More Algorithms on Approximate APSP

LRYP

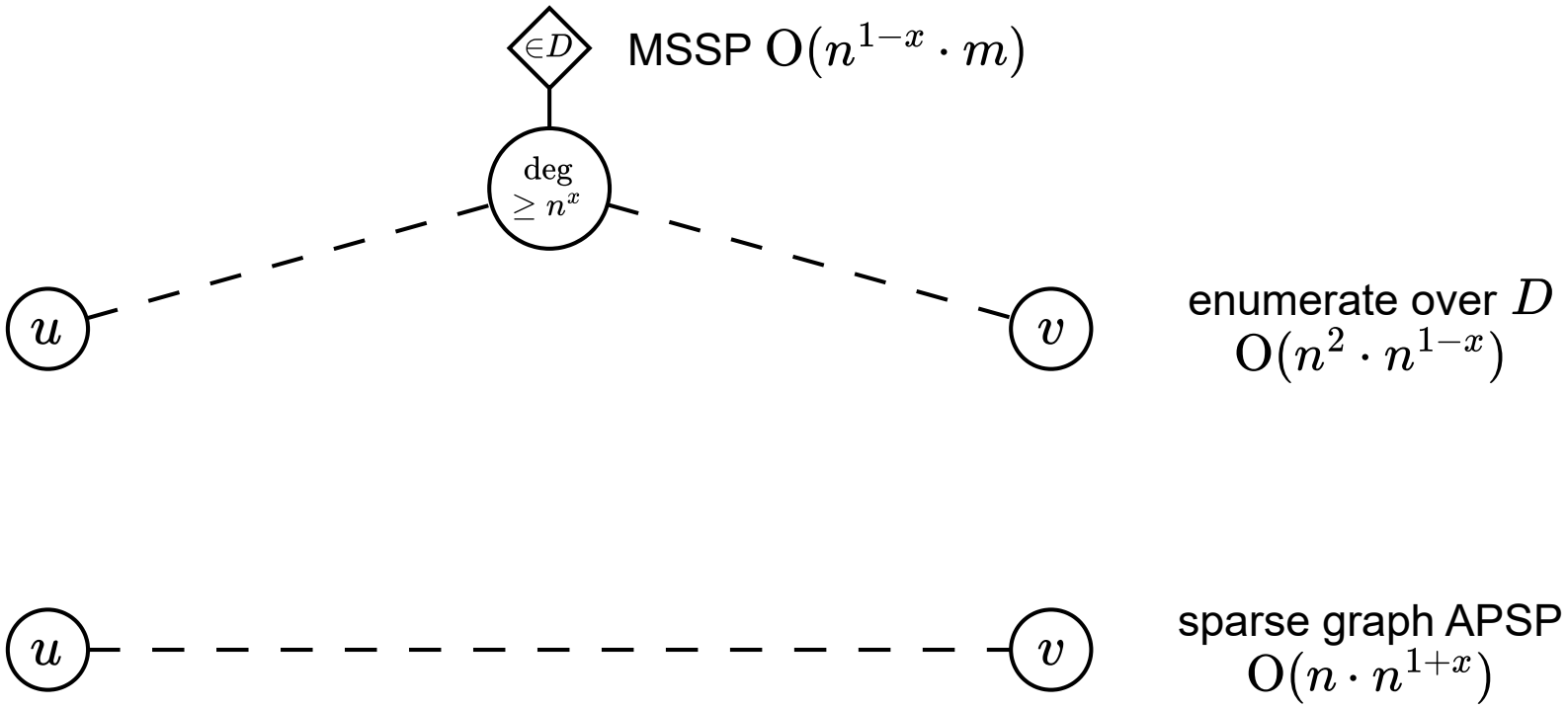
December 30, 2025

Outline

- 2-approximation for weighted graph
- 2-approximation for unweighted graph
- $+k$-approximation
- $+k$-approximation for short paths and 2-approximation for long paths

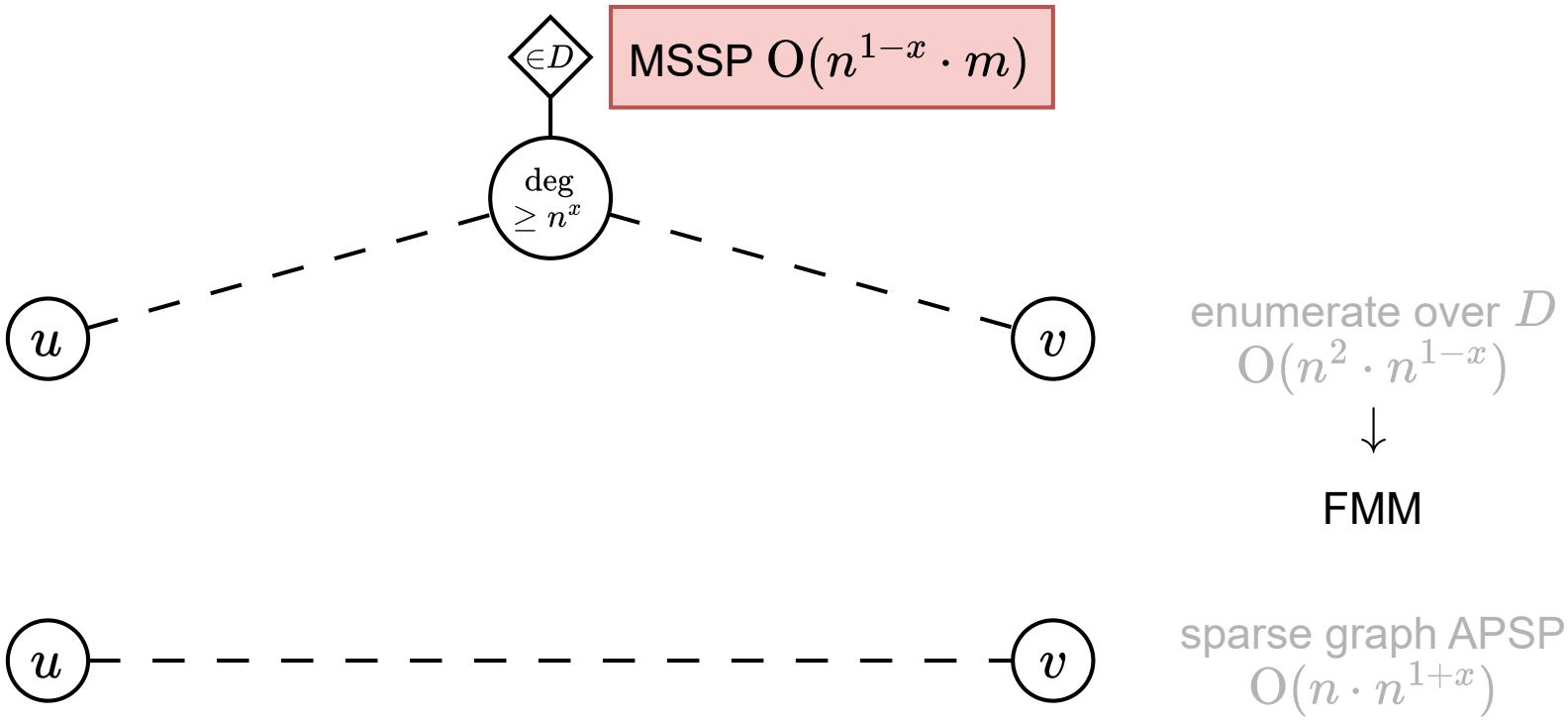
Convention: I use α-approximation for multiplicative approximation and $+\beta$-approximation for additive ones. (α, β)-approximation means $d(u, v) \leq \delta(u, v) \leq \alpha d(u, v) + \beta$. The big-O notation hides polylog factors.

Framework of naïve +2-approximation



Taking $x = 1/2$, we get the $O(n^{5/2})$ algorithm.

Optimizing +2-approximation



For now, let's focus on optimizing the part calculating SP starting from the covering vertices.

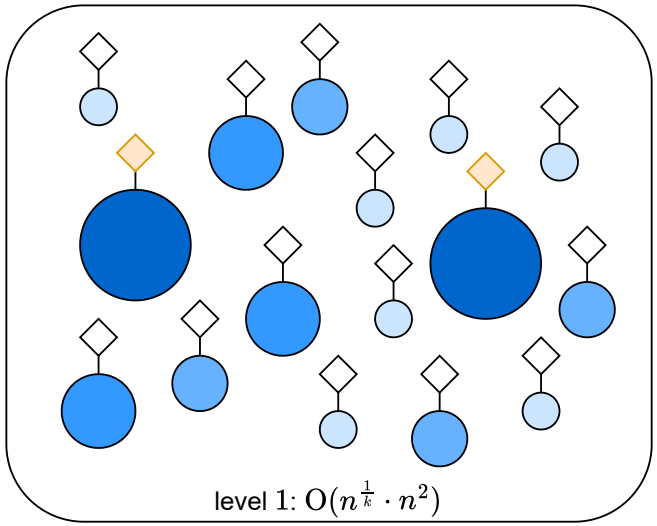
Optimizing MSSP

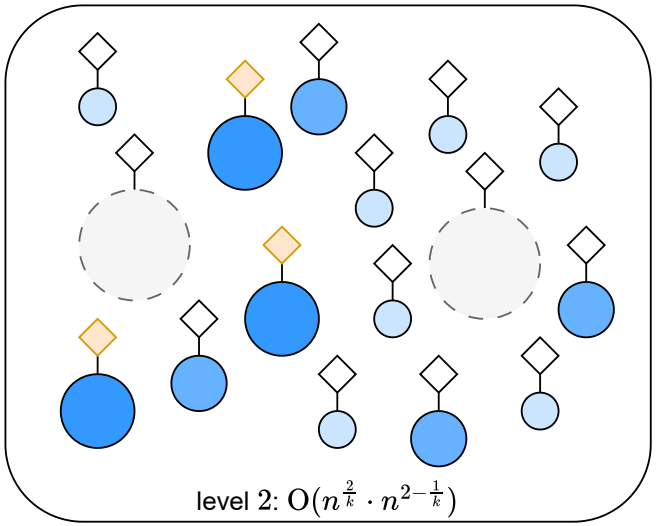
Consider the vertex with the largest degree on the SP from u to v (below denoted as $\pi(u, v)$). If the degree is between a and b, then:

- The covering set has size $O(n/a)$.
- The number of edges is $O(nb)$ since the edge between vertices with degree $> b$ can be ignored.

So the time is $O(n^2 b/a)$. We can divide the degrees into multiple levels and balance the time.

Optimizing MSSP

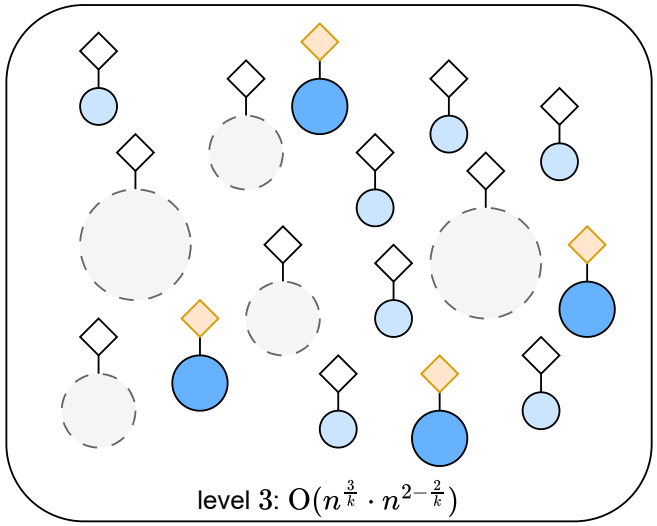


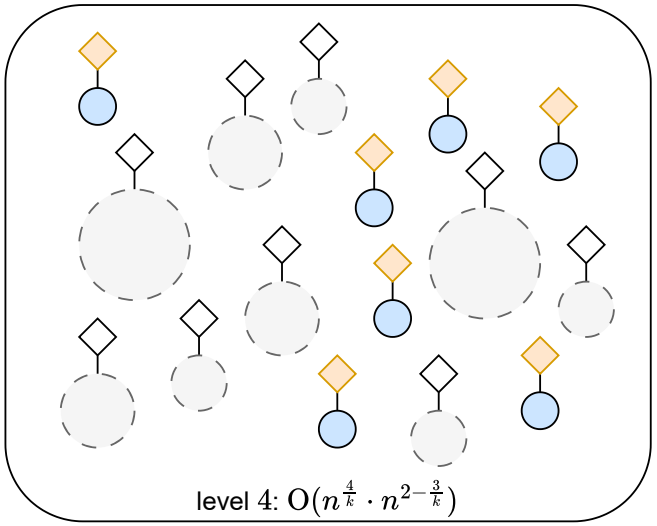


$\log n$ levels

$\downarrow$

$O(n^2)$





Framework on weighted graph

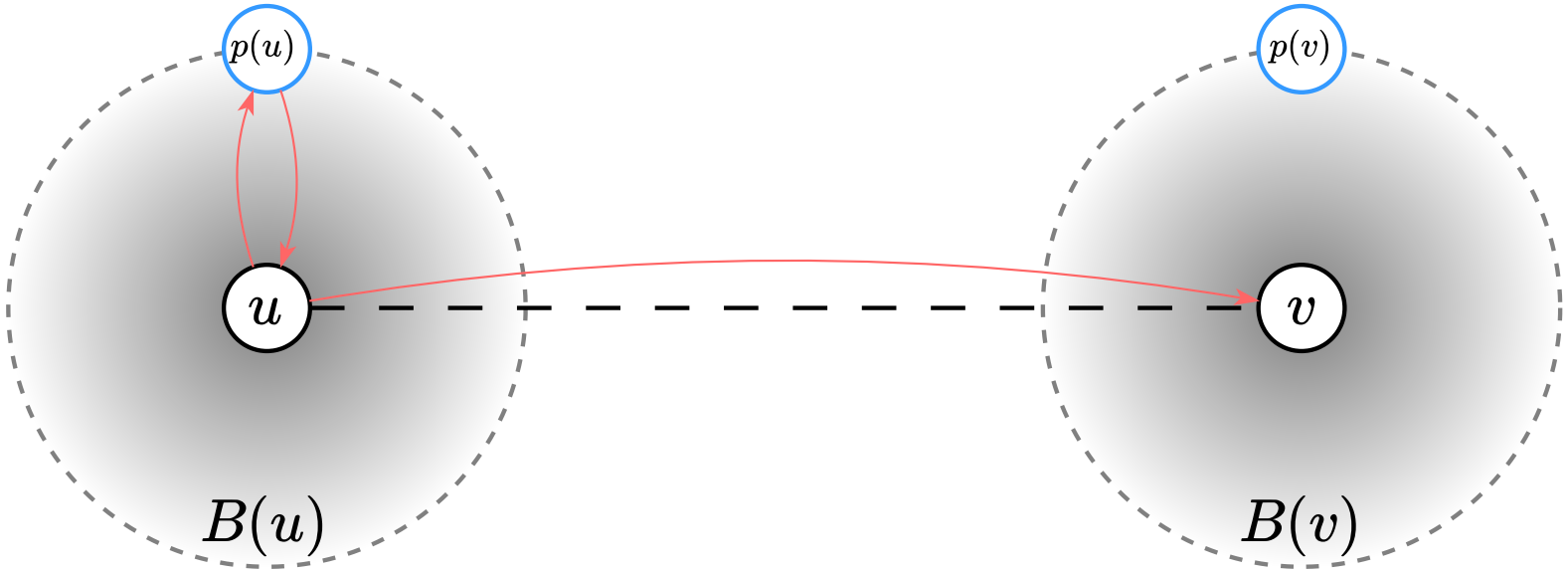
Sample a set of vertices D of size $\mathrm{O}(n^{1-x})$, to serve as "intermediate vertices" similar to the covering set in unweighted case.

For each u, calculate its nearest vertex in D, denoted as $p(u)$, and obtain the set of vertices closer to u than $p(u)$, together with u and $p(u)$ denoted as $B(u)$, called the bunch of u. Also, define the cluster of v as $C(v) = \{u \mid v \in B(u)\}$.

[Thorup, Zwick '01]: There is an algorithm that compute set D of size $\mathrm{O}(n^{1-x})$ in time $\mathrm{O}(mn^x)$, that makes every bunch and cluster's sizes bounded by $\mathrm{O}(n^x)$ w.h.p.

In contrast, one can easily design an **deterministic** algorithm that computes a hitting set in unweighted graph.

Framework on weighted graph

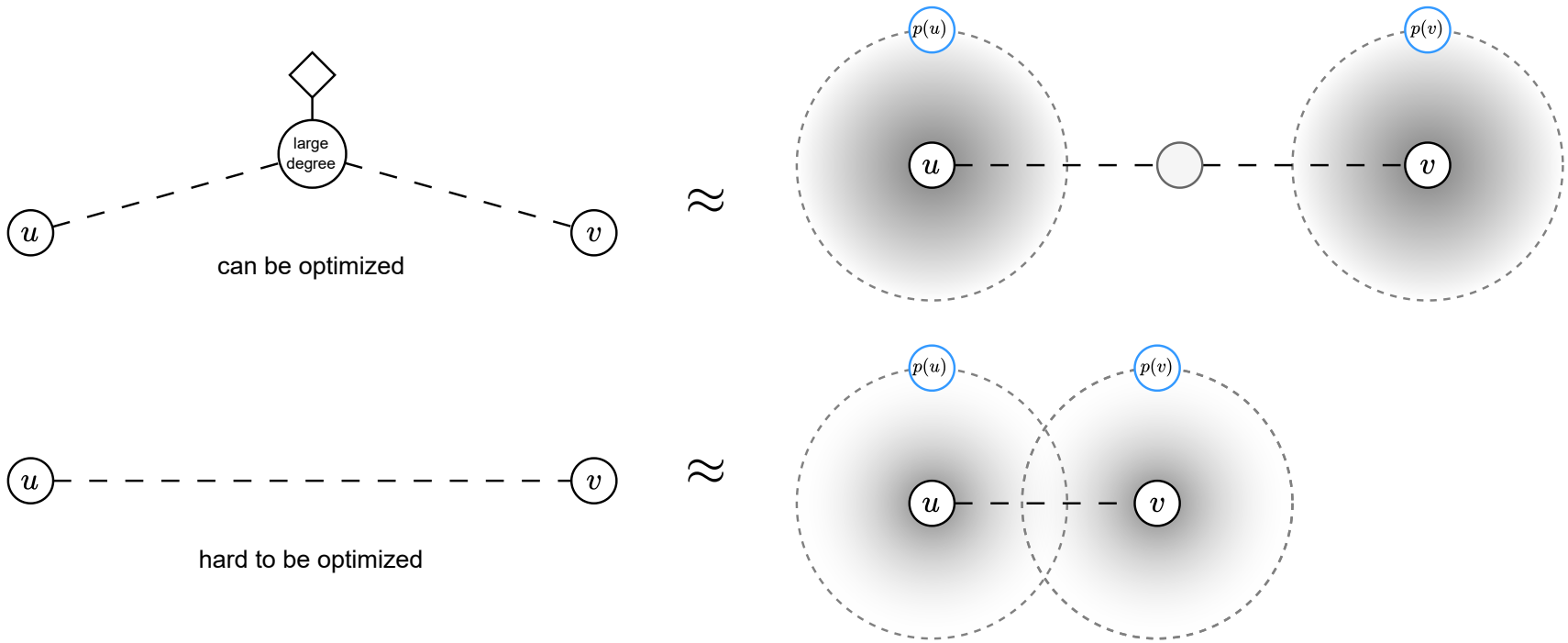


If, on $\pi(u, v)$, $\exists x \in V \setminus (B(u) \cup B(v))$, then

$$\min\{d(u, p(u)) + d(p(u), v), d(u, p(v)) + d(p(v), v)\}$$

is a 2-approximation of $d(u, v)$.

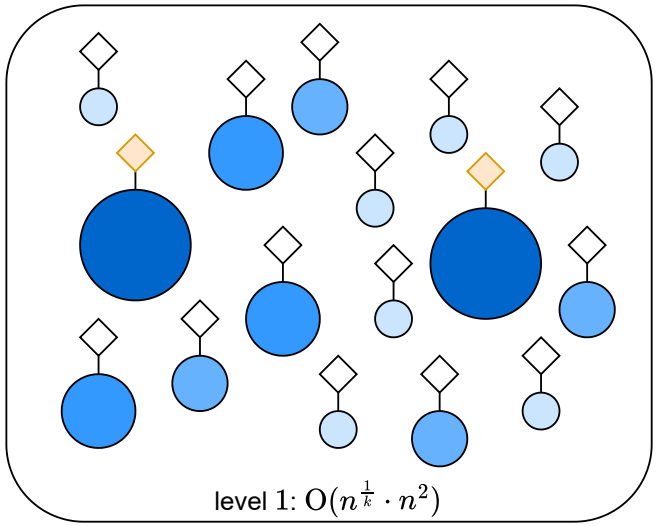
Analogy between weighted and unweighted case

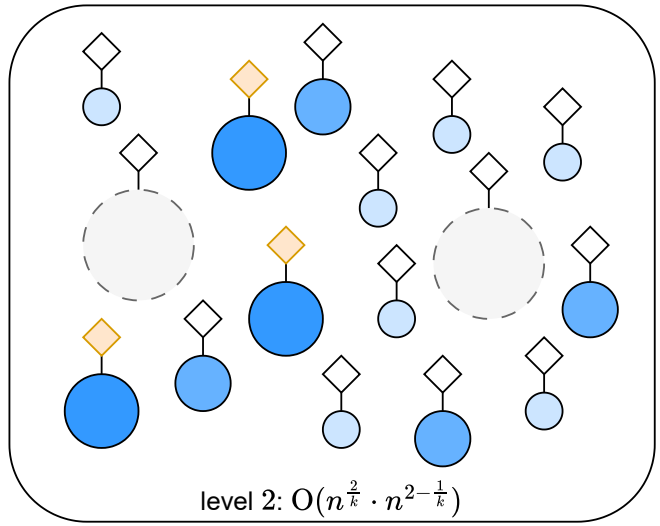


However, it's hard to deal with when the "intermediate vertex" cannot be utilized.

[Dory et al. '23]: $O(n^{2/3}m)$ for 2-approximation, $O(nm^{2/3})$ for $(2, W)$-approximation.

Analogy between weighted and unweighted case

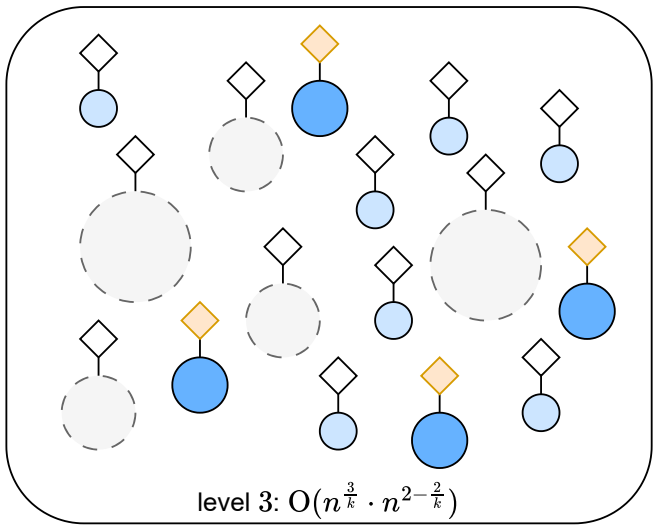


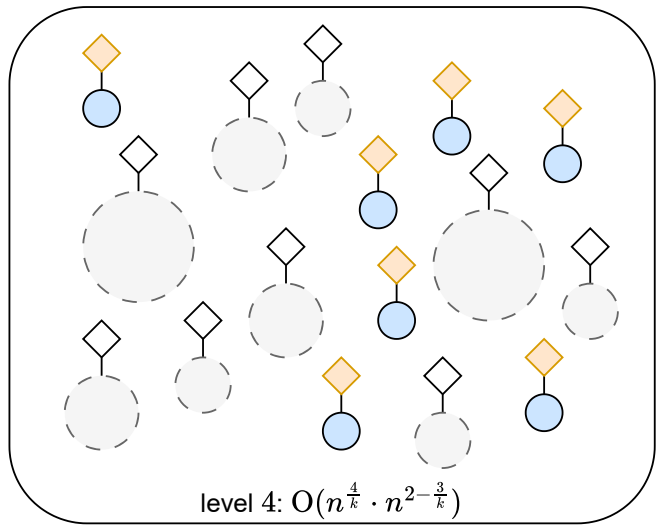


$\log n$ levels

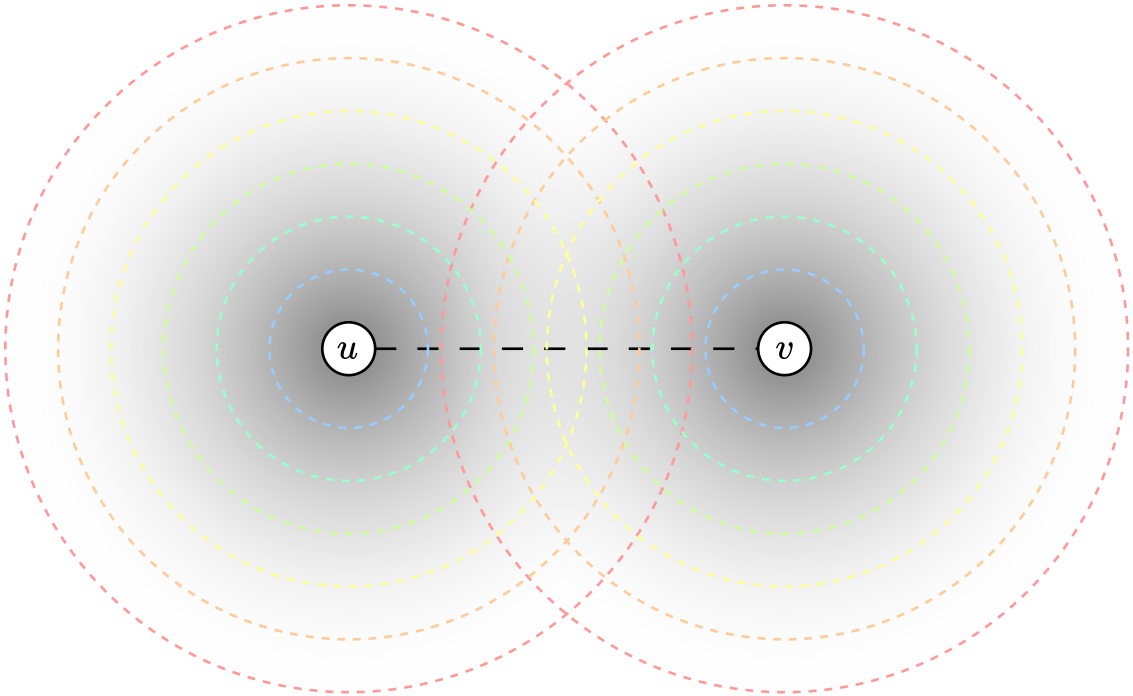
$\downarrow$

$O(n^2)$



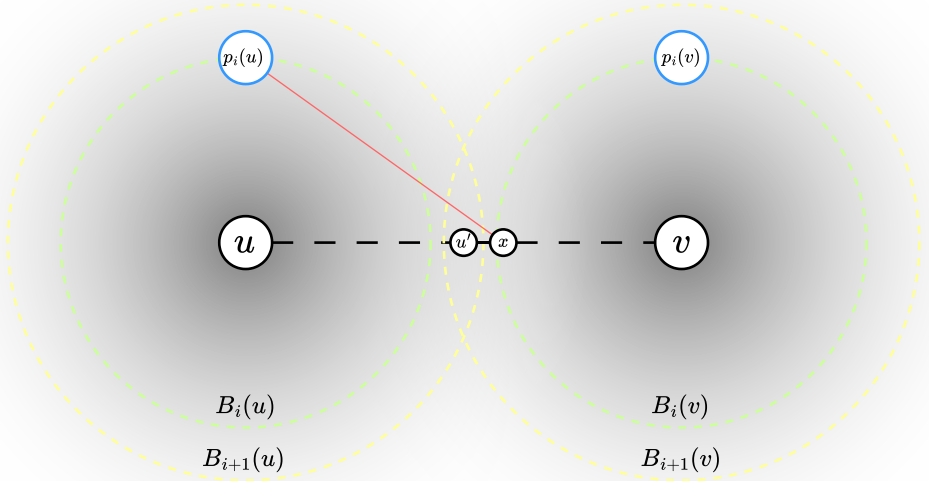


Hierarchical method for weighted case



Sample a chain of sets $V = D_0 \supseteq D_1 \supseteq \cdots \supseteq D_k$. Each level the set size halves and the bunch size doubles.

Hierarchical method for weighted case

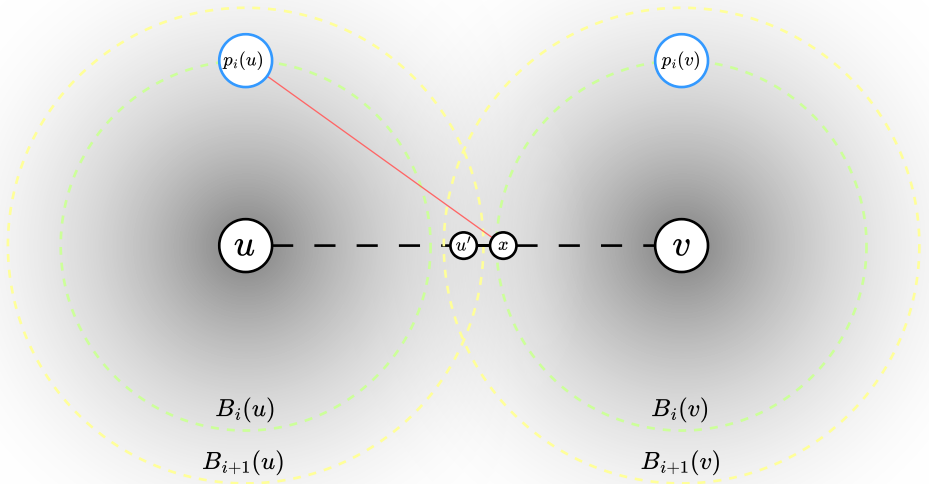


There exists some i, such that there's vertex $\in V \setminus (B_i(u) \cup B_i(v))$ on the SP, but not so in the next level.

The problem is, we want to avoid computing MSSP on the entire graph for $\{p_i(u)\} = D_i$. Recall the proof of 2-approximation:

$$d(u, p(u)) + \boxed{d(p(u), v)} \leq d(u, p(u)) + \boxed{d(p(u), u) + d(u, v)}.$$

Hierarchical method for weighted case



Let u' be the last vertex in $B_{i+1}(u)$ on the SP, and x be u''s successor. We only need to calculate $d(p_i(u), v)$ so that:

$$d(p_i(u), v) \leq d(p_i(u), u) + d(u, u') + w(u', x) + d(x, v),$$

rather than getting the exact distance.

Hierarchical method for weighted case

For $p_i(u)$, construct a graph, containing edges $(p_i(u), x)$ of weight

$$w(p_i(u), x) = \min_{u' \in B_{i+1}(u)} \{d(p_i(u), u) + d(u, u') + w(u', x)\}$$

and edges (v, x) in the original graph satisfying $w(v, x) \leq d(v, p_{i+1}(v))$. In this way, the path $p_i(u) \rightsquigarrow u \rightsquigarrow u' \to x \rightsquigarrow v$ is exactly covered.

Time complexity:

- Calculating $w(p_i(u), x)$: $\sum_{u'} \deg_{u'} |C_{i+1}(u)| = \mathrm{O}(m|C_{i+1}|)$.
- Dijkstra from $p_i(u)$: under some small adjustion we can regard D_{i+1} as randomly sampled. Thus the expected degree of each vertex is $\mathrm{O}(n/|D_{i+1}|)$ and the time will be $\mathrm{O}(n^2/|D_{i+1}|)$.

Overall it is

$$\sum_i \left(m|C_{i+1}| + \frac{n^2|D_i|}{|D_{i+1}|} \right) = \sum_i \left(\frac{mn}{|D_{i+1}|} + \frac{n^2|D_i|}{|D_{i+1}|} \right) = \mathrm{O}\left(\frac{mn}{|D_k|} + n^2 \right).$$

Hierarchical method for weighted case

Let $|D_k| = \mathrm{O}(n^{1-x})$. If $\pi(u, v)$ still includes "intermediate vertex" on level k, then run the normal MSSP. The time will be

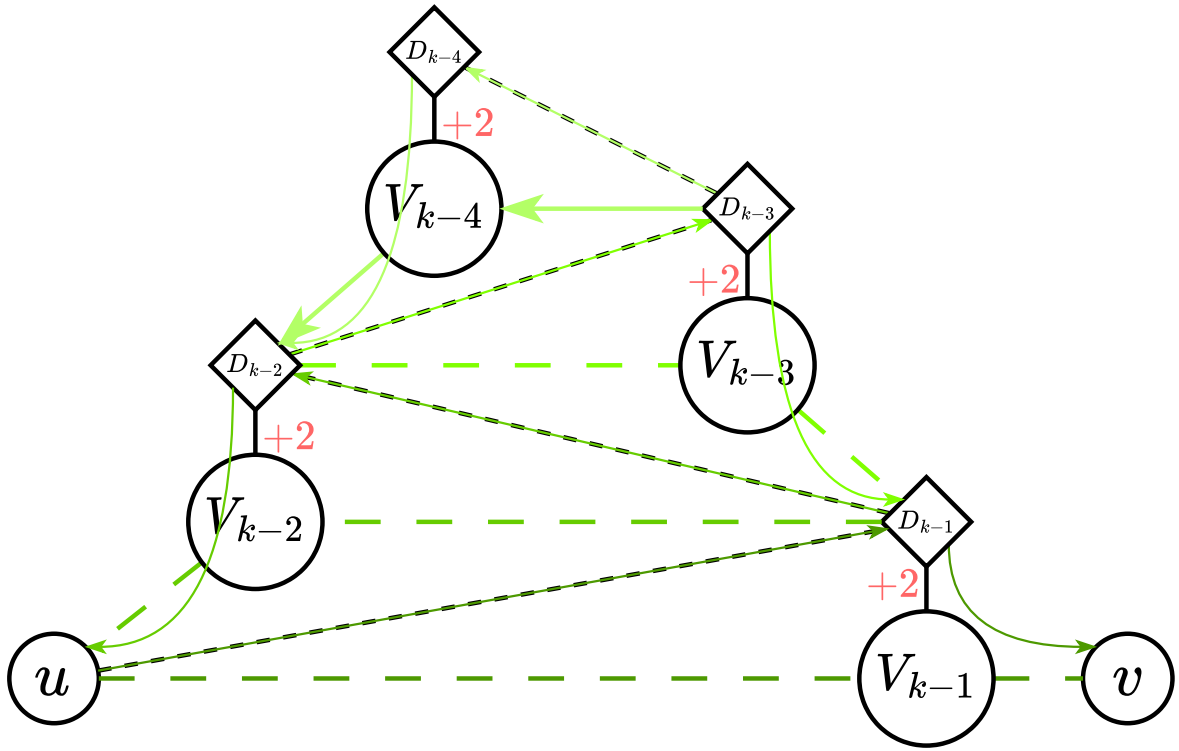
$$\mathrm{O}\left(mn^x + n^2 + mn^{1-x}\right) \to \mathrm{O}\left(n^{1/2}m + n^2\right).$$

This is the result of [Baswana, Kavitha '10]. It will be used as a subroutine in the later algorithms.

[Elkin, Neiman '22]: For $|S| = \mathrm{O}(n^r)$, $(1+\epsilon)$-approximate MSSP can be computed in $\mathrm{O}(m^{1+o(1)} + n^{\omega(1,r,1)}\epsilon^{-\mathrm{O}(1)} \log W)$ time.

Replace the normal MSSP with this, for dense graph, after balancing, we get $\mathrm{O}(n^{2.214}\epsilon^{-\mathrm{O}(1)} \log W)$ for $(2+\epsilon)$-approximation.

Layering for $+k$-approximation



Dividing the degree into k levels: $n = s_0 > s_1 > \cdots > s_k = 1$. For every vertex $u \in D_i$, run Dijkstra on $(V, \{u\} \times V \cup E^* \cup E_i)$, we can get $+2(i-1)$-approximation for $D_i \times V$. Thus there's $\mathrm{O}(n^{2+\frac{2}{k+2}})$ or $\mathrm{O}(n^{2-\frac{2}{k+2}} m^{\frac{2}{k+2}})$ $+k$-approximation for even k.

Faster algorithm for unweighted 2-approximation

A $+k$-approximation algorithm means 2-approximation for SP of length $\geq k$. This incentivizes us to divide cases both by degree and by path length. [Dory et al. '23]:

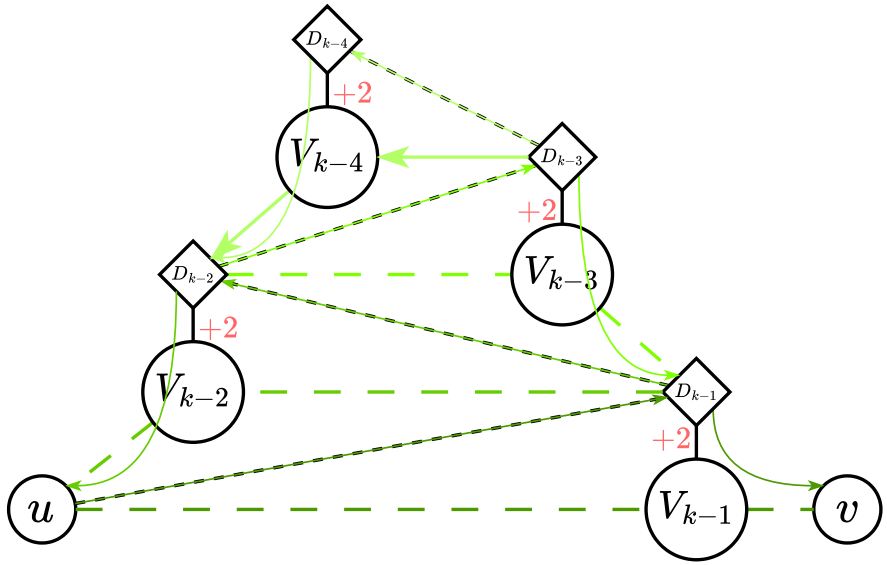
- $\pi(u, v)$ goes through a vertex of degree $\geq n^x$.
  - $d(u, v) \geq \log n$. Run $+\log n$-approximation, $\mathrm{O}(n^2)$.
  - $d(u, v) < \log n$. Bounded $(\min, +)$-MM can be calculated in $\mathrm{O}(Mn^\omega)$, so this part takes $\mathrm{O}(n^{\omega(1,1-x,1)})$ time.
- $\pi(u, v)$ only contains vertices of degree $< n^x$. Run the $\mathrm{O}(n^{1/2}m + n^2)$ weighted graph 2-approximation algorithm.

After balancing, we get $\mathrm{O}(n^{2.032})$.

Open problem: does 2-approximation have $\mathrm{O}(n^2)$ algorithm?

The above algorithm utilizes FMM and bunch construction. What about combinatorical algorithms, or deterministic algorithms?

Optimizing additive approximation

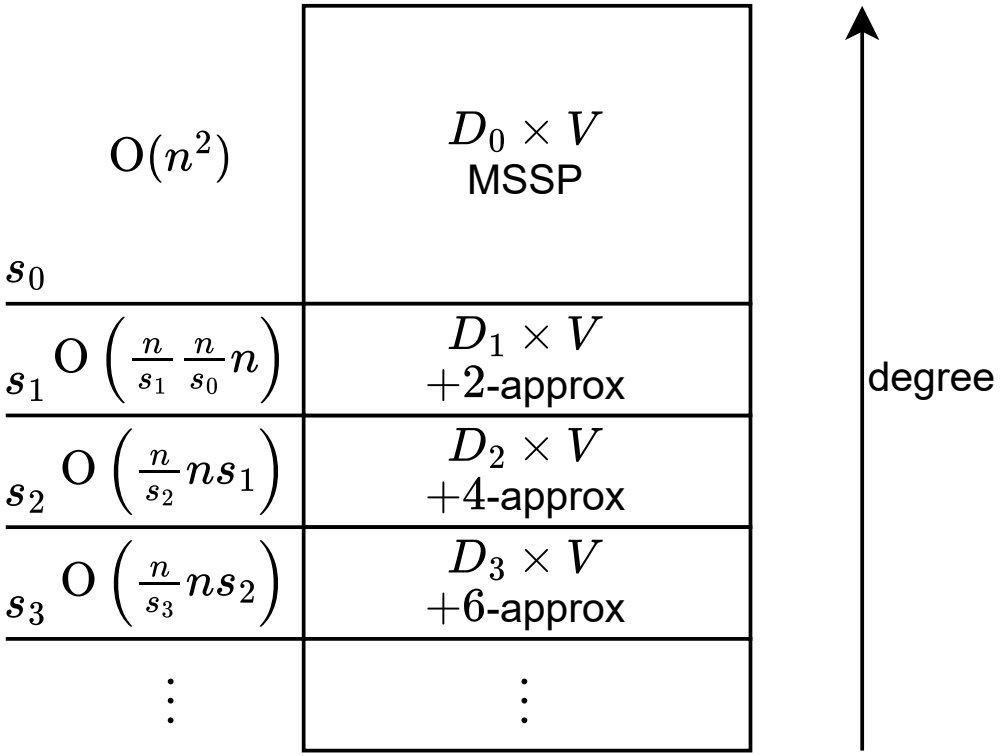


The proof of this method (hereinafter referred to as the new method) considers the last vertex on the SP with higher degree.

In contrast, the naïve method that directly enumerates over D, relies on any large-degree vertex on the SP, and always guarantees +2-approximation. Consider combining these two methods.

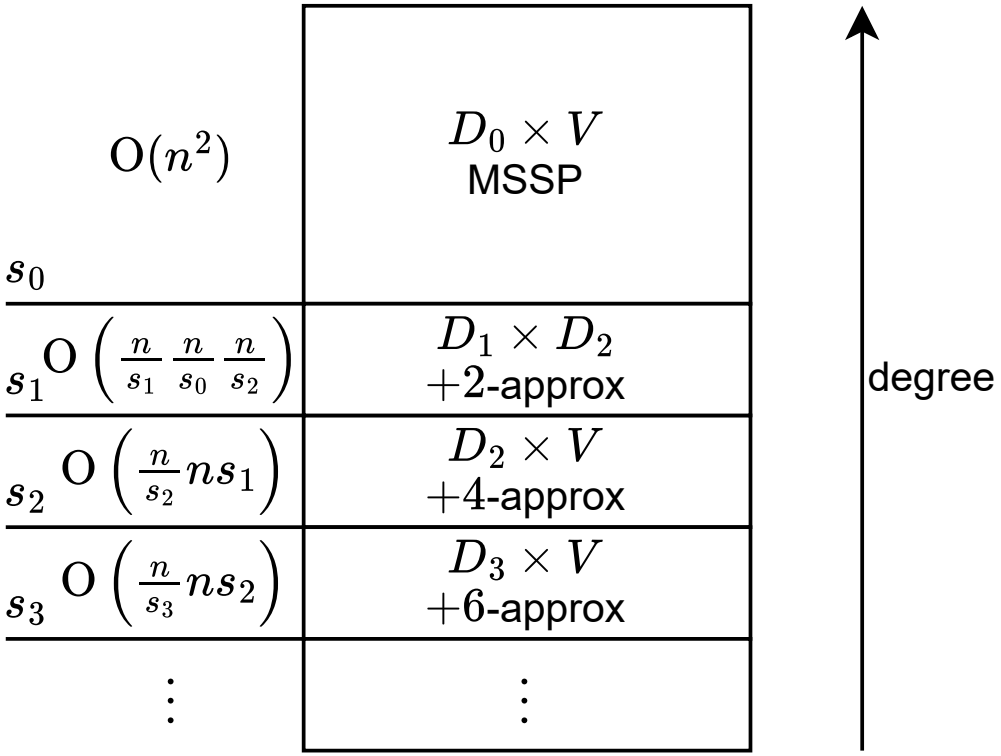
We've known that one layer for each method leads to $O(n^{7/3})$ +2-approximation. In the same paper [Dor, Halperin, Zwick '96], they proposed its generalized version.

Combining two methods



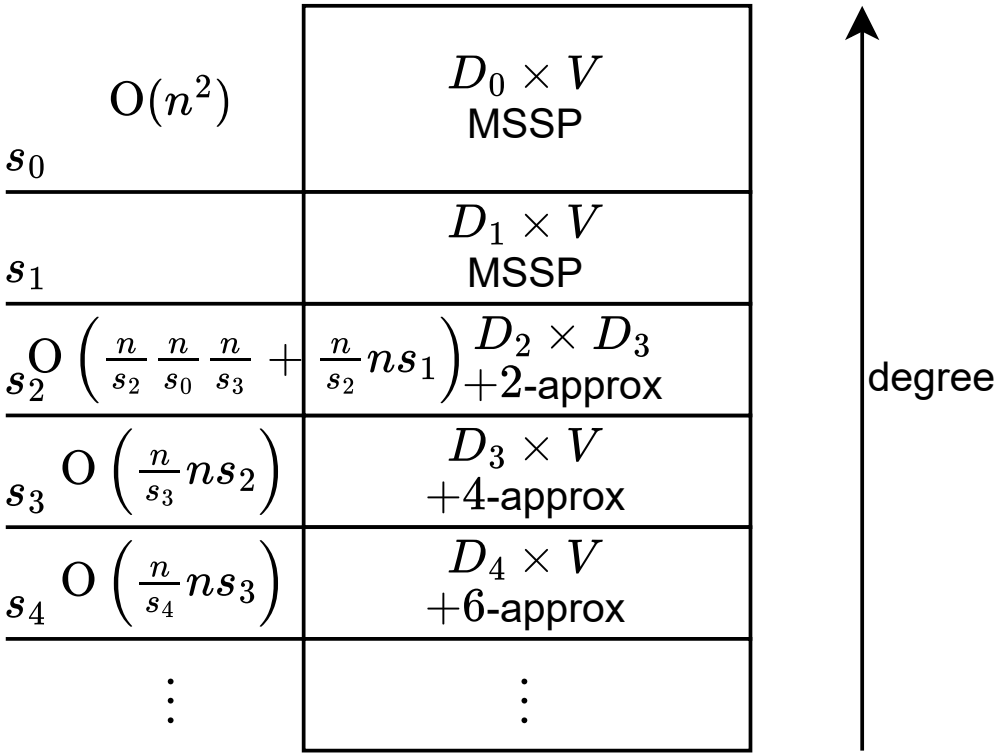
For the MSSP part, use the previous hierarchical method to achieve $O(n^2)$. For $+2$ part, it's a matrix multiplication. For the remaining part, use the new method.

Optimizing the combination



Since the +4 part only relies on D_1 as the intermediate vertices, we can reduce the dimension.

Optimizing the combination



If $\pi(u \in D_2, v \in D_3)$ only contains vertices of degree $< s_0$, we can use the new method. Otherwise run MM.

Optimizing the combination

For $+k$-approximation, we need $s_0 \sim s_{k/2+1}$, and the time is:

$$\mathrm{O}\left(\frac{n^3}{s_0 s_2 s_3} + \sum_{i=1}^{k/2} \frac{n^2 s_i}{s_{i+1}}\right),$$

which can be balanced as $\mathrm{O}(n^{2+\frac{2}{3k-2}})$. The original pseudocode of the algorithm is as follows. Concise but complicated to analyze!

Algorithm $\overline{\textbf{apasp}}_k$:

input: An unweighted undirected graph $G = (V, E)$.
output: A matrix $\{\hat{\delta}(u, v)\}_{u,v}$ of estimated distances.

For $i \leftarrow 1$ to $k-1$ let $s_i \leftarrow n^{1-\frac{i}{k}}$

For $i \leftarrow 1$ to $k-1$ let $V_i \leftarrow \{v \in V \mid \deg(v) \geq s_i\}$
For $i \leftarrow 2$ to k let $E_i \leftarrow \{(u, v) \in E \mid \deg(u) < s_{i-1} \text{ or } \deg(v) < s_{i-1}\}$

For $i \leftarrow 1$ to $k-1$ let $(D_i, E_i^*) \leftarrow \textbf{dominate}(G, s_i)$
$E_1 \leftarrow E$; $D_k \leftarrow V$; $E^* \leftarrow \cup_{1 \leq i < k} E_i^*$

For every $u, v \in V$ let $\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & \text{if } (u, v) \in E, \\ +\infty & \text{otherwise.} \end{cases}$

For $i \leftarrow 1$ to k do
For every $u \in D_i$ run $\textbf{dijkstra}\left((V, E_i \cup E^* \cup (\{u\} \times V)) \cup (\cup_{i+j_1+j_2 \leq 2k+1} D_{j_1} \times D_{j_2})\right), \hat{\delta}, u)$

Applying FMM to $+k$-approximation

[Zwick '98]: $(1+\epsilon)$-approximate $(\min,+)$-MM can be calculated in $\mathrm{O}(n^{\omega}\epsilon^{-1}\log M)$ time.

Applying this to the part where $\frac{n}{s_2} \times \frac{n}{s_0}$ matrix multiplies with $\frac{n}{s_0} \times \frac{n}{s_3}$, then balance, we get

Approx	Time
$(1+\epsilon, 4)$	$\mathrm{O}(n^{2.093}/\epsilon)$
$(1+\epsilon, 6)$	$\mathrm{O}(n^{2.058}/\epsilon)$
$(1+\epsilon, 8)$	$\mathrm{O}(n^{2.043}/\epsilon)$

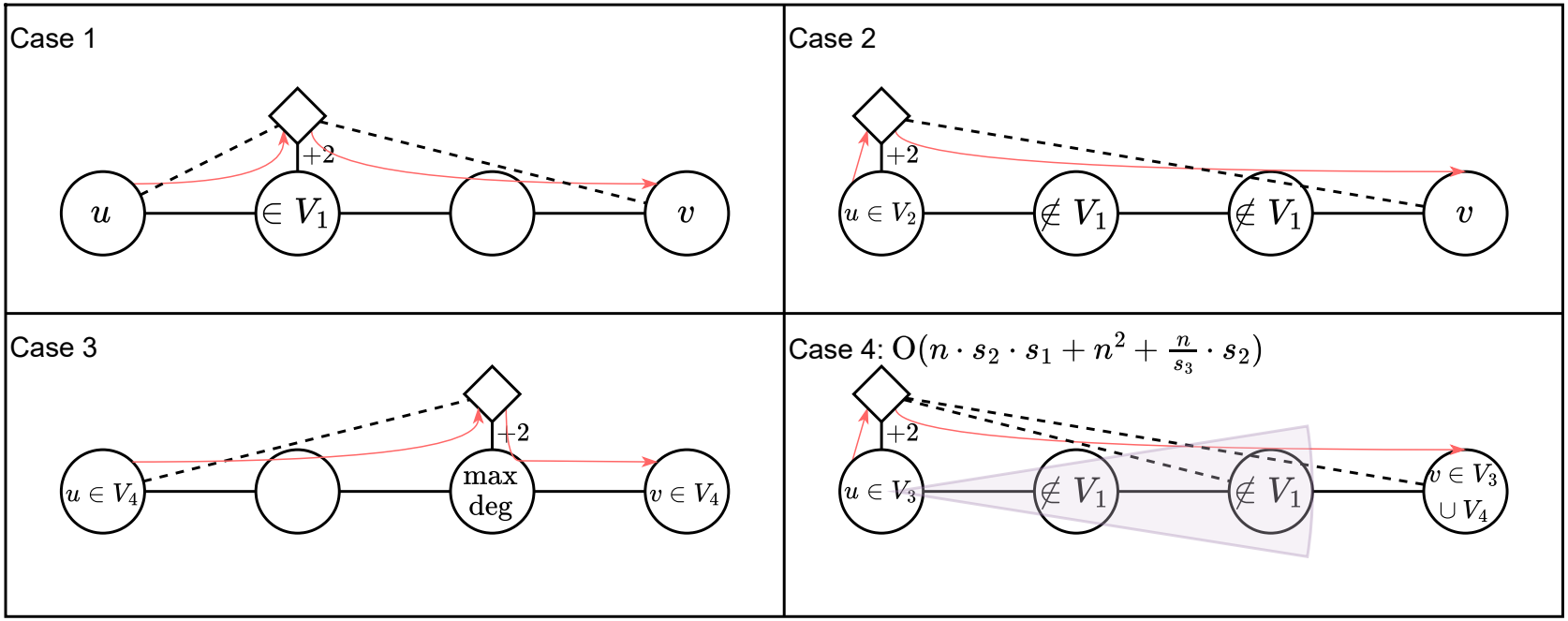
These times also apply to bounded length $+k$-approximation.

Applying FMM to $+k$-approximation

[Dürr '23] (generalization): For two matrices A, B of dimensions $n^\alpha \times n^\beta$ and $n^\beta \times n^\gamma$, and B is bounded difference, if one can add an arithmetic sequence to every row of B to make B monotone and bounded by n^μ, then $(\min, +)$-MM of them can be calculated in $\mathrm{O}(n^{(\alpha+\beta+\mu+\omega(\alpha,\beta,\gamma))/2})$ time.

Approx	Time
$+4$	$\mathrm{O}(n^{2.155})$
$+6$	$\mathrm{O}(n^{2.103})$
$+8$	$\mathrm{O}(n^{2.078})$

Additive approximation for short paths



[Roditty '23]: +2-approximation for length ≤ 3 can be computed in $O(\min\{n^{9/4}, n^{5/3}m^{1/3}\})$.

Case 1. Calculate distance for $D_1 \times V$ on the original graph.

Case 2. Calculate distance for $D_2 \times V$ on F_2.

Case 3. Symmetric for max degree vertex being nearer to u.

Case 4. BFS up to distance 2 for $u \in V_3$ on $V \setminus V_1$, update distance for $D_3 \times V$ and run Dijkstra from $x \in D_3$ on $\{x\} \times V \cup F_3$.

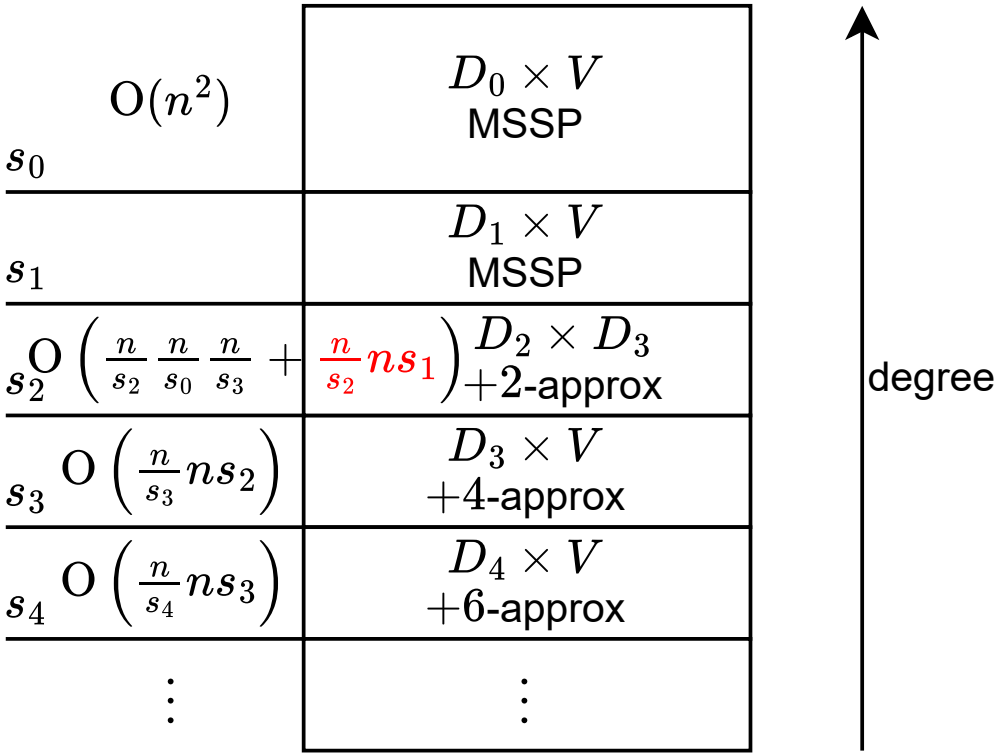
Additive approximation for short paths

Combining this $+2$-approximation for ≤ 3 and normal $+4$-approximation, we get a 2-approximation algorithm with the same time complexity. This implies that the current best combinatorical $(2,0)$-approximation algorithm is faster than $(1,2)$-approximation.

[Roditty '23]: For even $k \geq 4$, $+k$-approximation for length $\leq k+2$ can be computed in $\mathrm{O}(n^{2-\frac{2}{k+4}} m^{\frac{2}{k+4}})$. Thus, 2-approximation for length $\geq k$ can be computed in $\mathrm{O}(\min\{n^{2-\frac{2}{k+4}} m^{\frac{2}{k+4}}, n^{2+\frac{2}{3k-2}}\})$.

The basic strategy is still case-by-case analysis, using properties of intermediate vertices, and also running 2-deep BFS in one subcase.

2-approximation for long paths



Under the previous layering framework, one can show that, only utilizing the large degree vertices on $\pi(u, v)$ as intermediate vertices is enough. Thus, If we force $\pi(u, v)$ to contain some vertex of degree over s_0, then the $\frac{n}{s_2}ns_1$ part can be removed.

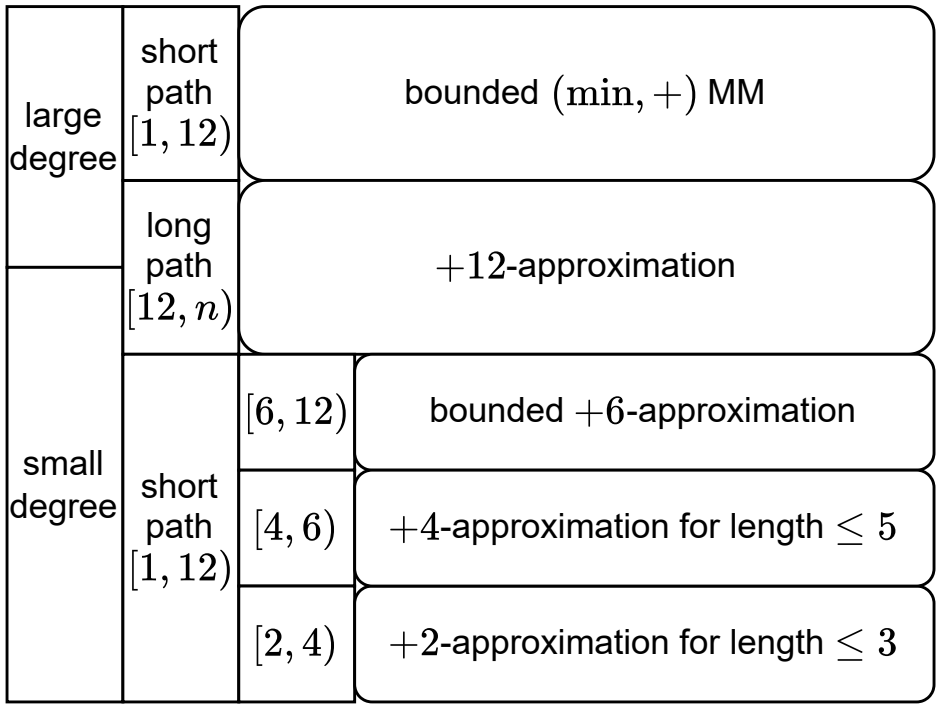
2-approximation for long paths

If $\pi(u, v)$ only contains relatively small degree vertices, we can utilize the weighted 2-approximation algorithm, whose time depends on m. Under such combination, the time is (s_1 simply doesn't appear)

$$\mathrm{O}\left(n^{\frac{3}{2}}s_0 + \frac{n^3}{s_0 s_2 s_3} + \sum_{i=2}^{k/2} \frac{n^2 s_i}{s_{i+1}}\right),$$

and can be balanced to $\mathrm{O}(n^{2+\frac{1}{2(k-1)}})$, which is better than $\mathrm{O}(n^{2+\frac{2}{3k-2}})$ for full approximation. Also it can be optimized by FMM.

Deterministic 2-approximation algorithm



$O(n^{\omega(1,1-x,1)})$

$O(n^{2+\frac{1}{17}})$

$O(n^{2.064})$

$O(n^{1+2x} + n^{1+\frac{3k-4}{k}x} + n^{2+\frac{x}{k}})$

The weighted 2-approximation algorithm contains randomness. [Saha, Ye '23] proposed a deterministic algorithm that calculate 2-approximation in $O(n^{2.072})$ time. The framework is shown above. It basically covers those approaches.

The bottleneck of 2-approximation APSP is likely to be the case of sparse short path.

Thanks!