Approximate All Pairs Shortest Paths [DHZ00]

All pairs shortest paths (APSP) is one of the most classical problems in algorithms. For unweighted graphs, the best time complexity for this problem is $O(M(n))$ where $M(n)$ is the time complexity of multiplying two $n \times n$ matrices. Usually $M(n)$ is denoted by $n^{\omega}$ where $\omega = 2.37$ (though it might get better tomorrow!).

So far, we mainly used approximation on shortest paths to improve the space of subgraphs, data structures, labels and routing tables. In this class, we will show how settling for approximate distances can considerably improve the run-time of computing these distances. We present an algorithm by Dor, Halperin and Zwick [DHZ00] that essentially solves APSP in nearly optimal time of $\tilde{O}(n^2)$, up to an additive distortion of $O(\log n)$ in the computed distances.

**Theorem 12.1** For every $k \geq 2$, given a $n$-vertex unweighted graph $G = (V,E)$, there exists an algorithm $\text{APSP}_k$ that computes a matrix $\{\delta(u,v)\}_{u,v}$ where $\text{dist}(u,v) \leq \delta(u,v) \leq \text{dist}(u,v,G) + 2(k-1)$, with time complexity of $\tilde{O}(n^{2+1/k})$.

We use the following two facts.

**Fact 12.2** (Single source distances) Consider an $n$-vertex weighted directed graph $G$ and let $s \in V(G)$ be an input vertex, which we call source. Dijkstra algorithm computes the shortest path tree with respect to $s$ (along with all $\{s\} \times V$ distances) in $O(m + n \log n)$ time.

Let $\Gamma(v)$ be the neighbors of vertex $v$ in $G$. Recall that a hitting set $S \subseteq V$ satisfies that $S \cap \Gamma(v) \neq \emptyset$ for every $v \in V'$. As discussed in Lecture 2, if the subset $V'$ consists of vertices with high-degree, then there exists a small hitting set which can be computed in linear time.

**Fact 12.3** (Small Hitting Sets) For every $\Delta \geq 1$, computing a hitting set $S$ of size $O(n \log n / \Delta)$ that hits all vertices with degree at least $\Delta$ can be done in $O(m)$ time.

Given a stretch parameter $k$, algorithm $\text{APSP}_k$ has $k$ phases. In each phase $i$, it computes sourcewise shortest path distances from a subset of sources $S_i$ in a graph $G_i$ (in fact, for each $s \in S_i$, we will define a different subgraph $G_i(s)$). As $i$ gets larger, the number of sources increases and the size of the subgraph $G_i$ decreases. Overall, we will be in a situation where $|S_i| \cdot |G_i| = O(n^{2+1/k})$ which is the time complexity for computing $S_i \times V$ distances by Fact 12.2, for every $i$.

We need some definitions. Consider the decreasing sequence $\Delta_0 \geq \Delta_1 \geq \Delta_2 \ldots \geq \Delta_{k-1} \geq \Delta_k$ of degree thresholds, where $\Delta_0 = n$, $\Delta_k = 1$ and $\Delta_i = n^{1/i/k}$ for $i \in \{1, \ldots, k-1\}$. Denote by $V_i = \{v \mid \deg(v,G) \geq \Delta_i\}$ the vertices with degree at least $\Delta_i$, and let $S_i$ be the hitting-set for all the vertices in $V_i$. By Fact 12.3, $|S_i| = O(n \log n / \Delta_i) = O(n^{1/k} \cdot \log n)$. For every $i \geq 1$, let $E_i = \{(u,v) \in E \mid \min\{\deg(u),\deg(v)\} \leq \Delta_{i-1}\}$. Thus, $E_1 = E(G)$ and $|E_i| \leq n \cdot \Delta_{i-1}$ for every $i$. Finally, for every $i \geq 1$, and $u \in V_i$, let $s_i(u)$ be an arbitrary vertex in $S_i \cap \Gamma(u)$, define $E_i^* = \{(u,s_i(u)) \mid u \in V_i\}$ and $E^* = \bigcup_i E_i^*$. We are now ready to describe algorithm $\text{APSP}_k$. See Fig. 12.1 for a complete description of the algorithm.

The weights $\tilde{W}$ of the edges in $G_i(s)$ are defined as follows: $\tilde{W}(s,v) = \delta(s,v)$ where $\delta(s,v)$ is the current estimate for the distance between $s$ and $v$. For any other edge $e = (u,v) \in G_i(s)$, it holds that $e \in G$ and $\tilde{W}(u,v) = 1$.

**Time complexity.** Computing $\bigcup_{i=1}^k S_i$ and the edges $E^*$ can be done in time $O(k \cdot n)$, by Fact 12.3. We next analyze the running time of phase $i$. The cardinality of the hitting set is bounded by $|S_i| = \ldots$
Algorithm APSP\(_k(G)\)

1. For every \(u, v \in V\), set \(\delta(u, v) = 1\) if \((u, v) \in E\), and \(\delta(u, v) = \infty\) otherwise.

2. For \(i = 1\) to \(k\), do:
   - For every \(s \in S_i\), compute the \(\{s\} \times V\) distances in the graph:
     \[G_i(s) = (V, E_i \cup E^* \cup (\{s\} \times V), W)\].
   - Update the entries of \(\delta(s, v)\) for every \(s, v \in S_i \times V\).

Fig. 12.1: APSP algorithm with additive approximation of \(2(k - 1)\)

\[O(n \log n / \Delta_i).\] The size of the subgraph \(G_i(s)\) for every \(s \in S_i\) is dominated by the size of the \(E_i\) edges where \(|E_i| = O(n \cdot \Delta_{i-1})\). Thus, by Fact 12.2, we get that all \(S_i \times V\) distances are computed in time \(\tilde{O}(|S_i| \cdot |E_i|) = \tilde{O}(n^{2+1/k})\).

**Stretch analysis.** Let \(\delta_i(u, v)\) be the estimate \(\delta(u, v)\) of the \(u-v\) distance after running Dijkstra from each \(s \in S_i\) in phase \(i\). We prove by induction on \(i\) that:
\[
\delta_i(u, v) \leq \text{dist}(u, v, G) + 2(i - 1), \forall u \in S_i, v \in V.
\]

For the base of the induction, consider \(i = 1\). Since \(E_1 = G\), the claim holds immediately. Assume that the claim holds up to phase \(i - 1\), and consider phase \(i\). Let \(\pi(u, v)\) be the \(u-v\) shortest path in \(G\).

**Case 1:** all edges on \(\pi(u, v)\) are in \(E_i\). This case is easy since the Dijkstra is computed in a graph that contains the shortest path edges.

**Case 2:** There exists \(w \in V_{i-1}\) on \(\pi(u, v)\). In the complementarity case, there must be a high degree vertex on the path \(\pi(u, v)\). Let \(w \in V_{i-1}\) be the closest vertex in \(V_{i-1}\) to \(v\) on the path \(\pi(u, v)\). Let \(w' = s_{i-1}(w)\), be the neighbor of \(w\) in \(S_{i-1}\). See Fig. 12.2 for an illustration. By induction assumption for \(i - 1\), we have that:
\[
\delta_{i-1}(u, w') \leq \text{dist}(u, w', G) + 2(i - 2) \leq \text{dist}(u, w, G) + 2i - 3,
\]

where the last inequality follows by the triangle inequality. We therefore have:
\[
\delta_i(u, v) \leq \delta_{i-1}(u, w') + 1 + \text{dist}(w, v, G) \leq \text{dist}(u, w, G) + 2i - 2 + \text{dist}(w, v, G) \leq \text{dist}(u, v, G) + 2(i - 1).
\]

Fig. 12.2: An illustration of case (2).

**References**