Graph Structure via Exact Gaussian Elimination

Aaron Schild
aschild@berkeley.edu

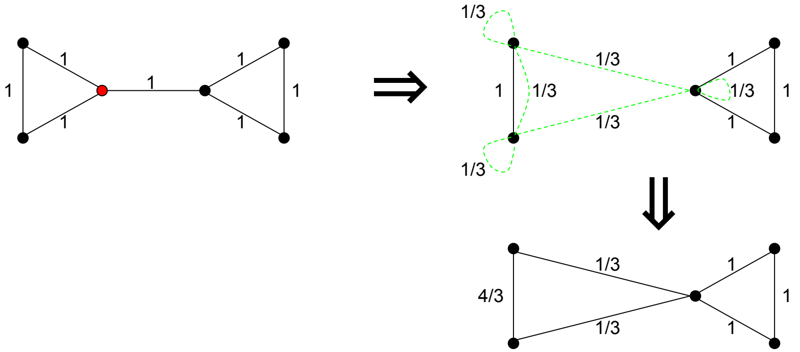
University of California, Berkeley

October 6, 2018

Relevant graph theoretic property of Schur complements

$H := \texttt{Schur}(G, S)$: a weighted graph with $V(H) = S$ and the property that the following two distributions are identical:

- the list of vertices visited by a random walk in H
- the list of vertices in S visited by a random walk in G



Goals of this talk

- Use our graph theoretic view of Schur complements to

Goals of this talk

- Use our graph theoretic view of Schur complements to
  - prove that electrical flows are good ℓ_2-oblivious routers

Goals of this talk

- Use our graph theoretic view of Schur complements to
  - prove that electrical flows are good ℓ_2-oblivious routers
  - sample uniformly random spanning trees in almost-linear time

Part I: Localization of Electrical Flows

Joint work with Satish Rao and Nikhil Srivastava

Types of flows

- Undirected graph G

Types of flows

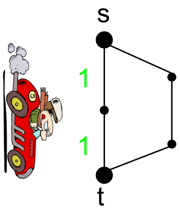
- Undirected graph G
- Two vertices $s, t \in G$ (always endpoints of an edge)

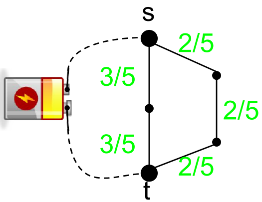
Types of flows

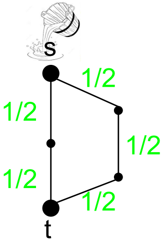
- Undirected graph G
- Two vertices $s, t \in G$ (always endpoints of an edge)
- $\mathbf{f}^{(p)}(s, t) \in \mathbb{R}^{E(G)}$: ℓ_p-minimizing s-t unit flow

Types of flows

- Undirected graph G
- Two vertices $s, t \in G$ (always endpoints of an edge)
- $\mathbf{f}^{(p)}(s,t) \in \mathbb{R}^{E(G)}$: ℓ_p-minimizing s-t unit flow
- Examples:
  - $\mathbf{f}^{(1)}(s,t)$ (s-t shortest path, left)
  - $\mathbf{f}^{(2)}(s,t)$ (s-t electrical flow, middle)
  - $\mathbf{f}^{(\infty)}(s,t)$ (proportional to s-t max flow, right)







Main question: how concentrated are electrical flows?

Concentration of flows

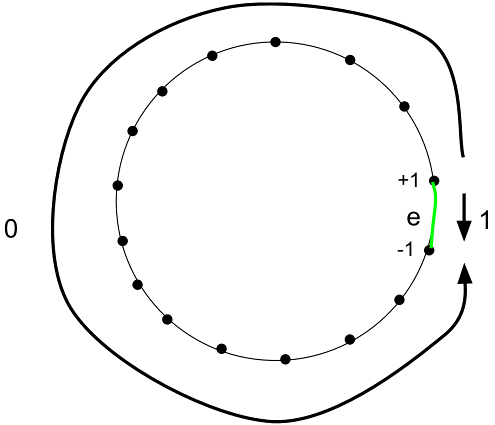
- Concentration of an edge e's ℓ_p flow: $\sum_{f \in E(G)} |\mathbf{f}_f^{(p)}(e)|$

Concentration of flows

- Concentration of an edge e's ℓ_p flow: $\sum_{f \in E(G)} |\mathbf{f}_f^{(p)}(e)|$
- Also the average length of flow paths

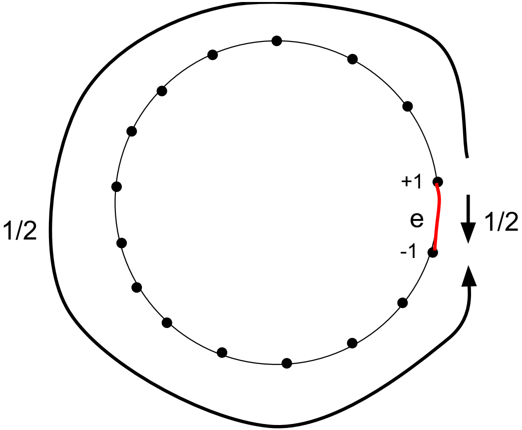
ℓ_1-flows (shortest paths) are concentrated

Total flow $= 1 \implies$ concentrated



ℓ_∞-flows (max flows) can be spread out

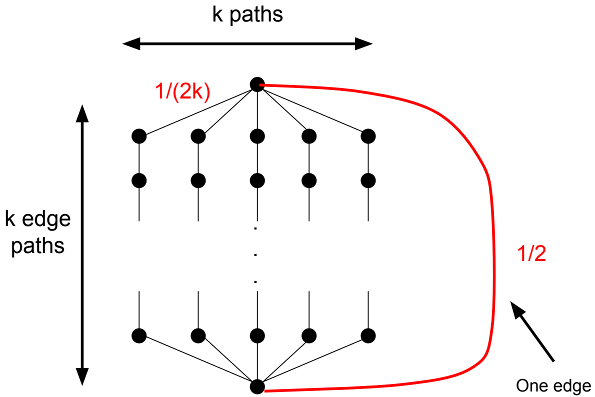
Total flow $= n/2 \implies$ spread out



ℓ_2-flows (electrical flows) can be spread out

$$k = \Theta(\sqrt{n})$$

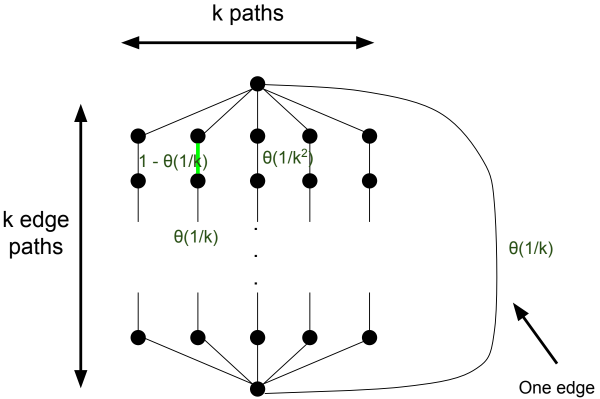
$$\text{Total flow} = \Theta(\sqrt{n}) \implies \text{spread out}$$



... but they aren't on average!

$$k = \Theta(\sqrt{n})$$

$$\text{Total flow} = \Theta(1) \implies \text{concentrated!}$$



Result: electrical flows concentrate on average

Theorem

In any unweighted graph G,

$$\sum_{e \in E(G)} \sum_{f \in E(G)} |\mathbf{f}_e^{(2)}(f)| \leq O(m \log^2 n)$$

Applications

- Electrical flows as oblivious routers

Applications

- Electrical flows as oblivious routers
  - ℓ_p-oblivious routing: given a set of demands $d_1, \ldots, d_k$, how well can routing demands independently do when compared with the optimal ℓ_p-multicommodity flow?

Applications

- Electrical flows as oblivious routers
  - ℓ_p-oblivious routing: given a set of demands $d_1, \ldots, d_k$, how well can routing demands independently do when compared with the optimal ℓ_p-multicommodity flow?
  - ℓ_∞-oblivious routing on edge-transitive graphs

Applications

- Electrical flows as oblivious routers
  - ℓ_p-oblivious routing: given a set of demands $d_1, \ldots, d_k$, how well can routing demands independently do when compared with the optimal ℓ_p-multicommodity flow?
  - ℓ_∞-oblivious routing on edge-transitive graphs
  - ℓ_2-oblivious routing on general graphs [HHN+08]

Applications

- Electrical flows as oblivious routers
  - ℓ_p-oblivious routing: given a set of demands $d_1, \ldots, d_k$, how well can routing demands independently do when compared with the optimal ℓ_p-multicommodity flow?
  - ℓ_∞-oblivious routing on edge-transitive graphs
  - ℓ_2-oblivious routing on general graphs [HHN$^+$08]
- Almost-linear time random spanning tree generation

Applications

- Electrical flows as oblivious routers
  - ℓ_p-oblivious routing: given a set of demands $d_1, \ldots, d_k$, how well can routing demands independently do when compared with the optimal ℓ_p-multicommodity flow?
  - ℓ_∞-oblivious routing on edge-transitive graphs
  - ℓ_2-oblivious routing on general graphs [HHN+08]
- Almost-linear time random spanning tree generation
  - Consider a process of the following form.
  - Fix two vertices s, t and repeatedly:
    - Compute s-t electrical flow
    - Change edge weight with minimum energy by constant factor

Applications

- Electrical flows as oblivious routers
  - ℓ_p-oblivious routing: given a set of demands $d_1, \ldots, d_k$, how well can routing demands independently do when compared with the optimal ℓ_p-multicommodity flow?
  - ℓ_∞-oblivious routing on edge-transitive graphs
  - ℓ_2-oblivious routing on general graphs [HHN+08]
- Almost-linear time random spanning tree generation
  - Consider a process of the following form.
  - Fix two vertices s, t and repeatedly:
    - Compute s-t electrical flow
    - Change edge weight with minimum energy by constant factor
  - Localization says that low energy edges remain low for a while

Applications

- Electrical flows as oblivious routers
  - ℓ_p-oblivious routing: given a set of demands $d_1, \ldots, d_k$, how well can routing demands independently do when compared with the optimal ℓ_p-multicommodity flow?
  - ℓ_∞-oblivious routing on edge-transitive graphs
  - ℓ_2-oblivious routing on general graphs [HHN$^+$08]
- Almost-linear time random spanning tree generation
  - Consider a process of the following form.
  - Fix two vertices s, t and repeatedly:
    - ★ Compute s-t electrical flow
    - ★ Change edge weight with minimum energy by constant factor
  - Localization says that low energy edges remain low for a while
  - Thus we can do fewer electrical flow computations

Linear algebraic version

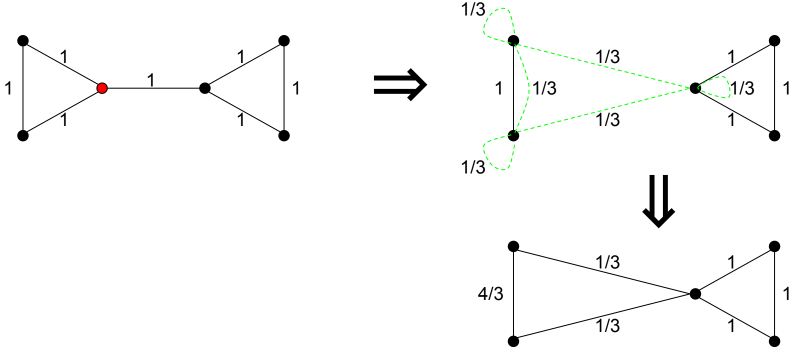
- Focus on unweighted graphs
- A: $n \times n$ adjacency matrix, $D = \text{diag}(A\mathbf{1})$: degree matrix
- $L = D - A$: Laplacian matrix of G with pseudoinverse L^+
- $b_e \in \mathbb{R}^n$: signed indicator of edge e
- In unweighted graphs, $\mathbf{f}_e^{(2)}(f) = |b_e^T L^+ b_f|$.

Restatement: $\sum_{e \in E(G)} \sum_{f \in E(G)} |b_e^T L^+ b_f| \leq O(m \log^2 n)$

Relevant graph theoretic property of Schur complements

$H := \texttt{Schur}(G, S)$: a weighted graph with $V(H) = S$ and the property that the following two distributions are identical:

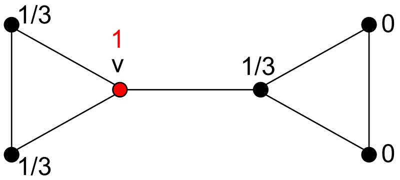
- the list of vertices visited by a random walk in H
- the list of vertices in S visited by a random walk in G



Projecting from G into $\texttt{Schur}(G, S)$

$$L^+ = P^T \begin{pmatrix} \texttt{Schur}(G,S)^+ & 0 \\ 0 & M^{-1} \end{pmatrix} P$$

- Applying Schur complement projection to a vector $\Leftrightarrow$ running a random walk until it hits S
- $P\mathbf{1}_v =$



Proof outline: high level

Goal: $\sum_{e \in E(G)} \sum_{f \in E(G)} |b_e^T L^+ b_f| \leq O(m \log^2 n)$

Proof outline: high level

Goal: $\sum_{e\in E(G)}\sum_{f\in E(G)}|b_e^T L^+ b_f| \leq O(m\log^2 n)$

- Pick a vertex, Schur complement it out, repeat (like Kyng-Sachdeva)

Proof outline: high level

Goal: $\sum_{e \in E(G)} \sum_{f \in E(G)} |b_e^T L^+ b_f| \leq O(m \log^2 n)$

- Pick a vertex, Schur complement it out, repeat (like Kyng-Sachdeva)
- Let x_i be the ith chosen vertex

Proof outline: high level

Goal: $\sum_{e \in E(G)} \sum_{f \in E(G)} |b_e^T L^+ b_f| \leq O(m \log^2 n)$

- Pick a vertex, Schur complement it out, repeat (like Kyng-Sachdeva)
- Let x_i be the ith chosen vertex
- Let $G_i := \texttt{Schur}(G, V(G) \setminus \{x_1, x_2, \dots, x_i\})$ with Laplacian L_i

Proof outline: high level

Goal: $\sum_{e \in E(G)} \sum_{f \in E(G)} |b_e^T L^+ b_f| \leq O(m \log^2 n)$

- Pick a vertex, Schur complement it out, repeat (like Kyng-Sachdeva)
- Let x_i be the ith chosen vertex
- Let $G_i := \texttt{Schur}(G, V(G) \setminus \{x_1, x_2, \dots, x_i\})$ with Laplacian L_i
- Let P_i be the Schur complement projection from G to G_i

Proof outline: high level

Goal: $\sum_{e\in E(G)}\sum_{f\in E(G)}|b_e^T L^+ b_f| \le O(m\log^2 n)$

- Pick a vertex, Schur complement it out, repeat (like Kyng-Sachdeva)
- Let x_i be the ith chosen vertex
- Let $G_i := \texttt{Schur}(G, V(G)\setminus\{x_1, x_2, \ldots, x_i\})$ with Laplacian L_i
- Let P_i be the Schur complement projection from G to G_i
- Let $\mathcal{V}_i = \sum_{e\in E(G)}\sum_{f\in E(G)}|(P_i b_e)^T L_i^+ (P_i b_f)|$

Proof outline: high level

Goal: $\sum_{e\in E(G)}\sum_{f\in E(G)}|b_e^T L^+ b_f| \le O(m\log^2 n)$

- Pick a vertex, Schur complement it out, repeat (like Kyng-Sachdeva)
- Let x_i be the ith chosen vertex
- Let $G_i := \texttt{Schur}(G, V(G)\setminus\{x_1,x_2,\ldots,x_i\})$ with Laplacian L_i
- Let P_i be the Schur complement projection from G to G_i
- Let $\mathcal{V}_i = \sum_{e\in E(G)}\sum_{f\in E(G)}|(P_i b_e)^T L_i^+(P_i b_f)|$
- **Suffices** to show that $\mathcal{V}_{i-1} \le \mathcal{V}_i + O(m(\log n)/(n-i))$

Subproblem for bounding increments

Subproblem for bounding increments

For a vertex set S, $v \in S$, and a vertex x in G, let

- $p_v^S(x)$ be the probability that random walk from x hits S at v

Subproblem for bounding increments

For a vertex set S, $v \in S$, and a vertex x in G, let

- $p_v^S(x)$ be the probability that random walk from x hits S at v

For an edge e with endpoints x and y, let

- $q_v^S(e) = |p_v^S(x) - p_v^S(y)|$

Subproblem for bounding increments

For a vertex set S, $v \in S$, and a vertex x in G, let

- $p_v^S(x)$ be the probability that random walk from x hits S at v

For an edge e with endpoints x and y, let

- $q_v^S(e) = |p_v^S(x) - p_v^S(y)|$

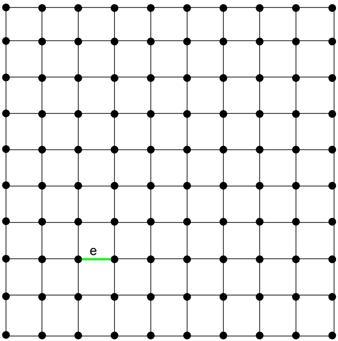
Lemma

$$\sum_{v \in S} \frac{\left(\sum_{e \in E(G)} q_v^S(e)\right)^2}{\sum_{e \in E(G)} q_v^S(e)^2} \leq O(m \log n)$$

Warmup: $S = V(G)$

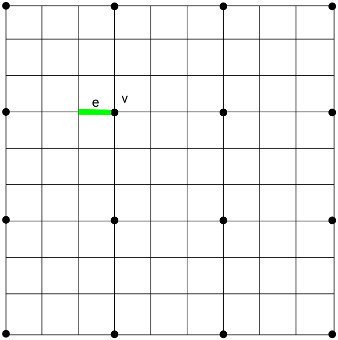
$q_v^S(e) = 1$ if v is an endpoint of e, 0 otherwise. Therefore,

$$\sum_{v \in S} \frac{(\sum_{e \in E(G)} q_v^S(e))^2}{\sum_{e \in E(G)} q_v^S(e)^2} = \sum_{v \in S} \frac{\deg(v)^2}{\deg(v)} = 2m$$



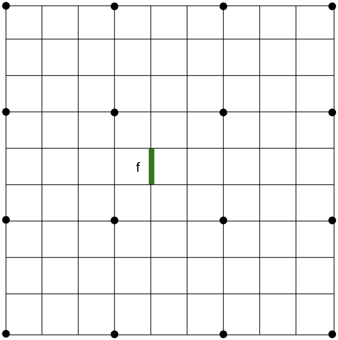
General S

Edges can only contribute substantially to a small number of terms



General S

Edges can only contribute substantially to a small number of terms



Lessons from Part I

Lessons from Part I

- electrical flows are concentrated on average

Lessons from Part I

- electrical flows are concentrated on average
- proof reduces to simpler objectives via vertex elimination

Lessons from Part I

- electrical flows are concentrated on average
- proof reduces to simpler objectives via vertex elimination
- probabilistic interpretation reasons about change in original graph

Part II: Random Spanning Trees

The weighted uniformly random spanning tree problem

Given an undirected graph G with weights (conductances) $\{c_e\}_{e\in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e\in E(T)} c_e$.

Motivation

Given an undirected graph G with weights (conductances) $\{c_e\}_{e \in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e \in E(T)} c_e$.

- Direct applications

Motivation

Given an undirected graph G with weights (conductances) $\{c_e\}_{e \in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e \in E(T)} c_e$.

- Direct applications
  - Symmetric [GSS11] and asymmetric [AGM$^+$10] traveling salesman
  - Dependent rounding

Motivation

Given an undirected graph G with weights (conductances) $\{c_e\}_{e \in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e \in E(T)} c_e$.

- Direct applications
  - Symmetric [GSS11] and asymmetric [AGM$^+$10] traveling salesman
  - Dependent rounding
  - Sparser cut sparsifiers [FH10, GRV09] and spectral sparsifiers [KS18]

Motivation

Given an undirected graph G with weights (conductances) $\{c_e\}_{e\in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e\in E(T)} c_e$.

- Direct applications
  - Symmetric [GSS11] and asymmetric [AGM+10] traveling salesman
  - Dependent rounding
  - Sparser cut sparsifiers [FH10, GRV09] and spectral sparsifiers [KS18]
  - Spectral sparsifiers via edge elimination [LS18]

Motivation

Given an undirected graph G with weights (conductances) $\{c_e\}_{e \in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e \in E(T)} c_e$.

- Direct applications
  - Symmetric [GSS11] and asymmetric [AGM$^+$10] traveling salesman
  - Dependent rounding
  - Sparser cut sparsifiers [FH10, GRV09] and spectral sparsifiers [KS18]
  - Spectral sparsifiers via edge elimination [LS18]
- Probability theory

Motivation

Given an undirected graph G with weights (conductances) $\{c_e\}_{e \in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e \in E(T)} c_e$.

- Direct applications
  - Symmetric [GSS11] and asymmetric [AGM$^+$10] traveling salesman
  - Dependent rounding
  - Sparser cut sparsifiers [FH10, GRV09] and spectral sparsifiers [KS18]
  - Spectral sparsifiers via edge elimination [LS18]
- Probability theory
- Sampling from determinantal point processes and correlated distributions

Motivation

Given an undirected graph G with weights (conductances) $\{c_e\}_{e \in E(G)}$ on its edges, sample a spanning tree T of G with probability proportional to $\prod_{e \in E(T)} c_e$.

- Direct applications
  - Symmetric [GSS11] and asymmetric [AGM$^+$10] traveling salesman
  - Dependent rounding
  - Sparser cut sparsifiers [FH10, GRV09] and spectral sparsifiers [KS18]
  - Spectral sparsifiers via edge elimination [LS18]
- Probability theory
- Sampling from determinantal point processes and correlated distributions
- Algorithmic applications of electrical flows

Matrix-based algorithms

m: number of edges
n: number of vertices
Idea: go through edges one by one and flip coins conditioned on prior choices

- Matrix-based algorithms (runtimes for weighted graphs)

Matrix-based algorithms

m: number of edges

n: number of vertices

Idea: go through edges one by one and flip coins conditioned on prior choices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83] $O(mn^\omega)$
  - [CMN96] $\tilde{O}(n^\omega)$

Matrix-based algorithms

m: number of edges
n: number of vertices
Idea: go through edges one by one and flip coins conditioned on prior choices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83] $O(mn^{\omega})$
  - [CMN96] $\tilde{O}(n^{\omega})$
  - [DKP$^+$17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$

Matrix-based algorithms

m: number of edges
n: number of vertices
Idea: go through edges one by one and flip coins conditioned on prior choices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83] $O(mn^\omega)$
  - [CMN96] $\tilde{O}(n^\omega)$
  - [DKP+17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$

Matrix-based algorithms

m: number of edges
n: number of vertices
Idea: go through edges one by one and flip coins conditioned on prior choices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83] $O(mn^{\omega})$
  - [CMN96] $\tilde{O}(n^{\omega})$
  - [DKP$^+$17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)

Matrix-based algorithms

m: number of edges

n: number of vertices

Idea: go through edges one by one and flip coins conditioned on prior choices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83] $O(mn^{\omega})$
  - [CMN96] $\tilde{O}(n^{\omega})$
  - [DKP+17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)
  - Quadratic for sparse graphs (☹)

Aldous-Broder: a random walk-based algorithm

Algorithm:

- Pick an arbitrary vertex $u \in V(G)$.
- Do a random walk until all vertices have been visited.
- Return the edges used to visit each vertex for the first time.

Aldous-Broder: a random walk-based algorithm

Algorithm:

- Pick an arbitrary vertex $u \in V(G)$.
- Do a random walk until all vertices have been visited.
- Return the edges used to visit each vertex for the first time.

Generates a weighted uniformly random spanning tree of G!

Aldous-Broder: a random walk-based algorithm

Algorithm:

- Pick an arbitrary vertex $u \in V(G)$.
- Do a random walk until all vertices have been visited.
- Return the edges used to visit each vertex for the first time.

Generates a weighted uniformly random spanning tree of G!

- Runtime: cover time, which can be $\Omega(mn)$ ☹

Aldous-Broder: a random walk-based algorithm

Algorithm:

- Pick an arbitrary vertex $u \in V(G)$.
- Do a random walk until all vertices have been visited.
- Return the edges used to visit each vertex for the first time.

Generates a weighted uniformly random spanning tree of G!

- Runtime: cover time, which can be $\Omega(mn)$ ☹
- Only need first visits (at most n such visits) ☺

History

m: number of edges
n: number of vertices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83]
  - [CMN96] $\tilde{O}(n^{\omega})$
  - [DKP+17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)
  - Quadratic for sparse graphs (☹)

History

m: number of edges

n: number of vertices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83]
  - [CMN96] $\tilde{O}(n^\omega)$
  - [DKP$^+$17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)
  - Quadratic for sparse graphs (☹)
- Random-walk-based algorithms (runtimes for unweighted graphs)

History

m: number of edges

n: number of vertices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83]
  - [CMN96] $\tilde{O}(n^{\omega})$
  - [DKP$^+$17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)
  - Quadratic for sparse graphs (☹)
- Random-walk-based algorithms (runtimes for unweighted graphs)
  - [Bro89, Ald90] $\tilde{O}(mn)$
  - [KM09] $\tilde{O}(m\sqrt{n})$
  - [MST15] $\tilde{O}(m^{4/3})$

History

m: number of edges

n: number of vertices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83]
  - [CMN96] $\tilde{O}(n^\omega)$
  - [DKP+17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)
  - Quadratic for sparse graphs (☹)
- Random-walk-based algorithms (runtimes for unweighted graphs)
  - [Bro89, Ald90] $\tilde{O}(mn)$
  - [KM09] $\tilde{O}(m\sqrt{n})$
  - [MST15] $\tilde{O}(m^{4/3})$
  - Polynomial dependence on weights (☹)

History

m: number of edges

n: number of vertices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83]
  - [CMN96] $\tilde{O}(n^\omega)$
  - [DKP+17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)
  - Quadratic for sparse graphs (☹)
- Random-walk-based algorithms (runtimes for unweighted graphs)
  - [Bro89, Ald90] $\tilde{O}(mn)$
  - [KM09] $\tilde{O}(m\sqrt{n})$
  - [MST15] $\tilde{O}(m^{4/3})$
  - Polynomial dependence on weights (☹)
  - Subquadratic running time (☺)

History

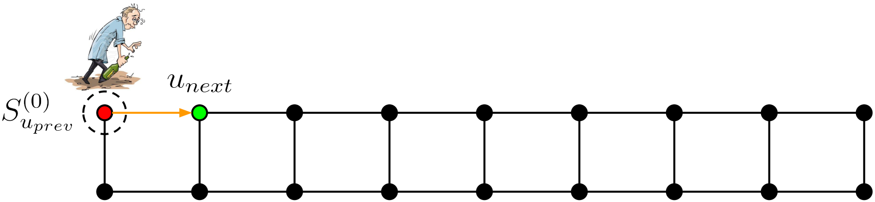
m: number of edges

n: number of vertices

- Matrix-based algorithms (runtimes for weighted graphs)
  - [Gue83]
  - [CMN96] $\tilde{O}(n^\omega)$
  - [DKP$^+$17] $\tilde{O}(n^{4/3}m^{1/2} + n^2)$
  - [DPPR17] $\tilde{O}(n^2\delta^{-2})$
  - No runtime dependence on weights (☺)
  - Quadratic for sparse graphs (☹)
- Random-walk-based algorithms (runtimes for unweighted graphs)
  - [Bro89, Ald90] $\tilde{O}(mn)$
  - [KM09] $\tilde{O}(m\sqrt{n})$
  - [MST15] $\tilde{O}(m^{4/3})$
  - Polynomial dependence on weights (☹)
  - Subquadratic running time (☺)
  - This work $\tilde{O}(m^{1+o(1)})$

Aldous-Broder remix

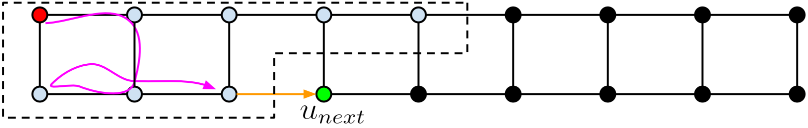
- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(0)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



[0] lachblogger.com

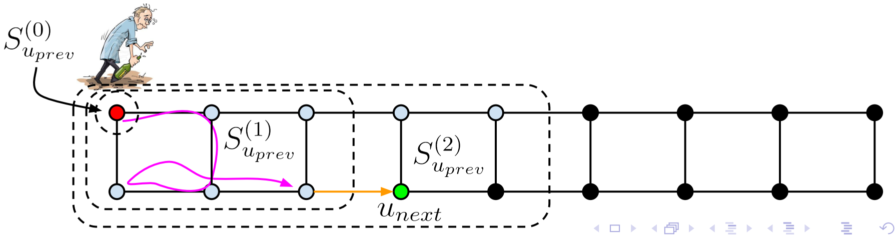
Wishful thinking

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - Sample the edge that the random walk starting at u uses to exit the set of visited vertices.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



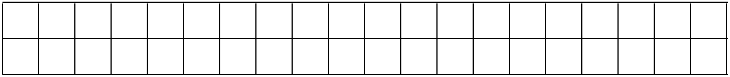
Shortcutting meta-algorithm

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and **pick** shortcutters $\{S_v^{(i)}\}_{i=1}^{\sigma_0}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - Let $i^* \in \{0, 1, \ldots, \sigma_0\}$ be the maximum value of i for which all vertices in $S_u^{(i)}$ have been visited.
  - **Sample** the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



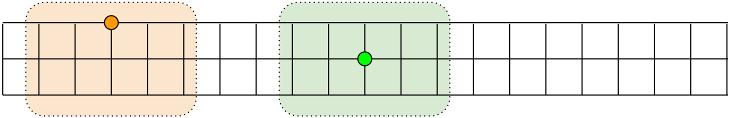
Example

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - Let $i^* \in \{0, 1, 2, 3\}$ be the maximum value of i for which all vertices in $S_u^{(i)}$ have been visited.
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



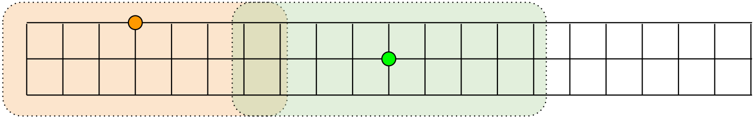
Example

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and **pick** shortcutters $S_v^{(1)}$ to be the 2-neighborhood of v for all $v \in V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - Let $i^* \in \{0, 1, 2, 3\}$ be the maximum value of i for which all vertices in $S_u^{(i)}$ have been visited.
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



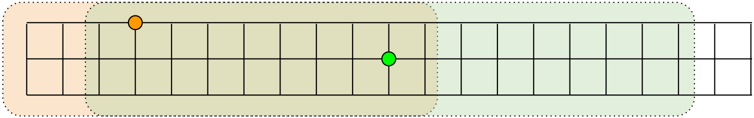
Example

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and **pick** shortcutters $S_v^{(2)}$ to be the 4-neighborhood of v for all $v \in V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - Let $i^* \in \{0, 1, 2, 3\}$ be the maximum value of i for which all vertices in $S_u^{(i)}$ have been visited.
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



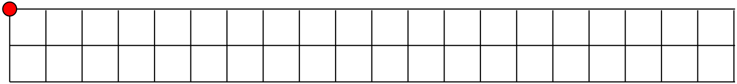
Example

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and **pick** shortcutters $S_v^{(3)}$ to be the 8-neighborhood of v for all $v \in V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - Let $i^* \in \{0, 1, 2, 3\}$ be the maximum value of i for which all vertices in $S_u^{(i)}$ have been visited.
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



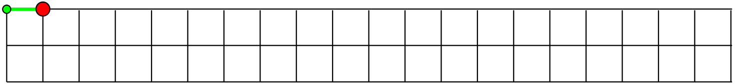
Example on walk step 1

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- **Pick** an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



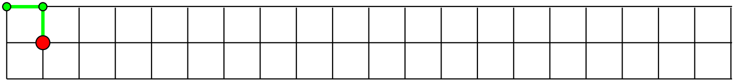
Example on walk step 2

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



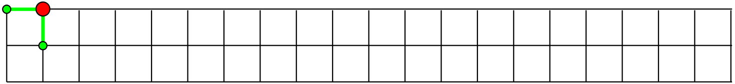
Example on walk step 3

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



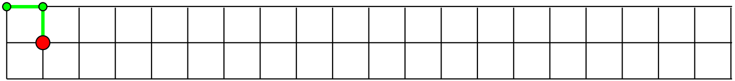
Example on walk step 4

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



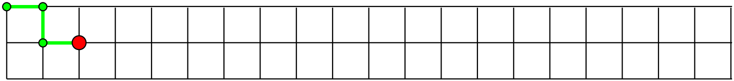
Example on walk step 5

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



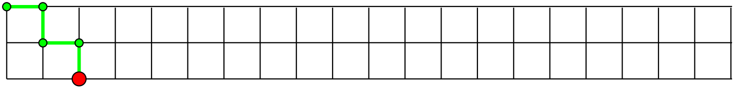
Example on walk step 6

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



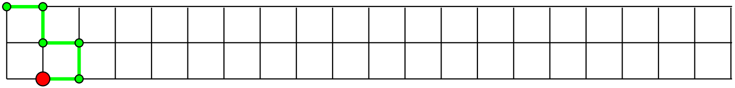
Example on walk step 7

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



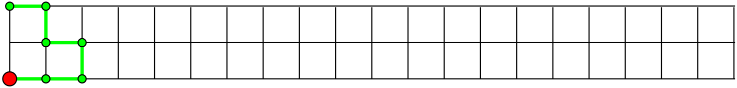
Example on walk step 8

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



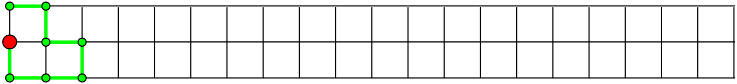
Example on walk step 9

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



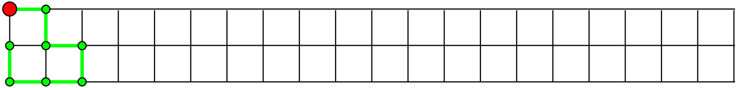
Example on walk step 10

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



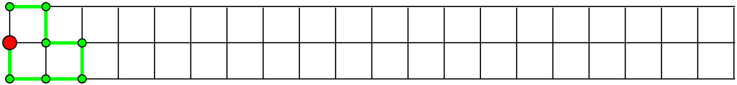
Example on walk step 11

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



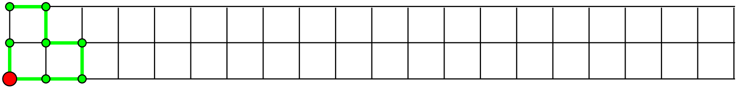
Example on walk step 12

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



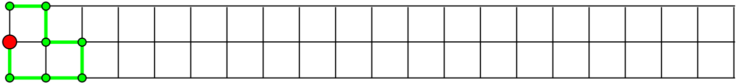
Example on walk step 13

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



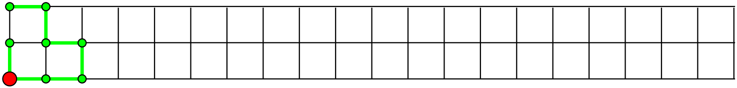
Example on walk step 14

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



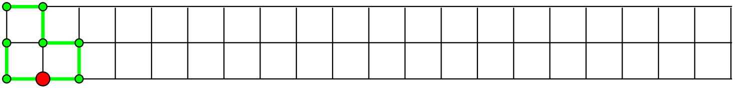
Example on walk step 15

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



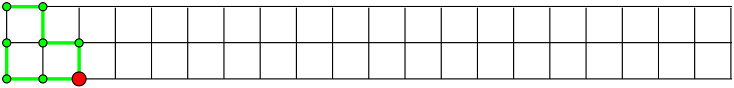
Example on walk step 16

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



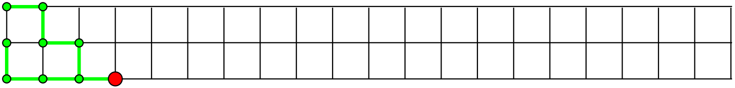
Example on walk step 17

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



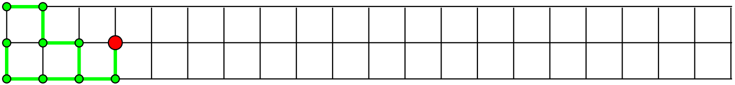
Example on walk step 18

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



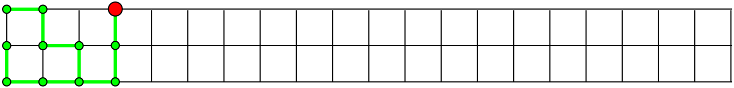
Example on walk step 19

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



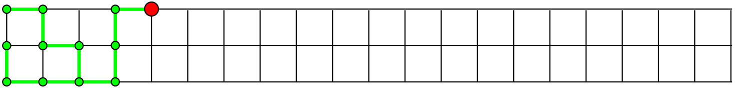
Example on walk step 20

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



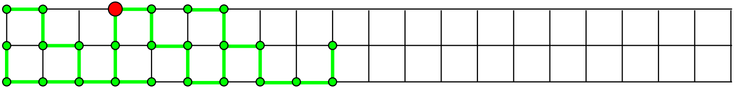
Example on walk step 21

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



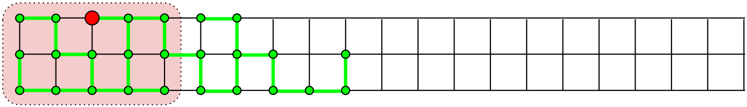
Example on walk step 64

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



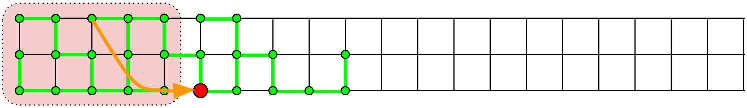
Example on walk step 65

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 1$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



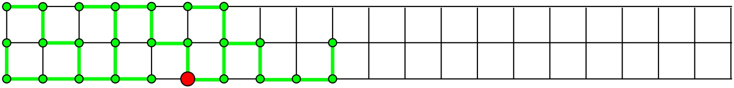
Example on walk step 65 $\to$ 120

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 1$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



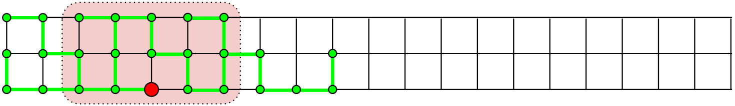
Example on walk step 120

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



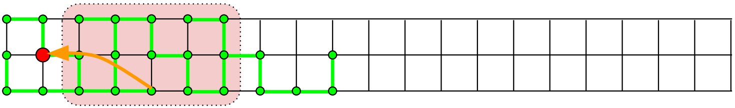
Example on walk step 121

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 1$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



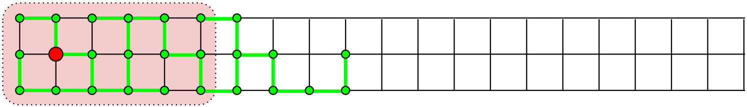
Example on walk step 121 $\to$ 127

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 1$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



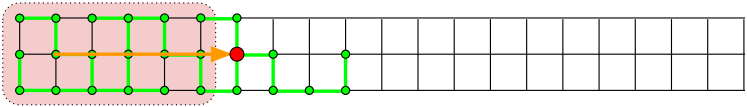
Example on walk step 127

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 2$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



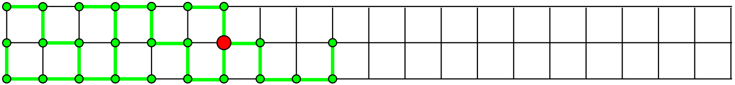
Example on walk step $127 \rightarrow 204$

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 2$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



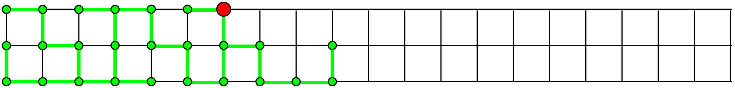
Example on walk step 204

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



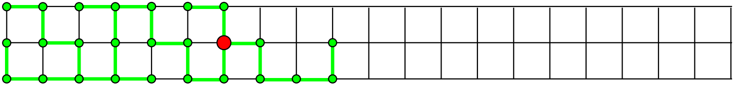
Example on walk step 205

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



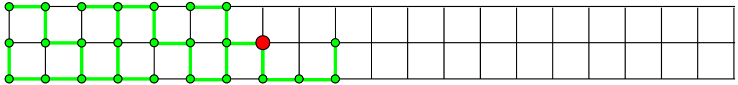
Example on walk step 206

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



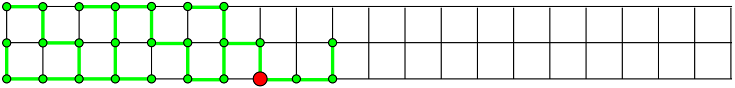
Example on walk step 207

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



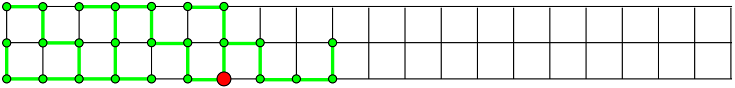
Example on walk step 208

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



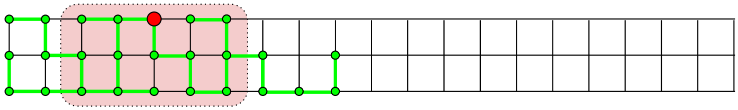
Example on walk step 209

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



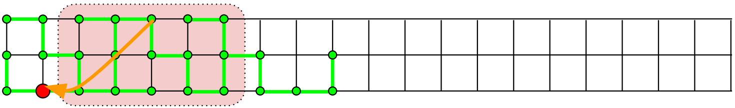
Example on walk step 213

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 1$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



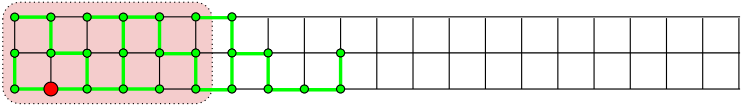
Example on walk step 213 $\rightarrow$ 246

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 1$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



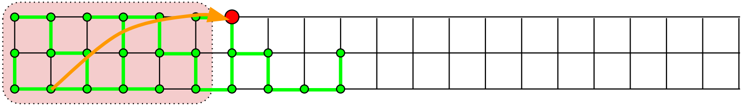
Example on walk step 246

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 2$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



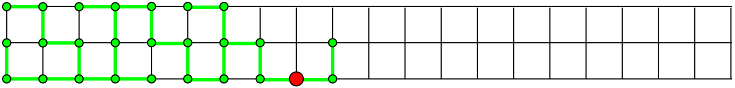
Example on walk step 246 $\rightarrow$ 307

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 2$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



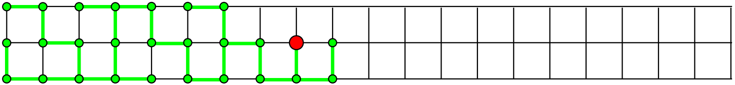
Example on walk step 313

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



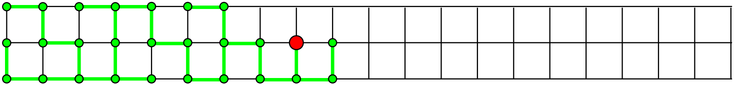
Example on walk step 314

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



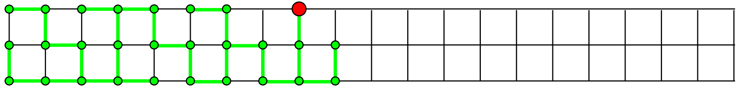
Example on walk step 316

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



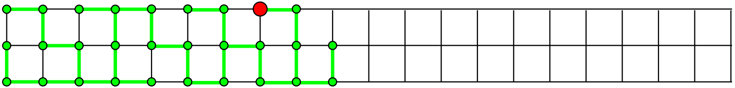
Example on walk step 317

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



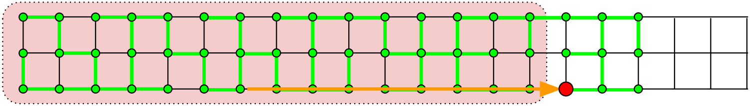
Example on walk step 318

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$
  and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 0$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



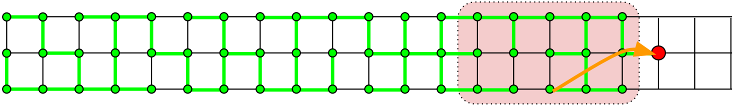
Example on walk step 485 $\to$ 821

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 3$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



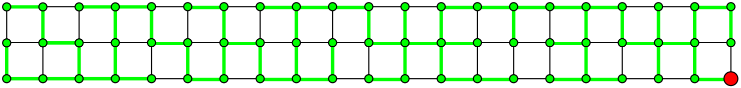
Example on walk step 821 $\rightarrow$ 826

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^{3}$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - $i^* \leftarrow 1$
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- Return the edges used to visit each vertex for the first time.



Example on walk step 875

- For each $v \in V(G)$, let $S_v^{(0)} = \{v\}$ and pick shortcutters $\{S_v^{(i)}\}_{i=1}^3$ with $v \in S_v^{(i)} \subseteq V(G)$
- Pick an arbitrary vertex $u \in V(G)$.
- While there is an unvisited vertex
  - **DONE**
  - Sample the edge that the random walk starting at u uses to exit $S_u^{(i^*)}$.
  - Replace u with the non-$S_u^{(i^*)}$ endpoint of this edge.
- **Return** the edges used to visit each vertex for the first time.



Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters
m: number of edges
n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]			$O(mn)$
[KM09]			$O(m\sqrt{n})$
[MST15]			$O(m^{4/3})$
This work			$O(m^{1+1/\sigma_0})$

Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters

m: number of edges

n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]	0		$O(mn)$
[KM09]			$O(m\sqrt{n})$
[MST15]			$O(m^{4/3})$
This work			$O(m^{1+1/\sigma_0})$

Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters

m: number of edges

n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]	0	N/A	$O(mn)$
[KM09]			$O(m\sqrt{n})$
[MST15]			$O(m^{4/3})$
This work			$O(m^{1+1/\sigma_0})$

Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters
m: number of edges
n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]	0	N/A	$O(mn)$
[KM09]	1		$O(m\sqrt{n})$
[MST15]			$O(m^{4/3})$
This work			$O(m^{1+1/\sigma_0})$

Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters

m: number of edges

n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]	0	N/A	$O(mn)$
[KM09]	1		$O(m\sqrt{n})$
[MST15]	2		$O(m^{4/3})$
This work			$O(m^{1+1/\sigma_0})$

Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters
m: number of edges
n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]	0	N/A	$O(mn)$
[KM09]	1		$O(m\sqrt{n})$
[MST15]	2		$O(m^{4/3})$
This work	$\tilde{\Theta}(\log \log n)$		$O(m^{1+1/\sigma_0})$

Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters
m: number of edges
n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]	0	N/A	$O(mn)$
[KM09]	1	Offline	$O(m\sqrt{n})$
[MST15]	2	Offline	$O(m^{4/3})$
This work	$\tilde{\Theta}(\log \log n)$		$O(m^{1+1/\sigma_0})$

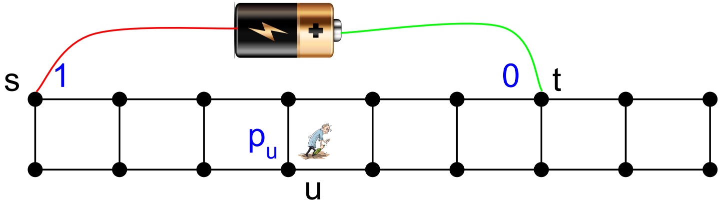
Summary of existing shortcutting-based algorithms

σ_0: number of shortcutters
m: number of edges
n: number of vertices

Algorithm	σ_0	Shortcutting method	Runtime
[Bro89, Ald90]	0	N/A	$O(mn)$
[KM09]	1	Offline	$O(m\sqrt{n})$
[MST15]	2	Offline	$O(m^{4/3})$
This work	$\tilde{\Theta}(\log \log n)$	Online	$O(m^{1+1/\sigma_0})$

Using Laplacian solvers to calculate hitting probabilities

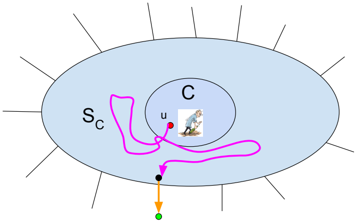
$$p_u = \Pr_u[\text{ random walk starting at } u \text{ hits } s \text{ before } t]$$



Can compute all p_us in $\tilde{O}(m)$ time! [ST14]

Shortcutting methods

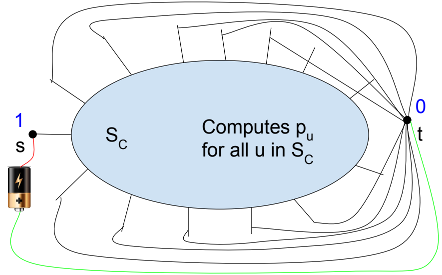
- **Given:** a shortcutter S_C with $C \subseteq S_C \subseteq V(G)$
- **Goal:** sample the S_C-escape edge for random walk starting at u



Shortcutting method	Preprocessing	Query

Offline shortcutting

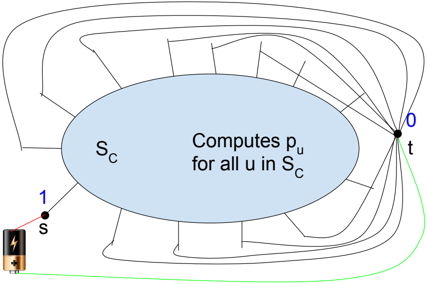
- **Given:** a shortcutter S_C with $C \subseteq S_C \subseteq V(G)$
- **Goal:** sample the S_C-escape edge for random walk starting at u



Shortcutting method	Preprocessing	Query
Offline	$\tilde{O}(\lvert E(S_C)\rvert \lvert\partial S_C\rvert)$	$O(\log n)$

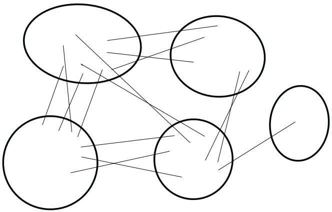
Offline shortcutting

- **Given:** a shortcutter S_C with $C \subseteq S_C \subseteq V(G)$
- **Goal:** sample the S_C-escape edge for random walk starting at u

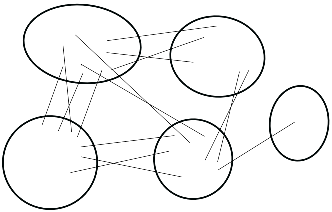


Shortcutting method	Preprocessing	Query
Offline	$\tilde{O}(\lvert E(S_C)\rvert \lvert\partial S_C\rvert)$	$O(\log n)$

Graph Partitioning



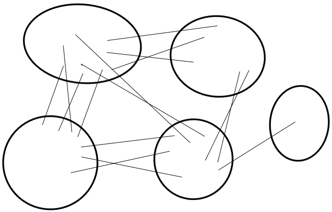
Graph Partitioning **via region growing**



Theorem (e.g. LR99)

A partition with diameter R and $O(m(\log m)/R)$ intercluster edges exists.

Graph Partitioning **via region growing**



Theorem (e.g. LR99)

A partition with diameter R and $O(m(\log m)/R)$ intercluster edges exists.

Applied throughout the metric embedding and LP rounding literature

Applying region growing to construct shortcutters [KM09]

Algorithm:

- Apply region-growing.
- Let $S_v^{(1)}$ be the unique cluster containing v.

Applying region growing to construct shortcutters [KM09]

Algorithm:

- Apply region-growing.
- Let $S_v^{(1)}$ be the unique cluster containing v.

Runtime:

- preprocessing time: (boundary)(cluster size) $\leq \tilde{O}((m/R)m) \leq \tilde{O}(m^2/R)$

Applying region growing to construct shortcutters [KM09]

Algorithm:

- Apply region-growing.
- Let $S_v^{(1)}$ be the unique cluster containing v.

Runtime:

- preprocessing time: (boundary)(cluster size) $\leq \tilde{O}((m/R)m) \leq \tilde{O}(m^2/R)$
- normal random walk steps: $\tilde{O}(mR)$

Applying region growing to construct shortcutters [KM09]

Algorithm:

- Apply region-growing.
- Let $S_v^{(1)}$ be the unique cluster containing v.

Runtime:

- preprocessing time: (boundary)(cluster size) $\leq \tilde{O}((m/R)m) \leq \tilde{O}(m^2/R)$
- normal random walk steps: $\tilde{O}(mR)$
- shortcut random walk steps: $\tilde{O}(m/R)n \leq \frac{m^2}{R}$

Applying region growing to construct shortcutters [KM09]

Algorithm:

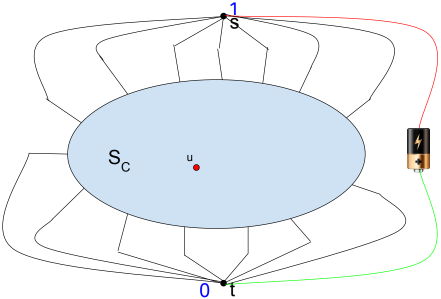
- Apply region-growing.
- Let $S_v^{(1)}$ be the unique cluster containing v.

Runtime:

- preprocessing time: (boundary)(cluster size) $\leq \tilde{O}((m/R)m) \leq \tilde{O}(m^2/R)$
- normal random walk steps: $\tilde{O}(mR)$
- shortcut random walk steps: $\tilde{O}(m/R)n \leq \frac{m^2}{R}$
- best tradeoff for $R = \sqrt{m}$.

Online shortcutting

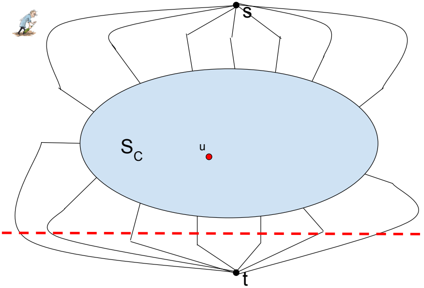
- **Given:** a shortcutter S_C with $C \subseteq S_C \subseteq V(G)$
- **Goal:** sample the S_C-escape edge for random walk starting at u



Shortcutting method	Preprocessing	Query
Offline	$\tilde{O}(\lvert E(S_C)\rvert\lvert\partial S_C\rvert)$	$O(\log n)$
Online	None	$\tilde{O}(\lvert E(S_C)\rvert)$

Online shortcutting

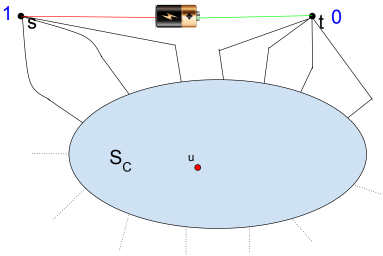
- **Given:** a shortcutter S_C with $C \subseteq S_C \subseteq V(G)$
- **Goal:** sample the S_C-escape edge for random walk starting at u



Shortcutting method	Preprocessing	Query
Offline	$\tilde{O}(\lvert E(S_C)\rvert \lvert\partial S_C\rvert)$	$O(\log n)$
Online	None	$\tilde{O}(\lvert E(S_C)\rvert)$

Online shortcutting

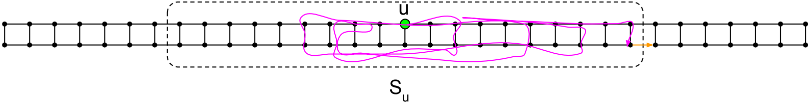
- **Given:** a shortcutter S_C with $C \subseteq S_C \subseteq V(G)$
- **Goal:** sample the S_C-escape edge for random walk starting at u



Shortcutting method	Preprocessing	Query
Offline	$\tilde{O}(\lvert E(S_C)\rvert \lvert \partial S_C \rvert)$	$O(\log n)$
Online	None	$\tilde{O}(\lvert E(S_C)\rvert)$

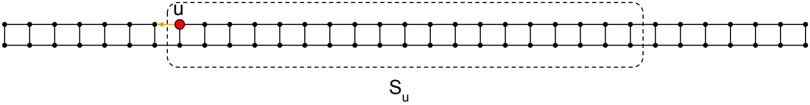
Using online shortcutting

Shortcutter work: $\tilde{O}(|E(S_u)|)$, random walk work $\Omega(|E(S_u)|^2)$ ☺



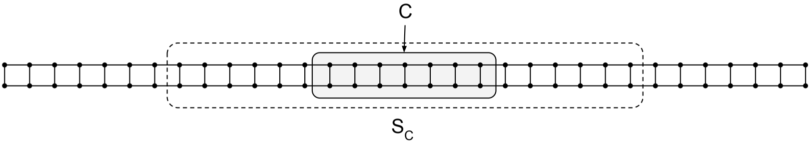
Using online shortcutting

Shortcutter work: $\tilde{\Omega}(|E(S_u)|)$, random walk work can be $O(1)$ ☹



Using online shortcutting

Core should be "well-separated" from the boundary of the shortcutter



Bounding work of online shortcutting

Bounding work of online shortcutting

Most random walk steps occur far away from an unvisited vertex

Bounding work of online shortcutting

Most random walk steps occur far away from an unvisited vertex

Lemma (Random walk bound)

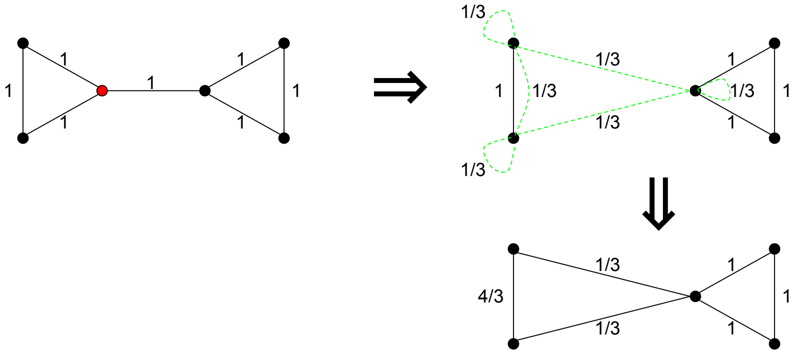
Consider a random walk starting at an arbitrary vertex in a graph I and an edge $\{u, v\} = f \in E(I)$. The

- *expected number of times the random walk traverses f from $u \to v$*
- *before the distance R-neighborhood of u is covered*
- *is at most $\tilde{O}(c_f R)$, where c_f is the conductance of the edge f*

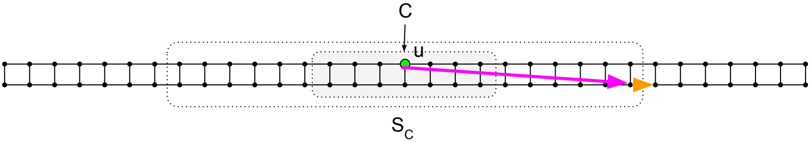
Recap of the Schur complement graph interpretation

$H := \texttt{Schur}(G, S)$: a weighted graph with $V(H) = S$ and the property that the following two distributions are identical:

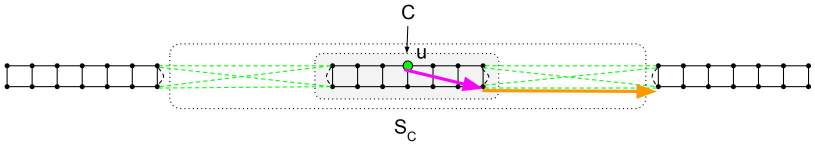
- the list of vertices visited by a random walk in H
- the list of vertices in S visited by a random walk in G



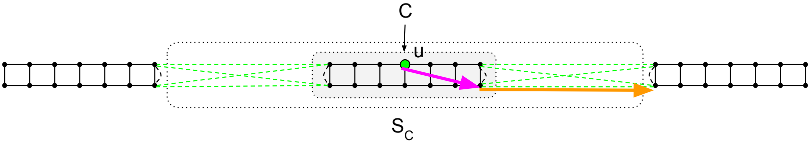
Charging shortcutter uses to Schur complement crossings



Charging shortcutter uses to Schur complement crossings

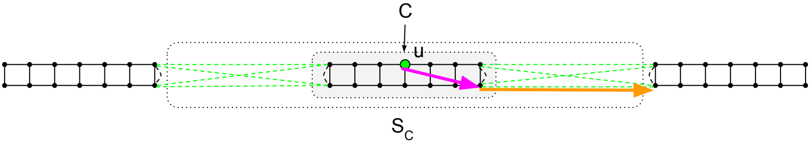


Charging shortcutter uses to Schur complement crossings



$R_i := m^{i/(\sigma_0+1)}$. To get almost-linear time,

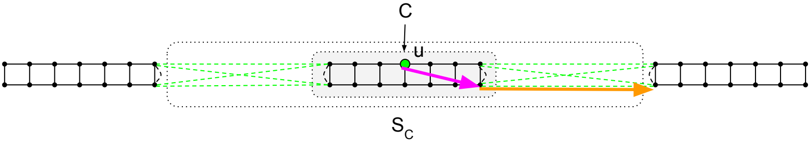
Charging shortcutter uses to Schur complement crossings



$R_i := m^{i/(\sigma_0+1)}$. To get almost-linear time,

- total green conductance $\leq \frac{1}{R_i}$ for $S_C^{(i)}$

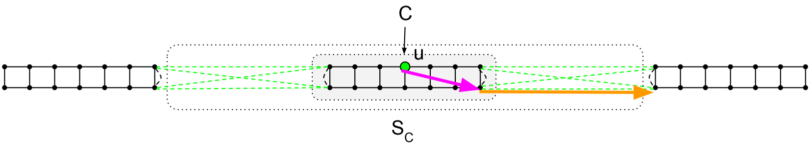
Charging shortcutter uses to Schur complement crossings



$R_i := m^{i/(\sigma_0+1)}$. To get almost-linear time,

- total green conductance $\leq \frac{1}{R_i}$ for $S_C^{(i)}$
- and distance to unvisited vertex $\leq R_{i+1}$

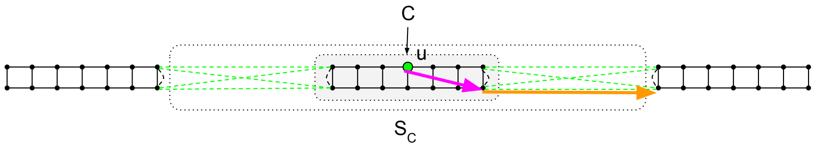
Charging shortcutter uses to Schur complement crossings



$R_i := m^{i/(\sigma_0+1)}$. To get almost-linear time,

- total green conductance $\leq \frac{1}{R_i}$ for $S_C^{(i)}$
- and distance to unvisited vertex $\leq R_{i+1}$
- uses/shortcutter
  $\leq$ (distance to unvisited vertex)(weight of edges)
  $\leq \tilde{O}(R_{i+1}/R_i) = m^{o(1)}$ by edge crossing bound

Charging shortcutter uses to Schur complement crossings



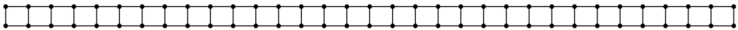
$R_i := m^{i/(\sigma_0+1)}$. To get almost-linear time,

- total green conductance $\leq \frac{1}{R_i}$ for $S_C^{(i)}$
- and distance to unvisited vertex $\leq R_{i+1}$
- uses/shortcutter
  $\leq$ (distance to unvisited vertex)(weight of edges)
  $\leq \tilde{O}(R_{i+1}/R_i) = m^{o(1)}$ by edge crossing bound
- work $\leq$ (total shortcutter size)(uses/shortcutter)
  $\leq (m^{1+o(1)})m^{o(1)} = m^{1+o(1)}$

Obtaining shortcutters that satisfy the first and third properties

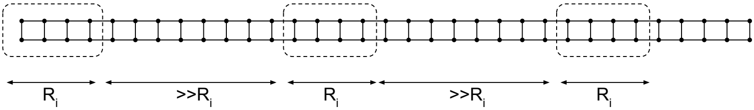
Obtaining shortcutters that satisfy the first and third properties

- Build cores first for each R_i, $i \in \{1, 2, \ldots, \sigma_0\}$ independently:



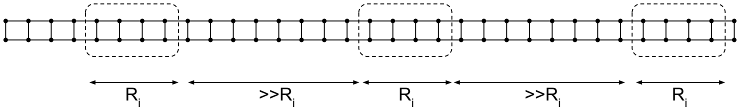
Obtaining shortcutters that satisfy the first and third properties

- Build cores first for each R_i, $i \in \{1, 2, \ldots, \sigma_0\}$ independently:
  - Cover the graph with $m^{o(1)}$ well-separated families in the effective resistance metric



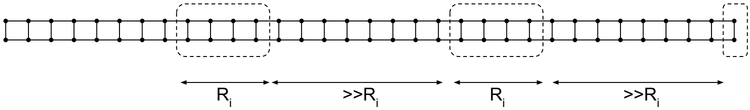
Obtaining shortcutters that satisfy the first and third properties

- Build cores first for each R_i, $i \in \{1, 2, \ldots, \sigma_0\}$ independently:
  - Cover the graph with $m^{o(1)}$ well-separated families in the effective resistance metric



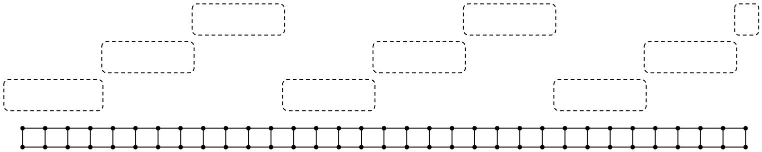
Obtaining shortcutters that satisfy the first and third properties

- Build cores first for each R_i, $i \in \{1, 2, \ldots, \sigma_0\}$ independently:
  - Cover the graph with $m^{o(1)}$ well-separated families in the effective resistance metric



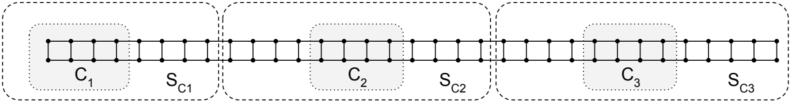
Obtaining shortcutters that satisfy the first and third properties

- Build cores first for each R_i, $i \in \{1, 2, \dots, \sigma_0\}$ independently:
  - Cover the graph with $m^{o(1)}$ well-separated families in the effective resistance metric
  - Construction similar to sparse neighborhood covers [ABCP99]



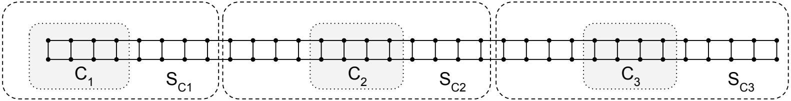
Obtaining shortcutters that satisfy the first and third properties

- Build cores first for each R_i, $i \in \{1, 2, \ldots, \sigma_0\}$ independently:
  - Cover the graph with $m^{o(1)}$ well-separated families in the effective resistance metric
  - Construction similar to sparse neighborhood covers [ABCP99]
- Build shortcutters around the cores



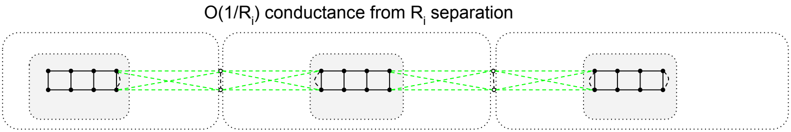
Obtaining shortcutters that satisfy the first and third properties

- Build cores first for each R_i, $i \in \{1, 2, \ldots, \sigma_0\}$ independently:
  - Cover the graph with $m^{o(1)}$ well-separated families in the effective resistance metric
  - Construction similar to sparse neighborhood covers [ABCP99]
- Build shortcutters around the cores
  - Voronoi diagram in probability space



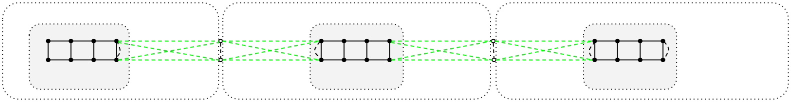
Obtaining shortcutters that satisfy the first and third properties

- First property: Schur complement conductance of $S_C^{(i)}$ is at most $\frac{m^{o(1)}}{R_i}$



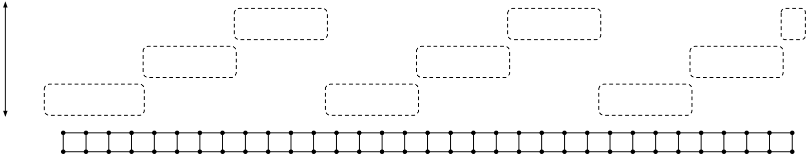
Obtaining shortcutters that satisfy the first and third properties

- First property: Schur complement conductance of $S_C^{(i)}$ is at most $\frac{m^{o(1)}}{R_i}$
  - Follows from well-separatedness



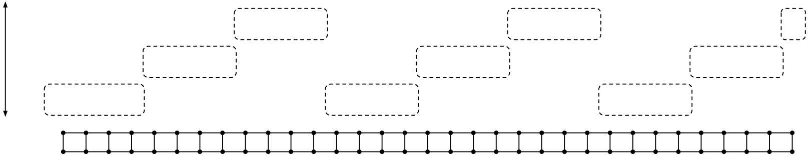
Obtaining shortcutters that satisfy the first and third properties

- First property: Schur complement conductance of $S_C^{(i)}$ is at most $\frac{m^{o(1)}}{R_i}$
  - Follows from well-separatedness
- Third property: Each vertex in G is only in $m^{o(1)}$ shortcutters



Obtaining shortcutters that satisfy the first and third properties

- First property: Schur complement conductance of $S_C^{(i)}$ is at most $\frac{m^{o(1)}}{R_i}$
  - Follows from well-separatedness
- Third property: Each vertex in G is only in $m^{o(1)}$ shortcutters
  - Follows from $\leq m^{o(1)}$ families



Lessons from Part II

- An $m^{1+o(1)}\alpha^{o(1)}$-time algorithm for generating weighted uniformly random spanning trees
- Overcame barriers in graph-partitioning based approaches from before by using Schur complements

Conclusion

- Probabilistic interpretation of Laplacian Gaussian elimination
  - Obtained a new ℓ_2 property of graphs
  - Random spanning trees in almost-linear time
- Paradigm relevant for other problems?

Questions?

Bibliography

Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg.
Near-linear time construction of sparse neighborhood covers.
SIAM J. Comput., 28(1):263–277, February 1999.

Arash Asadpour, Michel X. Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi.
An o(log n/ log log n)-approximation algorithm for the asymmetric traveling salesman problem.
In Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '10, pages 379–389, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

David J. Aldous.
The random walk construction of uniform spanning trees and uniform labelled trees.
SIAM J. Discret. Math., 3(4):450–465, November 1990.

A. Broder.