Fast Computation of Empirically Tight Bounds for the Diameter of Massive Graphs

Clémence Magnien [1], Matthieu Latapy [1] and Michel Habib [2]

Abstract

The diameter of a graph is among its most basic parameters. Since a few years, it moreover became a key issue to compute it for massive graphs in the context of complex network analysis. However, known algorithms, including the ones producing approximate values, have too high a time and/or space complexity to be used in such cases. We propose here a new approach relying on very simple and fast algorithms that compute (upper and lower) bounds for the diameter. We show empirically that, on various real-world cases representative of complex networks studied in the literature, the obtained bounds are very tight (and even equal in some cases). This leads to rigorous and very accurate estimations of the actual diameter in cases which were previously untractable in practice.

1 Context.

Throughout the paper, we consider a connected undirected unweighted graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. We denote by $d(u, v)$ the distance between u and v in G, by $\text{ecc}(v) = \max_u d(v, u)$ the eccentricity of v in G, and by $D = \max_{u,v} d(u, v) = max_v \text{ecc}(v)$ the diameter of G.

Computing all distances from one vertex to all the others (and thus its eccentricity) has $\Theta(m)$ time and space costs using a breadth-first search (BFS). In order to compute the diameter, one has to compute the distance between all pairs of vertices, which therefore has a $\Theta(n \cdot m)$ time and a $\Theta(m)$ space costs using n BFS. Using matrix products, one may achieve this in $O(n^{2.376} \cdot \text{polylog}(n))$ time and $\Theta(n^2)$ space [2, 17]. Therefore, the BFS approach is too slow for massive graphs, and the matrix approach remains too slow and has in addition a prohibitive space cost.

Methods have been proposed to compute the distances between all pairs of vertices without resorting to matrix products but faster than with BFS; see [12] for such an algorithm, in $\Theta\left(\frac{n^3}{logn}\right)$ time and $\mathcal{O}(n^2)$ space for dense graphs. See also [4] for an algorithm in $\mathcal{O}(n^2(\log\log n)^2/\log n) \subset o(n^2)$ time and $\mathcal{O}(n^2)$ space for sparse graphs. Again, such algorithms are too slow and too space consuming for practical use on massive graphs.

In [1, 8], the authors propose to compute *almost shortest paths* between all pairs of vertices, leading to an estimation $\overline{D}$ of D such that $\overline{D} \le D \le \overline{D} + 2$. The cost of this computation however is in $\Omega(n^2)$ time and $\Theta(n^2)$ space, which is still too expensive in our context. See [19] for a survey on the computation and approximation of diameters (and distances in general).

The authors of [16] use a different approach: they designed an algorithm for testing probabilistically whether the diameter of a graph G is below a given value D, or whether G is *close* to such a graph, in terms of the number of edges to add or remove to transform one graph into the other.

Some authors also designed methods for the approximation or exact computation of the diameter for some specific classes of graphs [13, 6, 7, 10, 9, 5], which we will use in the following.

Finally, current algorithms available to compute the diameter of a given graph have too high a computational cost; it is in practice impossible to obtain the exact value, or accurate estimations, of the diameter of massive graphs such as the ones encountered in complex network studies (which typically have hundreds of thousands vertices, and up to a few billions). Because of this, authors give ad-hoc estimations of the diameter (typically obtained as the maximal eccentricity found for

[1] LIP6 – CNRS and UPMC (`firstname.lastname@lip6.fr`)

[2] LIAFA – CNRS and Université Paris Diderot (`firstname.lastname@liafa.jussieu.fr`)

a given set of randomly chosen vertices), for which no rigorous assessment is done. Some authors also make no distinction between the diameter and the average distance, which they approximate using various heuristics.

A more rigorous approach consists in replacing the notion of diameter by other notions which capture similar properties while being easier to evaluate. In [15] for instance the authors consider the *effective diameter*, *i.e.* the value such that 90% of the pairs of vertices are at a distance lower than this value. Such approaches are appealing, but they are out of the scope of this paper, in which we show that the actual diameter may be estimated very accurately and very quickly in practice, even in massive graphs.

2 Computation methods.

We now present a series of very simple but powerful algorithms that give exact upper or lower bounds for the diameter. There is however no guarantee on how accurate these bounds are, except that if they are close from each other then the actual diameter is well estimated. The interest of this approach relies on the fact that the bounds we obtain are indeed close, which we study empirically in the next section.

Trivial bounds.

First notice that the eccentricity of any vertex v gives trivial bounds of the diameter: $\mathrm{ecc}(v) \leq D \leq 2 \cdot \mathrm{ecc}(v)$; we call these *trivial lower and upper bounds.* They can be computed in $\Theta(m)$ space and time.

Double sweep lower bound.

The quality of the trivial bounds obviously depends on the choice of the vertex v. In [6, 13] it is shown that, on some specific classes of graphs (including trees), if v is chosen such that $d(u, v) = \mathrm{ecc}(u)$ for a vertex u (*i.e.* v is among the vertices which are at maximal distance from u), then $D = \mathrm{ecc}(v)$. In these graphs, the diameter may therefore be computed by a BFS from any node u and then a BFS from a node at maximal distance from u, thus in $\Theta(m)$ space and time. In the general case, the value obtained in this way is different from the diameter, however, it always gives a better lower bound than than the trivial method from the same vertex (which gives $\mathrm{ecc}(u)$). We call bounds obtained in this way *double sweep lower bounds.*

Tree upper bound.

Now let us notice that the diameter of any spanning (*i.e.* containing all the vertices) connected subgraph of G is larger than or equal to the diameter of G; therefore, one may obtain upper bounds by considering appropriate subgraphs of G. As we have juste seen, it is easy to compute the diameter of any tree in $\Theta(m)$ time and space [13]. Spanning trees of G, in particular the ones having small diameters, therefore are good candidates for this. We will call a *tree upper bound* for the diameter any upper bound obtained as the diameter of a BFS tree from a vertex. It always is better than the corresponding trivial upper bound.

Iterating the bounds.

Finally, we obtain a set of very simple algorithms (all relying on a few BFS) which give lower and upper bounds of the diameter. Iterating them from different intial vertices makes it possible to obtain tighter and tighter bounds, with a linear cost for each step. In the context of massive graphs, the number of steps must therefore remain small, but it may be worth to make several iterations (as we will see in the next section).

The initial vertices may be chosen randomly, but one may also choose vertices more likely to produce tight bounds. We will do so for the tree upper bound: as highest degree vertices are intuitively likely to produce BFS trees with small diameter, we will consider vertices in decreasing order of degrees when iterating this algorithm. This is motivated by the fact that real-world complex networks are known to have some vertices with very high degree, see for instance [18, 11, 3]. We call this process the *highest degree tree upper bound*; in oppostion, we call iterating the tree upper bound from random vertices the *random tree upper bound*.

Notice that one may expect that iterating the computations for a long time will eventually lead to equal bounds, and thus to the exact computation of the diameter. There is no guarantee of this, however, as all the tree upper bounds may be strictly larger than D (if G is a cycle of n vertices, for instance, its diameter is $\lfloor n/2 \rfloor$ and the tree upper bound is $n-1$ whichever vertex one starts from). On the contrary, the double sweep lower bound will give the actual value of the diameter in some cases, and thus leads to the exact diameter when it is iterated on all vertices.

Implementation.

We have implemented the different heursitics described above [3]. The program is written in C, so as to be most efficient in central memory usage. Graphs are represented as adjacency arrays: to each vertex v is associated its degree $d^\circ(v)$ and an array of size $d^\circ(v)$ containing its neighbours. A graph with n vertices and m edges is therefore stored in a space equal to $2n + 2m$: two arrays of size n containing the degrees and the references to the adjacency arrays, and the adjacency arrays, with total size $2m$; this is close to optimal [4].

The only hardware requirement is that the graph thus stored should fits in central memory: if swapping occurs the performances of the program are degraded to such an extent that it cannot finish in a reasonable amount of time.

The main behaviour of the program consists in iterating the double sweep lower bound and highest degree tree upper bound until the difference between the best bounds obtained is lower than or equal to a given threshold value. The choice of this value may depend on many factors: the graph considered, the desired quality of the bounds, ... For all practical purposes however, setting this value to 5 should give a good estimate of the diameter in a short time.

The program also has an option allowing it to run until the diameter is estimated with a given precision p, *i.e.* until the best upper and lower bounds obtained, $\underline{D}$ and $\overline{D}$, are such that $(\overline{D} - \underline{D})/\underline{D} < p$.

Finally, the program gives the possibility to compute any of the five bounds for a given number of iteration. These options are useful for comparing the different heuristics.

Notice that, though all heuristics have a $\Theta(m)$ time complexity, they do not have the exact same computation time. Aproximately, one could say that computing a double sweep lower bound or tree upper bound takes twice as much time as computing a trivial lower bound or upper bound. As we will see in the next section, which presents experimental results for our heuristics, this does not play a big role for practical purposes: the double sweep lower bound and highest degree tree upper bound indeed provide very quickly a very good estimate for the diameter. These are therefore the heuristics to use in practice.

3 Experiments.

We tested our heuristics on a wide variety of real-world graphs coming from different contexts, as well as on different types of random graphs. In all cases, the double sweep lower bound and highest degree tree upper bound have yielded in a small number of iterations a good estimation of the diameter, and have performed better than the other heuristics.

To illustrate this, as well as compare the behaviours of the different bounds, we chose to present here detailed experimental results on a set of four real-world graphs, which may be considered as

[3] The program is available at `http://www-rp.lip6.fr/~magnien/Diameter/`.

[4] Our algorithms require that each link (u, v) is stored in a space equal to at least 2, u appearing in the adjacency array of v and conversely. This requires a space of at least $2m$ if one does not resort to compression techniques.

representative of the variety of cases met in complex network studies [5]:
an internet topology graph (INET) obtained from traceroutes ran daily in 2005 by *Skitter* [6] from several scattered sources to almost one million destinations, leading to 1 719 037 vertices and 11 095 298 edges;
a web graph (WEB) containing the 39 459 925 web pages (vertices) and 783 027 125 links (edges) collected in the `.uk` domain during a measurement conducted in 2005 by *WebGraph* [7];
a peer-to-peer graph (P2P) in which two peers are linked if one of them provided a file to the other in a measurement conducted on a large *eDonkey* server for a period of 47 hours in 2004 [8], leading to 5 792 297 vertices and 142 038 401 edges;
a traffic graph (IP) obtained from *MetroSec* [9] which captured each IP packet header routed by a given router during 24 hours, two IP addresses being linked if they appear in a packet as sender and destination, leading to 2 250 498 vertices and 19 394 216 edges.

On these graphs, we iterated the computation of all bounds: trivial lower bound, trivial upper bound, double sweep lower bound, random tree upper bound (all from randomly chosen nodes), and highest degree tree upper bound (from nodes chosen by decreasing order of degrees). The number of iteration was not the same for all the graphs: it should be as large as possible to provide much insight, but sufficiently small to run the experiment in a reasonable time. We finally ran 2 000 iterations for WEB graph, 5 000 for P2P, and 10 000 for IP and INET. As we will see, these numbers have little impact on our conclusions, if any.

	t.l.b.		d.s.l.b.		h.d.t.u.b.		r.t.u.b.		t.u.b.	
INET	29		**31**		**34**		34		38	
	998	0.0001	2	0.49	26	0.1002	23	0.0633	127	0.0011
P2P	8		**9**		**10**		10		10	
	210	0.0094	1	0.7005	1	0.039	120	0.0032	3237	0.0001
WEB	26		**32**		**33**		33		34	
	816	0.001	1	0.985	34	0.0015	46	0.0025	1572	0.0005
IP	9		**9**		**9**		9		10	
	5331	0.0001	1	0.989	4	0.0543	12	0.0346	6284	0.0001

Table 1: Best bounds obtained for our four graphs by iterating each computation method: trivial lower bound (t.l.b.), double sweep lower bound (d.s.l.b.), trivial upper bound (t.u.b.), random tree upper bound (r.t.u.b.), and highest degree tree upper bound (h.d.t.u.b.). In each case, we also give the number of iterations we had to perform to obtain this bound for the first time (left), as well as the frequency with which we obtained this bound (*i.e.* the number of times we obtained it divided by the number of iterations we ran) (right).

Table 1 summarises the results; it not only gives the results obtained in each case, but also some additional information on the performance of each method. We will give more details below, but one may already notice that the obtained bounds are tight even with the simplest computation methods: the worst case is the one of the trivial lower and upper bounds for INET, the best values they provide being respectively 29 and 38, which may already be considered as a relevant information on the actual diameter. The other bounds are tighter, and even equal (to 9) in the case of IP, thus giving the exact diameter in this case.

The obtained values are not the only meaningful parameter to evaluate the performance of the various methods; the frequency and the first time at which we obtain the given bound are also precious indicators which we give in Table 1. One may observe for instance that in the case of WEB not only the best trivial lower bound is 26 and the best double sweep lower bound is 32,

[5] These data sets are described in more details in [14]. Because they may be useful for other purposes, and because they are needed to reproduce our results, we provide them at `http://www-rp.lip6.fr/~magnien/Diameter/`.

[6] `http://www.caida.org/tools/measurement/skitter/`

[7] `http://webgraph.dsi.unimi.it/`

[8] `http://www-rp.lip6.fr/~latapy/P2P_data/`

[9] `http://www2.laas.fr/METROSEC/`

which is much better, but also the best trivial lower bound is obtained with frequency 0.001 while the other is obtained with frequency 0.985. In other words, in this case, the best trivial lower bound will be obtained approximately once every 1000 iterations, while the other, though it is much better, is obtained at almost every iteration.

Similar comments may be made on the trivial upper bound and the random tree upper bound: in the case of IP, for instance, one obtains 10 with frequency 0.0001 and 9 with frequency 0.0346, respectively. The random tree upper bound and the highest degree tree upper bound always give the same best value in our experiments, but the latter gives it with a much higher frequency, and thus with a lower expected number of iterations. Note however that in the case of WEB the difference is not as clear: though the highest degree tree upper bound reaches the best value rather quickly (34 steps), it does so with frequency 0.0015 while the random tree upper bound gives its best value with a frequency of 0.0025.

In order to deepen our understanding of the behavior of each proposed bounding method, we however need to observe not only the best obtained value but all the values obtained during the iteration. This may be done by studying for each upper bound computation method the *cumulative distribution* of the obtained values (*i.e.* for all k the number of times that the obtained lower bound was less than or equal to k) and for each lower bound computation method the *complementary* cumulative distribution of the obtained values (*i.e.* for all k the number of times that the obtained lower bound was greater than or equal to k).

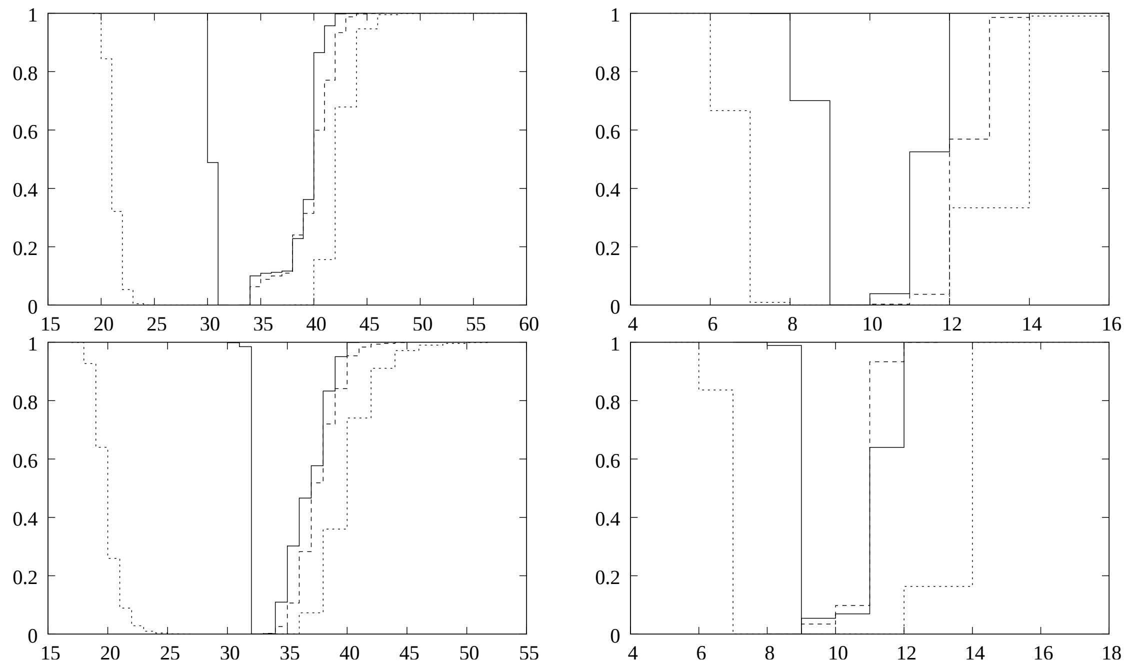

Figure 1: Cumulative distributions of the bounds obtained with all the different heuristics. For the lower bounds: trivial lower bound (dotted line, left) and double sweep lower bound (full line, left). For the upper bounds: trivial upper bound (dotted line, right), random tree upper bound (dashed line, right) and highest degree tree upper bound (full line, right). From left to right and top to bottom: INET, P2P, WEB and IP graphs.

These distributions are presented in Figure 1. They show for instance that the trivial lower bound for P2P gives a bound at least as good as 7 with probability 0.67. Likewise, still for P2P, the highest degree tree upper bound point at coordinates $(11, 0.53)$ means that it gives a bound at least as good as 11 with probability 0.53.

For each heuristic, the best obtained value is given by the point at which the corresponding plot meets the horizontal axis. If a method for the lower bound performs better than another, its plot will be on the right of the other's plot. Conversely, if a method for the upper bound performs better than another, its plot will be on the left of the other's plot. Notice that the plot for the double sweep lower bound meets the plots for both tree upper bounds at the value 9 in the case of the IP graph. This shows again that the exact value of the diameter was obtained for this graph.

Finally, these plots show that the observations above also hold for the general behavior of the studied methods. Concerning lower bounds, the double sweep lower bound performs much better than the trivial lower bound. Concerning upper bounds, the same is true for the random tree upper bound and the trivial upper bound. The highest degree tree upper bound performs in general slightly better than the random tree upper bound. In the case of WEB however, we observed that the random tree upper bound gives the value 33 with a higher frequency than the highest degree tree upper bound. This is not true in general for this graph: the highest degree tree upper bound performs better than the random tree upper bound in all other cases (the best value for both these methods is obtained with a probability too small to be observable on this plot). For IP, the converse is true: though the highest degree tree upper bound gives the best value with a higher frequency than the random tree upper bound, for other cases the random tree upper bound seems to perform better than the highest degree tree upper bound.

Conclusion.

In this note, we describe very simple methods based on BFS to give bounds on the diameter of a graph. The originality of this approach lies in the fact that there is a guarantee that the actual diameter is within the bounds we find, but there is no guarantee on the tightness of these bounds. In some cases, they may be very far of each other, and thus of little interest, if any. We however show empirically, through a representative set of experiments that we discuss, that one may expect the obtained bounds to be very tight in practice, while having a very small computational cost. In some cases, the obtained bounds may even be equal, thus giving the exact value of the diameter.

Up to our knowledge, this approach is the only known one able to give bounds in a reasonable time and space with such accuracy in many practical cases, including the ones we present.

This work opens several directions for research. Designing a method for estimating *beforehand* the number of iterations to run to obtain a good estimation of the diameter would be a very interesting result in this context. Our approach is complementary to the results developed in [16], which provide a way to estimate probabilistically whether a given graph has a diametre below a given value. Combining these two approaches would probably lead to interesting insight, and possibly provide a method for estimating where the diameter is situated within the obtained bounds. In a related manner, another key direction would be to obtain formal results on the tightness of the obtained bounds, which may be possible for instance in the case of random graphs. Finally, an interesting improvement to our results would be to remove (resp. add) edges to the graph under concern so as to obtain a graph from a class for which the exact diameter can be computed quickly (e.g., chordal graphs); this might give tighter bounds than the ones we currently obtain.

Acknowledgements. We thank all the colleagues who provided the data used in this note; no such work would be possible without their help. We thank Vincent Limouzy, as well as the anonymous referees, for helpful comments and references.

References

[1] Aingworth, Chekuri, and Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). In *ACM-SIAM SODA*, 1996.

[2] N. Alon, Z. Galil, O. Margalit, and M. Naor. Witnesses for boolean matrix multiplication and for shortest paths. In *IEEE FOCS*, 1992.

[3] A.Z. Broder, S.R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. L. Wiener. Graph structure in the web. *Computer Networks*, 33, 2000.

[4] Timothy M. Chan. All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 514 – 523, 2006.

[5] V. D. Chepoi and F. F. Dragan. Linear-time algorithm for finding a central vertex of a chordal graph. In *Proceedings of the second Annual European Symposium* Algorithms*, ESA'94, LNCS*, volume 855, 1994.

[6] D. Corneil, F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2-3), 2001.

[7] D.G. Corneil, F.F. Dragan, and E. Köhler. On the power of bfs to determine a graph's diameter. *Networks*, 42 (4), 2003.

[8] D. Dor, S. Halperin, and U. Zwick. All pairs almost shortest paths. *ECCC*, 4(040), 1997.

[9] Feodor F. Dragan. Estimating all pairs shortest paths in restricted graph families: a unified approach. *J. Algorithms*, 57(1):1–21, 2005.

[10] F.F. Dragan, F. Nicolai, and A. Brandstädt. Lexbfs-orderings and powers of graphs. In *Proceedings of the 22nd international workshop* Graph-Theoretic Concepts in Computer Science*, WG'96, LNCS*, volume 1197, 1997.

[11] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM*, 1999.

[12] T. Feder and R. Motwani. Clique partitions, graph compression, and speeding-up algorithms. In *ACM STOC*, 1991.

[13] G. Handler. Minimax location of a facility in an undirected tree graph. *Transportation Science*, 7 (3), 1973.

[14] Matthieu Latapy and Clémence Magnien. Complex network measurements: Estimating the relevance of observed properties. 2008. To appear in the proceedings of IEEE INFOCOM.

[15] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *ACM SIGKDD*, 2005.

[16] Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Struct. Algorithms*, 20(2):165–183, 2002.

[17] R. Seidel. On the all-pairs-shortest-path problem. In *ACM STOC*, 1992.

[18] D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393, 1998.

[19] U. Zwick. Exact and approximate distances in graphs – A survey. In *ESA*, volume 2161, 2001.