Sublinear-Time Sampling of Spanning Trees in the Congested Clique

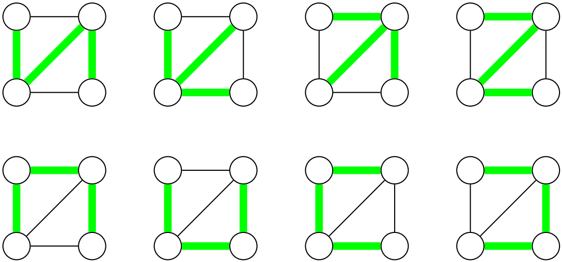
Sriram V. Pemmaraju, Sourya Roy, **Joshua Z. Sobel**

University of Iowa

June 2025

Random Spanning Trees

- Goal: Sample a spanning tree of a graph uniformly at random
- Purpose: Sparsification[1], Traveling Salesman[2], ...



[1] Goyal, Rademacher, Vempala. Expanders via random spanning trees. SODA '09

[2] Karlin, Klein, Gharan. A (slightly) improved approximation algorithm for metric tsp. STOC '21

Our Result

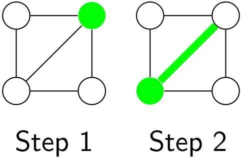
- First algorithm in Congested Clique
- Congest model existing $\tilde{O}(\sqrt{m}D)$ round algorithm[3]
- Main Result: $\tilde{\Theta}(n^{0.5+\alpha}) \approx (n^{0.657})$ rounds in Congested Clique
  Distributed matrix multiplication exponent[4] $\alpha \approx 0.157$
- Gives $\epsilon = \frac{1}{\text{poly}(n)}$ approximate sampling
- Exact sampling in $\tilde{\Theta}(n^{2/3+\alpha})$ rounds

[3] Das Sarma, Nanongkai, Pandurangan, Tetali. Efficient distributed random walks with applications. PODC '10

[4] Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela. Algebraic methods in the congested clique. PODC '15

Aldous-Broder Algorithm[6] [7]

- Take random walk starting from arbitrary vertex
- Collection of first visit edges (excluding start vertex) forms a uniform spanning tree
- Runtime is cover time
- Cover time is $O(n^3)$ [5]



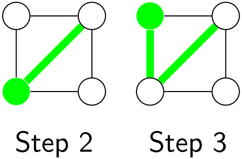
[5] Aleliunas, Karp, Lipton, Lovasz, Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. SFCS '79

[6] Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. SIAM Journal on Discrete Mathematics '90

[7] Broder. Generating random spanning trees. SFCS '89

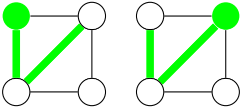
Aldous-Broder Algorithm

- Take random walk starting from arbitrary vertex
- Collection of first visit edges (excluding start vertex) forms a uniform spanning tree
- Runtime is cover time
- Cover time is $O(n^3)$



Aldous-Broder Algorithm

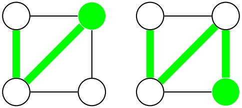
- Take random walk starting from arbitrary vertex
- Collection of first visit edges (excluding start vertex) forms a uniform spanning tree
- Runtime is cover time
- Cover time is $O(n^3)$



Step 3 Step 4

Aldous-Broder Algorithm

- Take random walk starting from arbitrary vertex
- Collection of first visit edges (excluding start vertex) forms a uniform spanning tree
- Runtime is cover time
- Cover time is $O(n^3)$



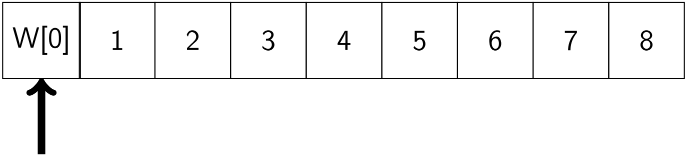
Step 4 Step 5

Top Down Walk Algorithm

- We will use a top-down "walk-filling" approach
- First pick start and end point of walk
- Recursively choose midpoints of walk
- Notation: $P[a, b]$ denotes transition matrix $a \to b$
- Precompute $P, P^2, ..., P^\ell$

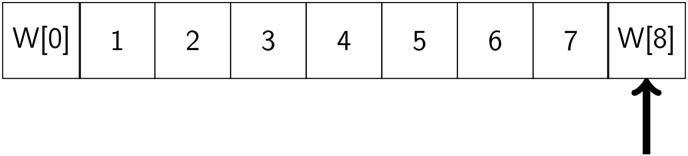
Top Down Walk Algorithm (Initialization)

-Pick arbitrary start vertex W[0]



Top Down Random Walk Algorithm (Initialization)

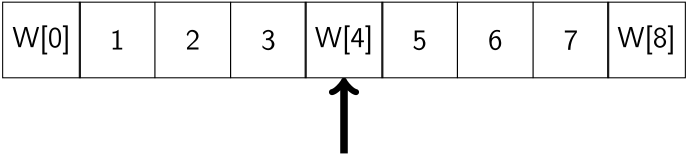
-Pick end vertex W[8]
-Distribution is $P^8[W[0], *]$



Top Down Random Walk Algorithm (Level 1)

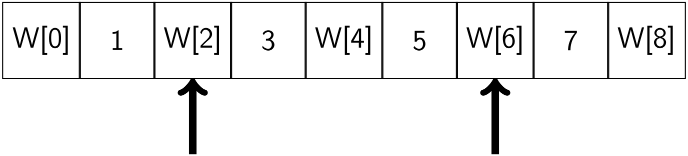
-Pick middle vertex W[4]

-Probability distribution $\propto P^4[W[0],*]P^4[*,W[8]]$



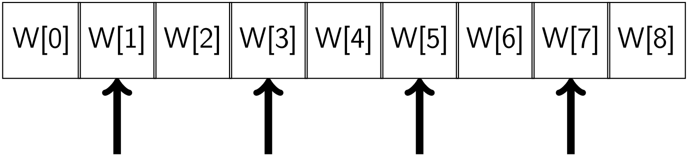
Top Down Random Walk Algorithm (Level 2)

-Pick vertices W[2],W[6]

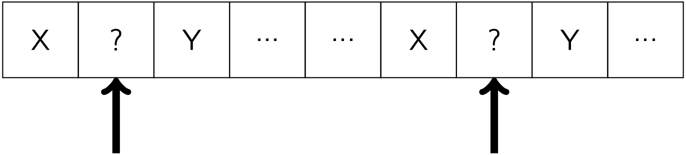


Top Down Random Walk Algorithm (Level 3)

-Pick remaining vertices



Key Insight 1



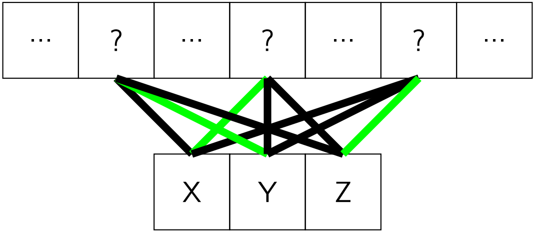
- Central machine M holds walk
- Midpoint distribution only depends on start/end points
- Assign one machine to generate midpoints for each start/end pair
- Collect midpoints at M for placement

Problem 1

- Problem: Can be $O(n^2)$ start/end pairs (too many to assign one to each machine)
- Solution: At any level truncate walk at $\sqrt{n}$-th distinct vertex***
- Result: We get random walk that visits exactly $\sqrt{n}$ vertices

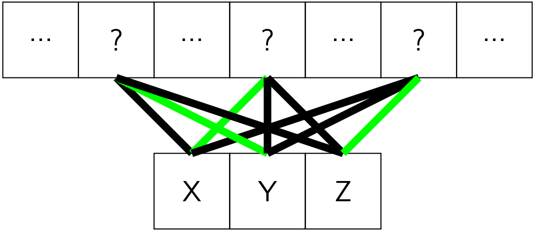
Problem 2

- Problem: Machine responsible for start/end pair (p, q) can't send sequence of midpoints to central machine.
- Compromise: Central machine receives multiset of all generated midpoints.
- Solution: Central machine resamples midpoint positions by drawing a random weighted perfect matching.



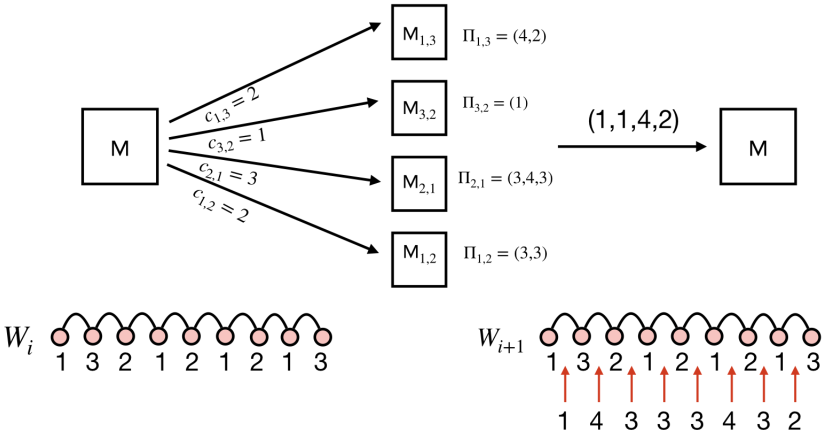
Sampling Matchings

Approximately sampling a random perfect matching can be done in polynomial time[8]



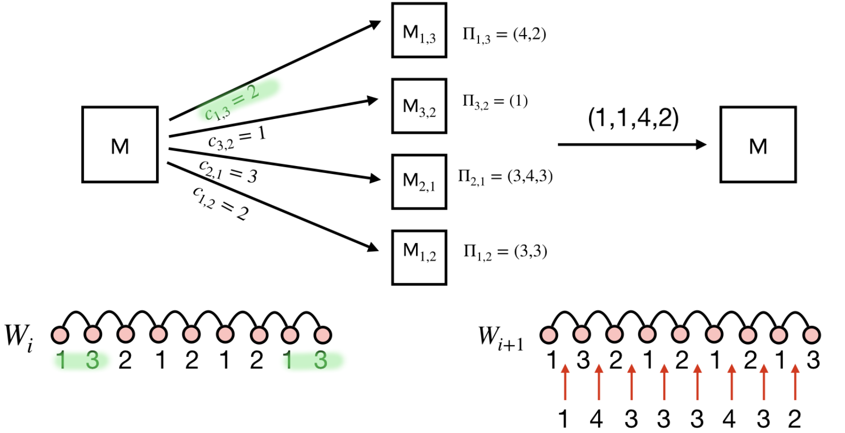
[8] Jerrum, Sinclair, Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. J. ACM '04

Overview

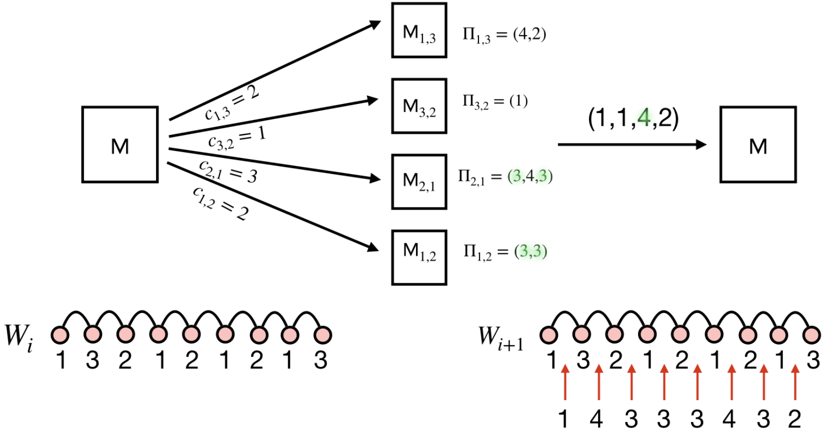


M
$c_{1,3} = 2$
$c_{3,2} = 1$
$c_{2,1} = 3$
$c_{1,2} = 2$
$M_{1,3}$
$\Pi_{1,3} = (4,2)$
$M_{3,2}$
$\Pi_{3,2} = (1)$
$M_{2,1}$
$\Pi_{2,1} = (3,4,3)$
$M_{1,2}$
$\Pi_{1,2} = (3,3)$
(1,1,4,2)
M
W_i
1 3 2 1 2 1 2 1 3
W_{i+1}
1 3 2 1 2 1 2 1 3
1 4 3 3 3 4 3 2

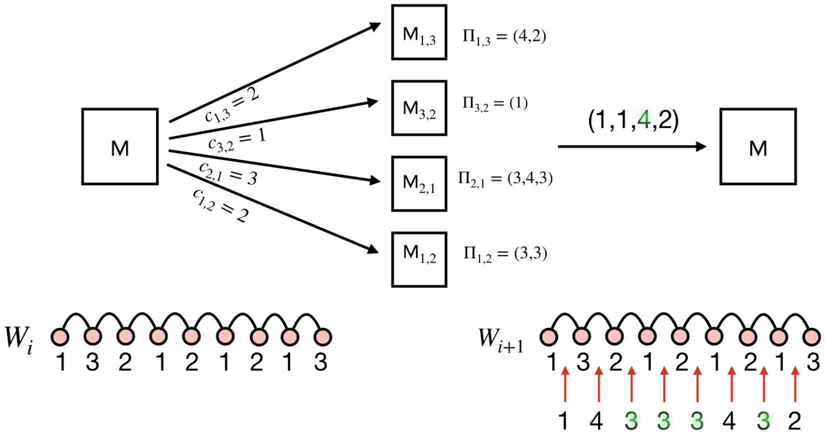
Overview


M
$c_{1,3} = 2$
$c_{3,2} = 1$
$c_{2,1} = 3$
$c_{1,2} = 2$
$M_{1,3}$
$\Pi_{1,3} = (4,2)$
$M_{3,2}$
$\Pi_{3,2} = (1)$
$M_{2,1}$
$\Pi_{2,1} = (3,4,3)$
$M_{1,2}$
$\Pi_{1,2} = (3,3)$
(1,1,4,2)
M
W_i
1 3 2 1 2 1 2 1 3
W_{i+1}
1 3 2 1 2 1 2 1 3
1 4 3 3 3 4 3 2

Overview

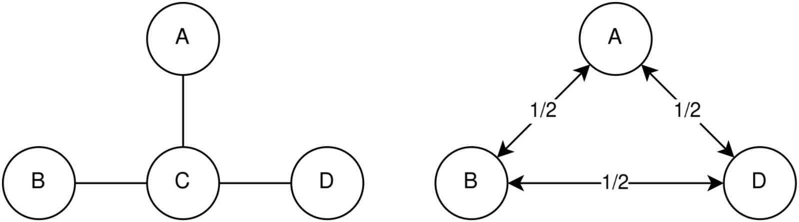

M
$c_{1,3} = 2$
$c_{3,2} = 1$
$c_{2,1} = 3$
$c_{1,2} = 2$
$M_{1,3}$
$\Pi_{1,3} = (4,2)$
$M_{3,2}$
$\Pi_{3,2} = (1)$
$M_{2,1}$
$\Pi_{2,1} = (3,4,3)$
$M_{1,2}$
$\Pi_{1,2} = (3,3)$
(1,1,4,2)
M
W_i
1 3 2 1 2 1 2 1 3
W_{i+1}
1 3 2 1 2 1 2 1 3
1 4 3 3 3 4 3 2

Overview


M
$c_{1,3} = 2$
$c_{3,2} = 1$
$c_{2,1} = 3$
$c_{1,2} = 2$
$M_{1,3}$
$M_{3,2}$
$M_{2,1}$
$M_{1,2}$
$\Pi_{1,3} = (4,2)$
$\Pi_{3,2} = (1)$
$\Pi_{2,1} = (3,4,3)$
$\Pi_{1,2} = (3,3)$
(1,1,4,2)
M
W_i
1 3 2 1 2 1 2 1 3
W_{i+1}
1 3 2 1 2 1 2 1 3
1 4 3 3 3 4 3 2

Problem 3

- Problem: Sampled walk only holds $\sqrt{n}$ distinct vertices
- Solution: Split algorithm into many phases
- Each phase visits new $\sqrt{n}$ vertices
- Use shortcutting technique[9]



[9] Kelner, Madry. Faster generation of random spanning trees. FOCS '09

Runtime

- $\sqrt{n}$ phases
- n^{α} time per phase
- Total Runtime: $\tilde{O}(n^{0.5+\alpha})$

Any Questions?