Brief Announcement: Distributed Single-Source Reachability

Mohsen Ghaffari
MIT
ghaffari@mit.edu

Rajan Udwani
MIT
rudwani@mit.edu

Abstract

In the *directed single-source reachability* problem, input is a directed graph $G = (V, E)$ and a source $s \in V$, and the objective is to identify nodes t for which there is a directed path in G from s to t. Recently Nanongkai [2] presented a distributed algorithm that solves this problem in $\tilde{O}(D + \sqrt{n}D^{1/2})$ rounds, where D and n respectively denote the network diameter and the number of nodes.

This note presents an algorithm that improves the round complexity to $\tilde{O}(D + \sqrt{n}D^{1/4})$, thus getting closer to the $\tilde{\Omega}(D + \sqrt{n})$ lower bound of Das Sarma et al. [1].

1 Problem

Given a directed graph $G = (V, E)$ and a source node s, we want a distributed algorithm that identifies nodes t for which there is a directed path from s to t. We use n and D to denote, respectively, the number of nodes and the *undirected* hop-diameter of G.

The communication model we use is the standard CONGEST model, where communications occur in synchronous rounds and per round, $O(\log n)$ bits can be sent over each edge. Moreover, following Nanongkai [2], we assume that the communication network itself is undirected.

2 Nanongkai's Reachability Algorithm

Nanongkai [2] gives a distributed algorithm for $1 + o(1)$ approximation of undirected weighted Single-Source Shortest Path (SSSP) problem, in $\tilde{O}(D + \min\{n^{2/3}, \sqrt{n}D^{1/4}\})$ rounds. He then notes that some of the components in his algorithm can be used for the single-source reachability problem, yielding a $\tilde{O}(D + \min\{n^{2/3}, \sqrt{nD}\})$ round algorithm. This reachability algorithm is roughly as follows:

1. **Sampling:** Every node v *samples* itself w.p. α/n where $\alpha = \max\{n^{1/3}, \sqrt{n/D}\}$. Let S be the set of sampled nodes and also add source s to S. We know $|S| = \tilde{O}(\alpha)$, w.h.p.

2. **h-hop Distances and Skeleton:** Every S-node runs an h-hop outgoing BFS, for $h = \Theta(\frac{n \log n}{\alpha}) = \tilde{\Theta}(\min\{n^{2/3}, \sqrt{nD}\})$. These BFSs are performed all (essentially) in parallel, in $\tilde{O}(h + n/h)$ rounds, using the standard *random delays* technique to control the congestion on edges. Given the range of h, this is equal to $\tilde{O}(h)$ rounds. At the end of the process, every node u knows for each sampled node $v \in S$ whether there is directed path with at most h-hops from v to u or not. Construct a skeleton graph $G' = (S, E_S)$ where for each two nodes $v, u \in S$, node v puts a directed edge from u to v if there is a directed path with at most h-hops from u to v. Using a similar method but with running the BFSs in the reverse direction, node u will also know if it has a directed path to v with at most h-hops. Thus, each node in S knows its incoming and outgoing G' neighbors.

3. **Total Broadcast, or BFS on Skeleton:** If $\alpha = n^{1/3}$, broadcast all the skeleton edges, in $\tilde{O}(D + \alpha^2)$ rounds. From these, each node t can locally figure out whether it is reachable from s. If $\alpha = \sqrt{n/D}$,

simulate α rounds of directed outgoing BFS from s on the skeleton G', in $O(D\alpha)$ rounds. Then broadcast the names of all sampled nodes reachable from s, in at most $\tilde{O}(D+\alpha)$ rounds.

3 An $\tilde{O}(D+\sqrt{n}D^{1/4})$ algorithm

In this section, we explain our algorithm which leads to the following result:

Theorem 3.1. *There is a distributed algorithm that solves the single-source reachability problem in $\tilde{O}(D+\min\{n^{2/3}, \sqrt{n}D^{1/4}, \})$ rounds of the* CONGEST *model, with high probability.*

We only describe the algorithm with round complexity $\tilde{O}(D+\sqrt{n}D^{1/4})$. Interleaving it with Nanongkai's $\tilde{O}(D+n^{2/3})$ algorithm, one can easily achieve the best of the two. We do steps 1 and 2 as above with $\alpha = \sqrt{n}/D^{1/4}$. Then, instead of running the full BFS on the Skeleton graph G' which takes $O(D\alpha)$ rounds to simulate, we will perform a BFS-like procedure with *active short-cutting*.

In the algorithm, we only talk about the sampled nodes S, and transmissions over the virtual graph G' will be broadcast to all nodes. We call a node *active* if it knows a path from s to itself, in G'. Initially, only s is active. The algorithm proceeds in *phases* and in each phase, a few more nodes become active. At the end, all S-nodes reachable from source s will be active and their ids will be publicly known. Hence, each node of the graph G can determine if it's reachable from s. The algorithm works as follows:

Reachability Algorithm on G', to be simulated via broadcasts on G:

1. Activate node s.
2. Repeat phases, where in each phase, the following steps occur in sequence:
   (a) *Announcing Newly Activated Nodes:* All nodes activated in the previous phase broadcast their ids. In the first phase, announce the id of s. If no id is announced, terminate.
   (b) *Shortcutting:* Every node with at most $\sqrt{D}$ inactive outgoing neighbors broadcasts the corresponding ($\leq \sqrt{D}$) outgoing edges.
   (c) *New Activations:* Each inactive node u that discovers that it is reachable from s becomes active. This happens if the union of the set of G'-edges broadcasted so far with the incoming edges of u, known to u itself, include a path from an active node to u.

Notice that the length of a phase, and even the length of each of the steps, is not fixed and it depends on the number of messages broadcast in the phase. However, simply by adding a step-start and step-end broadcast from s, which can be done in $O(D)$, we can keep all nodes consistently aware of which step is happening. Furthermore, the only steps that use communication are steps $2(a)$ and $2(b)$. Using standard broadcast routines, each of these steps can be implemented in $O(D+k)$ rounds, where k is the number of messages to be broadcast in that step.

Lemma 3.2. *Once the algorithm terminates, a node is active if and only if it is reachable from s.*

Proof. A node becomes active only if it discovers that there is a path from s to it, or from a node activated in an earlier phase. Hence, activated nodes are reachable from s. On the other hand, so long as there is a node u that is reachable from s but it has not become active, in each phase, at least one more node along the path from s to u becomes active in step $2(c)$. This means that the algorithm will not terminate at least until the node u has been activated as well. That is, all reachable nodes will be activated. □

Lemma 3.3. *The algorithm terminates after at most $\tilde{O}(D+\sqrt{n}D^{1/4})$ rounds.*

Proof. Clearly the i^{th} phase takes $O(D + b_i)$ rounds where b_i is the total number of messages broadcast in steps $2(b)$ and $2(c)$. That is, the number of nodes activated in phase $i-1$ plus the number of edges reported from nodes that have less than $\sqrt{D}$ inactive neighbors. Hence, the summation of the running time of all phases has an $O(D)$ term for each phase, plus each node can contribute at most $\sqrt{D}+1$ to the summation of b_i values over all phases. Thus, the total running time is at most

$$p \cdot O(D) + \tilde{O}(\sqrt{n}/D^{1/4}) \cdot (\sqrt{D}+1) = p \cdot O(D) + \tilde{O}(\sqrt{n}D^{1/4}),$$

where p is the total number of phases. Therefore, to complete the proof, it suffices to show that $p = \tilde{O}(\sqrt{n}/D^{3/4})$.

We prove this by roughly speaking showing that in each phase the size of the set of active nodes grows by at least $\sqrt{D}$, amortized. Hence, the total number of phases p is at most $\tilde{O}(\sqrt{n}/D^{1/4})/\sqrt{D} = \tilde{O}(\sqrt{n}/D^{3/4})$.

Consider an outgoing BFS in G' rooted in s and let us call the set of nodes at a given depth i of this BFS—i.e., distance i from s—its i^{th} layer and use L_i to denote this set. Define the first block to be the set of nodes in layers 0 to B_1, where B_1 is the layer number such that $\sum_{i=0}^{B_1} |L_i| \leq \sqrt{D}$ and $\sum_{i=0}^{B_1+1} |L_i| > \sqrt{D}$. Then, notice that in the first phase, all nodes of layers 0 to $B_1 - 1$ will announce all their outgoing edges (if $B_1 \geq 1$). This is simply because, these nodes cannot have any outgoing edge to layers above B_1. Therefore, at the end of phase one, all nodes of the first block are active, which means that at the end of phase 2, all nodes in blocks 0 to B_1+1 are active. That is, after 2 phases, at least $\sqrt{D}$ nodes became active. We can now repeat a similar argument for the next phases. Particularly, define B_j to be such that $\sum_{i=0}^{B_j} |L_i| \leq j\sqrt{D}$ and $\sum_{i=0}^{B_j+1} |L_i| > j\sqrt{D}$. Now that we know that by the end of phase 2, all nodes in layers 0 to $B_1 + 1$ are active, using a similar argument as above, we can conclude that by the end of phase 3, all nodes in layers 0 to B_2 are active, which means at the end of phase 4, all nodes in layers 0 to $B_2 + 1$ are active. Repeating this argument, we know all nodes in layers 0 to B_j are active by the end of phase $2j - 1$. Since there are only $\tilde{O}(\sqrt{n}/D^{1/4})$ nodes, there are only $\tilde{O}(\sqrt{n}/D^{3/4})$ blocks, which means by the end of phase $\tilde{O}(\sqrt{n}/D^{3/4})$, all reachable nodes are active. □

4 Discussion: Extension to the Weighted (Directed) Shortest Path problem

A natural question about the algorithm above is whether it can be modified to solve the more general problem of finding the shortest paths from s on a weighted virtual directed graph G', with similar runtime.

Note that as described above the algorithm need not even find paths from s with minimum hops to reachable nodes. In fact the very process of active shortcutting, which helps cut down the runtime, hampers this by possibly reaching nodes, that are otherwise a few hops away from s, through a longer path which was broadcast/revealed sooner due to shortcutting.

An intermediate step would be to extend the algorithm to find shortest paths on G' when all edge weights are equal. We claim without proof that in fact with a minor change, our algorithm can address this problem with the same runtime guarantee. The idea is essentially to not activate nodes until we are sure that the shortest path has been found, which would behave like a BFS for the virtual graph. However, the primary obstacle in generalizing this idea further is that our runtime analysis relies on the number of nodes in graph G' and this seems to obstructs us (or at least complicate) using ideas such as the *rounding technique* in [2, Algorithm 3.2].

References

[1] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 363–372, 2011.

[2] D. Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proc. of the Symp. on Theory of Comp. (STOC)*, 2014, to appear.