6.5210 Final Project

Yifan Kang, Alex Zhao

November 2024

§1 Project Proposal

We will focus on the Single-Source Shortest Path (SSSP) problem in undirected graphs with real non-negative edge weights.

Previously, we believed that using Dijkstra's algorithm with Fibonacci heaps to achieve a $O(m + n \log n)$ runtime was optimal. However, in 2023 **?**] discovered a randomized algorithm with a runtime of $O(m\sqrt{\log n \log\log n})$, which is faster than Fibonacci heap Dijkstra in the case where the average degree is smaller than $\sqrt{\log n}$.

The project would examine this new algorithm in detail and compare it with prior approaches, such as Fibonacci heap and binary heap Dijkstra. The focus will be synthesizing insights from the paper and related work, explaining the algorithm's key ideas, and making them more accessible to readers less familiar with these advanced algorithmic techniques.

§2 Overview of the randomized algorithm

The bottleneck of Dijkstra-based algorithms is the priority queue. Therefore, we break the original time bound by adding only a fraction of vertices into the priority queue. Let this subset of vertices be R. Using this subset, we find the shortest path to other vertices by forming disjoint **bundles** based on R. We introduced another concept of Ball(v) as the set of vertices closer to v than any vertices in R. We will later prove the fact that a vertex v will either have its shortest path through its bundle or through some vertex in its ball not too far from what we have previously calculated.

Definition 2.1. For all vertices $v \in V$, we find $b(v) = \text{argmin}_{u \in R} \text{dist}(v, u)$. $b(v)$ is its nearest neighbor in R.

Definition 2.2 (Ball)**.** $\text{Ball}(v) = \{\text{dist}(v, u) < \text{dist}(v, b(v)) | u \in V\}$ is the set of vertices closer than $b(v)$ to v.

See Figure 1.

Definition 2.3 (Bundle)**.** For $u \in R$, $\text{Bundle}(u) = \{b(v) = u | v \in V\}$ is the set of vertices v such that u is their nearest neighbor in R.

See Figure 2.

The algorithm can be divided into two parts: **bundle construction** and **bundle Dijkstra**.

Figure 1: Figure demonstrating example for a ball.

Figure 2: Figure demonstrating example for a bundle.

Algorithm 2.4 (Bundle Construction) — Proceeds as follows:

1. We randomly sample a subset of vertices R.
2. For all vertices $v \in V$, we find $b(v)$ by running Dijkstra from v until we get to a point in R.
3. Let $\text{Ball}(v)$ be the set of vertices closer than $b(v)$ to v. In the previous Dijkstra, we can get $\text{dist}(v, w)$ for all $w \in \text{Ball}(v) \cup \{b(v)\}$.

Definition 2.5 (Relax)**.** In our shortest path algorithm, relaxing a vertex v through u means update $d(v)$ by $\min(d(v), d(u) + w_{u,v})$. This definition can be extended to relaxing a vertex v through $u_0, u_1, \cdots u_k = v$, where we will update $d(v)$ by $\min(d(v), d(u_0) + \sum_{i=1}^{k} w_{u_{i-1},u_i})$. We use $\text{dist}(u, v)$ in place of $w_{u,v}$ if the value is known.

Algorithm 2.6 (Bundle Dijkstra) — Initially, we set $d(s) = 0$ and $d(v) = \infty$ for all $v \in V \setminus \{s\}$, and insert all $v \in R$ to a Fibonacci heap. When we pop a vertex u from the heap, we proceed as follows:

1.
   - For all $v \in \text{Bundle}(u)$, relax v by u.
   - For all $y \in \text{Ball}(v)$, relax v by y.
   - For all $z_2 \in \text{Ball}(v) \cup \{v\}$ and $z_1 \in N(z_2)$, relax v by z_1, z_2.
2. For all $x \in \text{Bundle}(u)$, update $y \in N(x)$ and $z_1 \in \text{Ball}(y)$. We relax y by x and z_1 by x, y.
3. When we update $v \notin R$, we relax $b(v)$ by v.

§2.1 Pseudocode

§3 Intuition

If you want a formal proof, skip to **??**.

We hold two propositions in parallel:

Proposition 3.1 (Distance Correctness)

The ith vertex drawn from R is the ith closest vertex to s, and its distance function is correct when drawn.

Figure 3: All nodes on the shortest path belong to bundles that were previously processed.

Figure 4: If x lies on the shortest path from $s \to y$ and if u_i has been processed but u_* has not, then $\text{dist}(s, u_*) \geq \text{dist}(s, u_i)$. Then $\text{dist}(x, y) \leq \text{dist}(x, u_*) + \text{dist}(y, u_i)$. Hence their balls overlap.

Proposition 3.2 (Bundle Correctness)

After Step 1, the distance function is correct for all $v \in \text{Bundle}(u)$ for all $u \in R$ already drawn from the priority queue.

Both for a base case and by way of example, we prove this for the 0th iteration.

Example 3.3 (Base case)

On the 0th iteration, we draw the source node s from the PQ.

To show **??**, s is trivially the closest node to s, and its distance function is correct.

To show **??**, note that for all $v \in \text{Bundle}(s)$ we already computed $\text{dist}(v, s)$ in the setup Dijkstra, so $d(v)$ is set to the correct value $\text{dist}(v, s)$.

Now we show **??** in generality. See **??**.
Now we show **??** in generality.

$$\begin{aligned} \text{dist}(s, x) + \text{dist}(x, u_*) + \text{dist}(u_i, y) &\geq \text{dist}(s, u_*) + \text{dist}(u_i, y) \\ &\geq \text{dist}(s, u_i) + \text{dist}(u_i, y) \\ &\geq \text{dist}(s, y) \\ &= \text{dist}(s, x) + \text{dist}(x, y). \end{aligned}$$

§4 Formal Proof

Proposition 4.1

test

Proof. test □

§5 Time complexity analysis

To ensure the time complexity of *Bundle Dijkstra*, we need to first transform the graph into a graph with small degrees. For a given graph G, we can construct G' in the following way:

Figure 5: If u_i has been processed but u_* has not, then the $\text{dist}(x, y) <= \text{dist}(x, u_*) + \text{dist}(y, u_i)$.

- For each vertex v and $w \in N(v)$, construct the vertex x_{vw}. Add 0-weight edges between these $|N(v)|$ vertices so that they form a cycle.
- For every edge $(u, v) \in G$, add an edge between x_{uv} and x_{vu} with weight w_{uv}.

We can see that $\text{dist}_{G'}(x_{uu'}, x_{vv'}) = \text{dist}_G(u, v)$ for all $u, v \in G$ and $u' \in N(u), v' \in N(v)$. Moreover, in this construction we have $O(m)$ vertices and each of them has degree at most 3.

§5.1 Bundle Construction

Without loss of generality, we assume that Dijkstra's algorithm breaks ties deterministically. Let S_v be the set of vertices we extracted when we run Dijkstra on v. When we extract a vertex u from the heap, the probability that $u \in R$ is $1/k$. As our Dijkstra terminates when we find such u, we have $\mathbb{E}[|S_v|] = k$. Moreover, by linearity of expectation we have:

$$\mathbb{E}[|S_v|^2] = \frac{1}{k} \cdot 1^2 + (1 - \frac{1}{k})(1^2 + 2\mathbb{E}[|S_v|] + \mathbb{E}[|S_v|^2])$$

Simplify and we get $\mathbb{E}[|S_v|^2] = 2k^2 - k = O(k^2)$. Let $f(x) = \sqrt{x} \log x$, we have $f''(x) = -\frac{\log(x)}{4x^{3/2}} \leq 0$ for $x \geq 1$. Then by Jensen's inequality,

$$\mathbb{E}[|S_v| \log |S_v|] \leq f(\mathbb{E}[|S_v|^2]) = O(k \log k)$$

Thus, the total runtime of all Dijkstra is

$$O(\sum_{v \in V \setminus R} \mathbb{E}[|S_v| \log |S_v|]) = O(mk \log k)$$

in expectation. The same bound applies to $\text{Ball}(v)$ since it is a subset of S_v.

§5.2 Bundle Dijkstra

The total runtime of extract-min of the Fibonacci heap is $O(|R| \log n)$. Every vertex $v \notin R$ only appears once in step 1 as v, in step 2 as x when $v \in \text{Bundle}(u)$. Since every vertex in the graph has a constant degree, it appears a constant number of times as y, a neighbor of x, in step 2. Similarly, one can argue that we have $O(|\text{Ball}(v)|)$ (z_1, z_2) pairs and $O(|\text{Ball}(v)|)$ z_1 in step 2. The overall runtime of the algorithm is thus

$$O(|R| \log n + \sum_{v \in V \setminus R} |\text{Ball}(v)|) = O(\frac{m}{k} \log k + mk \log k)$$

in expectation. Take $k = \sqrt{\frac{\log n}{\log \log n}}$, and we get:

$$O(\frac{m}{k} \log k + mk \log k) = O(m\sqrt{\frac{\log n}{\log \log n}} \log \log n) = O(m\sqrt{\log n \log \log n})$$

References

[] Ran Duan, Jiayi Mao, Xinkai Shu, and Longhui Yin. A Randomized Algorithm for Single-Source Shortest Path on Undirected Real-Weighted Graphs . In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 484–492, Los Alamitos, CA, USA, November 2023. IEEE Computer Society. URL `https://doi.ieeecomputersociety.org/10.1109/FOCS57990.2023.00035`.