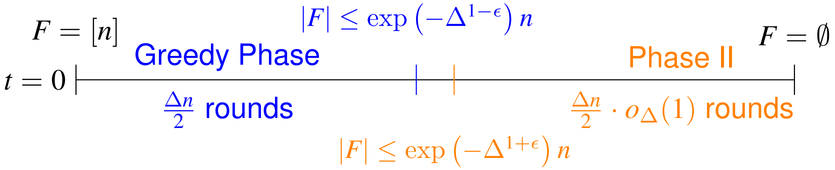
$F = [n]$ $\quad$ $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$ $\quad$ $F = \emptyset$

$t = 0$ $\quad$ Greedy Phase $\quad$ Phase II

$\frac{\Delta n}{2}$ rounds $\quad$ $\frac{\Delta n}{2} \cdot o_\Delta(1)$ rounds

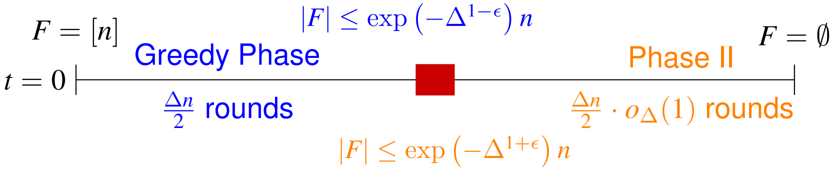
Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$
- Use ‘almost all’ rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$
- Swap remaining vertices in batches
  - Required time t^* dominated by first batch
  - $t^* \leq \frac{\Delta n}{2} \cdot o_\Delta(1) \iff |F| \leq \exp\left(-\Delta^{1+\epsilon}\right) n$



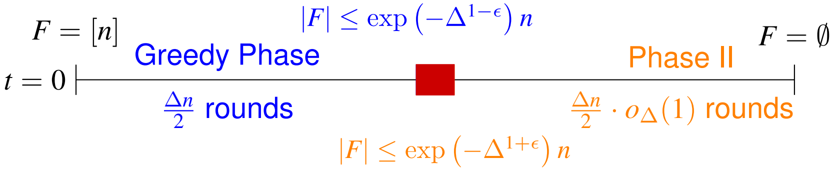
Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}(1 + o_\Delta(1))$
- Use ‘almost all’ rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$
- Swap remaining vertices in batches
  - Required time t^* dominated by first batch
  - $t^* \leq \frac{\Delta n}{2} \cdot o_\Delta(1) \iff |F| \leq \exp\left(-\Delta^{1+\epsilon}\right) n$



Recap

- $\tau(\mathcal{P}_H, \sigma) \leq \frac{\Delta n}{2}\left(1 + o_\Delta(1)\right)$
- Use ‘almost all’ rounds in greedy phase
  - $|F| \leq \exp\left(-\Delta^{1-\epsilon}\right) n$
- Swap remaining vertices in batches
  - Required time t^* dominated by first batch
  - $t^* \leq \frac{\Delta n}{2} \cdot o_\Delta(1) \iff |F| \leq \exp\left(-\Delta^{1+\epsilon}\right) n$
- Gap between two phases - requires more involved analysis



Open problems

- Improve $\tau(\mathcal{P}_H, \sigma)$ in the semirandom **star** process

Open problems

- Improve $\tau(\mathcal{P}_H, \sigma)$ in the semirandom **star** process

- Consider other models:

Open problems

- Improve $\tau(\mathcal{P}_H, \sigma)$ in the semirandom **star** process

- Consider other models:

  Each round, Builder is offered edges of

Open problems

- Improve $\tau(\mathcal{P}_H, \sigma)$ in the semirandom **star** process

- Consider other models:

  Each round, Builder is offered edges of

  - Random hamilton cycle

Open problems

- Improve $\tau(\mathcal{P}_H, \sigma)$ in the semirandom **star** process

- Consider other models:

  Each round, Builder is offered edges of
  - Random hamilton cycle
  - Random perfect matching

Open problems

- Improve $\tau(\mathcal{P}_H, \sigma)$ in the semirandom **star** process

- Consider other models:

  Each round, Builder is offered edges of
  - Random hamilton cycle
  - Random perfect matching
  - k random edges

Open problems

- Improve $\tau(\mathcal{P}_H, \sigma)$ in the semirandom **star** process

- Consider other models:

  Each round, Builder is offered edges of
  - Random hamilton cycle
  - Random perfect matching
  - k random edges

- Are there properties, for which semirandom **star** process is more efficient than semirandom **tree** process?