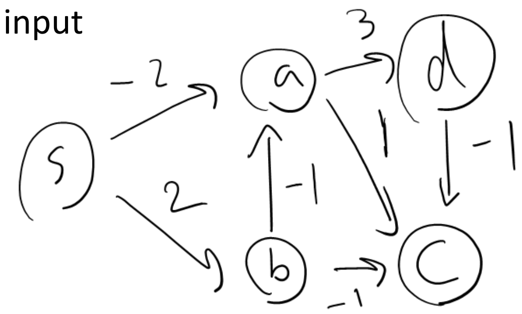
Negative-Weight Single-Source Shortest Paths

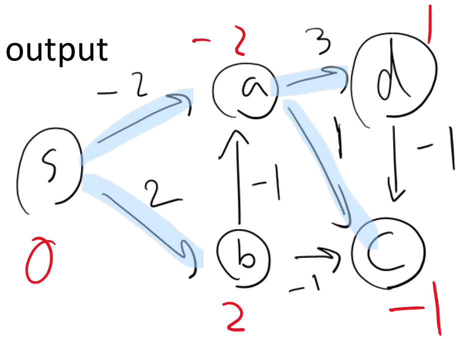
Danupon Nanongkai

MPI for Informatics

Single-Source Shortest Paths (SSSP)

- Input: Directed weighted graph $G = (V, E, w)$, source $s \in V$
  - Integer weights
- Output: distances $dist(s, v)$ from s to every $v \in V$





Notations
m = |E|, n=|V|
W is s.t. $w(e) \geq -W \; \forall e \in E$
$dist(u, v)$=length of shortest uv-path
$\tilde{O}$ hides polylog n

Textbook algorithms

Dijkstra
- Near-linear time ($O(m + n \log n)$ time)
- Restricted to **nonnegative** edge weights.

Bellman-Ford
- Work with **negative weights**
- **Far from** near-linear time (O(mn) time)

Research question: Fast algorithm for negative-weight SSSP?

History (m=#of edges, n=# vertices, w(e)$\geq -W$)

Classic (50s): O(mn) [Shimbel'55, Ford'56, Bellman'58, Moore'59]

Scaling techniques (80s-90s): $O(m\sqrt{n}\log W)$

Gabow'85 ($mn^{3/4}\log n$), Gabow-Tarjan'89 ($mn^{1/2}\log(nW)$), Goldberg'95 ($mn^{1/2}\log(W)$)

Special cases:

$\tilde{O}(m)$ time for planar graphs [Fakcharoenphol-Rao'06, Mozes-Wulff-Nilsen'10]

$\tilde{O}(n^{4/3}\log W)$ time for bounded-genus & minor-free graphs [Wulff-Nilsen'11]

Continuous Optimization + Dynamic Alg: $m^{4/3+o(1)}\log W$, $\tilde{O}(m+n^{1.5}\log W)$

- Cohen, Madry, Sankowski, Vladu'17 (($m^{10/7+o(1)}\log W$) , Axiotis, Madry, Vlady, 2020 ($m^{4/3+o(1)}\log W$), van den Brand, Lee, N, Peng, Saranurak, Sidford, Song, Wang 2020 ($m+n^{1.5}\log W$)

Our results (m=#of edges, n=# vertices, w(e)≥ $-W$)

(2022) $O(m\log^8(n)\ \log(W))$ time in expectation (without optimizing $\log^8$ term)

Bernstein, Nanongkai, Wulff-Nilsen: Negative-Weight Single-Source Shortest Paths in Near-linear Time.

(2023) $O(m\log^2(n)\log(nW)\,loglog(n))$ time in expectation

Bringmann, Cassis, Fischer: Negative-Weight Single-Source Shortest Paths in Near-Linear Time: Now Faster!

Codable. Teachable. Efficient in parallel, distributed, … [Ashvinkumar et al. 2023]

Related result: Min-cost flow in $m^{1+o(1)}log(W)$ time [Chen et al. 2022]

- Generalizes Negative SSSP, Bipartite matching, etc.
- Different techniques – e.g. continuous optimization & dynamic data structures

Key Tool: Low-Diameter Decomposition (LDD)

(SCC= Strongly connected component)

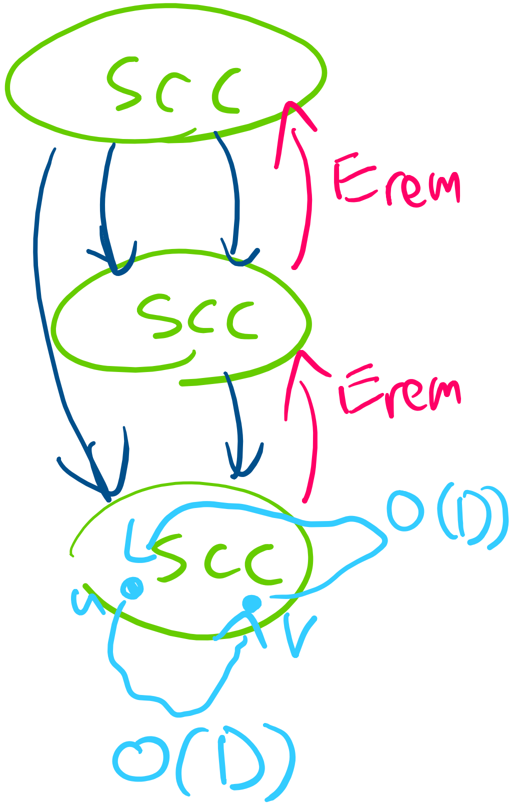
Definition of LDD(G,D)

<u>Input</u>: Directed graph $G = (V, E, w)$ with non-negative integer edge weight w and a positive integer D

<u>Output</u>: $E_{Rem} \subseteq E$ such that

1. each SCCs of $G \setminus E_{rem}$ has “weak diameter” O(D)
   - i.e. for u, v in the same SCC, $dist_G(u,v) = O(D)$ & $dist_G(v,u) = O(D)$
2. $\forall e \in E$, $\Pr[e \in E_{rem}] = O\left(\frac{w(e) \cdot (\log n)^2}{D} + n^{-8}\right)$.

<u>Runtime</u>: $\tilde{O}(m)$ in expectation.



<u>Remarks</u>: Probabilities may **not** be independent. E_{rem} is called E_{sep} in the previous version

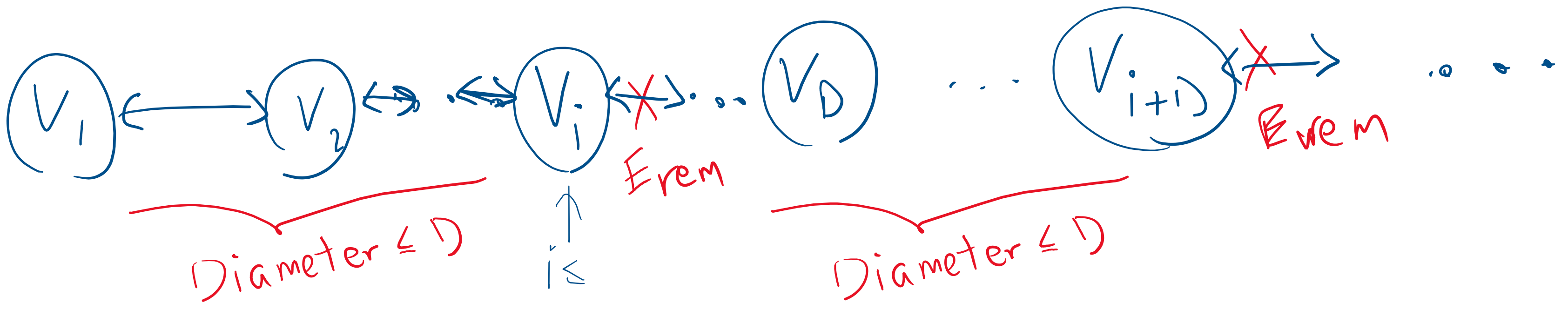
SCC = strongly-connected component

Example (1)

G = undirected path $(v_1, v_2, \ldots, v_n)$

<u>Getting LDD(G, D):</u>

- randomly select $i \in [1, D]$
- add edges $(v_i, v_{i+1}), (v_{i+D}, v_{i+D+1}), (v_{i+2D}, v_{i+2D+1})$, … to E_{rem}

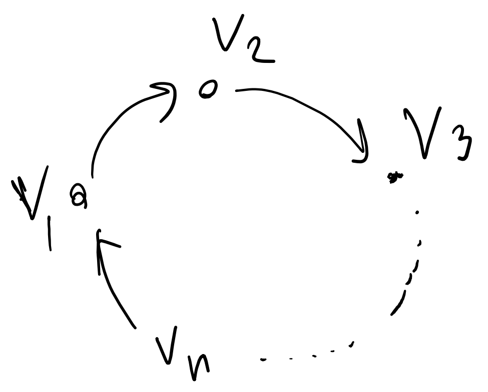


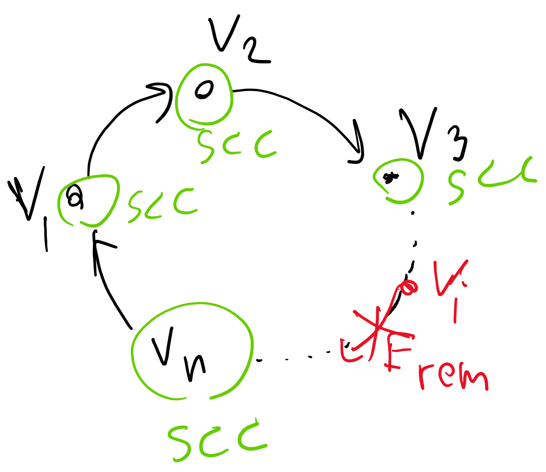
Example (2)

G = directed cycle $(v_1, v_2, \dots, v_n)$

<u>Getting LDD(G, D):</u> randomly add one edge to E_{rem}

→ Each node becomes an SCC





Our LDD algorithm

Algorithm FastLDD(G, D) // See explanation below

1. $G_0 \leftarrow G$. $E_{rem} \leftarrow \emptyset$ and $n \leftarrow |V(G)|$. // n does not change in the steps below.
2. For each $v \in V$:
   2.1 $k \leftarrow c \ln n$ for a big enough constant c.
   2.2 $S \leftarrow \{s_1, s_2, \dots s_k\}$ where s_i is a random node in V for every i. (Possible: $s_i = s_j$ for some $i \neq j$.)
   2.3 For each $s \in S$, compute $Ball_G^{in}(s, D)$ and $Ball_G^{out}(s, D)$ // $O(mk)$ time
   2.4 For each $v \in V$: // $O(nk)$ time
   - $|Ball_G^{in}(v, D) \cap S| \leftarrow \{s \in S \mid v \in Ball_G^{out}(s, D)\}$
   - $|Ball_G^{out}(v, D) \cap S| \leftarrow \{s \in S \mid v \in Ball_G^{in}(s, D)\}$

     2.5 For every $v \in V$, if $|Ball_G^{in}(v, D) \cap S| \leq (0.6)k$, mark v as **in-light**; else if $|Ball_G^{out}(v, D) \cap S| \leq (0.6)k$, mark v as **out-light**.
3. While G contains node v marked $*$**-light** for any $* \in \{in, out\}$:
   3.1 Sample $R_v \sim Geom(p)$ for $p = \min\{1, 20 \log(n)/D\}$. If $R_v > D/2$, let $R_v = D/2$.
   3.2 Compute $Ball_G^*(v, R_v)$. If $|Ball_G^*(v, R_v)| > 0.7n$, return $E_{rem} = E_{rem} \cup E(G)$ and terminate. // Claim: $Pr[terminate] \leq 1/n^{10}$
   3.3 $E'_{rem} \leftarrow LDD\left(G[Ball_G^*(v, R_v)], D\right)$. // Recurse
   3.4 $E_{rem} \leftarrow E_{rem} \cup \partial Ball_G^*(v, R_v) \cup E'_{rem}$
   3.4 $G \leftarrow G \setminus Ball_G^*(v, R_v)$.
4. Let v be any node in G. If $dist_{G_0}(v, u) > 2D$ or $dist_{G_0}(u, v) > 2D$ for any node u, return $E_{rem} = E_{rem} \cup E(G)$ and terminate. // Claim: $Pr[terminate] \leq 1/n^{10}$
   - (If $dist_{G_0}(v, u) \leq 2D$ and $dist_{G_0}(u, v) \leq 2D$, we can conclude that the weak diameter of G w.r.t. G_0 is $\leq 4D$.)

Return E_{rem}

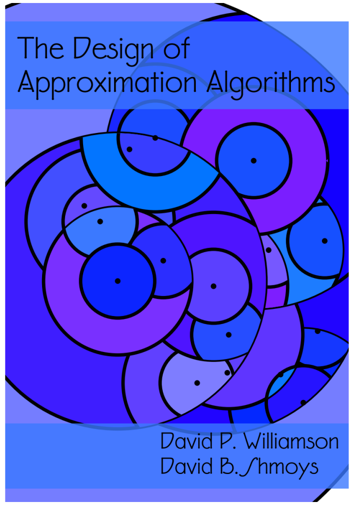
skip

Low-Diameter Decomposition!

Brief history of LDD

Undirected LDD: Many variants studied in many settings

e.g. [Awe85, AGLP89, AP92, ABCP92, LS93, Bar96, BGK+14, MPX13, PRS+18, FG19, CZ20, BPW20, FGdV21].



Highlights

- Distributed network synchronization [Awerbuch'85]
- Probabilistic tree embedding [Bartal'96]. Many lecture notes/books.

Directed LDD: Based on ball-carving technique [Linial-Saks'93,Bartal'96]

- Inefficient version in BGW'20.
- Only known applications: Dynamic directed SSSP [BGW'20], Negative SSSP.
- Open: More applications?

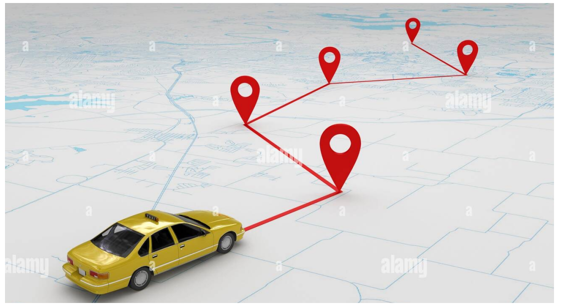
Lecture note:
https://bit.ly/3elW64i

Using LDD to solve negative SSSP

Need:

- Basics: Price/potential function, solving negative SSSP on DAG
- Solving negative SSSP fast when there are not “many” negative weights
    - Fun exercise – by combining Bellman-Ford and Dijkstra
- One crucial trick (see our talks on youtube)

Open



Electric car problem

- Battery can keep charge <B. Charge = 0 → car dies
- Lose charge on some roads, gain on others
- Min charge to go from s to t = ?

Negative SSSP:

- Strongly polynomial better than O(mn) [Bellman-Ford]?
- Deterministic algorithms

Broader problems:

- Alternative algorithms for problems solvable by min-cost flow?
  - Simple - Implementable/teachable?
- Beyond min-cost flow: non-bipartite matching, directed cut, vertex cut, disjoint spanning tree?