Determining the Diameter of Small World Networks

Frank W. Takes
Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands
ftakes@liacs.nl

Walter A. Kosters
Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands
kosters@liacs.nl

ABSTRACT

In this paper we present a novel approach to determine the exact diameter (longest shortest path length) of large graphs, in particular of the nowadays frequently studied small world networks. Typical examples include social networks, gene networks, web graphs and internet topology networks. Due to complexity issues, the diameter is often calculated based on a sample of only a fraction of the nodes in the graph, or some approximation algorithm is applied. We instead propose an exact algorithm that uses various lower and upper bounds as well as effective node selection and pruning strategies in order to evaluate only the critical nodes which ultimately determine the diameter. We will show that our algorithm is able to quickly determine the exact diameter of various large datasets of small world networks with millions of nodes and hundreds of millions of links, whereas before only approximations could be given.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory;
J.4 [**Computer Applications**]: Social and Behavorial Sciences

General Terms

Algorithms, Experimentation, Measurement

Keywords

diameter, small world networks, social networks

1. INTRODUCTION

The *diameter* of a graph is defined as the length of the longest shortest path between any two nodes in the graph. Exact algorithms for calculating this diameter traditionally require running an All Pairs Shortest Path (APSP) algorithm for each node in the network, ultimately returning the length of one of the longest shortest paths that was found. While this will indeed return the exact diameter of the graph, complexity is in the order $O(n^3)$ for general weighted graphs and $O(mn)$ for sparse unweighted graphs (with n vertices and m edges). This naive solution for obtaining the diameter is clearly not feasible in extremely large graphs with for example millions of vertices and a billion edges.

We will study the diameter of *small world networks*: sparse networks that are most typically characterized by an average distance between two random nodes that grows only proportionally to the logarithm of the total number of nodes in the network [8]. Examples of small world networks that are frequently studied are web graphs [3], internet topology networks [7] and gene networks [2], but perhaps nowadays most well-known are the social networks [17]. A common type of social network is the (explicit) online social network [15], with Facebook, LinkedIn and Twitter as well-known examples. Implicit social networks include telephone call graphs, e-mail graphs [9] and scientific collaboration networks [4].

The diameter is a relevant property of a network for many reasons. For example in social networks, the diameter could be an indication of how quickly information reaches everyone in the network in the worst-case. Within a scientific collaboration network, a high diameter may indicate that there are groups of researchers that are not working together very closely. In an internet routing network, the diameter could reveal something about the worst-case response time between any two machines in the network.

An advantage of studying the exact diameter is that we can observe the actual path that realizes the diameter, a piece of information that we do not get when for example an approximation algorithm is used, or when the diameter is estimated by looking at only a sample of the network.

The main contribution of this paper consists of a new algorithm for determining the exact diameter of small world networks. Based on various lower and upper bounds and critical node selection strategies, we improve upon the straightforward APSP algorithm as well as upon existing approximation algorithms, obtaining the exact diameter of networks with millions of nodes in a matter of seconds or minutes. The performance is empirically verified on various large datasets of small world networks.

The rest of the paper is structured as follows. Section 2 introduces some definitions and an analysis of our problem's complexity, after which we will discuss related work in Section 3. We will use Section 4 to outline our algorithm for deriving the diameter of a graph, and discuss performance results in Section 5. Finally, Section 6 concludes.

2. PRELIMINARIES

In this section we will first consider some basic definitions related to graphs, distances, eccentricity, graph diameter and shortest path problems, and then give some insight in the complexity of determining these measures. After that we will briefly discuss small world networks.

2.1 Definitions

A graph $G(V, E)$ consists of a set $V = \{1, 2, \ldots, n\}$ of vertices and a set of edges $E \subseteq V \times V$. We will call them *nodes* and *links*, respectively. Throughout the paper we will use n to denote the number of nodes $|V|$, and for the number of links $|E|$ we use m. The distance $d(v, w)$ between two nodes $v, w \in V$ is defined as the length of the shortest path from v to w. This paper deals with *unweighted* graphs, and we will assume that our graphs are *undirected*, meaning that $(v, w) \in E$ iff $(w, v) \in E$ and thus $d(v, w) = d(w, v)$. The *degree* of a node v in an undirected graph is simply defined as the number of links connecting to that node. Finally, we assume our graph is *connected*, implying that for each $v, w \in V$, $d(v, w)$ is a finite number. We are now ready to define the two most important concepts used in this paper, node *eccentricity* and graph *diameter*:

Definition 1. The *eccentricity* of a node $v \in V$ is denoted by $\varepsilon(v)$ and defined as $\max_{w \in V} d(v, w)$: the longest shortest path starting at node v.

Definition 2. The *diameter* of a graph G is denoted by $\Delta(G)$ and defined as $\max_{v,w \in V} d(v, w)$, or equivalently as the maximum eccentricity over all nodes: $\max_{v \in V} \varepsilon(v)$. It is the longest shortest path length between any pair of nodes.

For convenience, we define two combinatorial problems that are frequently addressed in this paper and are tightly related to eccentricity. First, the *Single-source Shortest Path* (*SSP*) problem is the problem that deals with finding all shortest paths from a single source vertex $v \in V$ to all other nodes in the graph. For non-sparse graphs this problem has the traditional time complexity of $O(n^2)$ (Dijkstra's shortest path algorithm). When the graph is sparse and therefore more easily stored using an adjacency list, time complexity is bounded by $O(m \ \log n)$. In our unweighted case this can even be reduced to $O(m)$, as a Breadth First Search (BFS) from the starting node is sufficient to find all shortest paths. In essence, solving the SSP problem for a node means that we have found the eccentricity of that particular node.

Next, we can define the *All Pairs Shortest Paths* (*APSP*) problem as the problem of finding the shortest paths between all pairs of nodes of the graph, which increases the previous time complexity by a factor n to $O(n^3)$ for general weighted graphs, and $O(mn)$ for our sparse unweighted graphs. In essence, the maximum distance value that the APSP algorithm obtains, is the maximum eccentricity over all nodes and thus equal to the diameter of the graph. So if we solve the APSP problem, we have found the diameter.

2.2 Small World Networks

In this paper we will specifically look at the diameter of *small world networks.* A good overview of algorithmic properties of these networks, of which we will discuss a few, is given in [8]. First of all, small world networks are generally *sparse*: the total number of links m is very small compared to the maximum number of links $n(n-1)/2$. This may cause the reader to believe that nodes typically have very long shortest paths between them, as there are very few links in general. However, a second interesting characteristic is that even though the network is very sparse, the average distance between two nodes is very small. More specifically, this distance grows only proportionally to the logarithm of the total number of nodes. Typically, the node degree distribution of a small world network follows a power law: there are only a few nodes with a very large amount of connections, the so-called *hubs*, and there are many nodes with relatively few connections. In essence, hubs are functioning as bridges to realize a low average shortest path length. Lastly, small world networks generally contain one very large connected component which contains the vast majority of the nodes.

3. RELATED WORK

Most *exact* algorithms for finding the diameter are actually implementations of matrix multiplication that solve the APSP problem and thus also find the diameter. While these algorithms work well and have time complexity $O(n^{2.376})$ [18], they usually suffer from large hidden constants, and are often very unpractical. A lot of work has been done on the *estimation* of the diameter [1, 6]. Such estimation algorithms typically determine the diameter of any type of graph with some very small additive error, using significantly less computation time than the APSP algorithm. A method which is claimed to be leading in approximating the diameter, is the Approximate Neighborhood Function (ANF) by Palmer et al. [16], which claims to have an empirically verified accuracy of 7%.

In work which does not focus on actually determining the diameter, but where it is found only as a static property of the dataset, a *sample* of the network is frequently used to determine the diameter [10, 15]. A heuristic algorithm to estimate the diameter is described in [13] by Leskovec et al. In this work, the diameter is determined by, starting from a random node, repeatedly selecting the farthest node, meanwhile keeping track of the highest distance so far. If this value no longer increases after a certain number of iterations, a lower bound on the diameter has been found.

Work has also been done on variants of the diameter, such as the *effective diameter*, which is defined as the 90-th percentile of the cumulative distribution of shortest path lengths. Though this measure may appear more robust, it is claimed that the diameter and the effective diameter tend to exhibit qualitatively similar behavior [11]. We believe that the exact diameter is an elementary graph property worth computing, and will therefore focus solely on an exact approach to determine the precise diameter.

4. ALGORITHM

We will start with some observations about the eccentricity of neighboring nodes and how they influence the diameter. Then we describe our actual algorithm called BoundingDiameters, which makes use of these observations to improve upon the APSP algorithm. Next we discuss the algorithm's complexity and some optimization techniques.

4.1 Observations

If we calculate the eccentricity $\varepsilon(v)$ for some node v, we know that for all nodes w with $d(v, w) = k$, their eccentricity $\varepsilon(w)$ lies between $\varepsilon(v) - k$ and $\varepsilon(v) + k$. The upper bound follows because any node w at distance k of v can get to v in exactly k steps, and then reach any other node in at most $\varepsilon(v)$ steps. The lower bound can be derived in the same way, by interchanging v and w in the previous statement. In the "best" case, w is on some path that realizes the eccentricity of v, and has an eccentricity $\varepsilon(w)$ of only $\varepsilon(v) - k$. This lower bound can of course never be less than k itself: if the shortest path between v and w is equal to k, then $\varepsilon(w)$ is at least equal to k. More formally:

Observation 1. **Node eccentricity bounds**
If a node $v \in V$ has eccentricity $\varepsilon(v)$, then for all nodes w we have $\max(\varepsilon(v) - d(v,w), d(v,w)) \leq \varepsilon(w) \leq \varepsilon(v) + d(v,w)$.

We know that the diameter of a graph is equal to the maximum eccentricity over all nodes. Therefore, the maximum lower bound on the eccentricity over all nodes, is also a lower bound for the diameter. Similarly, the maximum upper bound on the eccentricity over all nodes can be seen as an upper bound on the diameter. The upper bound can even be made more tight by observing that this bound can be at most twice as big as the smallest eccentricity upper bound over all nodes, as observed in [14]. These observations can be formalized as follows:

Observation 2. **Diameter bounds**
Let $\varepsilon_L(v)$ and $\varepsilon_U(v)$ denote currently known lower and upper bounds for the eccentricity of node $v \in V$. For the diameter $\Delta(G)$ of a graph G it holds that $\max_{v \in V} \varepsilon_L(v) \leq \Delta(G) \leq \min(\max_{v \in V} \varepsilon_U(v), 2 * \min_{v \in V} \varepsilon_U(v))$.

We will denote the above lower and upper bounds on the diameter by Δ_L and Δ_U, respectively.

4.2 Algorithm

The bounds mentioned above can be used to refine the original APSP algorithm by reducing the number of eccentricity calculations, as only nodes that can actually contribute to the diameter bounds are considered. Pseudo-code for our algorithm is given in Algorithm 1.

After setting some initial values (lines 3–7), in the main while-loop, our algorithm repeatedly selects a node v (line 9) from the candidate set W, which initially contains all the nodes. The various mechanisms for selecting the next node to be examined are outlined in Section 4.4. The algorithm then calculates the eccentricity of that node v (line 10), and uses the result to update the lower and upper bound of the graph diameter (Δ_L and Δ_U, lines 11–12), cf. Observation 2. Next, the eccentricity bounds of all nodes in the candidate set W are updated (line 14–15) according to Observation 1. Then we determine which nodes (including v) can be removed from the set of candidates (line 17). This can happen either because a node's eccentricity is already known since the lower and upper bounds are identical (which is always the case for the current node v), or because a node can no longer "contribute" to the diameter of the graph by increasing the lower bound or decreasing the upper bound (line 16). Note that because we calculated the eccentricity of v, we know for each node w the distance $d(v,w)$. Finally, the algorithm stops when all nodes have been examined, or when the lower bound is equal to the upper bound (line 8). It then returns the lower bound of the diameter, which at that point contains the real value of the diameter (line 21).

Algorithm 1 BOUNDINGDIAMETERS
```
 1: Input:   Graph $G(V,E)$
 2: Output:   Diameter of $G$
 3: $W \leftarrow V$;     $\Delta_L \leftarrow -\infty$;     $\Delta_U \leftarrow +\infty$;
 4: for $w \in W$ do
 5:     $\varepsilon_L[w] \leftarrow -\infty$;
 6:     $\varepsilon_U[w] \leftarrow +\infty$;
 7: end for
 8: while $\Delta_L \neq \Delta_U$ and $W \neq \emptyset$ do
 9:     $v \leftarrow$ SELECTFROM($W$);          // cf. Section 4.4
10:     $\varepsilon[v]$ = ECCENTRICITY($v$);
11:     $\Delta_L = \max(\Delta_L, \varepsilon[v])$;
12:     $\Delta_U = \min(\Delta_U, 2 * \varepsilon[v])$;
13:     for $w \in W$ do
14:         $\varepsilon_L[w] = \max(\varepsilon_L[w], \max(\varepsilon[v] - d(v,w), d(v,w)))$;
15:         $\varepsilon_U[w] = \min(\varepsilon_U[w], \varepsilon[v] + d(v,w))$;
16:         if ($\varepsilon_U[w] \leq \Delta_L$ and $\varepsilon_L[w] \geq \Delta_U/2$) or
                ($\varepsilon_L[w] = \varepsilon_U[w]$) then
17:             $W \leftarrow W - \{w\}$;
18:         end if
19:     end for
20: end while
21: return $\Delta_L$;
```

4.3 Complexity

Calculating the eccentricity of a node using the ECCENTRICITY() function (line 14) is the critical operation of our algorithm, as this requires running the previously mentioned $O(m)$ SSP algorithm. In the best case, we only have to calculate the eccentricity of *two* nodes v and w, only to find that $\varepsilon(w) = 2 * \varepsilon(v)$ (or vice versa), which means that the diameter is equal to $\varepsilon(w)$. In the worst case our algorithm needs to investigate the eccentricity of every single node, not improving the traditional APSP time complexity of $O(mn)$. An example of a graph in which *all* nodes have to be investigated in order to determine the diameter, is a graph where the nodes are connected through exactly one circle of edges, and will therefore all have identical eccentricity. In general, the eccentricity values of nodes in a network differ, and how much they differ will essentially determine the algorithm's performance, as larger differences in eccentricity values will result in tighter eccentricity bounds on surrounding nodes.

We claim that our algorithm specifically works well on small world networks, which we believe is due to power law degree distribution within such networks, as discussed in Section 2.2. A small world network has relatively few nodes with a very high degree (hubs), that will often (but not always) have a relatively low eccentricity value. The remainder of the nodes typically have a much lower degree, often (again, not always) resulting in a relatively high eccentricity value. Thus, due to the expected existence of a large diversity in eccentricity values of the nodes in a small world network, the bounds on the diameter will typically converge very quickly. Later on we will verify these claims empirically.

4.4 Selection Strategies

We will describe the different strategies that can be used to select the next node for which we want to calculate the eccentricity, essentially outlining the possible functionality of

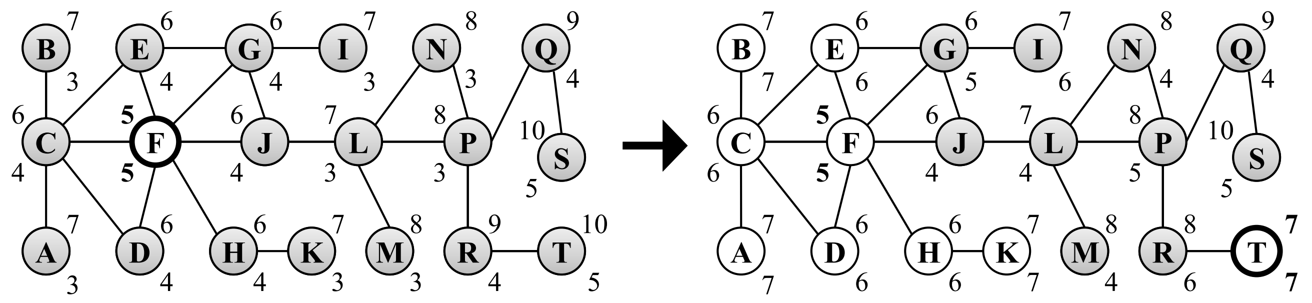

Figure 1: The path B-T realizes a diameter of length 7. Numbers beneath and above nodes denote lower and upper eccentricity bounds after first calculating the eccentricity of node F (left) and then node T (right).

the SELECTFROM() function that is called in line 12 of Algorithm 1. First notice how enumerating the nodes in some order, or selecting them at random, will in essence mean that we are executing the APSP algorithm, only now we discard nodes that can no longer contribute to the diameter bounds. While this will already improve upon the APSP algorithm, our main objective is to tighten the bounds of the diameter as quickly as possible in order to efficiently reduce the size of the set of candidate nodes. The strategy has to be easy to compute so that it does not influence the overall complexity.

The simplest strategy would be to select nodes based on their *degree*. This measure, known as *degree centrality*, is often suggested as a simple measure of the centrality of a node within a network, but is far from perfect with respect to eccentricity. For example, in Figure 1, node F has the highest degree (6 links) and eccentricity $\varepsilon(\text{F}) = 5$, whereas node J with lower degree 3 has eccentricity $\varepsilon(\text{J}) = 4$. The problem here is that degree centrality is merely a local measure. Therefore we suggest using the degree as a secondary selection mechanism only, for example to break ties in other methods, or to select the very first node to be examined.

4.4.1 Eccentricity bound difference

The *difference* between the lower and upper eccentricity bound could be an interesting feature, as it basically tells us how much we already know about the eccentricity of that node and its neighborhood. If for a certain node this difference is very big, determining its eccentricity may tighten the bounds of many nearby nodes.

4.4.2 Interchanging eccentricity bounds

Inspired by traditional branch-and-bound algorithms, we could choose to select nodes from the candidate set based on their their eccentricity lower bound or upper bound. To find nodes with high eccentricity, we select the node with the largest upper bound, and similarly we choose a node with a small lower bound to find nodes with a low eccentricity value. As our goal is to increase the lower bound and decrease the upper bound, we propose to *interchange* the selection of the node with the smallest lower bound and the node with the largest upper bound.

4.4.3 Repeated Farthest Distance

Another option is to select a node based on its distance to the previously investigated node, and then select a node with the highest distance. So essentially, starting from some initial node, we repeatedly select the farthest possible node. This is a variation of the heuristic presented in [13].

4.5 Example

We will give an example of how BOUNDINGDIAMETERS would determine the diameter of the graph depicted in Figure 1. As a selection strategy we alternately choose the largest upper bound and smallest lower bound, breaking ties by choosing the nodes with the highest degree. Any remaining ties are broken by choosing a random node. We will denote the a lower bound L and upper bound U by $[L; U]$.

The left situation of Figure 1 depicts the situation after the first iteration, where node F has been investigated. The diameter lower and upper bounds are now equal to $\Delta_L = \varepsilon(\text{F}) = 5$ and $\Delta_U = 2 * \varepsilon(\text{F}) = 10$, respectively. The current eccentricity bounds do not yet require us to remove any nodes from the candidate set. The eccentricity bounds after the second iteration are depicted in the right situation of Figure 1, where we have determined the eccentricity of node T, which is 7. The graph diameter now lies between $\Delta_L = 7$ and $\Delta_U = 10$. We can now remove A, B, C, D, E, H and K from the candidate list, as we have derived the exact eccentricity of these nodes (but without having calculated it explicitly). We can also remove node I and G, because they can no longer contribute to raising the lower bound or decreasing the upper bound.

For the third loop of our algorithm we calculate the eccentricity of node L, which is 4, lowering the diameter upper bound to $\Delta_U = 8$ and allowing us to update the eccentricity bounds to [4;5] for J, M and N, [5;5] for P, [4;6] for Q, [6;6] for R, [5;7] for S and finally [7;8] for the diameter Δ. We can now discard all nodes based on the same arguments as in the previous iteration, resulting in all nodes being visited, terminating the algorithm after only 3 eccentricity calculations, and returning the final value of the diameter: 7.

4.6 Optimizations

The size of the graph can be reduced by applying the following pruning strategy beforehand. For every node we can determine if removing its edges would disconnect the graph. If this is the case, and multiple identically structured small subgraphs remain, we can remove each but one of them, and still obtain the correct diameter value, assuming of course that a path that realizes the diameter of the graph does not run from one subgraph to another (pruned) subgraph. Therefore, the diameter of the pruned subgraph has to be smaller than $\Delta/2$. For example node C in Figure 1 is connected to two identical subtrees, namely nodes A and B. We could prune one of these subtrees, as they will both have identical eccentricity (bound) values. Similarly, P is connected to identical subtrees Q-S and R-T.

Dataset	Nodes	Links	Average Degree	Average Distance	Δ	Strategy 1	Strategy 2	Strategy 3	Nodes Pruned
AstroPhys [12]	17,903	396K	21	4.15	14	18	**9**	63	185
Enron [9]	33,696	362K	10	4.07	**13**	12	**11**	61	8,715
Flickr [15]	1,624,992	30.9M	18	5.38	24	10	**3**	7	553,242
Hyves	8,057,981	871M	112	4.75	25	40	**21**	44	446,258
LiveJournal [15]	5,189,809	97.4M	19	5.48	23	6	**3**	14	318,378
Orkut [15]	3,072,441	234M	76	4.16	10	357	**106**	389	27,429
Skitter [11]	1,696,415	22.2M	13	5.08	**31**	10	**4**	19	114,803
YouTube [15]	1,134,890	5.98M	5.3	5.32	24	2	**2**	2	399,553
Web [13]	855,802	8.64M	10	6.30	24	20	**4**	28	91,965
Wikipedia [5]	2,213,236	23.5M	11	4.81	18	21	**3**	583	947,582

Table 1: Comparison of different node selection strategies on various small world datasets.

5. EXPERIMENTS

This section starts with a brief description of our datasets, and then describes a measurement methodology for the different selection strategies described in Section 4.4. Next we will present and discuss the results of applying these strategies in our BoundingDiameters algorithm.

5.1 Datasets & Measurement Methodology

We will verify our algorithm on various datasets of small world networks. Characteristics of these datasets, such as the number of nodes, the number of directed links, the average degree, average node-to-node distance and the diameter Δ, are given in the first 6 columns of Table 1. Numbers are based solely on the largest connected component of each graph, therefore slight deviations from statistics presented in the original papers may be observed. The datasets Flickr, LiveJournal, YouTube, Web and Wikipedia are undirected versions of the original respective directed graphs.

In our experiments we will compare the three different selection strategies from Section 4.4, breaking ties as described in Section 4.5. The critical operation is the number of nodes for which we have to calculate the actual eccentricity, which will serve as our basis for comparison of the three strategies outlined in column 7 through 9 of Table 1. Note that the number of comparisons that the traditional APSP algorithm would perform, is equal to the second column, namely one eccentricity calculation for each node. Due to complexity issues (finding frequent subgraphs is NP-hard) we have chosen to only implement the simple optimization strategy (see Section 4.6) of pruning duplicate connected subgraphs consisting of one node. The number of pruned nodes using this technique is given in the last column of Table 1.

5.2 Results

The results in Table 1 clearly show that with only a few eccentricity calculations, our algorithm is able to determine the exact diameter of the datasets, with Strategy 2 as the best-performing node selection strategy. We expect this to be because interchanging the search for low and high eccentricity nodes basically means that we are interchanging the selection of a node in the dense and the peripheral part of the small world network, quickly lowering the upper bound and increasing the lower bound, respectively.

We believe the first strategy did not perform so well, because it only looked for unexplored areas of the graph, which is likely to be less effective when looking for extreme bound values that realize the diameter. The third method appeared to perform worse because it did not appear to find any low-eccentricity valued nodes that could lower the upper bound.

The last column shows how our optimization strategy is able to significantly reduce the size of the problem. This is not surprising; small world networks typically have many low degree nodes, and it is quite likely that many them of are linked to the same node, and can thus be pruned.

For the YouTube dataset, Strategy 2 only needs 2 eccentricity calculations to determine the diameter, demonstrating the best-case performance of our algorithm. It turned out that the node with the highest degree had eccentricity 12, causing nodes with bounds $[12; 24]$ to exist, while another node had eccentricity 24, realizing bounds of $[24; 24]$.

For the orkut dataset, a relatively large number of eccentricity calculations was needed to determine the diameter. To further analyze this, it could be interesting to look at how quickly the amount of candidate nodes decreases over time. Therefore we show the number of unvisited nodes and the lower and upper bounds on the diameter during the execution of our algorithm using Strategy 2 on the Orkut dataset in Figure 2. After 16 calculations, another 90 calculations were needed to decide whether the diameter was equal to 9 or 10, and the number of nodes to be examined only decreases by 1 or 2 after each eccentricity calculation. Apparently the remaining unvisited nodes are positioned in the graph in such a way that the calculation of the actual eccentricity of these nodes can not be avoided using the neighboring bounds. Though it takes a while to find the exact diameter, tight bounds on the diameter are actually quickly available.

Although not related to our performance measurement methodology, we mention that one node eccentricity calculation takes only a few seconds in our C++ implementation on a standard 3.2GHz machine with 10GB of memory. This means that we are able to determine the exact diameter in a matter of seconds (or a few minutes in case of the Orkut dataset), which is very reasonable compared to for example approximation algorithms such as ANF [16].

As an interesting side result it turned out that, very often, the actual eccentricity has been found for a large portion of the nodes in the graph when the algorithm has terminated, because these node's eccentricity lower and upper bounds had become equal. For example, for the Orkut dataset, $777,257$ actual eccentricity values (25%) were obtained, while only 106 values were explicity calculated. Also interesting to note is that the exact diameter that we obtain for the Enron and Skitter datasets deviates from the previously approximated values of 12 and 24, respectively.

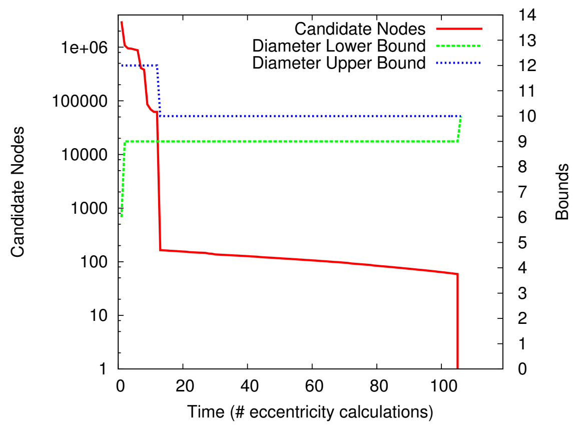

Figure 2: Orkut dataset – candidate nodes (left vertical axis; logarithmic) and lower and upper bounds (right vertical axis) vs. time (horizontal axis).

More information on the datasets used in this paper, the obtained diameter paths, and an implementation can be found at `www.liacs.nl/~ftakes/diameter/`.

6. CONCLUSION

We have shown that our algorithm, BoundingDiameters, is able to efficiently determine the exact diameter of small world networks, making use of lower and upper bounds on the eccentricity of the nodes and on the diameter itself. A proper selection strategy allows our algorithm to exploit the characteristic properties of small world networks. Moreover, we have outlined a pruning strategy which reduces the size of the problem. We have also shown that even when the diameter is not found very quickly, very tight bounds on the diameter are available after only a few iterations.

In future work we would like to investigate if our algorithm can be used to obtain the eccentricity of all nodes in the network, allowing us to study the exact eccentricity distribution of a graph. We also want to address the issue of determining the diameter of the strongly connected component of a directed graph. By using only the lower diameter bounds, significant improvements over the APSP algorithm are already observed. Last but not least, we hope to investigate how the exact diameter of small world networks behaves over time, and how our algorithm can be adjusted to adapt to changes in the network, i.e., the addition and deletion of nodes and links.

7. ACKNOWLEDGMENTS

This research is part of the COMPASS project, financed by NWO under grant number 612.065.926. We thank Hyves for making their anonymized friendship graph available.

8. REFERENCES

[1] D. Aingworth, C. Chekuri, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 547–553, 1996.

[2] R. Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118:4947–4957, 2005.

[3] R. Albert, H. Jeong, and A. Barabási. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999.

[4] A. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4):590–614, 2002.

[5] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia — A crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2009.

[6] D. Dor, S. Halperin, and U. Zwick. All pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

[7] S. Jin and A. Bestavros. Small-world characteristics of internet topologies and implications on multicast scaling. *Computer Networks*, 50(5):648–666, 2006.

[8] J. Kleinberg. The small-world phenomenon: An algorithm perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 163–170, 2000.

[9] B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. *Lecture notes in computer science 3201*, pages 217–226, 2004.

[10] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining*, pages 631–636, 2006.

[11] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining*, pages 177–187, 2005.

[12] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2, 2007.

[13] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[14] C. Magnien, M. Latapy, and M. Habib. Fast computation of empirically tight bounds for the diameter of massive graphs. *Journal of Experimental Algorithmics (JEA)*, 13:1–10, 2009.

[15] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM Conference on Internet Measurement*, pages 29–42, 2007.

[16] C. Palmer, P. Gibbons, and C. Faloutsos. ANF: A fast and scalable tool for data mining in massive graphs. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 81–90, 2002.

[17] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge Press, 1994.

[18] R. Yuster. Computing the diameter polynomially faster than APSP. *ArXiv e-prints, 1011.6181*, 2010.