Randomized Algorithms

Random Graphs

Joseph Chuang-Chieh Lin

Department of Computer Science & Engineering
National Taiwan Ocean University

Spring 2026


Outline

Why study random graphs?

- Many graph problems are NP-hard in the worst case:

Why study random graphs?

- Many graph problems are NP-hard in the worst case: Hamiltonian cycle, independent set, vertex cover, ...

Why study random graphs?

- Many graph problems are NP-hard in the worst case: Hamiltonian cycle, independent set, vertex cover, ...
- A natural average-case question is: are these problems hard on most graphs, or only on pathological (i.e., very particular and special) instances?

Why study random graphs?

- Many graph problems are NP-hard in the worst case: Hamiltonian cycle, independent set, vertex cover, ...
- A natural average-case question is: are these problems hard on most graphs, or only on pathological (i.e., very particular and special) instances?
- Random graph models turn this question into a probability problem.


Why study random graphs?

- Many graph problems are NP-hard in the worst case: Hamiltonian cycle, independent set, vertex cover, ...
- A natural average-case question is: are these problems hard on most graphs, or only on pathological (i.e., very particular and special) instances?
- Random graph models turn this question into a probability problem.
- **Goal of this lecture:** Connect graph algorithms with balls-and-bins and coupon-collector phenomena.


Outline

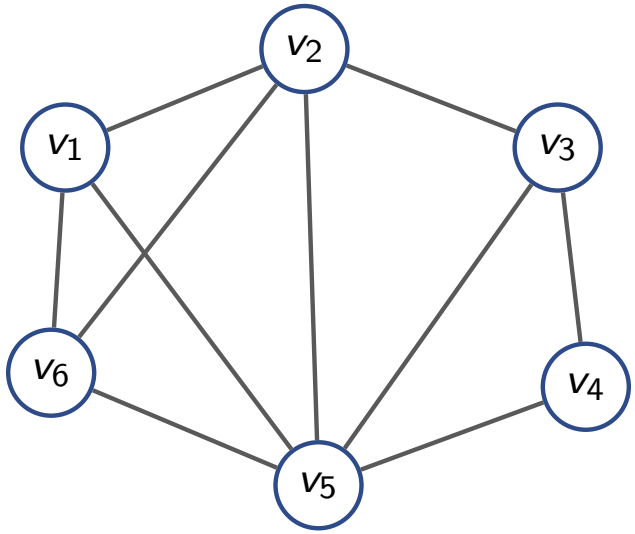
Two core random graph models

$G_{n,p}$

Each of the $\binom{n}{2}$ possible edges is included independently with probability p.

$G_{n,N}$

Exactly N edges are chosen uniformly from the $\binom{n}{2}$ possible edges.



Model $G_{n,p}$: definition and probability mass

Definition

In $G_{n,p}$, for a graph with **exactly m edges**,

$$\Pr(G) = p^m(1-p)^{\binom{n}{2}-m}.$$

Model $G_{n,p}$: definition and probability mass

Definition

In $G_{n,p}$, for a graph with **exactly m edges**,

$$\Pr(G) = p^m (1-p)^{\binom{n}{2}-m}.$$

- Equivalent generation procedure: inspect all $\binom{n}{2}$ possible edges one by one.
- Include each edge independently with probability p.
- Number of edges is a binomial random variable:

Model $G_{n,p}$: definition and probability mass

Definition

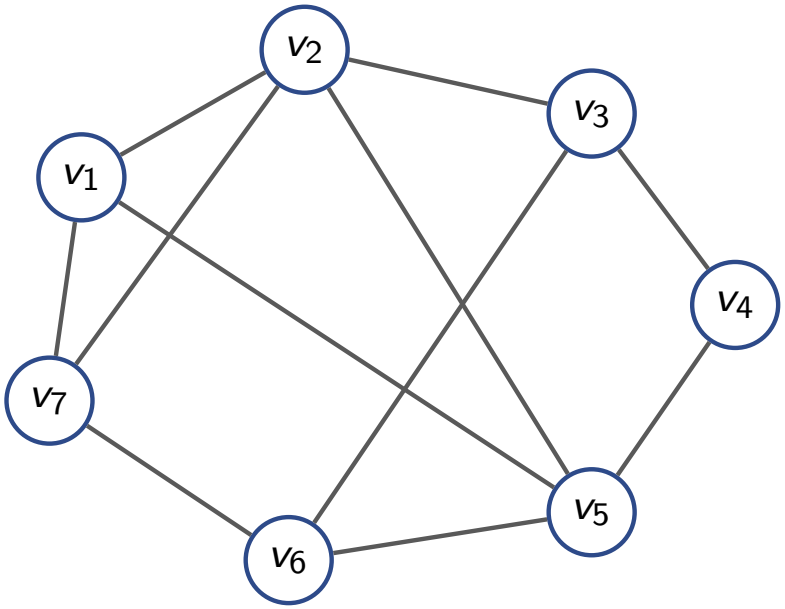
In $G_{n,p}$, for a graph with **exactly m edges**,

$$\Pr(G) = p^m (1-p)^{\binom{n}{2}-m}.$$

- Equivalent generation procedure: inspect all $\binom{n}{2}$ possible edges one by one.
- Include each edge independently with probability p.
- Number of edges is a binomial random variable:

$$X \sim B\left(\binom{n}{2}, p\right).$$

An Example of $G_{n,p}$



One sample from $G_{n,p}$:
each edge decided independently

- Independence is the defining convenience.
- Edge count fluctuates around its mean.
- Very amenable to Chernoff bounds.

Expectations

$$\mathbb{E}[X] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

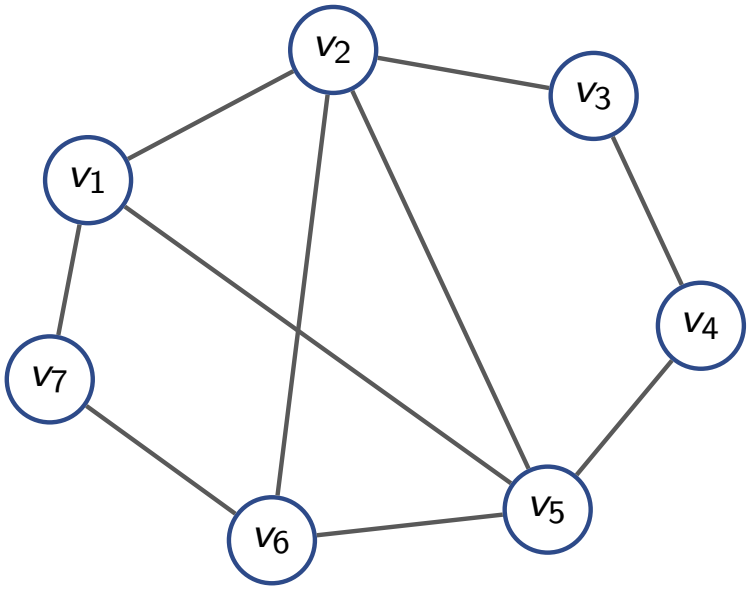
Model $G_{n,N}$: definition and generation

Definition

In $G_{n,N}$, every graph on n labeled vertices with exactly N edges is equally likely.

- There are $\binom{\binom{n}{2}}{N}$ such graphs.
- Generation procedure: start with no edges, then repeatedly choose a previously unused edge uniformly at random until exactly N edges have been added.
- Edge count is deterministic, but the chosen set is random.

An Example of $G_{n,N}$



One sample from $G_{n,N}$:
exactly N edges

- No independence between edge indicators.
- Convenient when the total number of edges is the natural control parameter.
- Very close to $G_{n,p}$ when $p \approx N/\binom{n}{2}$.

$G_{n,p}$ versus $G_{n,N}$

Feature	$G_{n,p}$	$G_{n,N}$
Edge count	random	fixed at N
Edge indicators	independent	dependent
Natural parameter	probability p	exact size N
Best tool	Chernoff on edge count	combinatorial monotonicity[1]
Relation	concentrated around $\binom{n}{2}p$	comparable to $G_{n,p}$ with $p \approx N/\binom{n}{2}$

[1] E.g., for a monotone increasing property, the probability it holds is naturally nondecreasing as N increases.

Expected number of edges and expected degree

In $G_{n,p}$

$$\mathbb{E}[|E|] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

- These formulas come directly from linearity of expectation.
- Clues for locating graph thresholds.

Expected number of edges and expected degree

In $G_{n,p}$

$$\mathbb{E}[|E|] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

- These formulas come directly from linearity of expectation.
- Clues for locating graph thresholds.

Example

If $(n-1)p \ll 1$, isolated vertices should be common; if $(n-1)p \gg \ln n$, connectivity-type properties become plausible.

Expected number of edges and expected degree

In $G_{n,p}$

$$\mathbb{E}[|E|] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

- These formulas come directly from linearity of expectation.
- Clues for locating graph thresholds.

Example

If $(n-1)p \ll 1$, isolated vertices should be common; if $(n-1)p \gg \ln n$, connectivity-type properties become plausible.

- For a vertex in $G_{n,p}$, $\Pr[v \text{ is isolated}] = (1-p)^{n-1}$

Expected number of edges and expected degree

In $G_{n,p}$

$$\mathbb{E}[|E|] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

- These formulas come directly from linearity of expectation.
- Clues for locating graph thresholds.

Example

If $(n-1)p \ll 1$, isolated vertices should be common; if $(n-1)p \gg \ln n$, connectivity-type properties become plausible.

- For a vertex in $G_{n,p}$, $\Pr[v \text{ is isolated}] = (1-p)^{n-1} \approx e^{-(n-1)p}$.

Expected number of edges and expected degree

In $G_{n,p}$

$$\mathbb{E}[|E|] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

- These formulas come directly from linearity of expectation.
- Clues for locating graph thresholds.

Example

If $(n-1)p \ll 1$, isolated vertices should be common; if $(n-1)p \gg \ln n$, connectivity-type properties become plausible.

- For a vertex in $G_{n,p}$, $\Pr[v \text{ is isolated}] = (1-p)^{n-1} \approx e^{-(n-1)p}$.
- The expected number of isolated vertices:

Expected number of edges and expected degree

In $G_{n,p}$

$$\mathbb{E}[|E|] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

- These formulas come directly from linearity of expectation.
- Clues for locating graph thresholds.

Example

If $(n-1)p \ll 1$, isolated vertices should be common; if $(n-1)p \gg \ln n$, connectivity-type properties become plausible.

- For a vertex in $G_{n,p}$, $\Pr[v \text{ is isolated}] = (1-p)^{n-1} \approx e^{-(n-1)p}$.
- The expected number of isolated vertices: $\approx ne^{-(n-1)p}$

Expected number of edges and expected degree

In $G_{n,p}$

$$\mathbb{E}[|E|] = \binom{n}{2} p, \quad \mathbb{E}[\deg(v)] = (n-1)p.$$

- These formulas come directly from linearity of expectation.
- Clues for locating graph thresholds.

Example

If $(n-1)p \ll 1$, isolated vertices should be common; if $(n-1)p \gg \ln n$, connectivity-type properties become plausible.

- For a vertex in $G_{n,p}$, $\Pr[v \text{ is isolated}] = (1-p)^{n-1} \approx e^{-(n-1)p}$.
- The expected number of isolated vertices: $\approx ne^{-(n-1)p}$
- ...

Graph properties and monotonicity

Graph property

A graph property depends only on the isomorphism class of the graph, not on the labels of the vertices.

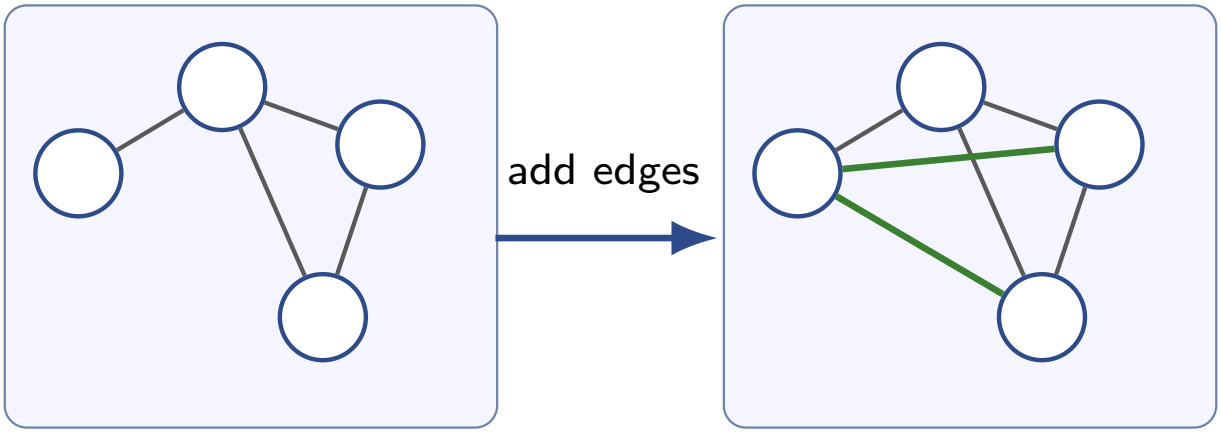
Monotone increasing

If $G = (V, E)$ has the property, then every $G' = (V, E')$ with $E \subseteq E'$ also has it.

Monotone decreasing

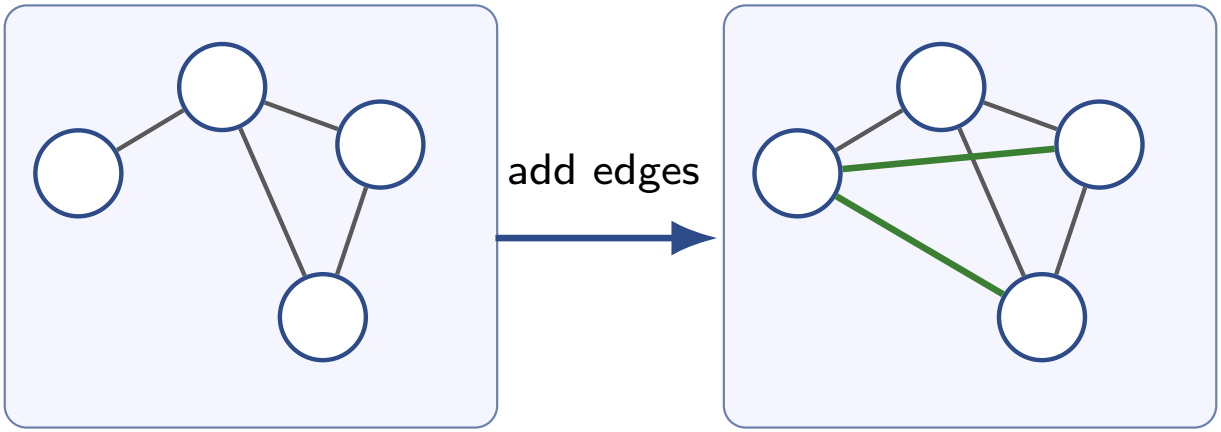
If $G = (V, E)$ has the property, then every $G' = (V, E')$ with $E' \subseteq E$ also has it.

Monotonicity picture



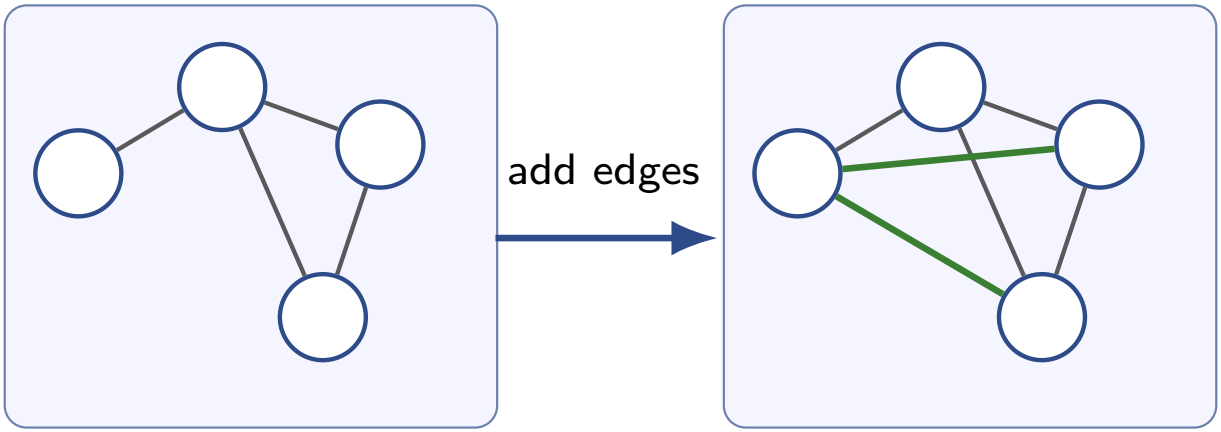
- Connectedness

Monotonicity picture



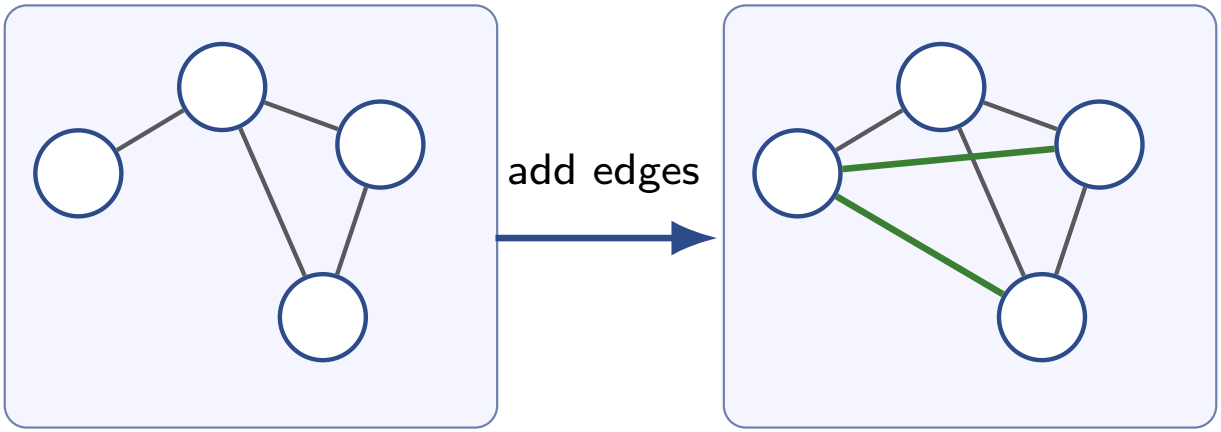
- Connectedness is monotone increasing.

Monotonicity picture



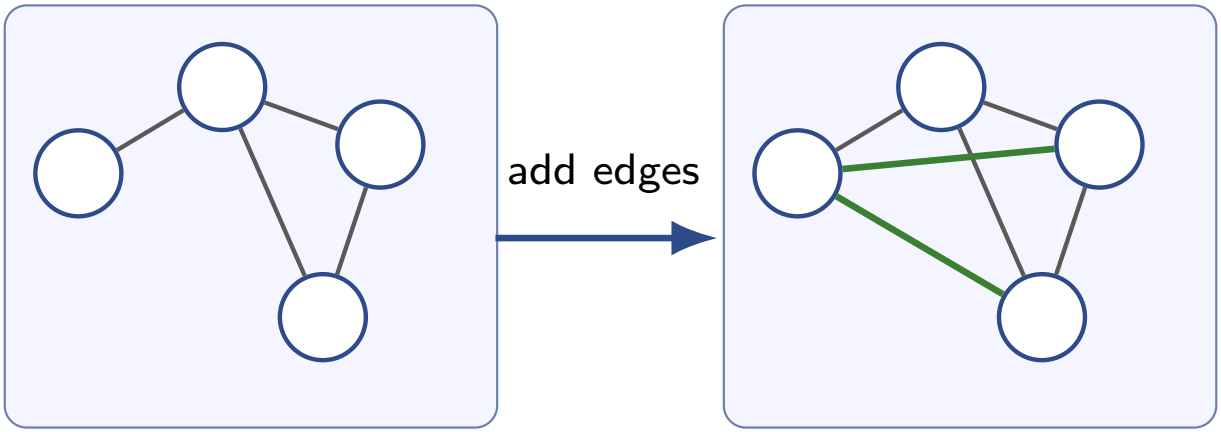
- Connectedness is monotone increasing.
- Acyclicity

Monotonicity picture



- Connectedness is monotone increasing.
- Acyclicity is monotone decreasing.
- "Being a tree"

Monotonicity picture



- Connectedness is monotone increasing.
- Acyclicity is monotone decreasing.
- "Being a tree" is not monotone.

Examples of monotone and non-monotone properties

Property	Type	Reason
Connected	increasing	adding edges cannot disconnect
Contains a k-vertex component	increasing	extra edges preserve that component
Contains no cycle	decreasing	removing edges cannot create a cycle
Is a tree	not monotone	adding one edge creates a cycle
Has a Hamiltonian cycle	increasing	once a cycle exists, extra edges do not destroy it

Why are $G_{n,p}$ and $G_{n,N}$ closely related?

- If $p = N/\binom{n}{2}$, then in $G_{n,p}$ the expected edge count is exactly N.
- The actual edge count in $G_{n,p}$ is sharply concentrated around this mean.
- Conditioned on having exactly N edges, a graph from $G_{n,p}$ is uniformly distributed over $G_{n,N}$.

Heuristic

Analyze in the model that is easier for the proof; transfer the result to the other model using monotonicity plus concentration.

Comparison for monotone properties

Lemma 1

For a monotone increasing graph property, let $P(n, N)$ be its probability in $G_{n,N}$ and $P(n, p)$ its probability in $G_{n,p}$. Define

$$p_+ = (1+\varepsilon)\frac{N}{\binom{n}{2}}, \qquad p_- = (1-\varepsilon)\frac{N}{\binom{n}{2}},$$

where $0 < \varepsilon < 1$ is constant. Then

$$P(n, p_-) - e^{-O(N)} \leq P(n, N) \leq P(n, p_+) + e^{-O(N)}.$$

Proof idea of Lemma 1: lower sampling level p_-

Let X be the number of edges in a graph drawn from G_{n,p_-}.

- Conditioned on $X = k$, the graph is uniform over $G_{n,k}$.
- Therefore

$$\begin{aligned} P(n, p_-) &= \sum_{k=0}^{\binom{n}{2}} P(n,k) \Pr[X = k] \\ &= \sum_{k \le N} P(n,k) \Pr[X = k] + \sum_{k > N} P(n,k) \Pr[X = k]. \end{aligned}$$

- For a monotone increasing property, if $k \le N$, then $P(n,k) \le P(n,N)$.

Hence

$$P(n, p_-) \le P(n,N) \Pr[X \le N] + \Pr[X > N]$$

Proof idea of Lemma 1: lower sampling level p_-

Let X be the number of edges in a graph drawn from G_{n,p_-}.

- Conditioned on $X = k$, the graph is uniform over $G_{n,k}$.
- Therefore

$$\begin{aligned} P(n, p_-) &= \sum_{k=0}^{\binom{n}{2}} P(n,k) \Pr[X = k] \\ &= \sum_{k \le N} P(n,k) \Pr[X = k] + \sum_{k > N} P(n,k) \Pr[X = k]. \end{aligned}$$

- For a monotone increasing property, if $k \le N$, then $P(n,k) \le P(n,N)$.

Hence

$$P(n,p_-) \le P(n,N) \Pr[X \le N] + \Pr[X > N] \le P(n,N) + \Pr[X > N].$$

Proof idea of Lemma 1: upper sampling level p_+

Similarly, if X is the number of edges in a graph from G_{n,p_+}, then

$$P(n,p_+) = \sum_{k<N} P(n,k)\Pr[X=k] + \sum_{k\geq N} P(n,k)\Pr[X=k].$$

So due to the monotone property,

$$P(n,p_+) \geq \Pr[X \geq N] P(n,N)$$

Proof idea of Lemma 1: upper sampling level p_+

Similarly, if X is the number of edges in a graph from G_{n,p_+}, then

$$P(n,p_+) = \sum_{k<N} P(n,k)\Pr[X=k] + \sum_{k\geq N} P(n,k)\Pr[X=k].$$

So due to the monotone property,

$$P(n,p_+) \geq \Pr[X \geq N]P(n,N) \geq P(n,N) - \Pr[X < N].$$

Proof idea of Lemma 1: upper sampling level p_+

Similarly, if X is the number of edges in a graph from G_{n,p_+}, then

$$P(n,p_+) = \sum_{k<N} P(n,k)\Pr[X=k] + \sum_{k\geq N} P(n,k)\Pr[X=k].$$

So due to the monotone property,

$$P(n,p_+) \geq \Pr[X \geq N]P(n,N) \geq P(n,N) - \Pr[X < N].$$

So the only remaining task is to show that the edge-count tails $\Pr[X > N]$ and $\Pr[X < N]$ are exponentially small in N.

Why Chernoff bounds appear here

Key observation

In $G_{n,p}$, the random variable $X = |E|$ is a sum of $\binom{n}{2}$ independent Bernoulli variables.

Therefore standard Chernoff inequalities imply

$$\Pr[X > (1+\delta)\,\mathbb{E}[X]] \le e^{-\delta^2\,\mathbb{E}[X]/3}, \quad \Pr[X < (1-\delta)\,\mathbb{E}[X]] \le e^{-\delta^2\,\mathbb{E}[X]/2}.$$

Why Chernoff bounds appear here

Key observation

In $G_{n,p}$, the random variable $X = |E|$ is a sum of $\binom{n}{2}$ independent Bernoulli variables.

Therefore standard Chernoff inequalities imply

$$\Pr[X > (1+\delta)\,\mathbb{E}[X]] \le e^{-\delta^2\,\mathbb{E}[X]/3}, \quad \Pr[X < (1-\delta)\,\mathbb{E}[X]] \le e^{-\delta^2\,\mathbb{E}[X]/2}.$$

- This is exactly the concentration bridge between $G_{n,p}$ and $G_{n,N}$.
- Monotonicity plus concentration is the recurring pattern.

The bounds for $\Pr[X > N]$ and $\Pr[X < N]$

$$\begin{aligned}\Pr[X > N] &= \Pr\left[X > \frac{1}{1-\epsilon}\mathbb{E}[X]\right] \\ &\leq \Pr[X > (1+\epsilon)\mathbb{E}[X]] \leq e^{-(1-\epsilon)\epsilon^2 N/3}.\end{aligned}$$

- Using $\frac{1}{1-\varepsilon} > 1+\varepsilon$ for $\varepsilon \in (0,1)$.

The bounds for $\Pr[X > N]$ and $\Pr[X < N]$

$$\begin{aligned}\Pr[X > N] &= \Pr\left[X > \frac{1}{1-\epsilon}\mathbb{E}[X]\right] \\ &\leq \Pr[X > (1+\epsilon)\mathbb{E}[X]] \leq e^{-(1-\epsilon)\epsilon^2 N/3}.\end{aligned}$$

- Using $\frac{1}{1-\varepsilon} > 1+\varepsilon$ for $\varepsilon \in (0,1)$.

$$\begin{aligned}\Pr[X < N] &= \Pr\left[X < \frac{1}{1+\epsilon}\mathbb{E}[X]\right] \\ &\leq \Pr\left[X < \left(1-\frac{\epsilon}{2}\right)\mathbb{E}[X]\right] \leq e^{-(1+\epsilon)\epsilon^2 N/8}.\end{aligned}$$

- Using $\frac{1}{1+\varepsilon} < 1-\frac{\varepsilon}{2}$ for $\varepsilon \in (0,1)$.

Isolated-vertex threshold in $G_{n,N}$

Theorem (Exercise)

Let

$$N = \frac{1}{2}(n \ln n + cn).$$

Then the probability that $G_{n,N}$ has no isolated vertices converges to

$$e^{-e^{-c}}$$

as $n \to \infty$.

- This mirrors the coupon-collector threshold for empty bins.
- Isolated vertices are the graph analog of bins that were never hit.

Balls-and-bins analogy

Balls and bins

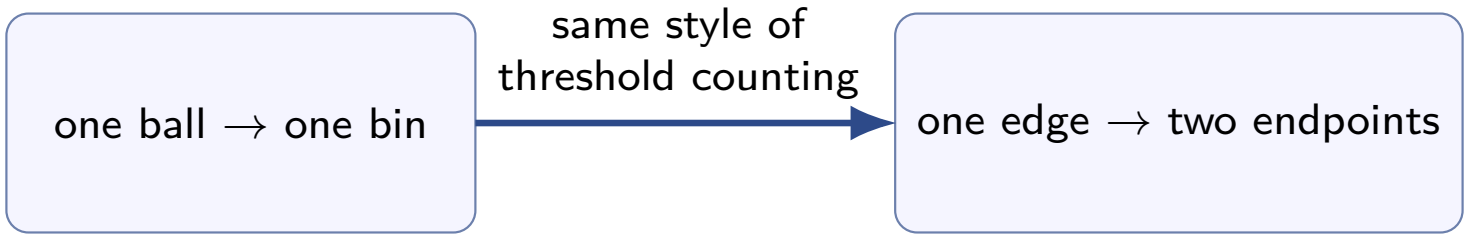
Throw one ball per trial into one bin.

- empty bin $\leftrightarrow$ coupon not seen

Random graphs

Throw one edge per trial into the graph.

- isolated vertex $\leftrightarrow$ endpoint never hit



Outline

Input presentation

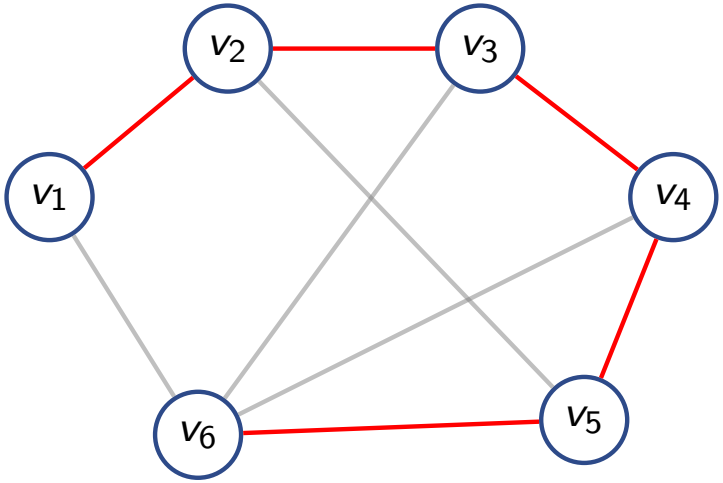
- We assume that our input is presented as a list of adjacent edges for each vertex in the graph.
- The edges of each list are given in a random order according to independent and uniform random permutations.

Hamiltonian paths and Hamiltonian cycles

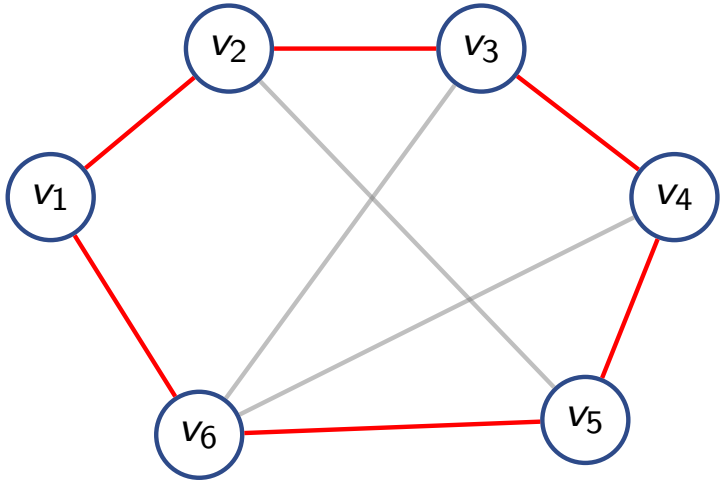
Definitions

A Hamiltonian path visits every vertex exactly once.

A Hamiltonian cycle visits every vertex exactly once and returns to the starting point.



Hamiltonian path

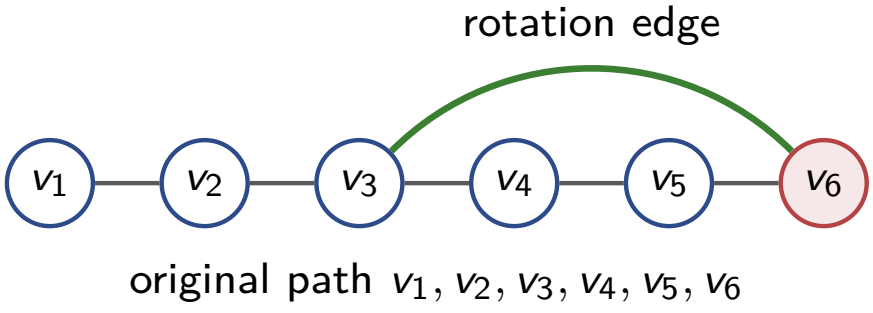


Hamiltonian cycle

Why is this application interesting?

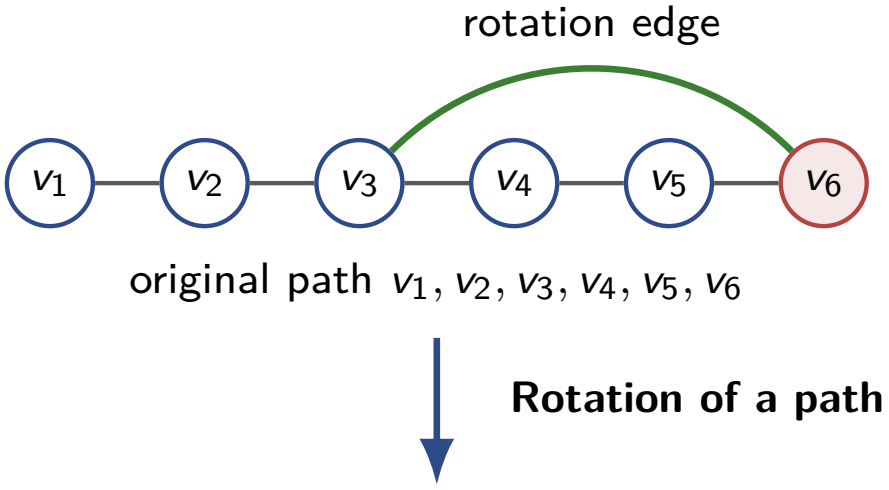
- In general graphs, computing a Hamiltonian cycle is NP-hard.
- Such worst-case complexity alone does not predict random-instance behavior.
- We show that it is not hard for suitably randomly selected graphs.
  - A simple randomized algorithm succeeds quickly with high probability.


Rotation: the key local operation



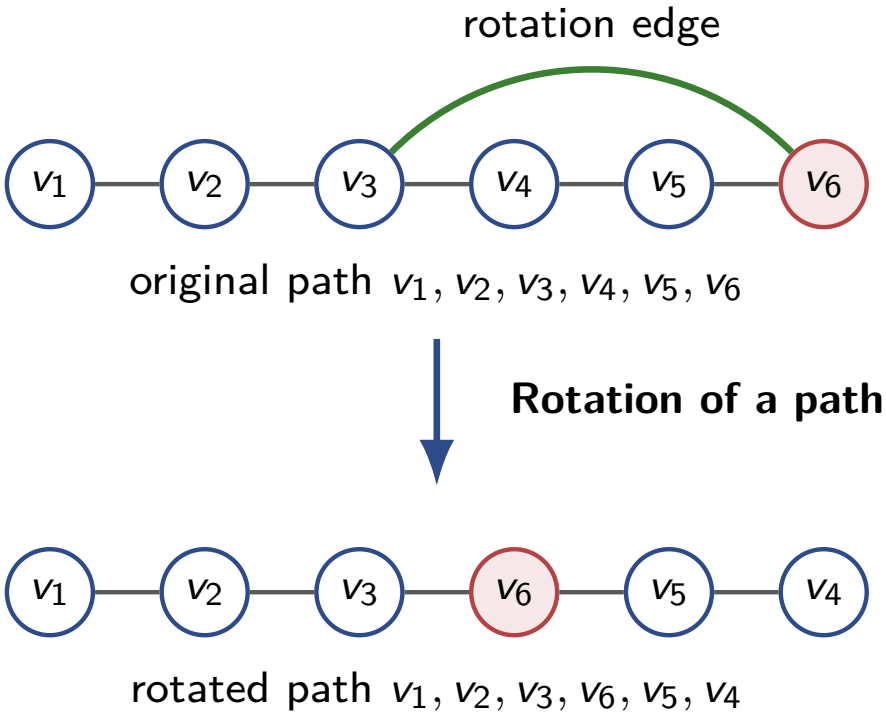
- A simple path $P = v_1, v_2, \ldots, v_k$ and an (v_k, v_i) exists in G, then we can rotate the suffix of P.
- $P' = v_1, v_2, \ldots, v_i, v_k, v_{k-1}, \ldots, v_{i+2}, v_{i+1}$.

Rotation: the key local operation



- A simple path $P = v_1, v_2, \ldots, v_k$ and an (v_k, v_i) exists in G, then we can rotate the suffix of P.
- $P' = v_1, v_2, \ldots, v_i, v_k, v_{k-1}, \ldots, v_{i+2}, v_{i+1}$.

Rotation: the key local operation



- A simple path $P = v_1, v_2, \ldots, v_k$ and an (v_k, v_i) exists in G, then we can rotate the suffix of P.
- $P' = v_1, v_2, \ldots, v_i, v_k, v_{k-1}, \ldots, v_{i+2}, v_{i+1}$.

The original natural algorithm (Algorithm 1)

1. Start from an arbitrary vertex and treat it as the head of the path.
2. Repeat the following steps:
   - ▷ Look at the first remaining edge adjacent to the head.
     - If it leads to a new vertex, extend the path.
     - If it leads to an internal path vertex, rotate.
3. Stop when the path closes to a Hamiltonian cycle or the head has no unused adjacent edge left.

Why is the original algorithm difficult to analyze?

Conditioning problem

Once the algorithm has already examined some edges in adjacency lists, the distribution of the remaining unseen edges is no longer the original unconditional one.

Why is the original algorithm difficult to analyze?

Conditioning problem

Once the algorithm has already examined some edges in adjacency lists, the distribution of the remaining unseen edges is no longer the original unconditional one.

- The algorithm reveals information adaptively.
- The future edge order depends on the past observations.
- That makes a direct probabilistic analysis messy.

Why is the original algorithm difficult to analyze?

Conditioning problem

Once the algorithm has already examined some edges in adjacency lists, the distribution of the remaining unseen edges is no longer the original unconditional one.

- The algorithm reveals information adaptively.
- The future edge order depends on the past observations.
- That makes a direct probabilistic analysis messy.

Strategy

Modify the algorithm so that its random choices remain analyzable by deferred decisions.


Modified algorithm: data structure

Each vertex v keeps two lists:

`used-edges`(v)

Edges adjacent to v that were used while v was the head.

`unused-edges`(v)

Adjacent edges not yet used from v's perspective.

- Initially, all `used-lists` are empty.
- The `unused-lists` are randomly generated in a controlled way.
- The algorithm may reverse, rotate using a used edge, or expose the next unused edge.

Modified algorithm: step 2(b) probabilities

Suppose the current path is $P = v_1, v_2, \ldots, v_k$ with head v_k.

At each iteration, do exactly one of:

1. reverse the path and make v_1 the head, with probability $1/n$;
2. pick a uniformly random edge from $\texttt{used-edges}(v_k)$ and rotate with it, with probability $|\text{used-edges}(v_k)|/n$;
3. take the first edge from $\texttt{unused-edges}(v_k)$, with the remaining probability.

- This is the technical trick that makes the next head exactly uniform.

Analytic input model for the unused lists

Assume that for each vertex v and each other vertex $u \neq v$,

- the edge (v, u) is initially placed on the `unused-edges` list of v independently with probability q;
  - The size of the unused-edge list: $\sim$ Binormial$(n-1, q)$.
- the resulting list is randomly ordered.

Asymmetry is allowed

An edge (u, v) may appear on u's list, on v's list, on both, or on neither.

This is not yet the usual $G_{n,p}$ model, but it is easier to analyze. Later we map back to $G_{n,p}$.

Why this input model is useful

- The `unused-edges` list of the current head was generated independently of the histories of the other vertices.
- Therefore the unseen part of that list still has a simple conditional distribution.
- This restores the principle of deferred decisions.

Big payoff

The next head can be treated as *uniformly* random over all n vertices.


Lemma 2

Let V_t be the head after step t of the modified algorithm. As long as the current head still has at least one unused edge,

$$\Pr[V_{t+1} = u \mid V_t = u_t, V_{t-1} = u_{t-1}, \ldots, V_0 = u_0] = \frac{1}{n}$$

for every vertex u.

Lemma 2

Let V_t be the head after step t of the modified algorithm. As long as the current head still has at least one unused edge,

$$\Pr[V_{t+1} = u \mid V_t = u_t, V_{t-1} = u_{t-1}, \ldots, V_0 = u_0] = \frac{1}{n}$$

for every vertex u.

Regardless of the entire past history, the next head is uniform on all vertices.

Outline

Proof of Lemma 2: Case 1 (reverse)

Suppose the current path is $P = v_1, v_2, \ldots, v_k$ with head v_k.

- The only way v_1 becomes the next head is by reversing the path.
- That event occurs with probability exactly $1/n$.

$$\Pr[V_{t+1} = v_1] = \frac{1}{n}.$$

One vertex done

The left endpoint already has the correct target probability.

Proof of Lemma 2: Case 2 (used-edge rotation)

If $(v_k, v_i) \in$ used-edges(v_k), then rotating with that edge makes v_{i+1} the new head.

- Step 2(b)-(2) is chosen with probability $|\text{used-edges}(v_k)|/n$.
- Conditional on that case, the used edge is chosen uniformly from the list.

Hence

$$\Pr[V_{t+1} = v_{i+1}] = \frac{|\text{used-edges}(v_k)|}{n} \cdot \frac{1}{|\text{used-edges}(v_k)|} = \frac{1}{n}.$$

Proof of Lemma 2: Case 3 (unused-edge step)

Now consider vertices reached via the next unseen edge in $\texttt{unused-edges}(v_k)$.

- The algorithm chooses this case with probability[2]

$$1 - \frac{1}{n} - \frac{|\text{used-edges}(v_k)|}{n}.$$

- The unseen edge is uniformly distributed among the $n - |\text{used-edges}(v_k)| - 1$ eligible neighbors.

[2] It's a bit like reverse-engineering.

Proof of Lemma 2: Case 3 (unused-edge step)

Now consider vertices reached via the next unseen edge in $\texttt{unused-edges}(v_k)$.

- The algorithm chooses this case with probability[2]

$$1 - \frac{1}{n} - \frac{|\text{used-edges}(v_k)|}{n}.$$

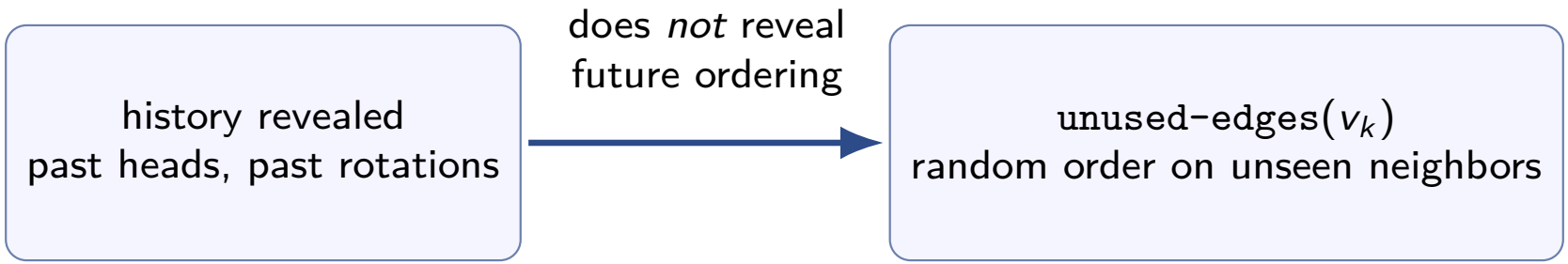
- The unseen edge is uniformly distributed among the $n - |\text{used-edges}(v_k)| - 1$ eligible neighbors.

Therefore each such target vertex u has probability

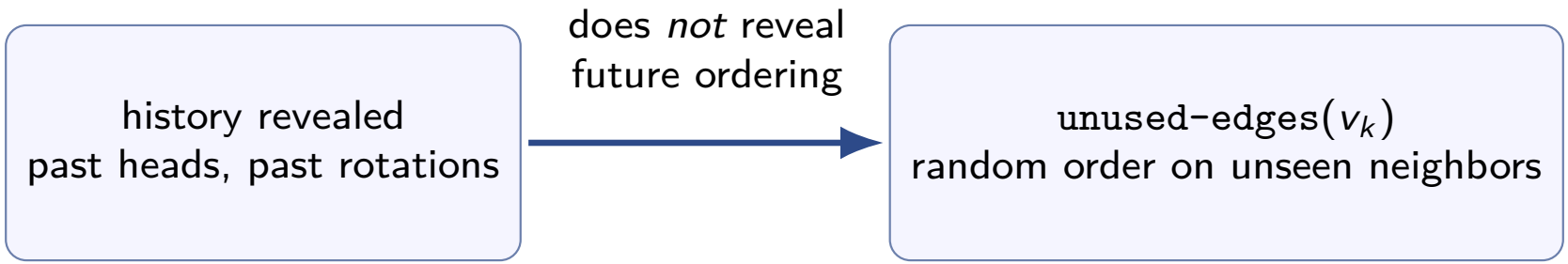
$$\left(1 - \frac{1}{n} - \frac{|\text{used-edges}(v_k)|}{n}\right) \frac{1}{n - |\text{used-edges}(v_k)| - 1} = \frac{1}{n}.$$

[2] It's a bit like reverse-engineering.

Deferred decisions: why the unused-edge step is uniform



Deferred decisions: why the unused-edge step is uniform



Principle of deferred decisions: conditioned on the past, the next unseen edge from v_k is still uniform among the remaining eligible neighbors.

- The past only reveals previously exposed edges.
- It does not bias the random ordering among unseen candidates of the current head.

Conclusion of Lemma 2

Combining the three cases:

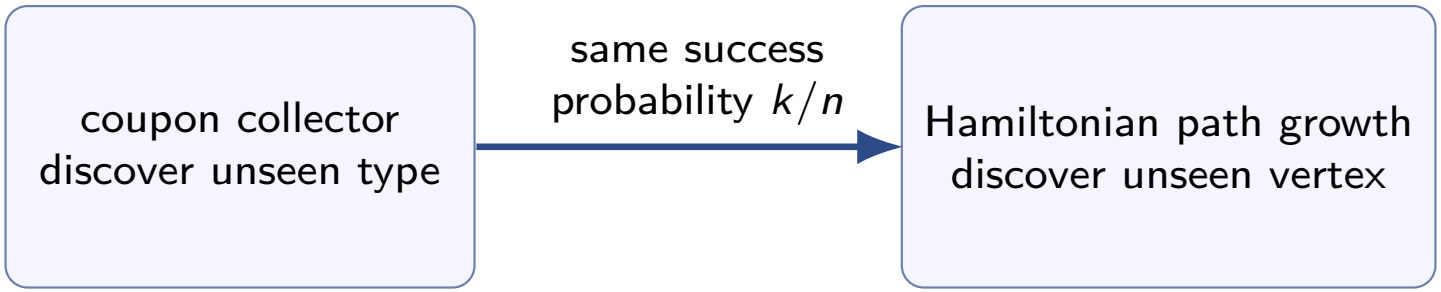
- v_1 gets probability $1/n$ from reversal;
- each reachable internal successor gets probability $1/n$;
- each outside vertex also gets probability $1/n$.

Key simplification

The head process behaves like repeated uniform sampling from the vertex set — as long as no unused list is exhausted.

This is what converts the analysis into coupon collector.


Coupon collector analogy



- When k vertices remain outside the path, a uniformly random next head extends the path with probability k/n.
- That is exactly the same success probability as collecting a new coupon type when k types are still missing.

From Hamiltonian path to Hamiltonian **cycle**

Once the path already contains all n vertices:

- a successful closing rotation occurs with probability $1/n$ at each step;
- therefore another coupon-collector-scale waiting time, $O(n \ln n)$, is enough to close the cycle with high probability.

Heuristic runtime

$$O(n \ln n) \text{ to build the path } + O(n \ln n) \text{ to close it } = O(n \ln n).$$

Theorem 2

Theorem 2

Suppose each edge (v, u) with $u \neq v$ is initially placed on v's unused list independently with probability

$$q \geq \frac{20 \ln n}{n}.$$

Then the modified algorithm finds a Hamiltonian cycle in $O(n \ln n)$ loop iterations with probability

$$1 - O(n^{-1}).$$

Proof plan for Theorem 2

Define two bad events:

E_1 : after $3n \ln n$ steps, no list emptied but no Hamiltonian cycle was found,

E_2 : some unused-edges list emptied during the first $3n \ln n$ steps.

Then

$$\Pr[\text{failure}] \leq \Pr[E_1] + \Pr[E_2].$$

So we just need to bound both terms by $O(1/n)$.

Bounding E_1: path construction phase

By Lemma 2, while no list is empty, the head is uniform on the n vertices at each step.

- Therefore the time to discover all vertices is exactly a coupon-collector process.
- The probability that more than $2n \ln n$ steps are needed to cover all n vertices equals the probability that some coupon type is missing after $2n \ln n$ samples.

For any fixed type,

$$\left(1 - \frac{1}{n}\right)^{2n \ln n} \leq e^{-2 \ln n} = \frac{1}{n^2}.$$

Union bound over n types gives probability at most $1/n$.

Bounding E_1: cycle-closing phase

After all vertices are already on the path:

- each step closes the path into a cycle with probability $1/n$;
- so the probability of failing to close within the next $n \ln n$ steps is

$$\left(1 - \frac{1}{n}\right)^{n \ln n} \leq e^{-\ln n} = \frac{1}{n}.$$

Hence

$$\Pr[E_1] \leq \frac{1}{n} + \frac{1}{n} = \frac{2}{n}.$$

Bounding E_2: split the event again

Recall: the size of an unused-list $\sim$ Binomial$(n-1, q)$.

Let

- E_{2a}: some vertex loses $\geq 9 \ln n$ edges from its unused list in the first $3n \ln n$ steps,
- E_{2b}: some vertex started with $< 10 \ln n$ unused edges.

Then

$$\Pr[E_2] \leq \Pr[E_{2a}] + \Pr[E_{2b}].$$

Bounding E_2: split the event again

Recall: the size of an unused-list $\sim$ Binomial$(n-1, q)$.

Let

- E_{2a}: some vertex loses $\geq 9 \ln n$ edges from its unused list in the first $3n \ln n$ steps,
- E_{2b}: some vertex started with $< 10 \ln n$ unused edges.

Then

$$\Pr[E_2] \leq \Pr[E_{2a}] + \Pr[E_{2b}].$$

Intuition

A list can become empty only if either it was too short initially or it was queried unusually often.


Bound on E_{2a}

Fix a vertex v. Let X be the number of times v is the head during the first $3n \ln n$ steps.

$$X \sim B(3n \ln n, 1/n), \qquad \mu = \mathbb{E}[X] = 3 \ln n.$$

Bound on E_{2a}

Fix a vertex v. Let X be the number of times v is the head during the first $3n \ln n$ steps.

$$X \sim B(3n \ln n, 1/n), \qquad \mu = \mathbb{E}[X] = 3 \ln n.$$

Using the Chernoff upper-tail bound with $\delta = 2$:

$$\Pr[X \geq 9 \ln n] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu}$$

Bound on E_{2a}

Fix a vertex v. Let X be the number of times v is the head during the first $3n \ln n$ steps.

$$X \sim B(3n \ln n, 1/n), \qquad \mu = \mathbb{E}[X] = 3 \ln n.$$

Using the Chernoff upper-tail bound with $\delta = 2$:

$$\Pr[X \geq 9 \ln n] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu} = \left(\frac{e^2}{27} \right)^{3 \ln n} \leq \frac{1}{n^2}.$$


Bound on E_{2a}

Fix a vertex v. Let X be the number of times v is the head during the first $3n \ln n$ steps.

$$X \sim B(3n \ln n, 1/n), \qquad \mu = \mathbb{E}[X] = 3 \ln n.$$

Using the Chernoff upper-tail bound with $\delta = 2$:

$$\Pr[X \geq 9 \ln n] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu} = \left(\frac{e^2}{27} \right)^{3 \ln n} \leq \frac{1}{n^2}.$$

Applying the union bound over all n vertices yields

$$\Pr[E_{2a}] \leq \frac{1}{n}.$$


Bound on E_{2b}

Let Y be the initial size of a vertex's unused list.

$$Y \sim B(n-1, q), \quad \mathbb{E}[Y] = (n-1)q \geq 19 \ln n$$

for sufficiently large n, since $q \geq 20 \ln n / n$.

Bound on E_{2b}

Let Y be the initial size of a vertex's unused list.

$$Y \sim B(n-1, q), \quad \mathbb{E}[Y] = (n-1)q \geq 19 \ln n$$

for sufficiently large n, since $q \geq 20 \ln n / n$.

Using the Chernoff lower-tail bound,

$$\Pr[Y \leq 10 \ln n] \leq e^{-(19 \ln n)(9/19)^2/2} \leq \frac{1}{n^2}.$$

Bound on E_{2b}

Let Y be the initial size of a vertex's unused list.

$$Y \sim B(n-1, q), \quad \mathbb{E}[Y] = (n-1)q \geq 19 \ln n$$

for sufficiently large n, since $q \geq 20 \ln n / n$.

Using the Chernoff lower-tail bound,

$$\Pr[Y \leq 10 \ln n] \leq e^{-(19 \ln n)(9/19)^2/2} \leq \frac{1}{n^2}.$$

Again union bound over all n vertices gives

$$\Pr[E_{2b}] \leq \frac{1}{n}.$$

Finish of Theorem 2

Combine the bounds:

$$\Pr[E_1] \leq \frac{2}{n}, \qquad \Pr[E_2] \leq \frac{2}{n}.$$

Therefore,

$$\Pr[\text{algorithm fails in } 3n \ln n \text{ iterations}] \leq \frac{4}{n}.$$

Conclusion

The modified algorithm finds a Hamiltonian cycle in $O(n \ln n)$ time with probability $1 - O(1/n)$.


Related back to $G_{n,p}$

Corollary 1

By initializing unused-edge lists appropriately, the modified algorithm finds a Hamiltonian cycle on a graph drawn from $G_{n,p}$ with probability $1 - O(1/n)$ whenever

$$p \geq \frac{40 \ln n}{n}.$$

How to transfer from $G_{n,p}$ to the list model

Choose $q \in [0, 1]$ such that

$$p = 2q - q^2$$

How to transfer from $G_{n,p}$ to the list model

Choose $q \in [0, 1]$ such that

$$p = 2q - q^2 \quad (= 1 - (1 - q)^2)$$

How to transfer from $G_{n,p}$ to the list model

Choose $q \in [0,1]$ such that

$$p = 2q - q^2 \quad (= 1 - (1-q)^2)$$

For every input edge $(u,v) \in G_{n,p}$, partition it independently as follows:

For each input edge $(u,v) \in G_{n,p}$:

- put on u's list only with prob. $\dfrac{q(1-q)}{2q-q^2}$
- put on v's list only with prob. $\dfrac{q(1-q)}{2q-q^2}$
- put on both lists with prob. $\dfrac{q^2}{2q-q^2}$

Why this partition works

For any oriented placement onto v's list,

$$p\left(\frac{q(1-q)}{2q-q^2}+\frac{q^2}{2q-q^2}\right)=q.$$

So each potential edge (u,v) lands on v's unused list with probability exactly q.

Why this partition works

For any oriented placement onto v's list,

$$p\left(\frac{q(1-q)}{2q-q^2}+\frac{q^2}{2q-q^2}\right)=q.$$

So each potential edge (u,v) lands on v's unused list with probability exactly q.

- The two endpoint placements are independent:
  $\because$ the prob. of landing on both lists is:

$$p\cdot\frac{q^2}{2q-q^2}=q^2.$$

- Different edges are treated independently.
- Thus, the list model assumptions of Theorem 2 are satisfied.

Why the threshold becomes $p \geq 40 \ln n/n$

From

$$p = 2q - q^2 \leq 2q,$$

we get

$$q \geq \frac{p}{2}.$$

Why the threshold becomes $p \geq 40 \ln n/n$

From

$$p = 2q - q^2 \leq 2q,$$

we get

$$q \geq \frac{p}{2}.$$

Therefore if

$$p \geq \frac{40 \ln n}{n},$$

then automatically

$$q \geq \frac{20 \ln n}{n},$$

$\Rightarrow$ precisely the hypothesis needed in Theorem 2.

Outline

Core takeaways

Random graph models

$G_{n,p}$ gives independence; $G_{n,N}$ gives a fixed edge count. Monotonicity plus Chernoff concentration ties them together.

Structural threshold (Exercise)

Around $N = \frac{1}{2}(n \ln n + cn)$, isolated vertices disappear with probability approaching $e^{-e^{-c}}$.

Algorithmic result

A modified rotation-based algorithm finds Hamiltonian cycles in $O(n \ln n)$ steps with high probability in sufficiently dense random graphs.


Discussion questions

1. Why is "being a tree" not monotone, while "being connected" is?
2. In Lemma 1, why do we need monotonicity to compare $P(n, k)$ with $P(n, N)$?
3. Which part of the Hamiltonian-cycle proof would break without Lemma 2?
4. Why is $3n \ln n$ a natural time budget?

Discussions