

A Combinatorial Algorithm
for All-Pairs Shortest Paths Problem
in Directed Vertex-Weighted Graphs
with Applications to Disc Graphs

Andrzej Lingas Dzmitry Sledneu

Lund University

All-Pairs Shortest Paths Problem

Problem

Given directed weighted graph $G = (V, E, \omega)$, $|V| = n$, $|E| = m$, $\omega : E \rightarrow [-\infty, +\infty]$, compute the *shortest distances* for *all pairs* of vertices.

Known Results

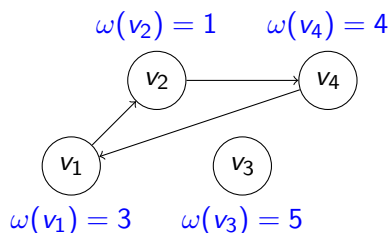
- ▶ Floyd, Warshall (1962): $O(n^3)$
- ▶ Johnson (1977): $O(nm + n^2 \log n)$
- ▶ ...
- ▶ Chan (2007): $O(n^3 \log^3 \log n / \log^2 n)$

Is there a **truly subcubic** ($O(n^{3-\epsilon})$ time) algorithm?

For some **special cases** there are.

Vertex-Weighted Graphs

Special case of edge-weighted graphs



$$G = (V, E, \omega),$$
$$\omega : |V| \rightarrow [-\infty, +\infty]$$

- ▶ Chan (2007): $O(n^{2.844})$
- ▶ Yuster (2009): $O(n^{2.842})$

Fast matrix multiplication, huge hidden constant.

Is there a truly subcubic **combinatorial** algorithm?

For some **special graphs** there are.

Highly Clustered Graphs

A – adjacency matrix of G .

$H(A)$ – complete weighted graph on the rows of A ,

$\omega(\text{row}_i, \text{row}_j) = \text{Hamming distance between } \text{row}_i \text{ and } \text{row}_j$
(n -dimensional metric space).

Definition

G is **highly clustered** if $MST(H(A)) = O(n^{2-\varepsilon})$,

where $MST(H(A))$ is a cost of the min spanning tree for $H(A)$.

Fact (Indyk (2000))

A $\Theta(\log n)$ -approximate minimum spanning tree for $H(A)$ can be computed by a Monte Carlo algorithm in time $\tilde{O}(n^2)$.

Related Problem: Boolean Matrix Multiplication

Special case of APSP – undirected, unweighted graph.

- ▶ Coopersmith, Winograd (1990): $O(n^{2.376})$
(fast matrix multiplication)
- ▶ Bansal, Williams (2009): $O(n^3 \log^2 \log n / \log^{9/4} n)$
(combinatorial)
- ▶ Björklund, Lingas (2001): $\tilde{O}(cn + n^2)$,
where $c = \min\{MST(H(A)), MST(H(B^T))\}$
(combinatorial, truly subcubic for highly clustered graphs)

Björklund et al. algorithm cannot be used for transitive closure, since highly clusterness goes away in power graphs.

Mixed Matrix Products

Definition

Matrix $C \in M_n([-\infty, +\infty])$ is a **mixed right product** of $A \in M_n([-\infty, +\infty])$ and $B \in M_n(\{0, 1\})$,

$$\text{if } C[i, j] = \begin{cases} \min_{1 \leq k \leq n} \{A[i, k] \mid B[k, j] = 1\}, & \exists k \text{ s.t. } B[k, j] = 1 \\ +\infty, & \textit{otherwise.} \end{cases}$$

Example

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, B = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, C = \begin{pmatrix} +\infty & 1 \\ +\infty & 3 \end{pmatrix}$$

Mixed Matrix Products

Definition

Matrix $C \in M_n([-\infty, +\infty])$ is a **mixed left product** of $B \in M_n(\{0, 1\})$ and $A \in M_n([-\infty, +\infty])$,

$$\text{if } C[i, j] = \begin{cases} \min_{1 \leq k \leq n} \{A[k, j] \mid B[i, k] = 1\}, & \exists k \text{ s.t. } B[i, k] = 1 \\ +\infty, & \textit{otherwise.} \end{cases}$$

Example

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, B = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, C = \begin{pmatrix} 3 & 4 \\ 3 & 4 \end{pmatrix}$$

Fast Computation of Right Mixed Matrix Product

1. Compute the $O(\log n)$ -approximate MST of $H(B^T)$.
Fix some traversal $(col_1, col_2, \dots, col_n)$ of the tree.
2. For $j = 1, \dots, n$ compute

$$Ins[col_j] = \begin{cases} \{k \mid B[k, col_j] = 1\}, & k = 1, \\ \{k \mid B[k, col_j] - B[k, col_{j-1}] = 1\}, & k > 1 \end{cases}$$
$$Del[col_j] = \begin{cases} \emptyset, & k = 1, \\ \{k \mid B[k, col_{j-1}] - B[k, col_j] = 1\}, & k > 1 \end{cases}$$

Note: $|Del[col_j]| + |Ins[col_j]|$ equals to the Hamming distance between col_j and col_{j-1} .

Fast Computation of Right Mixed Product

3. For $row = 1, \dots, n$

3.1 *Heap.Init()*

3.2 For $j = 1, \dots, n$

For all $k \in Ins[col_j]$ do *Heap.Insert*($A[i, k]$)

For all $k \in Del[col_j]$ do *Heap.Delete*($A[i, k]$)

$$C[row, col_j] = \begin{cases} +\infty, & \text{Heap.IsEmpty() = true,} \\ \text{Heap.Min()}, & \text{otherwise.} \end{cases}$$

Running Time

Lemma

Algorithm runs in time $\tilde{O}(n^2 + nMST(H(B^T)))$.

Proof.

Step 1 (MST computation) – $\tilde{O}(n^2)$.

Step 2 (*Ins, Del* computation) – $O(n^2)$.

Step 3 (product computation) – $\tilde{O}(nMST(H(B^T)))$.



APSP for Vertex-Weighted Graphs

$G = (V, E, \omega)$, $|V| = n$, $\omega : V \rightarrow [0, +\infty]$, $A \in M_n(\{0, 1\})$ – adjacency matrix.

Let $D \in M_n([0, +\infty])$, where $D(v, u)$ is a shortest distance between v and u .

Let $D^{(i)} \in M_n([0, +\infty])$, where $D^{(i)}(v, u)$ is a shortest distance between v and u on paths of at most i edges.

$$D^{(0)}(v, u) = \begin{cases} 0, & v = u, \\ +\infty, & \textit{otherwise}, \end{cases}$$
$$D^{(1)}(v, u) = \begin{cases} \omega(v) + \omega(u), & (v, u) \in E, \\ +\infty, & \textit{otherwise} \end{cases}$$

Reduction to Mixed Matrix Product

Lemma

$D^{(i+1)}$ can be computed from $D^{(i)}$ and right mixed product of $D^{(i)}$ and A (left mixed product of A and $D^{(i)}$) in time $O(n^2)$.

Proof.

$$D^{(i+1)}[v, u] = \min\{\{D^{(i)}[v, u]\} \cup \{\min_{1 \leq k \leq n} \{D^{(i)}[v, k] + \omega(u) \mid A[k, u] = 1\}\}\}$$



Hitting Set

A set Q – of all q -vertex shortest paths can be constructed by backtracking from $D^{(0)}, \dots, D^{(q-1)}$ in time $O(qn^2)$.

Fact (Hitting set greedy heuristic)

*Given Q – a collection of $O(n^2)$ paths, where each path consists of exactly q vertices, we can find a subset B_Q of size $O((n/q) \log n)$ called **bridging set** that hits all paths in Q and we can do it in time $O(qn^2)$.*

Algorithm for APSP

1. Compute $D^{(0)}, \dots, D^{(q-1)}$.
2. Compute Q – a set of all q -vertex shortest paths.
3. Compute B_Q – $O((n/q) \log n)$ bridging set for Q .
4. Run Dijkstra's algorithm for all vertices from B_Q .
5. Run Dijkstra's algorithm for all from B_Q in reversed graph.
6. For all pairs of vertices (v, u) , $v \notin B_Q$, $u \notin B_Q$ set

$$D[v, u] = \min\{D^{(q-1)}[v, u], \min_{k \in B_Q} \{D[v, k] + D[k, u] - \omega(k)\}\}.$$

Running Time

Theorem

All-pairs shortest paths for a directed graph with nonnegative vertex weights can be computed by a combinatorial randomized algorithm in time $\tilde{O}(n^2\sqrt{n+c})$, where $c = \min\{MST(H(A)), MST(H(A^T))\}$.

Proof.

Step 1 ($D^{(0)}, \dots, D^{(q-1)}$ computation) – $\tilde{O}(q(cn + n^2))$.

Steps 2-3 (Q and B_Q computation) – $O(qn^2)$.

Steps 4-5 (Dijkstra's algorithm) – $\tilde{O}((n/q)n^2)$.

Step 6 – $\tilde{O}((n/q)n^2)$.

Overall – $\tilde{O}(q(cn + n^2))$, set $q = \sqrt{\frac{n^3}{cn+n^2}}$.



Uniform Disk Graphs of Bounded Density

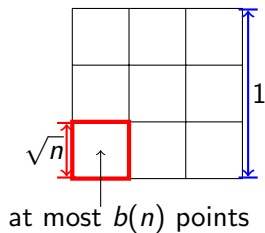


Figure: $b(n)$ -dense point set

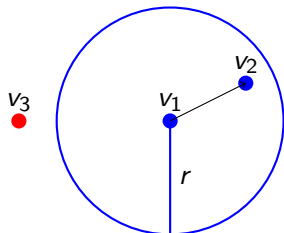


Figure: $(v_1, v_2) \in E, (v_1, v_3) \notin E$

APSP for Vertex-Weighted Uniform Disk Graphs

Theorem

The APSP problem for a vertex-weighted uniform disk graph induced by a $b(n)$ -dense point set can be solved by a combinatorial algorithm in time

$$\tilde{O}(\sqrt{r}n^{2.75}\sqrt{b(n)}).$$

Corollary

For vertex-weighted uniform disk graphs of unit radius induced by $O(1)$ -dense point set APSP can be solved in time

$$\tilde{O}(n^{2.75}).$$

Open Questions

- ▶ More graph classes with property

$$c = \min\{MST(H(A)), MST(H(A^T))\} = O(n^{3-\varepsilon}).$$

- ▶ Generalization for edge-weighted graphs.

Thank You!

Questions?