Distributed Algorithms
All-Pairs Shortest Paths

Maruth Goyal

UT Austin

Spring 2021

APSP Setup

- Suppose that we are interested in knowing all of the following:
  - Length of the shortest path $\delta(u, v)$ for all $u, v \in V$
  - The number of shortest paths between any two vertices σ_{uv}
  - For each pair of vertices u, v, all the predecessors of v along shortest paths from u

APSP Setup

- Suppose that we are interested in knowing all of the following:
  - Length of the shortest path $\delta(u, v)$ for all $u, v \in V$
  - The number of shortest paths between any two vertices σ_{uv}
  - For each pair of vertices u, v, all the predecessors of v along shortest paths from u
- Useful to compute certain graph measures
  - eg: Betweeness Centrality: fraction of all shortest paths in the graph that pass through the given vertex. Measures "importance".

APSP Setup

- Suppose that we are interested in knowing all of the following:
  - Length of the shortest path $\delta(u, v)$ for all $u, v \in V$
  - The number of shortest paths between any two vertices σ_{uv}
  - For each pair of vertices u, v, all the predecessors of v along shortest paths from u
- Useful to compute certain graph measures
  - eg: Betweeness Centrality: fraction of all shortest paths in the graph that pass through the given vertex. Measures "importance".
- Assume graph is unweighted and directed.
- Assume every node knows n, the number of processors
- Goal: every processor v should know the following at the end:
  1. $\delta(u, v)$ for all $u \in V$
  2. σ_{uv} for all $u\ inV$
  3. $P_u(v)$ for all $u \in V$

APSP

- We will study the algorithm(s) from *"Distributed Algorithms for Directed Betweenness Centrality and All Pairs Shortest Paths"* [Pontecorvi and Ramachandran, 2018]
- We will be working in the CONGEST model.

APSP

- We will study the algorithm(s) from *"Distributed Algorithms for Directed Betweenness Centrality and All Pairs Shortest Paths"* [Pontecorvi and Ramachandran, 2018]
- We will be working in the CONGEST model.
- They present a distributed APSP algorithm which runs in $\min(n + O(D), 2n)$ rounds, and $mn + O(m)$ messages.
  - Improves over prior work [Lenzen and Peleg, 2013] which had $2n$ rounds, and up to $2mn$ messages.

APSP Algorithm

Algorithm Sketch

1. Initialize state
2. In each round r:

APSP Algorithm

Algorithm Sketch

1. Initialize state
2. In each round r:
    1. If the distance $\delta(u, v)$ and σ_{uv} have converged for some u, send out $(\delta(u, v), u, \sigma_{uv})$.

APSP Algorithm

Algorithm Sketch

1. Initialize state
2. In each round r:
    1. If the distance $\delta(u, v)$ and σ_{uv} have converged for some u, send out $(\delta(u, v), u, \sigma_{uv})$.
    2. Process incoming messages; update state to reflect current best known δ, σ values

APSP Algorithm

Issue #1

What state do we need to track?

Issue #1

What state do we need to track?

State

- At each vertex v, maintain a list L_v.
  - Stores tuples of the form $(\delta(u,v), u)$ in lexicographically increasing order.
  - Initially just $[(0, v)]$
- Lazily maintain $\delta(u,v)$ and $\sigma(u,v)$. Initially, $\delta(v,v) = 0$, $\sigma(v,v) = 1$.

APSP Algorithm

Algorithm Sketch

1. Initialize state ✓
2. In each round r:
   1. **If the distance $\delta(u,v)$ and σ_{uv} have converged for some u, send out $(\delta(u,v), u, \sigma_{uv})$.**
   2. Process incoming messages; update state to reflect current best known δ, σ values

APSP Algorithm

Issue #2

How do we know $\delta(u, v), \sigma(u, v)$ have converged?

APSP Algorithm

Issue #2

How do we know $\delta(u, v), \sigma(u, v)$ have converged?

Claim

If at a round r, there is a u such that $r = d(u, v) + \ell_v^r(d(u, v), u)$, then $\delta(u, v)$ and $\sigma(u, v)$ have converged, where $\ell_v^r(\delta(u, v), u)$ is the index of $(\delta(u, v), u)$ in L_v at round r, and $d(u, v)$ is the current distance estimate.

APSP Algorithm

Claim

If at a round r, there is a u such that $r = d(u,v) + \ell_v^r(d(u,v), u)$, then $\delta(u,v)$ and $\sigma(u,v)$ have converged, where $\ell_v^r(\delta(u,v), u)$ is the index of $(\delta(u,v), u)$ in L_v at round r, and $d(u,v)$ is the current distance estimate.

APSP Algorithm

Claim

If at a round r, there is a u such that $r = d(u,v) + \ell_v^r(d(u,v),u)$, then $\delta(u,v)$ and $\sigma(u,v)$ have converged, where $\ell_v^r(\delta(u,v),u)$ is the index of $(\delta(u,v),u)$ in L_v at round r, and $d(u,v)$ is the current distance estimate.

Lemma

If an entry $(d(s,v),s)$ is inserted in L_v at position k in round r, then $d(s,v) + k > r$

APSP Algorithm

Claim

If at a round r, there is a u such that $r = d(u,v) + \ell_v^r(d(u,v),u)$, then $\delta(u,v)$ and $\sigma(u,v)$ have converged, where $\ell_v^r(\delta(u,v),u)$ is the index of $(\delta(u,v),u)$ in L_v at round r, and $d(u,v)$ is the current distance estimate.

Lemma

If an entry $(d(s,v),s)$ is inserted in L_v at position k in round r, then $d(s,v) + k > r$

Proof of Claim.

- By induction on hops h b/w s and v in D_{sv}, dag of shortest paths between them. Base case $h = 0$ trivial by intialization.
- Suppose D_{sv} has $h+1$ hops. Then, for any shortest path, consider v's predecessor u, whose shortest path has at most h hops.

□

APSP Algorithm

Claim

If at a round r, there is a u such that $r = d(u,v) + \ell_v^r(d(u,v),u)$, then $\delta(u,v)$ and $\sigma(u,v)$ have converged, where $\ell_v^r(\delta(u,v),u)$ is the index of $(\delta(u,v),u)$ in L_v at round r, and $d(u,v)$ is the current distance estimate.

Proof of Claim Cont'd.

- By induction on hops h b/w s and v in D_{sv}, dag of shortest paths between them. Base case $h = 0$ trivial by intialization.
- Suppose D_{sv} has $h+1$ hops. Then, for any shortest path, consider v's predecessor u, whose shortest path has at most h hops.
- By IH, u will send $(\delta(s,u),u)$ at some round r, and by lemma v will insert it at position k satisfying $r < k + \delta(s,v)$ if not already present, and update state appropriate if it is.

APSP Algorithm

Claim

If at a round r, there is a u such that $r = d(u,v) + \ell_v^r(d(u,v),u)$, then $\delta(u,v)$ and $\sigma(u,v)$ have converged, where $\ell_v^r(\delta(u,v),u)$ is the index of $(\delta(u,v),u)$ in L_v at round r, and $d(u,v)$ is the current distance estimate.

Proof of Claim Cont'd.

- By induction on hops h b/w s and v in D_{sv}, dag of shortest paths between them. Base case $h = 0$ trivial by intialization.
- Suppose D_{sv} has $h+1$ hops. Then, for any shortest path, consider v's predecessor u, whose shortest path has at most h hops.
- By IH, u will send $(\delta(s,u),u)$ at some round r, and by lemma v will insert it at position k satisfying $r < k + \delta(s,v)$ if not already present, and update state appropriate if it is.
- This holds for all predecessors of v in D_{sv}, and by lemma all updates occur in rounds before final value is sent out.

APSP Algorithm

Lemma

If an entry $(d(s, v), s)$ *is inserted in* L_v *at position* k *in round* r*, then* $d(s, v) + k > r$

Proof of lemma.

- By induction. Round $r = 1$, $d(s, v) = 1$ necessary, and $k \geq 1$, thus $d(s, v) + k \geq 2 \geq 1 = r$.

$\square$

APSP Algorithm

Lemma

If an entry $(d(s, v), s)$ *is inserted in* L_v *at position* k *in round* r*, then* $d(s, v) + k > r$

Proof of lemma.

- By induction. Round $r = 1$, $d(s, v) = 1$ necessary, and $k \geq 1$, thus $d(s, v) + k \geq 2 \geq 1 = r$.
- Suppose r first round where $(d(s, v), s)$ inserted s.t $d(s, v) + k \leq r$. Then, if this message arrived from u, it must've satisfied $d(s, u) + i > r - 1$ by IH.

□

APSP Algorithm

Lemma

If an entry $(d(s,v),s)$ is inserted in L_v at position k in round r, then $d(s,v)+k>r$

Proof of lemma.

- By induction. Round $r=1$, $d(s,v)=1$ necessary, and $k\geq 1$, thus $d(s,v)+k\geq 2\geq 1=r$.
- Suppose r first round where $(d(s,v),s)$ inserted s.t $d(s,v)+k\leq r$. Then, if this message arrived from u, it must've satisfied $d(s,u)+i>r-1$ by IH.
- Thus, $d(s,u)+1+i>r$, and so $d(s,v)+i>r$. To complete proof, observe $k\geq i$ since all in position $1,\ldots,i-1$ must've been sent to v before round r.

$\square$

APSP Algorithm

Algorithm Sketch

1. $L_v := [(0, v)]$, $\delta(v, v) := 0$, $\sigma(v, v) = 1$ ✓
2. In each round r:
   1. If there is some u such that $r = d(u, v) + \ell_v^r(d(u, v), u)$ then send out $(\delta(u, v), u, \sigma_{uv})$ to $\mathrm{out}(v)$ ✓
   2. Process incoming messages; update state to reflect current best known δ, σ values

APSP Algorithm

Issue #3

How do we process incoming messages?

APSP Algorithm

Issue #3

How do we process incoming messages?

Processing Incoming Messages

Suppose that at round r we receive a message $(\delta(s,u), s, \sigma(s,u))$

1. If $(d(s,v), s) \notin L_v$
   1. Add $(d(s,v), s)$ to L
   2. $d(s,v) := \delta(s,u) + 1, \quad \sigma(s,v) := \sigma(s,u),\ P_s(v) := \{u\}$
2. Else if there's $(d(s,v), s) \in L_v$ such that $d(s,v) = \delta(s,u) + 1$
   1. Update $\sigma(s,v) := \sigma(s,v) + 1, \quad P_s(v) := P_s(v) \cup \{u\}$
3. Else if there's $(d(s,v), s) \in L_v$ such that $d(s,v) > \delta(s,u) + 1$
   1. Replace $(d(s,v), s)$ appropriately such that $d(s,v) := \delta(s,u) + 1$, $\sigma(s,v) := \sigma(s,u),\ P_s(v) := \{u\}$

APSP Algorithm

Algorithm Sketch

1. $L_v := [(0, v)]$, $\delta(v, v) := 0$, $\sigma(v, v) = 1$ ✓
2. In each round r:
   1. If there is some u such that $r = d(u, v) + \ell_v^r(d(u, v), u)$ then send out $(\delta(u, v), u, \sigma_{uv})$ to $\text{out}(v)$ ✓
   2. If $(d(s, v), s) \notin L_v$ ✓
      1. Add $(d(s, v), s)$ to L
      2. $d(s, v) := \delta(s, u) + 1$, $\sigma(s, v) := \sigma(s, u)$, $P_s(v) := \{u\}$
   3. Else if there's $(d(s, v), s) \in L_v$ such that $d(s, v) = \delta(s, u) + 1$
      1. Update $\sigma(s, v) := \sigma(s, v) + 1$, $P_s(v) := P_s(v) \cup \{u\}$
   4. Else if there's $(d(s, v), s) \in L_v$ such that $d(s, v) > \delta(s, u) + 1$
      1. Replace $(d(s, v), s)$ appropriately such that $d(s, v) := \delta(s, u) + 1$, $\sigma(s, v) := \sigma(s, u)$, $P_s(v) := \{u\}$

Are we done?

APSP Algorithm

Algorithm Sketch

1. $L_v := [(0, v)]$, $\delta(v, v) := 0$, $\sigma(v, v) = 1$ ✓
2. In each round r:
   1. If there is some u such that $r = d(u, v) + \ell_v^r(d(u, v), u)$ then send out $(\delta(u, v), u, \sigma_{uv})$ to $\text{out}(v)$ ✓
   2. If $(d(s, v), s) \notin L_v$ ✓
      1. Add $(d(s, v), s)$ to L
      2. $d(s, v) := \delta(s, u) + 1, \quad \sigma(s, v) := \sigma(s, u), \; P_s(v) := \{u\}$
   3. Else if there's $(d(s, v), s) \in L_v$ such that $d(s, v) = \delta(s, u) + 1$
      1. Update $\sigma(s, v) := \sigma(s, v) + 1, \quad P_s(v) := P_s(v) \cup \{u\}$
   4. Else if there's $(d(s, v), s) \in L_v$ such that $d(s, v) > \delta(s, u) + 1$
      1. Replace $(d(s, v), s)$ appropriately such that $d(s, v) := \delta(s, u) + 1$, $\sigma(s, v) := \sigma(s, u)$, $P_s(v) := \{u\}$

Termination

How do we know when to terminate?

APSP Algorithm

Termination

How do we know when to terminate?

Strategy #1

We claimed that convergence occurs when $r = d(s, v) + \ell_v^r(d(s, v), s)$.
But,

$$\max_{s,v,r} [d(s, v) + \ell_v^r(d(s, v), s)] \leq 2n$$

Thus we can have every processor run and terminate after $2n$ rounds.

APSP Algorithm

Algorithm Sketch

1. $L_v := [(0, v)]$, $\delta(v, v) := 0$, $\sigma(v, v) = 1$ ✓
2. In each round $1 \leq r \leq 2n$:
    1. If there is some u such that $r = d(u, v) + \ell_v^r(d(u, v), u)$ then send out $(\delta(u, v), u, \sigma_{uv})$ to $\text{out}(v)$ ✓
    2. If $(d(s, v), s) \notin L_v$ ✓
        1. Add $(d(s, v), s)$ to L
        2. $d(s, v) := \delta(s, u) + 1, \quad \sigma(s, v) := \sigma(s, u),\ P_s(v) := \{u\}$
    3. Else if there's $(d(s, v), s) \in L_v$ such that $d(s, v) = \delta(s, u) + 1$
        1. Update $\sigma(s, v) := \sigma(s, v) + 1, \quad P_s(v) := P_s(v) \cup \{u\}$
    4. Else if there's $(d(s, v), s) \in L_v$ such that $d(s, v) > \delta(s, u) + 1$
        1. Replace $(d(s, v), s)$ appropriately such that $d(s, v) := \delta(s, u) + 1$, $\sigma(s, v) := \sigma(s, u)$, $P_s(v) := \{u\}$

APSP Analysis

- In order to fully show correctness, and analyze the complexity of the algorithm we also need the following lemma(s)

APSP Analysis

- In order to fully show correctness, and analyze the complexity of the algorithm we also need the following lemma(s)

Lemma

At each vertex v, the distance values in the sequence of messages sent by v are non-decreasing

Proof.

- By contradiction. Suppose v sends d_{sv} in round r, and then d in a later round with $d < d_{sv}$.
- Then, d must have been received in round $r' \geq r$, as o/w would've been sent before d_{sv}.

$\square$

APSP Analysis

- In order to fully show correctness, and analyze the complexity of the algorithm we also need the following lemma(s)

Lemma

At each vertex v, the distance values in the sequence of messages sent by v are non-decreasing

Proof.

- By contradiction. Suppose v sends d_{sv} in round r, and then d in a later round with $d < d_{sv}$.
- Then, d must have been received in round $r' \geq r$, as o/w would've been sent before d_{sv}.
- If d_{sv} inserted at k, then d must be inserted at $k' \leq k$.

$\square$

APSP Analysis

- In order to fully show correctness, and analyze the complexity of the algorithm we also need the following lemma(s)

Lemma

At each vertex v, the distance values in the sequence of messages sent by v are non-decreasing

Proof.

- By contradiction. Suppose v sends d_{sv} in round r, and then d in a later round with $d < d_{sv}$.
- Then, d must have been received in round $r' \geq r$, as o/w would've been sent before d_{sv}.
- If d_{sv} inserted at k, then d must be inserted at $k' \leq k$.
- But then $d + k' < d_{sv} + k = r \leq r'$. This contradicts earlier lemma ($r' < d + k'$).

□

APSP Analysis

- In order to fully show correctness, and analyze the complexity of the algorithm we also need the following lemma(s)

Lemma

At each vertex v, the distance values in the sequence of messages sent by v are non-decreasing

- Implies at most one message sent per source vertex by each vertex.

APSP Analysis

Complexity

1. Round complexity: $2n$
2. Communication Complexity: $\mathcal{O}(mn)$

APSP Analysis

Complexity

1. Round complexity: $2n$
2. Communication Complexity: $\mathcal{O}(mn)$

- Can we improve round complexity?
- $2n$ coarse upper bound. For certain graphs nodes might terminate early and thus be wasting time!

APSP Analysis

Complexity

1. Round complexity: $2n$
2. Communication Complexity: $\mathcal{O}(mn)$

- Can we improve round complexity?
- $2n$ coarse upper bound. For certain graphs nodes might terminate early and thus be wasting time!
- Idea: When a node is finished, maybe it should notify others and somehow stop early.

APSP Early Termination

Issue #4

How do we terminate early safely?

Early Termination

- Perform leader election to elect v_1, the vertex with smallest UID.
- Have v_1 run BFS (in parallel with everything else) to construct a spanning tree.
- When a node v and all of its children finish (i.e., $|L_v| = n$), it notifies its parent
- Once all of v_1's children finish, broadcast stop message to all children.

APSP Early Termination

Issue #4

How do we terminate early safely?

Early Termination

- Perform leader election to elect v_1, the vertex with smallest UID.
- Have v_1 run BFS (in parallel with everything else) to construct a spanning tree.
- When a node v and all of its children finish (i.e., $|L_v| = n$), it notifies its parent
- Once all of v_1's children finish, broadcast stop message to all children.

Complexity

1. Round complexity (claim): $\min\left(2n, n + \mathcal{O}(D)\right)$
2. Communication Complexity: $\mathcal{O}(mn + 4m)$

APSP Early Termination

Claim

For a strongly connected digraph G with bounded diameter $D < n/5$ the algorithm terminates in $n + \mathcal{O}(D)$ rounds.

APSP Early Termination

Claim

For a strongly connected digraph G with bounded diameter $D < n/5$ the algorithm terminates in $n + \mathcal{O}(D)$ rounds.

Proof.

v_1 receives its final stopping message exactly D rounds after the last vertex has finished. In particular, the last vertex is a vertex on one end of a path of length D. The final message for this vertex to receive is scheduled at round $n + D$. Thus, within $n + 3D$ rounds, all termination messages are sent. □

References

Lenzen, C. and Peleg, D. (2013).
Efficient distributed source detection with limited bandwidth.
In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 375–382.

Pontecorvi, M. and Ramachandran, V. (2018).
Distributed algorithms for directed betweenness centrality and all pairs shortest paths.
arXiv preprint arXiv:1805.08124.

Questions?

Thank You!