A new application of Orthogonal Range Searching for computing Giant Graph Diameters

Guillaume Ducoffe

A new application of Orthogonal Range Searching for computing Giant Graph Diameters

Guillaume Ducoffe
National Institute for Research and Development in Informatics,
The Research Institute of the University of Bucharest ICUB,
University of Bucharest, Faculty of Mathematics and Computer Science
Romania.
guillaume.ducoffe@ici.ro

Abstract

A well-known problem for which it is difficult to improve the textbook algorithm is computing the graph diameter. We present two versions of a simple algorithm (one being Monte Carlo and the other deterministic) that for every fixed h and unweighted undirected graph G with n vertices and m edges, either correctly concludes that $diam(G) < hn$ or outputs $diam(G)$, in time $\mathcal{O}(m + n^{1+o(1)})$. The algorithm combines a simple randomized strategy for this problem (Damaschke, *IWOCA'16*) with a popular framework for computing graph distances that is based on range trees (Cabello and Knauer, *Computational Geometry'09*). We also prove that under the Strong Exponential Time Hypothesis (SETH), we cannot compute the diameter of a given n-vertex graph in truly subquadratic time, even if the diameter is an $\Theta(n/\log n)$.

2012 ACM Subject Classification Theory of computation → Shortest paths, Theory of computation → Problems, reductions and completeness

Keywords and phrases Graph diameter; Orthogonal Range Queries; Hardness in P; FPT in P.

Digital Object Identifier 10.4230/OASIcs.SOSA.2019.12

Funding This work was supported by the Institutional research programme PN 1819 "Advanced IT resources to support digital transformation processes in the economy and society - RESINFO-TD" (2018), project PN 1819-01-01"Modeling, simulation, optimization of complex systems and decision support in new areas of IT&C research", funded by the Ministry of Research and Innovation, Romania. This work was also supported by a grant of Romanian Ministry of Research and Innovation CCCDI-UEFISCDI. project no. 17PCCDI/2018.

1 Introduction

We refer to [5] for any undefined terminology. Graphs in this study are finite, simple, connected and unweighted. For every graph $G = (V, E)$, let $n := |V|$ and $m := |E|$. The distance $dist_G(u, v)$ between any two vertices $u, v \in V$ is defined as the minimum number of edges on a uv-path in G. A *layer* is any set $L_i(v) := \{u \in V \mid dist_G(u, v) = i\}$ for some $v \in V$ and integer $i \geq 0$. Finally, the eccentricity of vertex v, denoted $ecc_G(v)$, is equal to $\max_{u \in V} dist_G(u, v)$, and the diameter of G, denoted $diam(G)$, is equal to $\max_{v \in V} ecc_G(v)$.

Computing the diameter of a graph is a fundamental problem with countless applications in computer science and other domains. As every undergraduate student (should) know, this problem can be solved in roughly quadratic time by running a single-source shortest-path algorithm from every vertex of the graph. It has been asked repeatedly whether one could improve on this textbook algorithm, in order to achieve truly subquadratic-time computation of the diameter. Unfortunately, the answer to that question seems to be 'No' [15]. Specifically, under SETH it is already hard to recognize *split graphs* with diameter 2, and this holds even

if their clique-number is an $\Theta(\log n)$ [6][1]. Small diameter graphs are usually said to be the hardest case for the problem, in the sense that as the diameter gets polynomial in n we can obtain an almost optimal $(1 - n^{-\mathcal{O}(1)})$-approximation in truly subquadratic time [3, 10]. However, the *exact* computation of such "giant" graph diameters has been less studied.

In [11], Damaschke asked whether one can compute the graph diameter in nearly linear time assuming it is a large fraction of the number of vertices. His question seems relevant to the study of chain-like structures, *e.g.*, in road networks or chain molecules. Specifically, we consider the following problem in this note:

▶ Problem 1 (h-DIAMETER).
Input: A graph $G = (V, E)$; a constant $h \in (0; 1)$.
Output: The exact diameter of G if it is at least hn (otherwise, any value $< hn$).

As a partial answer to Problem 1, Damaschke presented a deterministic linear-time algorithm for the special case $h > 1/2$. The latter is based on the decomposition of a graph by its biconnected components and a delicate removal procedure of irrelevant subgraphs. As noted by Damaschke himself, qualitatively different methods are needed to solve the general case. In [11], he also presented a Monte Carlo $\mathcal{O}(m + n \log n)$-time algorithm for the case $h > 1/3$. The probability of a correct result depends on the simultaneity of several random events. Recently in a master's thesis [4] some of Damaschke's students generalized his ideas to any h, thereby obtaining a (not so simple) $\mathcal{O}(n^2)$-time algorithm for the general case. The "big-oh" notation hides a large constant-factor in h.

1.1 Our results

We answer positively to the open question of [11]. Specifically, we first present a Monte Carlo algorithm that runs in time $\mathcal{O}(\frac{1}{h} \cdot (m + 2^{\mathcal{O}(\frac{1}{h})} n^{1+o(1)}))$ and that, given as entry a graph G such that $diam(G) \geq hn$, outputs with constant probability its diameter (Theorem 3).

- For that, we follow the same general (and quite intuitive) strategy as in [11, 4]: finding a separator of size $\mathcal{O}(1/h)$ that disconnects the two ends of some arbitrary diametral path. Such a separator can be easily computed, with constant probability, by performing a BFS from a random vertex v and choosing a smallest layer $L_i(v) := \{u \in V \mid dist_G(u, v) = i\}$ in the range $i \in \{hn/3, \ldots, 2hn/3\}$. Although a similar approach was used in [11, 4] our proofs are, in our opinion, cleaner and more direct than the ones given in these previous works. In particular, the correctness of our algorithm only depends on the random choice of the starting vertex v.
- Then, once we computed a separator as described above, we are left with computing the maximum distance between two vertices it disconnects. Previous works [11, 4] reduce this computation to a new problem called LARGEST MIXED SUM. Instead, we can directly apply a popular framework for computing graph distances that is based on a *range tree* [8]. Doing so, we give a new simple application of this textbook data-structure.
- Finally, we remark that in order to make our algorithm deterministic, it suffices to run a BFS from every vertex v into some (hn/c)-distance dominating set, for some small constant c (instead of picking this vertex randomly). We end up observing that such a distance dominating set of size $\left(\frac{1}{h}\right)^{\mathcal{O}(1)}$ can be computed by using a few BFS (Theorem 5).

[1] The logarithmic bounds on the clique-number are not explicitly stated in [6]. Nevertheless, they can be deduced from an easy application of the Sparsification Lemma, as noted *e.g.* in [1] for similar constructions.

For nonconstant h, our algorithm still outperforms the textbook algorithm for DIAMETER provided $h = \omega(1/\log n)$. Perhaps surprisingly, we prove this is (conditionally) optimal: any truly subquadratic algorithm for computing the diameter of a given n-vertex graph would falsify SETH, even if the diameter is an $\Theta(n/\log n)$ (Theorem 6).

2 Preliminaries

In what follows is a simple observation that is the cornerstone of our algorithms: in any consecutive subsequence of $\Theta(n)$ layers in a BFS-tree, there must be one of size $\mathcal{O}(1)$.

▶ **Lemma 1.** *Let $G = (V, E)$ be a graph of order n, let $0 < p < q < r < 1$, and let $v \in V$ have $ecc_G(v) \geq rn$. There exists $i \in \{\lceil pn \rceil, \ldots, \lfloor qn \rfloor\}$ such that the layer $L_i(v) := \{u \in V \mid dist_G(u, v) = i\}$ contains $< \frac{1-r}{q-p} + 1$ vertices.*

Proof. The number of layers $L_i(v)$ to be considered is:

$$\lfloor qn \rfloor - \lceil pn \rceil + 1 = \lfloor (q-p)n \rfloor + 1 > (q-p)n.$$

We also know that there are $\geq (rn - \lfloor qn \rfloor) + \lceil pn \rceil \geq (r - q + p)n$ vertices that are not contained into any of these layers. Therefore, the maximum number of vertices a smallest such layer can contain is:

$$< \frac{(1 - r + q - p)n}{(q-p)n} = \frac{1-r}{q-p} + 1.$$

◀

Then, assume one such layer disconnects the two ends of an arbitrary diametral path of the graph. In order to compute the diameter of the graph, we only need to compute the maximum distance between two vertices this layer disconnects. This is a routine that naturally appears in the computation of the diameter and the Wiener index of *bounded treewidth* graphs [1, 7, 8], and as such there already exists a standard method for solving this problem:

▶ **Proposition 1** (implicit in [7])**.** Let $G = (V, E)$ be a graph and $S \subseteq V$ be a separator, where $|S| \leq k$. We can compute $D_S := \max\{dist_G(x, y) \mid S \text{ is an } xy\text{-separator}\}$ in time $\mathcal{O}(k \cdot (m + \binom{k+1+\lceil \log n \rceil}{k+1} 2^k n))$, that is in $\mathcal{O}(k \cdot (m + 2^{\mathcal{O}(k)} n^{1+o(1)}))$.

For the sake of completeness, let us give some intuition of how this above problem relates to *range trees*. Let $S = \{s_1, s_2, \ldots, s_k\}$ and $C_1, C_2, \ldots, C_\ell$ be the connected components of $G \setminus S$. For every $i \in \{1, \ldots, k\}$ we search for the furthest pair x, y such that: (a) S is an xy-separator, and (b) $dist_G(x, s_i) + dist_G(s_i, y) = dist_G(x, y)$, as follows. We first map every $v \in C_j$ to the $(k+1)$-dimensional point $P^i_v = (p^i_{v,0}, p^i_{v,1}, \ldots, p^i_{v,k})$, where $p^i_{v,0} = j$, $p^i_{v,t} = dist_G(v, s_t) - dist_G(v, s_i)$ for every $t \geq 1$, and we associate to this point the value $f_i(v) = dist_G(v, s_i)$. To understand why, note that if $x \in C_j, y \in C_{j'}$ are such that $j < j'$ and $dist_G(x, s_i) + dist_G(s_i, y) = dist_G(x, y)$, then we have $dist_G(x, s_i) + dist_G(s_i, y) \leq dist_G(x, s_t) + dist_G(s_t, y) \iff -p^i_{x,t} \leq p^i_{y,t}$ for any $t \geq 1$. Therefore, given $x \in C_j$, in order to find a furthest vertex $y \in \bigcup_{j' > j} C_{j'}$ from x, it suffices to compute a point P^i_y such that:

$$p^i_{x,0} < p^i_{y,0};\ -p^i_{x,t} \leq p^i_{y,t} \text{ for every } t \in \{1, \ldots, k\};\ \text{and } f_i(y) \text{ is maximized.}$$

The $(k+1)$-dimensional range tree is a classical data-structure that can be used in order to solve this above computation. Specifically, given that there are $|V \setminus S| = \mathcal{O}(n)$ points

P_v^i to store, we can construct such a range tree in time $\mathcal{O}(k\binom{k+1+\lceil \log n \rceil}{k+1}n)$ in such a way that for every $x \notin S$, the corresponding query (computation of P_y^i) can be answered in time $\mathcal{O}(2^k\binom{k+1+\lceil \log n \rceil}{k+1})$ [14]. We stress that the analysis of this construction is involved, but its implementation is quite straightforward (*e.g.*, see [7] for details).

Comparison with previous work. Damaschke's students solved a variant of the above problem in $\mathcal{O}(kn^2)$-time by reducing it to a new problem they called Largest Mixed Sum [4]. Roughly, their solution consists in a brute force range searching. We use range trees in order to improve their running time, although in doing so we sacrifice analytical simplicity. A potential drawback of our algorithms compared to [4] is that the range tree data-structure has relatively high preprocessing and storage costs, that make it less practical for moderate values of k [2]. Some way to address this issue could be the use of alternative data-structures for range searching [2, 9].

3 Monte Carlo algorithm

We present in this section a simple algorithm for the computation of graph diameters that are at least a fixed fraction of the number of vertices. Unlike previous works [11, 4], we use randomization only to choose the starting vertex of our BFS run (Algorithm GiantDiameter). Our algorithm is correct assuming this starting vertex is sufficiently close to an end of some (arbitrary) diametral path. We prove next that it happens with constant probability.

▶ **Lemma 2.** *Let $G = (V, E)$ be a graph and assume $diam(G) \geq hn$. For every vertex $v \in V$ that is drawn u.a.r., the following holds with probability $\geq 2h/3$: There exists a diametral pair $x, y \in V$ such that $dist_G(x, v) \leq hn/3$ (and so, $dist_G(y, v) \geq 2hn/3$).*

Proof. Fix an arbitrary *diametral* path P with ends $x, y \in V$. Every vertex $v \in V(P)$ such that either $dist_G(v, x) \leq hn/3$ or $dist_G(v, y) \leq hn/3$ satisfies the desired property, and there are exactly $2hn/3$ such vertices. ◀

GiantDiameter

Input: graph $G = (V, E)$, h.

Output: a lower-bound on $diam(G)$.

1: Let $v \in V$ picked u.a.r.
2: **if** $ecc_G(v) < 2hn/3$ **then**
3: **return** $ecc_G(v)$. *// this occurs with proba. $\leq 1 - 2h/3$ if $diam(G) \geq hn$.*
4: Find a layer $i \in \{\lceil hn/3 \rceil, \ldots, \lfloor 2hn/3 \rfloor\}$ s.t. $|L_i(v)| \leq 3/h$.
5: Compute $D_i := \max\{dist_G(x, y) \mid L_i(v)$ is an xy-separator$\}$.
6: **return** $\max\{D_i\} \cup \{ecc_G(u) \mid u \in L_i(v)\}$.

▶ **Proposition 2.** Let $G = (V, E)$ be a graph and assume $diam(G) \geq hn$. Algorithm GiantDiameter correctly computes $diam(G)$ with probability $\geq 2h/3$.

Proof. By Lemma 1 (applied with $p = h/3$ and $q = r = 2h/3$), a small-size layer $L_i(v)$ as requested by the algorithm always exists if $ecc_G(v) \geq 2hn/3$. Then, the algorithm is correct if there exists a diametral pair $x, y \in V$ such that there is no connected component of $G \setminus L_i(v)$ that both contains x, y (possibly, $x \in L_i(v)$ or $y \in L_i(v)$). This is always the case if $\min\{dist_G(v, x), dist_G(v, y)\} \leq hn/3$ (and so, $\max\{dist_G(v, x), dist_G(v, y)\} \geq 2hn/3$), and by Lemma 2 the latter happens with probability $\geq 2h/3$. ◀

The bottleneck of Algorithm GiantDiameter is the computation of D_i. Using Proposition 1, we can conclude as follows:

▶ **Theorem 3.** *Let h be a fixed constant. In time $\mathcal{O}(\frac{1}{h} \cdot (m + 2^{\mathcal{O}(\frac{1}{h})} n^{1+o(1)}))$, we can either conclude a given graph G has diameter $< hn$, or compute its diameter, with probability of correctness $\geq 2h/3$.*

Proof. We run Algorithm GiantDiameter, whose output is correct with probability $\geq 2h/3$ by Proposition 2. The dominant step for the algorithm is the computation of D_i, that can be done in time $\mathcal{O}(\frac{1}{h} \cdot (m + 2^{\mathcal{O}(\frac{1}{h})} n^{1+o(1)}))$ by Proposition 1. ◀

As usual, the probability of correctness can be increased to $1 - n^{-\mathcal{O}(1)}$ by running Algorithm GiantDiameter $\mathcal{O}(\log n/h)$ times and outputting the maximum distance we obtained.

4 Deterministic algorithm

Next, we show how to derandomize our algorithm from the previous section. We recall that we use randomization only to choose the starting vertex v of some BFS run. Furthermore, the latter vertex v is correctly chosen if it is at a distance $\leq hn/3$ from an end of some arbitrary diametral path. Hence, instead of choosing v at random, we can try *all* the vertices contained into some (hopefully small) $(hn/3)$-distance dominating set. We prove next that there always exists such a set of size polynomial in $1/h$.

▶ **Lemma 4.** *Let $G = (V, E)$ be a graph. In $\mathcal{O}(m + n)$-time, we can output a set S where $|S| = \mathcal{O}(1/h)$ and such that $dist_G(v, S) \leq hn/3$ for every vertex $v \in V$.*

Proof. We use the constructive proof of Meir and Moon on k-distance dominating sets in trees [13]. Specifically, let T be an arbitrary spanning tree of G. Such a tree can be computed in $\mathcal{O}(n + m)$-time by using, say, a breadth-first search. For any integer $k \geq 0$, we will explain next how to construct a k-distance dominating set of size $\left\lceil \frac{n}{k+1} \right\rceil$ for T (and so, also for G) in time $\mathcal{O}(n)$. By setting $k = \lfloor hn/3 \rfloor$, this will prove the lemma.

For that, we first compute the two ends of a diametral path in T, that can be easily done in time $\mathcal{O}(n)$ using two BFS [12]. Let x be any one of these two ends. If $n \leq k + 1$ or more generally, $diam(T) \leq k$ then, we can output $S = \{x\}$. Otherwise, we compute in time $\mathcal{O}(n)$ a breadth-first search from x in T. For every $i \in \{0, 1, 2, \ldots, k\}$, let $S_i := \{v \in V \mid dist_T(x, v) \equiv i \pmod{k+1}\}$. Note that $S_i \neq \emptyset$ for every i since we assume $diam(T) > k$. Furthermore as proved in [13, Theorem 5], S_i is a k-distance dominating set of T for any fixed i. Since there are exactly $k + 1$ possibilities for i, there exists a i_0 such that $|S_{i_0}| \leq \left\lfloor \frac{n}{k+1} \right\rfloor$. We output $S = S_{i_0}$. ◀

▶ **Theorem 5.** *Let h be a fixed constant. In time $\mathcal{O}(\frac{1}{h^2} \cdot (m + 2^{\mathcal{O}(\frac{1}{h})} n^{1+o(1)}))$, we can either conclude a given graph G has diameter $< hn$, or compute its diameter.*

Proof. We apply Lemma 4 in order to compute an $(hn/3)$-distance dominating set S of size $\mathcal{O}(1/h)$. Then, we apply Algorithm GiantDiameter for every $v \in S$, and we output the maximum distance we obtain after these $|S|$ runs. ◀

5 Conditional Lower-bound

Our GiantDiameter algorithm still runs in truly subquadratic time if $h = \omega(1/\log n)$. It would be interesting to compute in truly subquadratic time the *exact* diameter of a graph when it is an $\Theta(n/\log n)$. We prove that it cannot be done under standard complexity assumptions.

▶ **Theorem 6.** *Under SETH, there exists a function $h(n) = \Theta(1/\log n)$ such that the following problem cannot be solved in time $\mathcal{O}(n^{2-\varepsilon})$, for any $\varepsilon > 0$: Given an n-vertex graph G, either correctly decide $diam(G) < h(n)n$, or compute $diam(G)$.*

Proof. Let $G = (K \cup S, E)$ be a n-vertex split graph such that K induces a clique of size $|K| = \mathcal{O}(\log n)$ and S induces a stable set. We construct a graph $G' = (V', E')$ as follows. The vertex-set of G' is partitioned into two disjoint copies S^0, S^1 of the stable set S and n disjoint copies $K_1, K_2, \ldots, K_n$ of the clique K. For every $s \in S$ we denote by s^0 and s^1 the respective copies of s in S^0 and S^1; in the same way, for every $v \in K$ and $i \in \{1, 2, \ldots, n\}$ we denote by v^i the copy of vertex v in K_i. Furthermore, every copy K_i induces a clique, we add an edge $\{v^i, v^{i+1}\}$ for every $v \in K, 1 \leq i < n$, and the two edges $\{s_0, v^1\}, \{s_1, v^n\}$ for every $s \in S, v \in K$ such that $\{s, v\} \in E$. By construction, G' has order $\mathcal{O}(n \log n)$ and size $\mathcal{O}(n \log^2 n)$. We also have:

- For $u, v \in K$, $dist_{G'}(u^i, v^j) = |j - i|$ if $u = v$, and $|j - i| + 1$ otherwise.
- For $s \in S, v \in K$, $dist_{G'}(s_0, v^i) = dist_{G'}(s_1, v^{k-1+i}) = i$ if $\{s, v\} \in E$, and $i + 1$ otherwise.
- For $s, s' \in S$, $dist_{G'}(s_0, s'_0) = dist_{G'}(s_1, s'_1) = dist_G(s, s') \leq 3$.
- For $s, s' \in S$, $dist_{G'}(s_0, s'_1) = n - 1 + \max\{2, dist_G(s, s')\}$.

As a result, $n + 1 \leq diam(G') \leq n - 1 + diam(G) \leq n + 2$. In particular, $diam(G') = n - 1 + diam(G)$. This implies that computing $diam(G')$ is $\tilde{\mathcal{O}}(n)$-time equivalent to computing $diam(G)$, that cannot be done in truly subquadratic time under SETH [6, 15]. ◀

Acknowledgements

We wish to thank the referees for their careful reading of the first version of this manuscript, and their useful comments.

References

1 A. Abboud, V. Vassilevska Williams, and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *SODA*, pages 377–391. SIAM, 2016. URL: `https://arxiv.org/abs/1506.01799`.

2 J. Bentley and J. Friedman. Data structures for range searching. *ACM Computing Surveys (CSUR)*, 11(4):397–409, 1979.

3 P. Berman and S. Kasiviswanathan. Faster approximation of distances in graphs. In *WADS*, pages 541–552. Springer, 2007.

4 B. Block and M. Milakovic. Computing diameters in slim graphs. Master's thesis, Chalmers University of Technology, University of Gothenburg, Sweden, 2018. URL: `http://publications.lib.chalmers.se/records/fulltext/255208/255208.pdf`.

5 J. A. Bondy and U. S. R. Murty. *Graph theory*. Grad. Texts in Math., 2008.

6 M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016. URL: `https://arxiv.org/abs/1407.4972`.

7 K. Bringmann, T. Husfeldt, and M. Magnusson. Multivariate analysis of orthogonal range searching and graph distances parameterized by treewidth. In *IPEC*. LIPIcs, 2018. to appear. URL: `https://arxiv.org/abs/1805.07135`.

8 S. Cabello and C. Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Computational Geometry*, 42(9):815–824, 2009.

9 T. Chan. Orthogonal Range Searching in Moderate Dimensions: k-d Trees and Range Trees Strike Back. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7226`.

10 S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. Tarjan, and V. Vassilevska Williams. Better approximation algorithms for the graph diameter. In *SODA*, pages 1041–1052. SIAM, 2014.

11 P. Damaschke. Computing giant graph diameters. In *IWOCA*, pages 373–384. Springer, 2016.

12 C. Jordan. Sur les assemblages de lignes. *J. Reine Angew. Math*, 70(185):81, 1869.

13 A. Meir and J. Moon. Relations between packing and covering numbers of a tree. *Pacific Journal of Mathematics*, 61(1):225–233, 1975.

14 L. Monier. Combinatorial solutions of multidimensional divide-and-conquer recurrences. *Journal of Algorithms*, 1(1):60–74, 1980.

15 L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC*, pages 515–524. ACM, 2013.