CS 270: Combinatorial Algorithms and Data Structures Spring 2023

Lecture 1 — January 17, 2023

Prof. Jelani Nelson *Scribe: Altan Haan*

1 Single-Source Shortest Path

In this lecture, we will cover the single-source shortest path problem, with a focus on the recent work by Bernstein, Nanongkai, and Wulff-Nilsen [BNW22]. We begin by covering some mathematical background, before jumping into a sketch of the algorithm.

1.1 General Background

Definition 1.1. A *weighted directed graph* is a tuple $G = (V, E, w)$ where V is a set of vertices, E is a set of directed edges of the form (u, v) where $u, v \in V$, and $w : E \to \mathbb{R}$ gives the edge weights. The *length* of a path $P = (v_0, \ldots, v_r)$ is naturally defined as $w(P) = \sum_{i=1}^{r} w(v_{i-1}, v_i)$. Lastly, we will frequently use $m = |E|$ and $n = |V|$ to denote the number of edges and vertices respectively.

Definition 1.2. The *single-source shortest path (SSSP)* task takes as input a directed weighted graph $G = (V, E, w)$ and a *source vertex* $s \in V$, and outputs a directed tree rooted at s such that for all $v \in V$ the shortest path from $s \to v$ is given by the (unique) path in the tree from $s \to v$.

For the result in [BNW22], we further constrain w such that for all $e \in E$, $w(e) \in \mathbb{Z}$ and $-W \le w(e) \le W$ for some $W \ge 0$.

Remark 1.3. We have seen two algorithms from undergrad algorithms for solving this problem:

- **Dijkstra's algorithm**, which runs in $O(m+n\log n)$ time under the assumption that $w(e) \ge 0$ for all $e \in E$; and
- **Bellman-Ford**, which runs in $O(mn)$ time for unrestricted edge weights (whilst detecting negative weight cycles if any exist).

Definition 1.4. For convenience, we write $\tilde{O}(f)$ in place of $O(f \cdot (\log f)^k)$ for some sufficient k.

1.2 Price Functions and Scaling

Definition 1.5. A *price function* is a function $\varphi : V \to \mathbb{R}$, which defines an associated transformed weight function $w_\varphi(u, v) = w(u, v) + \varphi(u) - \varphi(v)$.

Remark 1.6. For any path $P = (v_0, \ldots, v_r)$ we have $w_\varphi(P) = w(P) + \varphi(v_0) - \varphi(v_r)$, by a simple telescoping argument (the prices of internal vertices cancel out). It follows that $w_\varphi(P) = w(P)$ if P is a cycle, as $v_0 = v_r$.

Remark 1.7. As all paths P from $s \to v$ have $w_\varphi(P) = w(P) + \varphi(s) - \varphi(v)$, the shortest path $P^* = \arg\min_P(w(P) + \varphi(s) - \varphi(v)) = \arg\min_P w(P)$ remains unchanged.

With these observations, we can formulate a new strategy for solving SSSP: if we can find a φ such that $w_\varphi(e) \geq 0$ for all $e \in E$, then we can utilize the much cheaper Dijkstra's algorithm on the transformed weights. We'll call these "good" price functions. But first, we need to show that such φ's do in fact exist.

Claim 1.8. There exists a $\varphi : V \to \mathbb{R}$ such that $w_\varphi(e) \geq 0$ for all $e \in E$, **if and only if** G has no negative cycles.

Proof. The forward direction is simple: supposing we have such a φ, then by Remark 1.6 we know any cycle C satisfies $w(C) = w_\varphi(C) \geq 0$.

For the other direction, assume there are no negative cycles and define $\varphi(v) = d_G(s, v)$ where $d_G(s, v)$ is the length of the shortest path from $s \to v$ in G. Then for any $(u, v) \in E$, we have $w_\varphi(u, v) = w(u, v) + d_G(s, u) - d_G(s, v)$. By the triangle inequality, we also know that $d_G(s, v) \leq d_G(s, u) + w(u, v)$. Together it follows that

$$w_\varphi(u, v) \geq w(u, v) + d_G(s, u) - (d_G(s, u) + w(u, v)) = 0.$$

□

Now that we know φ exists under the standard SSSP assumption, we state and prove a theorem which allows us to take some shortcuts on the way to utilzing Dijkstra's algorithm.

Theorem 1.9 (Goldberg [Gol95])**.** *Suppose we have an algorithm* `solve` *which takes* G *and returns a good* φ *in* $T(m)$ *time, under the assumption that* $w(e) \geq -1$ *for all* $e \in E$*. Then there is an algorithm* `solve`* *that runs in* $O(T(m) \log W)$ *time in the general case of* $w(e) \geq -W$*. Note that* $w(e)$ *is assumed to be integral.*

Proof. Without loss of generality, we round up $W = 2^k$ to the nearest power of 2. We proceed by strong induction on k.

Base case. This follows trivially from just calling `solve`.

Inductive case. Define $\hat{w}(e) = \lceil w(e)/2 \rceil$ and $\hat{G} = (V, E, \hat{w})$. Then, let $\hat{\varphi} = \texttt{solve}^*(\hat{G})$, and define $\varphi' = 2\hat{\varphi}$. Note that $\hat{\varphi}$ is well-defined as $\lceil w(e)/2 \rceil \geq -2^{k-1}$. Furthermore,

$$w(e) \geq 2\lceil w(e)/2 \rceil - 1 = 2\hat{w}(e) - 1.$$

It follows that for any $(u, v) \in E$,

$$\begin{aligned} w_{\varphi'}(u, v) = w(u, v) + \varphi'(u) - \varphi'(v) &\geq 2\hat{w}(u, v) - 1 + 2\hat{\varphi}(u) - 2\hat{\varphi}(v) \\ &= 2(\hat{w}(u, v) + \hat{\varphi}(u) - \hat{\varphi}(v)) - 1 \\ &= 2\hat{w}_{\hat{\varphi}}(u, v) - 1 \\ &\geq -1, \end{aligned}$$

where the last inequality is due to the goodness of $\hat{\varphi}$ with respect to $\hat{w}$.

This shows that $G_{\varphi'}$ (i.e. G with the transformed weight under φ') satisfies the conditions for `solve`, and so the final price function is given by $\varphi = \texttt{solve}(G_{\varphi'}) + \varphi'$ as $G_\varphi = (G_{\varphi'})_{\texttt{solve}(G_{\varphi'})}$. □

1.3 Bernstein–Nanongkai–Wulff-Nilsen (BNWN)

We now sketch out the high-level approach taken by BNWN. First we'll give the formal statement.

Theorem 1.10 (Bernstein–Nanongkai–Wulff-Nilsen)**.** *Let $G = (V, E, w)$ such that $-W \leq w(e) \leq W$ for some $W \geq 0$ and $w(e) \in \mathbb{Z}$ for all $e \in E$, and let $s \in V$ be the source. Then there is an algorithm which computes SSSP in $\tilde{O}(m \log W)$ time.*

We also give some definitions used in the proof sketch.

Definition 1.11 (Weak diameter)**.** Let $G = (V, E, w)$ and $S \subseteq V$ be a strongly connected component. Then the *weak diameter* of S is defined as $\max_{u,v \in S} d_G(u, v)$.

Definition 1.12. For a given source s and target vertex v, we denote the minimum number of negative edges on any shortest path from $s \to v$ by $\eta_G(v)$.

Proof sketch (Theorem 1.10). First, we augment G with a dummy source node σ by defining $G_\sigma = (V_\sigma, E_\sigma, w_\sigma)$, where

$$\begin{aligned} V_\sigma &= V \cup \{\sigma\}, \\ E_\sigma &= E \cup \{(\sigma, v) \mid v \in V\}, \\ w_\sigma(u, v) &= \begin{cases} 0 & \text{if } u = \sigma, \\ w(u, v) & \text{otherwise.} \end{cases} \end{aligned}$$

Note that solving SSSP on G_σ with σ gives the solution for G with s, so for the rest of the sketch we'll just refer to G_σ as G.[1]

We now cover the subroutines utilized by the full algorithm.

$\mathsf{LowDiameterDecomp}(G, D)$. This subroutine takes a graph G *with nonnegative edge weights* and $D \geq 0$ computes a set $E^{\text{rem}} \subseteq E$ such that:

(1) Each strongly connected component (SCC) of $G \setminus E^{\text{rem}}$ has *weak diameter* less than or equal to D. That is, for all u, v in the SCC, $d_G(u, v) \leq D$ (where the distance is taken in the original graph containing E^{rem}).

(2) The probability of an edge being in E^{rem} satisfies

$$P(e \in E^{\text{rem}}) \leq O\left(\frac{w(e) \log^2 n}{D} + n^{-10}\right).$$

This subroutine runs in $\tilde{O}(m)$ time.

$\mathsf{FixDAGEdges}(G, P)$. This subroutine takes in a graph G and vertex partitioning P such that (1) each partitioned subgraph has no negative weight edges and (2) the graph obtained by contracting each partition to a single vertex is a DAG. Under these assumptions, FixDAGEdges outputs a good price function in $O(m + n)$ time.

[1] Assuming $n \ll m$, the time complexity remains unchanged.

ElimNeg(G). This subroutine takes a graph G with constant out-degree and outputs a good price function in $O(\log n \sum_{v \in V}(1 + \eta_G(v)))$ time. We briefly remark that any graph can be transformed into one with constant out-degree by replacing non-constant vertices with zero-weight cycles as depicted below:

ScaleDown(G, Δ, B). The subroutine takes a graph G such that $\eta(G) = \max_{v \in V} \eta_G(v) \leq \Delta$ and $w(e) \geq -2B$ for all $e \in E$, and returns a price function φ such that $\varphi(e) \geq -B$ for all $e \in E$.

Main algorithm. At a high level, ScaleDown invokes LowDiameterDecomp, FixDAGEdges, and ElimNeg. ScaleDown is in turn invoked repeatedly by the main algorithm. Intuitively, LowDiameterDecomp produces partitions with small η values.

```
def main($G = (V, E, w)$):
    $B \leftarrow 2n$ // without loss of generality, round $n$ up to pow of 2
    $\overline{w} \leftarrow B \cdot w$
    $\overline{G} \leftarrow (V, E, \overline{w})$
    $\varphi_0 \leftarrow 0$
    for i from 1 to $\log_2 B$:
        $\psi_i \leftarrow \mathsf{ScaleDown}(\overline{G}_{\varphi_{i-1}}, n, B/2^i)$
        $\varphi_i \leftarrow \varphi_{i-1} + \psi_i$
    for each $e \in E$:
        $w^*(e) \leftarrow \overline{w}_{\varphi_{\log_2 B}}(e) + 1$
    run Dijkstra on $(V, E, w^*)$
```

□

References

[BNW22] Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 600–611, 2022.

[Gol95] Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, 1995.