Brief Announcement: Distributed 3/2-Approximation of the Diameter

Preliminary version of a brief announcement to appear at DISC'14

Stephan Holzer*
MIT
holzer@mit.edu

David Peleg †
Weizmann Institute
david.peleg@weizmann.ac.il

Liam Roditty ‡
Bar-Ilan University
liamr@macs.biu.ac.il

Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch

Contact: Stephan Holzer, +1-617-258-8682, http://people.csail.mit.edu/holzer/

Abstract

We present an algorithm that computes a 3/2-approximation of the diameter of a graph. This algorithm takes time $\mathcal{O}(\sqrt{n \log n} + D)$ in the CONGEST model, where in each synchronous round, every node can transmit a different (but short) message to each of its neighbors. Due to a lower bound stated for graphs of small diameter in [2] this algorithm is optimal. We extend this algorithm to compute a $(3/2 + \varepsilon)$-approximation of the diameter in time $\mathcal{O}(\sqrt{\frac{n}{D\varepsilon} \log n} + D)$.

1 Introduction

The diameter is one of the most fundamental properties of a graph. This is true especially in distributed computing, where it is used to define which problems are local (as their runtime is independent of the diameter) and problems that are global (as their runtime is lower bounded by the diameter). Independently of each other the authors of [3] and [7] presented algorithms that compute the diameter in time $\mathcal{O}(n)$. This runtime matches a lower bound of [2] stated for networks of small diameter. Besides this algorithm, the authors of [3] and [7] also state algorithms to approximate the diameter. E.g. [3] provides a 3/2-approximation in time $\mathcal{O}(n^{3/4} + D)$ that was later improved by [5] to $\mathcal{O}(\sqrt{n} \log n + D)$. This matches a lower bound of $\tilde{\Omega}(\sqrt{n} + D)$ derived in [2] for any $(3/2 - \varepsilon)$-approximation in small diameter networks (and constant ε). However, when the $(3/2 - \varepsilon)$-approximation lower bound of [2] is analyzed in more detail and generalized to networks of arbitrary diameter (and arbitrary ε) it turns into a $\Omega(\sqrt{(n/D)\varepsilon} + D)$ lower bound.

In this brief announcement we present an algorithm to 3/2-approximate the diameter in time $\mathcal{O}(\sqrt{n \log n} + D)$ that we obtain by combining results of [3, 7] with ideas from [8]. This solution is a factor $\sqrt{\log n}$ faster than the one achieved in [5] and uses a different approach. Our different approach is of interest as we show how to extend it to compute a $(3/2 + \varepsilon)$-approximation to the diameter in time $\mathcal{O}(\sqrt{n/(D\varepsilon) \log n} + D)$. Thus we essentially match the $\Omega(\sqrt{(n/D)\varepsilon} + D)$ lower bound (when the approximation factor is allowed to differ by a small value).

*Part of this work has been done at ETH Zurich. At MIT the author was supported by the following grants: AFOSR Contract Number FA9550-13-1-0042, NSF Award 0939370-CCF, NSF Award CCF-1217506, NSF Award number CCF-AF-0937274.

†Supported in part by grants from the Israel Science Foundation, the United-States - Israel Binational Science Foundation and the Israel Ministry of Science.

‡Work supported by the Israel Science Foundation (grant no. 822/10)

2 Model and Basic Definitions

Model: The CONGEST model [6] is a message passing model with limited bandwidth. In this model a network is represented by an undirected unweighted graph $G = (V, E)$ where nodes V correspond to processors (computers or routers). Two nodes are connected by an edge from set E if they can communicate directly with each other. Each node in V has a unique identifier (ID) in the range of $\{1, \ldots, 2^{\mathcal{O}(\log |V|)}\}$. Nodes initially know the IDs of nodes in their immediate neighborhood. Communication over edges in E is synchronous. Every node can send $B = \mathcal{O}(\log |V|)$ bits of information over all its edges in one round of communication. A node can send different messages of size B to each of its neighbors and receive different messages from each of its neighbors in every round. We are interested in time complexity, i.e., the number of communication rounds required by a distributed algorithm to solve a problem. Subsequently, internal computations are neglected.

We denote the number $|V|$ of nodes of a graph by n, and the number $|E|$ of its edges by m. For simplicity, for $u \in V$, we sometimes use u also to refer to u's ID, when this is clear from the context. Let us denote by $d(u, v)$ the (hop-)distance of nodes u and v in G, which is the length of a shortest u-v path in G. A k-dominating set for a graph G is a subset $\mathcal{DOM}$ of vertices with the property that for every $v \in V$ there is some $u \in \mathcal{DOM}$ at distance of at most k to v.

Definition 1 (Eccentricity, diameter)**.** *The eccentricity of a node $u \in V$ is $ecc(u) := \max_{v \in V} d(u, v)$, namely, the maximum distance to any other node in the graph. The diameter $D := \max_{u \in V} ecc(u) = \max_{u,v \in V} d(u, v)$ of a graph G is the maximum distance between any two nodes of the graph.*

Definition 2 (Approximation)**.** *Given an optimization problem P, denote by OPT the value of the optimal solution for P and by sol_A the value of the solution of an algorithm A for P. Let $\rho \geq 1$. We say A computes a ρ-approximation (multiplicative approximation) for P if $OPT \leq sol_A \leq \rho \cdot OPT$ for any input.*

Definition 3 (APSP, S-SP)**.** *Let $G = (V, E)$ be a graph. The all pairs shortest paths (APSP) problem is to compute the shortest paths between any pair of vertices in $V \times V$. In the S-Shortest Paths (S-SP) problem, we are given a set $S \subseteq V$ and need to compute the shortest paths between any pair of vertices in $S \times V$.*

3 A 3/2-Approximation to the Diameter

We describe an $\mathcal{O}(\sqrt{n \log n} + D)$-time algorithm that computes a 3/2-approximation to the diameter. This algorithm is based on a sequential algorithm that was recently presented in [8] which in turn extends an algorithm of Aingworth et al. [1].

Let $C_k(w)$ denote the set of k closest vertices to w visited by a (partial) BFS starting in w that stops after visiting k nodes (ties are broken arbitrarily, e.g. by lexicographical order in the tree's topology). This set $C_k(w)$ is computed only for a single vertex w (e.g. with smallest ID). Algorithm 1 presented below is a distributed version of the non-distributed algorithm of [8]. The authors of [8] provide more intuition behind Algorithm 1 and a proof of correctness.

Theorem 1. *Algorithm 1 computes a 3/2-approximation of the diameter w.h.p. in $\mathcal{O}(\sqrt{n \log n} + D)$ time.*

Proof. In [8], Theorem 1, it is stated that Algorithm 1 of [8] computes the desired approximation. Our Algorithm 1 is Algorithm 1 of [8] adapted to the distributed setting. We analyze the runtime of our algorithm: The first step can be done locally by every node and w.h.p. creates a set S of size $\Theta((n/s) \log n)$. In step two, we compute S-SP in time $\mathcal{O}(|S| + D) = \mathcal{O}((n/s) \log n + D)$ by using the S-SP algorithm from Section 6.1. in [3]. The results of this can be used by each

Algorithm 1 Distributed version of [8] as executed by each node $v \in G$.
Output: 3/2-approximation to the diameter of G

1: **each** node v **joins** set S with probability $\frac{\log n}{s}$;
2: **compute** a BFS from **each** node in S;
3: **for every** $v \in V$, **compute** $p_S(v) :=$ the closest node in S to v;
4: $w := \arg\max_{v \in V} d(v, p_S(v))$;
5: **compute** a BFS tree from w as well as $C_s(w)$;
6: **for every** $v \in C_s(w)$, **compute** a BFS tree from v;
7: **return** the maximum depth of any BFS tree that was computed;

node internally to solve step 3 without communication. The node w in Line 4 can be found by max-aggregation in time $\mathcal{O}(D)$. Computing BFS_w and $C_s(w)$ in Line 5 can be done in $\mathcal{O}(D)$ as well. To compute $C_s(w)$ node w essentially aggregates information on how many nodes are in each level of the BFS_w as e.g. done in Algorithm `Diam_DOM` in [4]. From this information w computes an i such that $|N_i(w)| < s \leq |N_{i+1}(w)|$. Next each node at level i tells its parent how many nodes at level $i+1$ are in its subtree. Accordingly the nodes in level $i-1$ proceed in the same way and so on. Based on this information, exactly $s - |N_i(w)|$ nodes can be selected in level $i+1$ in time $\mathcal{O}(D)$.

The next line can be realized by computing $C_s(w)$-SP in time $\mathcal{O}(s+D)$. The return value can be found by a max-aggregation in time $\mathcal{O}(D)$. Thus the total time complexity is $\mathcal{O}(s + (n/s)\log n + D)$. By choosing $s := \sqrt{n \log n}$ we obtain the desired runtime of $\mathcal{O}(\sqrt{n \log n} + D)$. □

4 A $(3/2+\varepsilon)$-Approximation to the Diameter

Theorem 2. *For any $0 < \varepsilon \leq 1/3$, a $(3/2+\varepsilon)$-approximation of the diameter can be computed w.h.p. in $\mathcal{O}\left(\sqrt{\frac{n}{D}\varepsilon^{-1}\log n} + D\right)$ time.*

To show this result, we extend and modify Algorithm 1 slightly by using ideas of the algorithm to $(1+\varepsilon)$-approximate the diameter presented in [3], Section 6.2. First we compute a 2-approximation D' of D by executing a BFS from the node with smallest ID . Next we set $\varepsilon' := \frac{\varepsilon}{3/2-\varepsilon}$ and compute a $\frac{\varepsilon'}{8}D'$-dominating set $\mathcal{DOM}$ of size at most $8n/(\varepsilon' D')$ in Line 3. To compute this dominating set we can use Algorithm `Diam_DOM` presented in [4]. Now we execute Algorithm 1 restricted to the nodes in $\mathcal{DOM}$ in the sense that other nodes only implicitly participate (mainly by forwarding messages) as described below. In more detail, we first compute a set $S \subset \mathcal{DOM}$ by asking each node in $\mathcal{DOM}$ to join S with probability $\frac{\log n}{s}$, where s is chosen later. Therefore S is of size $\Theta\left(\frac{|\mathcal{DOM}|\cdot\log n}{s}\right) = \Theta(\frac{n\log n}{\varepsilon' D' s})$ w.h.p.. Next, for each $v \in S$, we compute $BFS(v)$ in graph G. Based on this information each node $v \in \mathcal{DOM}$ can compute $p_S(v)$, a closest node in S to v. By a max-aggregation convergecast started in the node with smallest ID, we can identify a node $w \in \mathcal{DOM}$ of largest distance to S, that is a node $w \in \mathcal{DOM}$ such that $d(w, p_S(w)) \geq d(u, p_S(u))$ for all nodes $u \in \mathcal{DOM}$. Next we compute a set $C_s^{\mathcal{DOM}}(w)$, which is defined to be a set that consists of s closest nodes in $\mathcal{DOM}$ to w. This computation is done in a similar way to the (partial) BFS in previous section. Then we perform a $BFS(u)$ for each $u \in C_s^{\mathcal{DOM}}(w)$. The algorithm returns $\frac{3}{2(1+\varepsilon')}$ times the maximal depth of any BFS that was computed during the execution.

The proof of Theorem 2 can be found in the appendix and will be included in a full version of this brief announcement.

References

[1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.

[2] S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks Cannot Compute Their Diameter in Sublinear Time. In Y. Rabani, editor, *Proceedings of the 23rd annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012* , pages 1150–1162, 2012.

[3] S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In D. Kowalski and A. Panconesi, editors, *Proceedings of the 31st annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 2012, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 355–364, 2012.

[4] S. Kutten and D. Peleg. Fast distributed construction of small k-dominating sets and applications. *Journal of Algorithms*, 28(1):40–66, 1998.

[5] C. Lenzen and D. Peleg. Efficient distributed source detection with limited bandwidth. In P. Fatourou and G. Taubenfeld, editors, *Proceedings of the 32nd annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 2013, Montreal, Quebec, Canada, July 22-24, 2013*, pages 375–382, 2013.

[6] D. Peleg. *Distributed computing: a locality-sensitive approach.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, USA, 2000.

[7] D. Peleg, L. Roditty, and E. Tal. Distributed algorithms for network diameter and girth. In A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 660–672. Springer, Berlin & Heidelberg, Germany, 2012.

[8] L. Roditty and V. V. Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *Proceedings of the 45th annual ACM Symposium on Theory of Computing, STOC 2013, Palo Alto, California, USA, June 1-4, 2013*, pages 515–524, 2013.

Appendix

A Proof of Theorem 2

Algorithm 2 Distributed version of [8] as executed by each node $v \in G$.
Input: accuracy parameter ε Output: $(3/2+\varepsilon)$-approximation to the diameter of G

1: **compute** and **broadcast** $D' := 2 \cdot ecc(ID_{\min})$;
2: $\varepsilon' := \frac{\varepsilon}{3/2-\varepsilon}$; $k := \lfloor \varepsilon' D'/8 \rfloor$;
3: **compute** $\mathcal{DOM} :=$ k-dominating set of size at most $\max\{1, \lfloor n/(k+1) \rfloor\}$;
4: v **joins** set S with probability $\frac{\log n}{s}$;
5: **compute** a BFS from each node in S;
6: **For every** $v \in V$, **compute** $p_S(v) :=$ the closest node in S to v;
7: $w := \arg\max_{v \in V} d(v, p_S(v))$;
8: **compute** a BFS tree from w as well as $C_s^{\mathcal{DOM}}(w)$;
9: **For every** $v \in C_s^{\mathcal{DOM}}(w)$, **compute** a BFS tree from v;
10: **return** $\frac{3}{2(1+\varepsilon')}$ times the maximum depth of any BFS tree that was computed;

We already know from Theorem 5 in [8] that Algorithm 1 produces a 3/2-approximation when executed on G. We follow along the lines of their proof and adopt it to our modified algorithm to show that it computes a $(3/2+\varepsilon)$-approximation to the diameter.

Lemma 1. *Let $G = (V, E)$ be a graph with diameter $D = 3h + z$, where $h \geq 0$ and $z \in \{0, 1, 2\}$. The maximal depth of any BFS tree that was computed in Algorithm 2 is at least $2h(1-\varepsilon')$ w.h.p..*

Proof. Let $a, b \in V$ such that $d(a, b) = D$. Then there are nodes $a', b' \in \mathcal{DOM}$ such that $d(a', b') \geq D - 2k = D(1 - \frac{\varepsilon'}{2})$. Let $w \in \mathcal{DOM}$ be a vertex that satisfies $d(w, p_S(w)) \geq d(u, p_S(u))$ for all nodes $u \in \mathcal{DOM}$.

- Case 1, $(d(w, p_S(w)) \leq h)$: Then $d(a', p_S(a')) \leq h$. As the algorithm computes $BFS(v)$ for every $v \in S$, it follows that $BFS(p_S(a'))$ is computed as well. Since $ecc(a')$ is at least $D - k = D(1 - \varepsilon'/4)$, it follows that $ecc(P_S(a')) \geq ecc(a') - h = 2h + z - \frac{\varepsilon'}{4}(3h + z) \geq 2h(1 - \varepsilon')$ as required.

- Case 2, $(d(w, p_S(w)) > h)$: We can also assume that $ecc(w) < 2h(1-\varepsilon')$ since the algorithm computes $BFS(w)$ and if $ecc(w) \geq 2h(1 - \varepsilon')$ then it computes a BFS tree of depth at least $2h(1-\varepsilon')$ as required. Since $ecc(w) < 2h(1 - \varepsilon')$ it follows that $d(w, b') < 2h(1 - \varepsilon')$. Moreover, since $d(w, p_S(w)) > h$ we can conclude that S hits $C_s^{\mathcal{DOM}}(w)$ w.h.p., that is $S \cap C_s^{\mathcal{DOM}}(w) \neq \emptyset$. Therefore it must be the case that $C_s^{\mathcal{DOM}}(w)$ contains a node at distance greater h from w, and hence $N_h(w) \subseteq C_s^{\mathcal{DOM}}(w)$. This implies that there is a vertex $w' \in C_s^{\mathcal{DOM}}(w)$ on the path from w to b' such that $d(w, w') = h$ and hence $d(w', b') < 2h(1 - \varepsilon') - h = h - 2\varepsilon'$. Since $d(a', b') \geq D(1 - \frac{\varepsilon'}{2}) = (3h + z)(1 - \frac{\varepsilon'}{2})$, we also have that $d(a', w') \geq d(a', b') - d(w', b') > 2h(1 - \varepsilon')$. The algorithm computes $BFS(u)$ for every $u \in C_s^{\mathcal{DOM}}(w)$, and in particular, it computes $BFS(w')$, which has depths at least $d(a, w') \geq 2h(1 - \varepsilon')$.

□

Proof. (of Theorem 2). Correctness follows from Lemma 1 combined with the choice of ε' and the requirement that $\varepsilon \leq 1/3$: By multiplying the depth of the deepest BFS performed with $\frac{3}{2(1-\varepsilon')}$, we obtain an estimate $\hat{D}$ such that $D \leq \hat{D} \leq (\frac{3}{2} + \varepsilon)D$.

Analyzing the runtime is almost the same as in Theorem 1. We only need to add $O(D)$ for computing the k-dominating set in Line 1. The total runtime is $O(|S|+s+D) = O(\frac{n}{\varepsilon' D' s}+s+D)$. Choosing $s := \sqrt{\frac{n \log n}{\varepsilon' D'}}$ yields the desired runtime. □