1 +2-Approximate APSP in $\tilde{O}(n^{7/3})$ time

Last time we discussed an algorithm that in $\tilde{O}(n^{3/2})$ time obtained additive +2 approximate All Pairs Shortest Paths in unweighted undirected graphs. Today we will present a faster algorithm that achieves the same error guarantee but runs in $\tilde{O}(n^{7/3})$ time. Notably, this running time is faster than the current known bounds on $n \times n$ matrix multiplication! (Note: you can solve APSP exactly in $O(n^\omega)$ time where we currently only know that $\omega < 2.373$.)

Theorem 1.1 (Dor, Halperin, Zwick)**.** *There is an algorithm which, given an n-vertex undirected unweighted graph, runs in time $\tilde{O}(n^{7/3})$ and computes estimates $d'(u,v)$ satisfying $d(u,v) \leq d'(u,v) \leq d(u,v)+2$ for all $u,v \in V$.*

The algorithm sets two thresholds $n^{1/3}$ and $n^{2/3}$ and considers all nodes of degree $< n^{1/3}$ low degree, those of degree $> n^{2/3}$ high degree and the rest as medium degree. One also takes two hitting sets: S that hits the neighborhoods of the high degree nodes and T that hits those of the medium degree nodes. S can be handled just as in the previous lecture: run BFS from every node in S and then for every u, v whose shortest path contains a high degree node, $\min_{s\in S} d(u,s) + d(s,v)$ is a +2-approximate distance. The rest of the algorithm takes care of the shortest paths that have no high degree nodes.

First, one creates a graph where there are no edges between any two high degree nodes. This graph is sparse - it only has $O(n \cdot n^{2/3})$ edges, so that one can afford to compute BFS from each node in T in it. This computes for every $t \in T, v \in V$ the length of a shortest $t-v$ path among these that contain no high degree node. Now, if we could compute $\min_{t\in T} d(u,t) + d(t,v)$ efficiently, we would be done with all paths that have a medium degree node, and those that don't can be handled by doing APSP in a graph on $n^{4/3}$ edges in $O(n^{7/3})$ time. However, computing $\min_{t\in T} d(u,t) + d(t,v)$ seems to require $\Omega(n^2|T|)$ time which is too costly. To remedy this, one computes shortest paths in n different graphs. See Algorithm 1.

Proof of Thm. 1.1. We first show the approximation bounds $d(u,v) \leq d'(u,v) \leq d(u,v) + 2$. As before, the lower bound is trivial. For the upper bound, from our discussion above, it suffices to consider the case that the shortest path γ joining vertices $x, y \in V$ involves only vertices of $L \cup M$, but not L alone.

In this case, let z be the vertex in $\gamma \cap M$ which is furthest from x, and let t be the chosen vertex of $T \cap N(z)$ for which (z,t) was added to E_x. By the triangle inequality for d_x we have $d_x(x,y) \leq d_x(x,t) + d_x(t,z) + d_x(z,y)$; the middle term $d_x(t,z)$ is simply 1. The path from z to y involves only L-incident edges, so $d_x(z,y) = d(z,y)$. Lastly $d_x(x,t) \leq d_{\text{mid}}(x,t) \leq d_{\text{mid}}(x,z) + d_{\text{mid}}(z,t)$ (by the triangle inequality for d_{mid}), and this equals $d(x,z) + d(z,t) = d(x,z) + 1$ by our assumption that γ involves only $L \cup M$. Combining these inequalities, $d_x(x,y) \leq d_x(x,t) + d_x(t,z) + d_x(z,y) \leq d_{\text{mid}}(x,t) + 1 + d(z,y) \leq d(x,z) + d(z,y) + 2 = d(x,y) + 2$, where the last equality holds because P was assumed to be a shortest path.

We now check that the algorithm has the claimed runtime: Computation of S takes time $O(n^{5/3}\log n)$; computation of T takes time $O(n^{4/3}\log n)$. Running BFS from all $s \in S$ takes time $O(|S|n^2) = O(n^{7/3}\log n)$. The graph G_{mid} has $|E_{\text{mid}}| = O(n^{5/3})$ edges, so running BFS in G_{mid} from all $t \in T$ takes time $O(|T||E_{\text{mid}}|) = O(n^{7/3}\log n)$. Forming $K_\circ$ takes time $O(|L|n^{1/3} + |S||V|) = O(n^{4/3}\log n)$. For each $x \in V$, forming K_x takes time $O(|T| + |M|) = O(n)$ if we do not re-compute $K_\circ$ (although we can afford to). Then $|E_x| = O(|L|n^{1/3} + |S|n + |T| + |M|) = O(n^{4/3}\log n)$, so running Dijkstra from x in K_x takes time $O(n^{4/3}\log n)$. Running over $x \in V$ gives final runtime $O(n^{7/3}\log n)$ as claimed. □

Algorithm 1: AAPSP-DHZ($G = (V, E)$)

$L \leftarrow \{v \in V : |N(v)| \leq n^{1/3}\}$;$H \leftarrow \{v \in V : |N(v)| > n^{2/3}\}$; $M \leftarrow V \setminus (L \cup H)$;
$S \leftarrow$ hitting set for $(N(v) : v \in H)$, $|S| = O(n^{1/3} \log n)$;
$T \leftarrow$ hitting set for $(N(v) : v \in M)$, $|T| = O(n^{2/3} \log n)$;
foreach $s \in S$ **do**
  BFS(s) to compute $d(s, v)$ for each $v \in V$;
$G_{\text{mid}} \leftarrow G$ with only edges incident to $L \cup M$;
foreach $t \in T$ **do**
  BFS$_{\text{mid}}(t)$ (BFS in G_{mid}) to compute $d_{\text{mid}}(t, v)$ for all $v \in V$;
form new graph $K_\circ = (V, E_\circ)$ with edge weights w:
foreach $u \in L$ **do**
  add to $E_\circ$ all edges $(u, v) \in E$, setting $w(u, v) = 1$;
foreach $s \in S$ **do**
  foreach $v \in V$ **do**
    add edge (s, v) to $E_\circ$ and set $w(s, v) = d(s, v)$;
foreach $x \in V$ **do**
  form new graph $K_x = (V, E_x)$ with edge weights w: initialize $K_x \leftarrow K_\circ$
  foreach $t \in T$ **do**
    add edge (x, t) to E_x and set $w(x, t) = d_{\text{mid}}(x, t)$;
  foreach $z \in M$ **do**
    add a single edge (z, t) to E_x for some $t \in T \cap N(z)$ and set $w(z, t) = 1$;
  compute (exact) SSSP from x in K_x (Dijkstra) to obtain distances $d_x(x, v)$ for all $v \in V$;
ouput $(d'(x, y) : x, y \in V)$ where $d'(x, y) = d_x(x, y) \wedge d_y(y, x)$

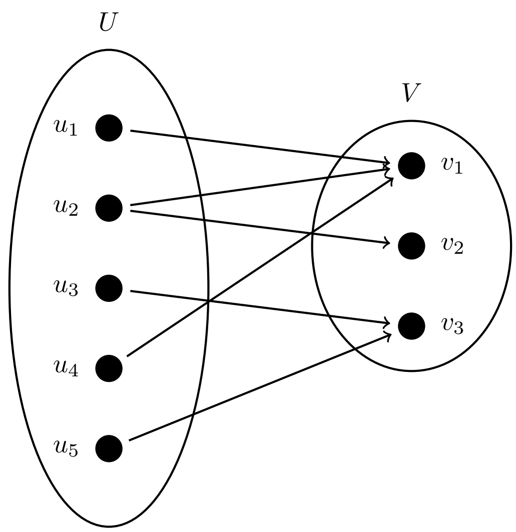
2 Graph Spanners

Our goal is to compress information about distances in a graph by looking at distances within a subgraph.

Definition 2.1. *An (α, β)-spanner of $G = (V, E)$ is a subgraph $H = (V, E_H)$, $E_H \in E$, such that $\forall u, v \in V$,*

$$d(u, v) \leq d_H(u, v) \leq \alpha d(u, v) + \beta.$$

If $\beta = 0$, then it's a mutiplicative spanner (α-spanner); if $\alpha = 1$, then it's an additive spanner ($+\beta$-spanner). In general, directed graphs don't contain sparse spanners. An example is shown in the figure below, where G is a directed bipartite graph, with all its edges leave nodes in set U and incident on nodes in set V. In this case, any spanner with finite (α, β) must contain all edges in G. As a result, we will focus on undirected graphs in this class.



3 Multiplicative Spanners

Theorem 3.1. *Let $k \geq 1$ be an integer, then every n-node graph G with weighted edges contains a $(2k-1)$-spanner with $O\left(n^{1+\frac{1}{k}}\right)$ edges.*

Conjecture 3.1. (Erdős girth conjecture) *For integer $k \geq 1$ and sufficiently large n, there exist n-node graphs of girth $\geq 2k+2$ with $\Omega\left(n^{1+\frac{1}{k}}\right)$ edges.*

Claim 3.1. *The Erdős girth conjecture implies that the bound in Theorem 3.1 is tight, and so there exists some graph G on n nodes such that any $2k-1$ spanner has $\Omega\left(n^{1+\frac{1}{k}}\right)$ edges.*

Proof of Claim 2.1. Let G be an unweighted graph on n edges with girth $2k+2$ and $\Omega\left(n^{1+\frac{1}{k}}\right)$ edges. We'll show that G has no non-trivial $2k-1$ spanners.
Assume there exists some subgraph $H \neq G$ that is a $2k-1$ spanner for G. Choose some edge $(u,v) \in E - E_H$. By the definition of a spanner, $d_H(u,v) \leq (2k-1)d(u,v) = 2k-1$. Therefore there exists some path P in E_H connecting u,v with length at most $2k-1$. However, adding (u,v) to P then completes a cycle in G of length at most $2k$; since G has girth at least $2k+2$, this is a contradiction. This proves the claim. □

Proof of Theorem 3.1. We can generate a sufficient $2k-1$ spanner using the Create-Spanner algorithm. We prove the correctness of this algorithm with the following three claims.

Algorithm 2: Create-Spanner(G)

```
E_H ← ∅.
foreach (u, v) ∈ E in non-decreasing order do
    if d_H(u, v) > (2k − 1)w(u, v) then
        E_H ← E_H ∪ (u, v)
Return H.
```

Claim 3.2. *H s a $(2k-1)$-spanner, i.e.,$\forall u,v \in V, d_H(u,v) \leq (2k-1)d(u,v)$.*

Claim 3.3. *H has girth greater than $2k$.*

Claim 3.4. *Any n-node graph with girth greater than $2k$ has $O\left(n^{1+\frac{1}{k}}\right)$ edges.*

Proof of Claim 3.2. Let u,v be vertices in V, and P be their shortest path in G. For each edge (x,y) in P, either:

- $(x,y) \in E_H$
- There is some path in H between x,y of length at most $(2k-1)w(x,y)$. If no such path exists, then (x,y) would have been added to E_H in Create-Spanner when it was considered.

Therefore

$$d_H(u,v) \leq \sum_{(x,y)\in P} d_H(x,y) \leq \sum_{(x,y)\in P} (2k-1)w(x,y) = (2k-1)w(P) = (2k-1)d(u,v).$$

□

Proof of Claim 3.3. Assume H has a cycle C of length $\leq 2k$. Let (u,v) be the highest weighted edge of C. Then (u,v) is the last edge in C added to E_H, satisfying

$$\sum_{\substack{(x,y)\in C,\\ (x,y)\neq(u,v)}} w(x,y) > (2k-1)w(u,v)$$

On the other hand, the path $C\backslash(u,v)$ has length at most $(2k-1)w(u,v)$, giving

$$\sum_{\substack{(x,y)\in C,\\ (x,y)\neq(u,v)}} w(x,y) \leq (2k-1)w(u,v)$$

This gives a contradiction. □

Proof of Claim 3.4. Let H be any graph with girth greater than $2k$ and at least $10n^{1+\frac{1}{k}}$ edges. Modify the graph by repeatedly removing any nodes of degree $\leq \lceil n^{\frac{1}{k}} \rceil$, and any edges incident to that node, until no such nodes exist. The total number of edges removed in this way at most $2n \cdot \lceil n^{\frac{1}{k}} \rceil$, which means that at least $8n^{1+\frac{1}{k}}$ edges remain (and so the graph is not empty).

The minimum degree of the resulting subgraph is greater than $\lceil n^{\frac{1}{k}} \rceil$. However, by homework problem **1.3b**, this means that the subgraph has girth at most $2k$, and therefore the original graph does as well. This is a contradiction. □

The subgraph returned by Create-Spanner is a $(2k-1)$-spanner by Claim 3.2, and has $O\left(n^{1+\frac{1}{k}}\right)$ edges by Claim 3.3, Claim 3.4. This completes the proof of the theorem. □