PRINCETON UNIV. F'24 COS 597B: RECENT ADVANCES IN GRAPH ALGORITHMS

Lecture 1: Shortest Paths in Near-Linear Time [BNW22]

Lecturer: *Huacheng Yu* Scribe: *Frederick Qiu*

1 Problem Setting

The Shortest Paths Problem. Let $G = (V, E)$ be a digraph with (possibly negative) integer edge weights w. Let $n = |V|$ and $m = |E|$, and let W be the magnitude of the most negative edge weight, i.e., $w(e) \geq -W$ for all $e \in E$. Fix a source vertex s.

We want to output a shortest-path tree from s, or output that there exists a negative weight cycle.

Theorem 1. *There exists a randomized algorithm which either detects a negative weight cycle or outputs the shortest-path tree from s in $\tilde{O}(m \log W)$ time.*[1]

For simplicity, we will assume the graph is reach-able from s and that there are no negative cycles, meaning we want to output, for all $v \in V$, the value $\text{dist}_{G,w}(s, v)$.

2 Preliminaries

Recall that *Bellman-Ford* is an algorithm which solves negative-weight shortest-paths in $O(m\ell)$ time, where ℓ is the largest number of edges in a shortest path.

Also recall that *Dijkstra's algorithm* is an algorithm which solves *non*-negative-weight shortest paths in $\tilde{O}(m)$ time.

2.1 Price Functions

Definition 1 (Price Function). *Let $\phi : V \to \mathbb{R}$ be a* price function *on the vertices. Then we define the weights w_ϕ by $w_\phi(u, v) := w(u, v) + \phi(u) - \phi(v)$.*

Observation 1. *For any price function ϕ and vertices $u, v \in V$, the shortest path from u to v with respect to weights w is the same as the shortest path from u to v with respect to weights w_ϕ.*

Proof. Consider any path $v_1 \to \cdots \to v_\ell$. The length of this path with respect to w_ϕ is

$$\sum_{i=1}^{\ell-1} \Big(w(v_i, v_{i+1}) + \phi(v_i) - \phi(v_{i+1}) \Big) \quad = \quad \phi(v_1) + \Big(\sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) \Big) - \phi(v_\ell) \ ,$$

where the summation on RHS is just the length with respect to w. Therefore, the length of any path from u to v with respect to w_ϕ is just the length with respect to w, plus the constant $\phi(u) - \phi(v)$, meaning the shortest path between u and v is the same with respect to w and w_ϕ. □

[1] $\tilde{O}$ hides polylog factors.

Claim 2. *There always exists ϕ such that $w_\phi(e) \geq 0$ for all $e \in E$.*

Proof. Choose $\phi(\cdot) = \text{dist}_{G,w}(s', \cdot)$ for any $s' \in V$. Then for any edge (u, v), we have

$$w_\phi(u, v) \quad = \quad w(u, v) + \text{dist}_{G,w}(s, u) - \text{dist}_{G,w}(s, v) \enspace .$$

Since a path from s to v is the path from s to u plus the edge (u, v), we must have $w(u, v) + \text{dist}_{G,w}(s, u) \geq \text{dist}_{G,w}(s, v)$, the length of the shortest path from s to v. Thus, $w_\phi(u, v) \geq 0$. □

3 Proof Sketch of Main Theorem

The crux of the proof lies in the following claim:

Proposition 3. *There exists a subroutine running in $\tilde{O}(m)$ time which finds a price function ϕ such that $w_\phi(e) \geq -W/2$ for all $e \in E$.*

The idea is then to run this subroutine on the problem instance $O(\log(nW))$ times in succession to ensure that the resulting edge weights are all $\geq -1/n$. We then round the remaining negative edge weights to 0 and run Dijkstra's algorithm on the resulting non-negative-weight graph.

Claim 4. *Suppose all path lengths are integers. If all edge weights are $\geq -1/n$, then rounding all negative-weight edge weights to* 0 *results in a valid shortest-path tree with respect to the original weights.*

Proof. Because no path can contain n or more edges, the difference in distance of a path before and after rounding is $\leq (n-1)/n < 1$. As all path lengths were integers before rounding, no shortest path can become longer than a non-shortest path after rounding. □

Combining this and the fact that price functions do not change shortest paths, the output of Dijkstra's algorithm on the modified edge weights would be a valid shortest-path tree for the original edge weights. It therefore remains to prove Theorem 3.

3.1 Low Diameter Decomposition

Our subroutine will make use of the following concept.

Definition 2 (Low Diameter Decomposition)**.** *For a digraph $H = (V, E)$ with weights w and a parameter $D > 0$, a* low diameter decomposition *(LDD) is a random variable $E^{rem} \subseteq E$ satisfying:*

1. *All strongly connected components (SCC) of $H \setminus E^{rem}$ have* weak diameter *at most D, meaning for all u, v in the same SCC, the shortest path from u to v has length at most D.*
2. $\Pr[e \in E^{rem}] \leq \tilde{O}(\max\{0, w(e)\}/D + n^{-10})$ *for all $e \in E$. Note that these events do not need to be independent.*[2]

[2]In fact, it is impossible in some cases for these events to be independent. Consider a graph with large cliques $V_1, \ldots, V_{2D}$ with edge weights 0. Between V_i and V_{i+1} create a complete bipartite graph with edge weights 1. Then if each edge between cliques is deleted independently w.p. $1/D$, we have w.h.p. that edges will exist between each clique, meaning the entire graph will be one SCC with diameter more than D.

Example 1. *Suppose we have a line graph. An LDD with parameter D could generate E^{rem} by randomly adding one of the first D edges, then adding every D^{th} edge afterwards.*

Lemma 5. *LDD can be solved in $\tilde{O}(m)$ time.*

Proof. See next lecture. $\square$

3.2 The Subroutine (Simpler Version)

Now, let us return to the subroutine. For simplicity of notation, Let $W = 2$.

Define the weights w^{+1} by $w^{+1}(e) = w(e) + 1$. Our approach will be to find a price function ϕ such that $w^{+1}_{\phi}(e) \geq 0$ for all $e \in E$. Then we have that $w_{\phi}(e) \geq -1$ for all $e \in E$, so we will have completed the subroutine.

Next lecture, we'll see how to execute the subroutine in $\tilde{O}(m)$ time. This lecture, we will give a subroutine running in $\tilde{O}(m\sqrt{n})$ which computes ϕ such that $w_{\phi}(e) \geq -1$ for *most* edges $e \in E$.

1. Apply LDD on digraph G with weights w^{+1} and parameter $D = \sqrt{n}$.
2. Order the SCC left to right topologically (ignoring the edges E^{rem}). Denote the edges inside a single SCC by *internal* edges, the edges going from left to right across SCCs by *forward* edges, and the edges going from right to left across SCCs (all such edges are in E^{rem}) by *backward* edges.
3. Attach a dummy source s' connected to every vertex by an edge with weight 0, and set $\phi(v)$ to be the distance from s' to v using only internal edges.
4. Number the SCC from left to right, and for a vertex v in SCC i, set $\phi(v) := \phi(v) - Wi$.

Proposition 6. *The above subroutine can be run in $\tilde{O}(m\sqrt{n})$ time and generates ϕ such that $w_{\phi}(e) \geq -1$ for all internal and forward edges.*

Proof. Notice that step (3) makes all internal edge weights non-negative (with respect to w^{+1}), and step (4) makes all forward edge weights non-negative (with respect to w^{+1}) without altering internal edge weights (because the prices for vertices within an SCC change by the same amount), so the subroutine works as claimed.

Step (1) can be run in $\tilde{O}(m)$ time by Theorem 5. Steps (2) and (4) are efficient book-keeping. Step (3) can be run in $O(m\ell)$ time, where ℓ is the largest number of edges in a shortest path using only internal edges.

Claim 7. *For any $v \in V$, the shortest path from s' to v using only internal edges has at most $\sqrt{n}$ edges.*

Proof. If the shortest path from s' to v is not just the edge (s', v), then it must start with some edge (s', u) for $u \neq v$ in the same SCC. Then the shortest path from u to v using only internal edges must have negative length.

Suppose for contradiction that the shortest path from u to v has at least $\sqrt{n}$ edges. Since u, v are in the same SCC, there also exists a shortest path from v to u with length at most $\sqrt{n}$ (since we took an LDD with parameter $D = \sqrt{n}$). Therefore, there is a cycle (with respect to w^{+1}) from u to v to u with at least $\sqrt{n}$ edges and length $< \sqrt{n}$.

However, this means that with respect to the original weights w, there is a negative cycle from u to v to u, a contradiction. Therefore, the shortest path from u to v has at most $D-1$ edges, so the shortest path from s' to v has at most D edges. $\square$

Thus, $\ell = \sqrt{n}$ and so Step (3) can be run in $O(m\sqrt{n})$ time as desired. $\square$

We will show how to handle the backward edges in the next lecture, using the second property of LDD to show that the number of backward edges in any shortest path is low in expectation, allowing for an efficient combination of Dijkstra's algorithm and Bellman-Ford to be used.

References

[BNW22] Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 600–611. IEEE, 2022.