A note on the size of query trees

Shai Vardi∗

August 3, 2017

Abstract

We consider query trees of graphs with degree bounded by a constant, d. We give simple proofs that the size of a query tree is constant in expectation and $2^{O(d)}\log n$ w.h.p.

1 Introduction

Let $G = (V, E)$ be an undirected graph whose degree is bounded by a constant $d$. We assume that $|V| = n$ is large: $d \ll n$. Let $r : V \rightarrow [0, 1]$ be a ranking function that assigns each vertex a real number between 0 and 1, uniformly at random. We call $r(v)$ $v$’s rank. Vertex ranks induce an orientation of the originally undirected edges - if $r(v) \leq r(u)$, the edge is oriented from $v$ to $u$; in case of equality, the edge is bi-directional.

A query tree $T_v$ is the set of vertices that are reachable from $v$ after the edges have been oriented (strictly speaking, it is not necessarily a tree, but we use the term “query tree” for consistency with e.g., [2, 3]).

The aim of this note is to give a simple proof that the query tree has size $2^{O(d)}\log n$ w.h.p.

Theorem 1.1. Let $G = (V, E)$ be a graph whose degree is bounded by $d$ and let $r : V \rightarrow [0, 1]$ be a function that assigns to each vertex $v \in V$ a number between 0 and 1 independently and uniformly at random. Let $T_{\text{max}}$ be the size of the largest query tree of $G$: $T_{\text{max}} = \max\{|T_v| : v \in V\}$. Then, for $L = 4(d + 1)$,

$$\Pr[|T_{\text{max}}| > 2^L \cdot 15L \log n] \leq \frac{1}{n^2}.$$ 

The proof of Theorem 1.1 is based on a proof in [5], and employs a quantization of the rank function. Let $f$ denote a quantization of $r$ (in other words $r(u) \geq r(v) \Rightarrow f(u) \geq f(v)$); let $T_v^f$ denote the query tree with respect to $f$. Then $T_v \subseteq T_v^f$. Therefore it suffices to bound $|T_v^f|$.

In Section 5, we give a brief discussion on query trees. The reader is referred to [7] for an introduction to local computation algorithms and role query trees play therein, and to [3] for an introduction to query trees and their use in the analysis of sublinear approximation algorithms.

2 Preliminaries

We denote the set $\{0, 1, \ldots, m\}$ by $[m]$. Logarithms are base $e$. Let $G = (V, E)$ be a graph. For any vertex set $S \subseteq V$, denote by $N(S)$ the set of vertices that are not in $S$ but are neighbors of some vertex in $S$: $N(S) = \{N(v) : v \in S\} \setminus S$. The length of a path is the number of edges it contains.

∗California Institute of Technology, Pasadena, CA, USA. E-mail: svardi@caltech.edu.
For a set $S \subseteq V$ and a function $f : V \to \mathbb{N}$, we use $S \cap f^{-1}(i)$ to denote the set $\{v \in S : f(v) = i\}$.

Let $G = (V, E)$ be a graph, and let $f : V \to \mathbb{N}$ be some function on the vertices. An adaptive vertex exposure procedure $A$ is one that does not know $f$ a priori. $A$ is given a vertex $v \in V$ and $f(v)$; $A$ iteratively adds vertices from $V \setminus S$ to $S$: for every vertex $u$ that $A$ adds to $S$, $f(u)$ is revealed immediately after $u$ is added. Let $S^t$ denote $S$ after the addition of the $t^{\text{th}}$ vertex. The following is a simple concentration bound whose proof is given for completeness.

**Lemma 2.1.** Let $G = (V, E)$ be a graph, let $L > 0$ be some constant, let $c = 15L$, and let $f : V \to [L]$ be a function chosen uniformly at random from all such possible functions. Let $A$ be an adaptive vertex exposure procedure that is given a vertex $v \in V$. Then, for any $\ell \in [L]$, the probability that there is some $t$, $c \log n \leq t \leq n$ for which $|S^t \cap f^{-1}(\ell)| > \frac{2|S^t|}{L}$ is at most $\frac{1}{n^2}$.

**Proof.** Let $v_j$ be the $j^{\text{th}}$ vertex added to $S$ by $A$, and let $X_j$ be the indicator variable whose value is 1 iff $f(v_j) = \ell$. For any $t \leq n$, $\mathbb{E}\left[\sum_{j=1}^{t} X_j\right] = \frac{t}{L}$. As $X_i$ and $X_j$ are independent for all $i \neq j$, by the Chernoff bound, for $c \log n \leq t \leq n$,

$$\Pr\left[\sum_{j=1}^{t} X_j > \frac{2t}{L}\right] \leq e^{\frac{-t}{2L}} \leq e^{-5 \log n}.$$ 

A union bound over all possible values of $t : c \log n \leq t \leq n$ completes the proof. \qed

### 3 Expectation

We first show that the expected size of a query tree is a constant depending only on $d$.

**Theorem 3.1 ([4]).** Let $G = (V, E)$ be a graph whose degree is bounded by $d$ and let $r : V \to [0, 1]$ be a function that assigns to each vertex $v \in V$ a number between 0 and 1 independently and uniformly at random. Let $T_v$ be the size of the query tree of some vertex $v \in V$. Then $\mathbb{E}[|T_v|] \leq e^d$, where the expectation is over the random choices of $r$.

**Proof.** Let $k > 0$ be an integer. For any path of length $k$ originating from $v$, the probability that the path is monotone decreasing is $\frac{1}{(k+1)!}$. There are at most $d^k$ such paths. Hence, by the union bound, the expected number of monotone paths of length $k$ originating from $v$ is at most $\frac{d^k}{(k+1)!}$, and the expected number of vertices in these paths is at most $\frac{(k+1)d^k}{(k+1)!} = \frac{d^k}{k!}$. Therefore, the expected total number of vertices in monotone non-increasing paths is at most

$$\sum_{k=0}^{\infty} \frac{d^k}{k!} = e^d,$$

which is an upper bound on the expected size of the query tree. \qed

### 4 Concentration

For the concentration bound, let $r : V \to [0, 1]$ be a function chosen uniformly at random from all such possible functions. Partition $[0, 1]$ into $L = 4(d + 1)$ segments of equal measure, $I_1, \ldots, I_L$. For every $v \in V$, set $f(v) = \ell$ if $r(v) \in I_\ell$ ($f$ is a quantization of $r$).
Consider the following method of generating two sets of vertices: \( T \) and \( R \), where \( T \subseteq R \). For some vertex \( v \), set \( T = R = \{ v \} \). Continue inductively: choose some vertex \( w \in T \), add all \( N(w) \) to \( R \) and compute \( f(u) \) for all \( u \in N(w) \). Add the vertices \( u \) such that \( u \in N(w) \) and \( f(u) \geq f(w) \) to \( T \). The process ends when no more vertices can be added to \( T \). \( T \) is the query tree with respect to \( f \), hence \( |T| \) is an upper bound on the size of the actual query tree (i.e., the query tree with respect to \( r \)). However, it is difficult to reason about the size of \( T \) directly, as the ranks of its vertices are not independent. The ranks of the vertices in \( R \), though, are independent, as \( R \) is generated by an adaptive vertex exposure procedure. \( R \) is a superset of \( T \) that includes \( T \) and its boundary, hence \(|R|\) is also an upper bound on the size of the query tree.

We now define \( L + 1 \) “layers” - \( T_{\leq 0}, \ldots, T_{\leq L} \): \( T_{\leq \ell} = T \cap \bigcup_{i=0}^{\ell} f^{-1}(i) \). That is, \( T_{\leq \ell} \) is the set of vertices in \( T \) whose rank is at most \( \ell \). (The range of \( f \) is \([L]\), hence \( T_{\leq 0} \) will be empty, but we include it to simplify the proof.)

**Claim 4.1.** Set \( L = 4(d+1), c = 15L \). Assume without loss of generality that \( f(v) = 0 \). Then for all \( 0 \leq i \leq L - 1 \),

\[
\Pr[|T_{\leq i}| \leq 2^i c \log n \land |T_{\leq i+1}| \geq 2^{i+1} c \log n] \leq \frac{1}{n^i}.
\]

**Proof.** For all \( 0 \leq i \leq L \), let \( R_{\leq i} = T_{\leq i} \cup N(T_{\leq i}) \). Note that

\[
R_{\leq i} \cap f^{-1}(i) = T_{\leq i} \cap f^{-1}(i), \tag{1}
\]

because if there had been some \( u \in N(T_{\leq i}), f(u) = i \), \( u \) would have been added to \( T_{\leq i} \).

Note that \( |T_{\leq i}| \leq 2^i c \log n \land |T_{\leq i+1}| \geq 2^{i+1} c \log n \) implies that

\[
|T_{\leq i} \cap f^{-1}(i+1)| > \frac{|T_{\leq i+1}|}{2} \tag{2}
\]

In other words, the majority of vertices \( v \in T_{\leq i+1} \) must have \( f(v) = i + 1 \).

Given \( |T_{\leq i+1}| > 2^{i+1} c \log n \), it holds that \( |R_{\leq i+1}| > 2^{i+1} c \log n \) because \( T_{\leq i+1} \subseteq R_{\leq i+1} \). Furthermore, \( R_{\leq i+1} \) was constructed by an adaptive vertex exposure procedure and so the conditions of Lemma 2.1 hold for \( R_{\leq i+1} \). From Equations (1) and (2) we get

\[
\Pr[|T_{\leq i}| \leq 2^i c \log n \land |T_{\leq i+1}| \geq 2^{i+1} c \log n] \leq \Pr \left[ |R_{\leq i+1} \cap f^{-1}(i+1)| > \frac{|T_{\leq i+1}|}{2} \right]
\]

\[
\leq \Pr \left[ |R_{\leq i+1} \cap f^{-1}(i+1)| > \frac{2|R_{\leq i+1}|}{L} \right]
\]

\[
\leq \frac{1}{n^i},
\]

where the second inequality is because \( |R_{\leq i+1}| \leq (d+1)|T_{\leq i+1}| \), as \( G \)'s degree is at most \( d \); the last inequality is due to Lemma 2.1. \( \square \)

**Lemma 4.2.** Set \( L = 4(d+1) \). Let \( G = (V, E) \) be a graph with degree bounded by \( d \), where \( |V| = n \). For any vertex \( v \in G \), \( \Pr \left[ T_v > 2^L \cdot 15L \log n \right] < \frac{1}{n^3} \).

**Proof.** To prove Lemma 4.2, we need to show that, for \( c = 15L \),

\[
\Pr[|T_{\leq L}| > 2^L c \log n] < \frac{1}{n^3}.
\]
We show that for $0 \leq i \leq L$, $\Pr[|T_{\leq i}| > 2^i c \log n] < \frac{1}{n^2}$, by induction. For the base of the induction, $|S_0| = 1$, and the claim holds. For the inductive step, assume that $\Pr[|T_{\leq i}| > 2^i c \log n] < \frac{1}{n^2}$. Then

$$\Pr[|T_{\leq i+1}| > 2^{i+1} c \log n] = \Pr[|T_{\leq i+1}| > 2^{i+1} c \log n : |T_{\leq i}| > 2^i c \log n] \Pr[|T_{\leq i}| > 2^i c \log n] + \Pr[|T_{\leq i+1}| > 2^{i+1} c \log n : |T_{\leq i}| \leq 2^i c \log n] \Pr[|T_{\leq i}| \leq 2^i c \log n].$$

From the inductive step and Claim 4.1, using the union bound, the lemma follows.

Applying a union bound over all the vertices gives the size of each query tree is $O(\log n)$ with probability at least $1 - 1/n^2$, completing the proof of Theorem 1.1.

5 Discussion

Query trees were introduced by Nguyen and Onak [3], where they bounded their expected size. Mansour et al. [2], studying query trees in the context of local computation algorithms [6] (see [1] for a recent survey), showed that their size is at most $O(\log n)$ w.h.p. The proof presented above is adapted from [5] - the proof is simpler and more elegant than that of [2]. Furthermore, in order to generate the random order required in the proof, it suffices to have a random function $f : V \to [L]$, where $L$ is a constant. This, combined with the fact the relevant set is of size at most $O(\log n)$ w.h.p., allows us to use a random seed of length only $O(\log n)$ to generate such an $f$. See [5, 7] for details.

Acknowledgments We thank Guy Even for suggesting that a short note such as this might be informative and for his useful comments.

References

[1] Reut Levi and Moti Medina. A (centralized) local guide. Bulletin of EATCS, 2(122), 2017.
[2] Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In Proc. 39th International Colloquium on Automata, Languages and Programming (ICALP), pages 653–664, 2012.
[3] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 327–336, 2008.
[4] Krzysztof Onak. New Sublinear Methods in the Struggle Against Classical Problems. PhD thesis, MIT, 2010.
[5] Omer Reingold and Shai Vardi. New techniques and tighter bounds for local computation algorithms. J. Comput. Syst. Sci., 82(7):1180–1200, 2016.
[6] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In Proc. 2nd Symposium on Innovations in Computer Science (ICS), pages 223–238, 2011.
[7] Shai Vardi. Designing Local Computation Algorithms and Mechanisms. PhD thesis, Tel Aviv University, Tel Aviv, Israel, 2015.