Finding the Diameter in Real-World Graphs

Experimentally Turning a Lower Bound into an Upper Bound

Pierluigi Crescenzi[1], Roberto Grossi[2], Claudio Imbrenda[2], Leonardo Lanzi[1], and Andrea Marino[1]

[1] Dipartimento di Sistemi e Informatica, Università di Firenze
[2] Dipartimento di Informatica, Università di Pisa

Abstract. The diameter of an unweighted graph is the maximum pairwise distance among its connected vertices. It is one of the main measures in real-world graphs and complex networks. The *double sweep* is a simple method to find a lower bound for the diameter. It chooses a random vertex and performs two breadth-first searches (BFSes), returning the maximum length among the shortest paths thus found. We propose an algorithm called *fringe*, which uses few BFSes to find a *matching* upper bound for almost all the graphs in our dataset of 44 real-world graphs. In the few graphs it cannot, we perform an exhaustive search of the diameter using a cluster of machines for a total of 40 cores. In all cases, the diameter is surprisingly *equal* to the lower bound found after very few executions of the double sweep method. The lesson learned is that the latter can be used to find the diameter of real-world graphs in many more cases than expected, and our fringe algorithm can quickly validate this finding for most of them.

1 Introduction

The diameter D of an unweighted undirected graph $G = (V, E)$ is the maximum pairwise distance $D = \max_{u,v \in V} d(u, v)$ between connected vertices, where $d(u, v)$ denotes the number of edges found along the shortest path from u to v (or vice versa). We assume that G is connected, otherwise we consider its largest connected component to define its diameter. A large body of experimental study on real-world graphs and complex (e.g. social) networks (see, for example, [4]) uses the diameter of the underlying graph as one of the relevant measures to analyze, along with the degree distribution, the size evolution, the local density, the clustering coefficient, and so on.

The textbook algorithm to compute the diameter requires n breadth-first searches (BFSes), each taking $O(n + m)$ time, where $n = |V|$ is the number of vertices and $m = |E|$ is the number of edges in G. This method is too expensive since there might be many graphs to process, with most of them having large size. This situation is well known and several approaches have been proposed (see, for example, [16] and the introduction of [12]). These algorithms require $\Omega(n^2/\text{polylog}\,(n))$ time (and, sometimes, $\Omega(n^2/\text{polylog}\,(n))$ space).

M. de Berg and U. Meyer (Eds.): ESA 2010, Part I, LNCS 6346, pp. 302–313, 2010.

A useful practical alternative is to execute a suitable small number of BFSes to empirically establish some lower and upper bounds, denoted as L and U respectively, such that $L \leq D \leq U$ holds. This is not merely an arbitrary heuristics, since we obtain the actual value of D for the input graph G when $L = U$. For this reason, we call it a *self-checking heuristics*: although there is no guarantee of success for every feasible input, a self-checking heuristics allows the user to certificate that the output is the desired one, namely, the diameter in our case. The drawback that there is no guarantee of success is compensated by the following advantages:

(a) The self-checking heuristics requires few BFSes in practice, and thus its complexity is linear [12].

(b) An empirical upper bound is possible, whereas probabilistic or average-case practical approaches can provide just a lower bound [7] since the diameter is a *worst-case* measure.

(c) Large graphs can be analyzed, since BFS has a good external-memory implementation [14] and works on graphs stored in compressed format [3].

What if the self-checking heuristics terminates returning two values L and U such that $L < U$ holds? The requirement is that this case should be quite rare, so that one can resort to one of the costly algorithms that are always guaranteed to find the diameter.

The *double sweep* is a simple method described in [12,6] that is very effective in finding a lower bound for the diameter: choose a random vertex r, run a BFS at r, and find a vertex a at maximum distance from r; then, run a BFS at a and find a vertex b at maximum distance from a; return the length of the shortest path from a to b.

Let $\mathrm{ecc}(u) = \max_{v \neq u} d(u, v)$ be *eccentricity* of $u \in V$; equivalently, $\mathrm{ecc}(u)$ is the height of the BFS-tree rooted at u. Hence, the double sweep method computes $\mathrm{ecc}(a)$. It is iterated over a small number of experiments, so that it returns L as the maximum $\mathrm{ecc}(a)$ over the vertices a that are identified by each iteration. A good heuristics is to mark the already identified as, so that they are always chosen to be distinct one from the other.

Some authors noted that the above method empirically finds the diameter for some classes of graphs (see, for instance, [8]). In this paper, we also show that the value found by double sweep is always better than the lower bound provided with the graphs available in [10]. Note that there are cases in which double sweep can badly fail as shown in Section 4. However, having more than one iteration seems to help in practice.

To our knowledge, the best upper bound for a self-checking heuristics is described in [12]. It can be very close to L, but it rarely matches: in this paper, we will see that in the case of 44 networks, it matches only 12 times. We will discuss more about these methods in Section 4.

Our results. In this paper, we introduce a simple but powerful algorithm, called *fringe*, that experimentally provides a *matching* upper bound U (i.e. $U = L$) in many cases. The idea behind our self-checking heuristics is that of using the double sweep as a starting point that identifies the three vertices r, a and b. Not only we use its value L as a lower bound, but we then consider the vertex u that is halfway along the shortest path connecting a and b to find an upper bound U. We use u as if it were the "center" of the graph G (whatever "center" means). Clearly, there are situations in which u is far away from the real center of G, and we provide a refinement of the above idea to deal with these situations. We refer the reader to the notion of fringe introduced in Section 2, where the description of our algorithm is given.

What is interesting is that using a dataset of 44 real-world graphs described in Section 3, our experiments in Section 4 show that the fringe algorithm provably finds a matching upper bound in all graphs except few of them (just 5 in number). For the latter graphs, the upper bounds are tighter or no worse than the best known in the literature. We measure the performance of the methods by counting how many BFSes are executed, which is a realistic indicator of the running time.

Even more interesting, we ran long-term experiments with five 24GB-RAM machines for a total of 40 cores, to compute (using the textbook algorithm) the diameter for the few graphs for which the fringe algorithm did not find a matching upper bound. To our surprise, the diameter of these graphs is $D = L$ in all cases, where L is the lower bound found by the double sweep [12,6].

The lesson learned in our large-scale experiments is the following. Run the double sweep method to have an educated guess of what is the diameter of a large graph. Use our fringe algorithm to validate this fact. In the few cases in which $L < U$, run any $\Omega(n^2/\text{polylog}\,(n))$-time algorithm with guaranteed result.

Our study indicates two further lines of research. The former is related to the weighted graphs, and seeks for simple practical methods to find (nearly) tight lower and upper bounds for the *weighted* diameter, in which the distance $d(\cdot,\cdot)$ takes into account the edge weights. The latter poses the question of finding a theoretical $o(n^2/\text{polylog}\,(n))$-time algorithm that can check if a given value D is the diameter of the input graph G. Note that we only require the algorithm to produce a binary answer. This is enough, since we can employ it for $D = L, L+1, \ldots, U$ since L and U are usually quite close each other.

2 The Fringe Algorithm

Here we describe our *fringe* method to improve the upper bound U and possibly match the lower bound L obtained by the *double sweep* method of [12,6]. The full code that combines the latter two methods, improving either L or U (or both) whenever possible, is reported in `diameter.algoritmica.org`.

We consider an unweighted undirected graph $G = (V, E)$. We also assume that G is connected (if not, G denotes its largest connected component). For any vertex $u \in V$, let T_u denote an unordered BFS-tree obtained by starting the BFS traversal of G from u. We recall that the eccentricity $\text{ecc}(u)$ is the height of T_u, and that $2 \cdot \text{ecc}(u) \geq \text{diam}(G)$.

We also denote the diameter of a subgraph $G' \subseteq G$ by $\text{diam}(G')$, where $G' = (V', E')$, $V' \subseteq V$, and $E' \subseteq E$. If $V' = V$ and G' is connected, G' satisfies the property that $\text{diam}(G') \geq \text{diam}(G)$ since the edges in $E \setminus E'$ are not taken into account. We will use this fact for $G' \equiv T_u$, namely, $\text{diam}(T_u) \geq \text{diam}(G)$.

We define the *fringe* of u, denoted $F(u)$, as the set of vertices $v \in V$ such that $d(u, v) = \mathrm{ecc}(u)$. We observe that these vertices correspond to the leaves of T_u having maximum depth. This notion is central to our algorithm since we are interested in vertices u of real-word graphs for which $|F(u)|$ is not so large. Let $B(u) = \max_{z \in F(u)} \mathrm{ecc}(z)$. We now define the upper bound $U(u)$ computed by the fringe algorithm starting from a certain node u.

$$U(u) = \begin{cases} 2 \cdot \mathrm{ecc}(u) - 1 & \text{if } |F(u)| > 1 \text{ and } B(u) = 2 \cdot \mathrm{ecc}(u) - 1 \\ 2 \cdot \mathrm{ecc}(u) - 2 & \text{if } |F(u)| > 1 \text{ and } B(u) < 2 \cdot \mathrm{ecc}(u) - 1 \\ \mathrm{diam}(T_u) & \text{otherwise} \end{cases} \tag{1}$$

Lemma 1. *$U(u) \geq D$, where D is the diameter of G.*

Proof. Since $D \equiv \mathrm{diam}(G)$, it is always upper bounded by $\mathrm{diam}(T_u)$. The *default* case is therefore $U(u) = \mathrm{diam}(T_u)$. We now consider some cases explicitly, and see if they can improve over the default case. If the fringe $F(u)$ contains just one vertex, that vertex belongs to $\mathrm{diam}(T_u)$, and the default case holds. Hence, we assume that $F(u)$ contains more than one vertex, and consider $B(u)$.

When $B(u) = 2 \cdot \mathrm{ecc}(u)$, we actually have $D = B(u)$: There must exist two vertices in $F(u)$ that (a) are among the leaves of T_u, and (b) have the root of T_u as their lowest common ancestor, and (c) are hit by the double sweep at u, which finds that $L = 2 \cdot \mathrm{ecc}(u)$. Since $B(u) = \mathrm{diam}(T_u)$ here, the default case covers this situation.

When $B(u) = 2 \cdot \mathrm{ecc}(u) - 1$, we also find $D = B(u)$: In this case, we have $d(x, y) \leq 2 \cdot \mathrm{ecc}(u) - 1$ for any two distinct vertices x, y such that $x \in F(u)$ or $y \in F(u)$ (possibly both). Suppose by contradiction that $D \geq 2 \cdot \mathrm{ecc}(u)$. This implies that there must exist two distinct vertices $x', y' \in V \setminus F(u)$ such that $d(x', y') \geq 2 \cdot \mathrm{ecc}(u)$. But since they are both not in the fringe of u, their connecting path inside T_u is of length at most $2 \cdot \mathrm{ecc}(u) - 2$, thus contradicting the fact that their distance $d(x', y')$ is longer. Also, the double sweep at u finds that $L = 2 \cdot \mathrm{ecc}(u) - 1$, so it cannot be $D < 2 \cdot \mathrm{ecc}(u) - 1$. Hence, we set $U(u) = B(u)$.

When $B(u) < 2 \cdot \mathrm{ecc}(u) - 1$, we can set $U(u) = 2 \cdot \mathrm{ecc}(u) - 2$ since the diameter can be equal or smaller. The argument is analogous to the previous case: Suppose by contradiction that $D \geq 2 \cdot \mathrm{ecc}(u) - 1$. This implies that there must exist two distinct vertices $x', y' \in V \setminus F(u)$ such that $d(x', y') \geq 2 \cdot \mathrm{ecc}(u) - 1$. But this is a contradiction since $d(x', y') \leq 2 \cdot \mathrm{ecc}(u) - 2$. □

We have the ingredients of the fringe algorithm for finding an upper bound U:

1. Let r, a, and b be the vertices identified by double sweep (using two BFSes).
2. Find the vertex u that is halfway along the path connecting a and b inside the BFS-tree T_a (with ties broken arbitrarily).
3. Compute the BFS-tree T_u and its eccentricity $\mathrm{ecc}(u)$.
4. If $|F(u)| > 1$, find the BFS-tree T_z for each $z \in F(u)$, and compute $B(u)$:
   - If $B(u) = 2 \cdot \mathrm{ecc}(u) - 1$, return $2 \cdot \mathrm{ecc}(u) - 1$.
   - If $B(u) < 2 \cdot \mathrm{ecc}(u) - 1$, return $2 \cdot \mathrm{ecc}(u) - 2$.
5. Return the diameter $\mathrm{diam}(T_u)$.

Theorem 1. *The fringe algorithm correctly computes an upper bound for the diameter of the input graph G, using at most $|F(u)| + 3$ breadth-first searches.*

The proof of Theorem 1 immediately follows from Lemma 1 and by inspection of the fringe algorithm. We can use a slack parameter $F_{\max}$ in practice: if $|F(u)| > F_{\max}$, we skip all the $|F(u)|$ BFSes and simply return $\mathrm{diam}(T_u)$. Since we iterate the algorithm over several choices of r, a, b, u, this guarantees that we never exceed $F_{\max} + 3$ BFSes per iteration. Additionally, we avoid calculating $U(u)$ when $2 \cdot \mathrm{ecc}(u) - 2$ is not an improvement over the best upper bound found in previous iterations. We refer the reader to the full code in our website.

3 The Datasets

We chose our set of real-word graphs, so as to cover the largest taxonomy as possibile. The list of graphs, their features, and their source are reported in Table 1 (Note that some graphs have been made undirected, even though they were originally directed.) Here, we simply list them: a detailed description of each graph can be found in our website (`diameter.algoritmica.org`).

Social networks. In on-line social networks, nodes represent people and edges represent interactions between people: Epinions social network (`soc-Epinions1`), LiveJournal social network (`soc-LiveJournal1`), Slashdot social network, November 2008 (`soc-Slashdot0811`), Wikipedia vote network (`wiki-Vote`), Slashdot social network, February 2009 (`soc-Slashdot0902`).

Communication networks. In communication networks, nodes represent people and edges represent communication among them: Enron email network (`email-Enron`), EU email communication network (`email-EuAll`), Wikipedia Talk network (`wiki-Talk`).

Citation networks. In this kind of network, nodes represent papers and edges represent citations: Patent citation network (`cit-Patents`), CiteSeer (`CiteSeer`), Hep-th Citation Graph(`hep-th-citations-MAX`).

Collaboration networks. In the collaboration networks, nodes represent people and edges represent collaborations: Astro Physics collaboration network (`ca-AstroPh`), Condense Matter collaboration network (`ca-CondMat`), General Relativity and Quantum Cosmology collaboration network (`ca-GrQc`), High Energy Physics - Phenomenology collaboration network (`ca-HepPh`), High Energy Physics - Theory collaboration network (`ca-HepTh`), Actor collaboration network (`IMDB`), DBLP co-author network (`dblp20080824-MAX`), free software development collaboration network (`advogato`).

Web graphs. In Web graphs, nodes represent webpages and edges are hyperlinks: (`web`) (uk-2005) [5], and [2] (`arabic-2005`), (`cnr-2000`),(`eu-2005`), (`in-2004`), (`indochina-2004`), (`it-2004`), (`sk-2005`).

Product co-purchasing networks. In these cases, nodes represent products and edges link commonly co-purchased products: Amazon product co-purchasing network, March 02 2003 (`amazon0302`), Amazon product co-purchasing network,

Table 1. The data set (the fifth and the sixth column refer to the largest connected component)

Network	Acronym	Nodes	Edges	Nodes l.c.c.	Edges l.c.c.	Source
advogato	ADVO	7418	96074	5272	91806	[13]
amazon0302	AMA1	262111	1799582	262111	1799582	[10]
amazon0312	AMA2	400727	4699736	400727	4699736	[10]
Amazon0505	AMA3	410236	4878872	410236	4878872	[10]
arabic-2005	ARAB	22743881	1107806146	22634275	1104463734	[2]
as-skitter	ASSK	1696415	22190596	1694616	22188418	[10]
ca-AstroPh	CAAS	18771	396100	17903	393944	[10]
ca-CondMat	CACO	23133	186878	21363	182572	[10]
ca-GrQc	CAGR	5241	28968	4158	26844	[10]
ca-HepPh	CAH1	12006	236978	11204	235238	[10]
ca-HepTh	CAH2	9875	51946	8638	49612	[10]
cit-HepPh	CIT1	34546	841840	34401	841654	[10]
cit-HepTh	CIT2	27770	704646	27400	704116	[10]
cit-Patents	CITP	3774768	33037894	3764117	33023480	[10]
citeseer	CITE	259217	1064080	220997	1010654	[11]
cnr-2000	CNR2	325557	5477938	325557	5477938	[2]
dblp20080824-MAX	DBLP	511163	3742140	511163	3742140	[15]
dip20090126-MAX	DIP2	19928	82404	19928	82404	[15]
email-Enron	EMA1	36691	367660	33695	361620	[10]
email-EuAll	EMA2	265214	731138	224832	681588	[10]
eu-2005	EU20	862664	32276936	862664	32276936	[2]
HC-BIOGRID	HCBI	4039	20642	4039	20642	[15]
hep-th-citations-MAX	HEPT	27400	704042	27400	704042	[15]
imdb	IMDB	908830	75177226	880455	74989272	[9]
in-2004	IN20	1353703	26252344	1353703	26252344	[2]
indochina-2004	INDO	7414758	301969638	7320539	298109708	[2]
it-2004	IT20	41290577	2054949790	41290577	2054949790	[2]
itdk0304-rlinks-undirected	ITDK	192244	1218132	190914	1215220	[15]
p2p-Gnutella31	P2PG	62586	295782	62561	295754	[10]
p2p	P2P	5380578	284076802	5380491	284076702	[5]
roadNet-CA	ROA1	1965206	5533214	1957027	5520776	[10]
roadNet-PA	ROA2	1088092	3083796	1087562	3083028	[10]
roadNet-TX	ROA3	1379917	3843320	1351137	3758402	[10]
sk-2005	SK20	50634118	3620101486	50634118	3620101486	[2]
soc-Epinions1	SOCE	75879	811478	75877	811476	[10]
soc-LiveJournal1	SOCL	4847571	86739236	4843953	86725498	[10]
soc-sign-epinions	SOC1	131827	1423564	119130	1409144	[10]
soc-sign-Slashdot090221	SOC2	82140	1000960	82140	1000960	[10]
soc-Slashdot0811	SOC3	77360	1092972	77360	1092972	[10]
soc-Slashdot0902	SOC4	82168	1165064	82168	1165064	[10]
trust	TRUS	49288	762434	49288	762434	[13]
web	WEB	39454463	1566054250	39252879	1562879784	[5]
wiki-Talk	WIK1	2394385	9319128	2388953	9313362	[10]
wiki-Vote	WIK2	7115	201522	7066	201470	[10]

March 12 2003 (amazon0312), Amazon product co-purchasing net, May 05 2003 (amazon0505).

Internet peer-to-peer networks. In peer-to-peer networks, nodes represent computers and edges represent communication among them: Gnutella peer-to-peer network, August 31 2002 (p2p-Gnutella31), [5](p2p).

Road networks. In this case, intersections and endpoints are represented by nodes and the roads connecting these intersections or road endpoints are represented by undirected edges: California road network (roadNet-CA), network of Pennsylvania (roadNet-PA).

Autonomous systems graphs. These are graphs of the Internet: Autonomous systems by Skitter (as-Skitter), Router topology (itdk0304-rlinks-undirected).

Signed networks. These are networks with positive and negative edges such as friend/foe, trust/distrust, and so on: Epinions social network (soc-sign-epinions), Slashdot social network, February 2009 (soc-sign-Slashdot090221).

Biological networks. These graphs refer to databases of physical and genetic and biological interactions: (HC-BIOGRID), Database of Interacting Proteins (dip20090126-MAX).

4 Experiments

In this section we describe both the experimental setting and the obtained results. All the code, the data set and the logs of the experiments are available at diameter.algoritmica.org. The data set is distributed according to a unique format, which is the same adopted by [12]: to this aim, we have implemented several conversion utilities that are also available at the previously specified URL.

Setting. We used 5 machines with 8 cores each (Intel Xeon CPU X5570 at 2.93GHz), where each core has 8 MB shared cache with a 24 GB shared memory. The operating system is CentOS 5.3 with a Linux kernel version 2.6.18 and gcc version 4.1.2. This huge computational power was mainly aimed at allowing us to run the textbook algorithms for computing the diameter when our fringe algorithm does not provide a tight upper bound.

Upper bound computation methods. We compared an implementation of our fringe algorithm, called fub, with three methods for computing upper bounds on the diameter. The first two methods, called rtub and hdtub respectively, are taken from [12]. In particular, rtub selects a random node r and returns the diameter of T_r, while hdtub chooses r as one of the nodes with the highest degree. The third method, called mtub, is a simplification of fub, in which we simply returns the diameter of T_u.

Experiment description. For each of the 44 networks described in Section 3 and for each of the above four upper bound computation methods, we have performed 10 experiments. Each experiment consisted of 10 executions of the algorithm at hand: for each execution, $F_{\max}$ is set to 50000 (in other words, no practical bound is set to the number of BFSes to be performed).

Results. The results of our experiments are summarized in Table 2, where each row corresponds to one network. The first column indicates the acronym of the network, while the second column shows the corresponding diameter D. The third and the fourth columns indicate the lower bound L computed by the double sweep method and the number X_L of its executions returning L as output. Moreover, we report the value S_L which has been computed as follows. For any i with $1 \leq i \leq X_L$, let s_i be the minimum number of breadth-first searches required by the i-th experiment returning L as output for the first time: S_L is equal to $\sum_{i=1}^{X_L} s_i/X_L$.[1] In other words, S_L indicates how quickly the double sweep algorithm converges to its best lower bound.

For each of the four upper bound algorithms, we report in the table the best upper bound U and the number X_U of executions returning U as output. Moreover, we report the value S_U which has been computed analogously to S_L. Namely, for any i with $1 \leq i \leq X_U$, let s_i be the minimum number of breadth-first searches required by the i-th experiment returning U as output for the first time: S_U is equal to $\sum_{i=1}^{X_U} s_i/X_U$. In other words, S_U indicates how quickly the algorithm converges to its best upper bound. Finally, in the case of the `fub` method, we also report B, which is the maximum size $|F(u)|$ of the fringe: by Theorem 1, the value $B+3$ indicates the maximum number of breadth-first searches that are required by a single execution of the algorithm.

In the table, for each row we highlight the best upper bound computed by any of the four methods for the corresponding network.

Discussion. The first impressive observation is that the two columns D and L are identical, that is, the value returned by the double sweep method is actually an upper bound too. By looking at columns X_L and S_L, moreover, we can observe that this bound is frequently and quickly achieved. Indeed, almost for every network the tight bound is computed at every experiment (that is, 10 times) and most of the times (that is, 31 times) two breadth-first searches are sufficient to compute this bound. In any case, at most 15 breadth-first searches are required (in the case of a road network).

The second important observation is that, in almost all networks, the tightness of L can be experimentally proved in a very efficient way by using our fringe algorithm. Indeed, by looking at columns U, X_U, and S_U corresponding to `fub`, we can observe that (1) only 7 values of U are not highlighted (that is, `fub` almost always computes the best upper bound), (2) the empirical probability of `fub` returning the best upper bound is high (on the average 0.93), and (3) the number S_L of breadth-first searches required by `fub` in order to prove the tightness of

[1] Notice that since, for each experiment, we perform 10 executions of the double sweep algorithm, then $2 \leq S_L \leq 20$.

Table 2. Summary of experimental results (in the case of the starred upper bound values, by performing a slightly greater number of executions, we have been able to compute either a tight upper bound (in the case of CAH2 and of DBLP) or an almost tight upper bound (12 in the case of P2PG)))

		dslb			fub				mtub			hdtub			rtub		
	D	L	X_L	S_L	U	X_U	S_U	B	U	X_U	S_U	U	X_U	S_U	U	X_U	S_U
ADVO	9	9	10	2	**9**	10	6	9	**9**	10	6	**9**	10	4	**9**	2	24
AMA1	38	38	10	2	**38**	10	3	7	**38**	10	4	**38**	10	12	**38**	2	8
AMA2	20	20	10	2	**20**	10	18	9	22	10	12	21	10	4	21	2	30
AMA3	22	22	10	2	**22**	10	3	11	**22**	10	4	**22**	10	4	**22**	3	21
ARAB	47	47	10	2	**47**	10	9	6	48	10	4	51	10	8	50	2	14
ASSK	31	31	10	12	**31**	10	5	3	32	10	4	34	10	28	34	5	33
CAAS	14	14	10	2	**14**	10	11	4	15	10	9	15	10	12	16	2	28
CACO	15	15	10	2	**15**	7	31	9	16	8	30	17	10	4	17	2	26
CAGR	17	17	10	2	**17**	10	23	15	18	10	18	19	10	4	18	2	10
CAH1	13	13	10	2	**13**	10	50	63	14	10	16	15	10	4	15	3	25
CAH2	18	18	10	2	20*	10	4	2	20	10	4	20	10	16	20	2	30
CIT1	14	14	10	5	**14**	10	13	3	15	10	8	15	10	20	16	9	14
CIT2	15	15	10	2	**15**	10	20	8	16	10	17	16	10	20	16	1	24
CITP	26	26	10	2	28	10	12	4	30	10	4	29	10	8	31	1	40
CITE	52	52	10	2	**52**	10	3	11	**52**	10	4	54	10	4	**52**	2	16
CNR2	34	34	10	2	**34**	9	11	3	**34**	9	19	39	10	4	35	10	10
DBLP	22	22	10	2	24*	2	27	25	24	1	21	24	10	8	25	4	13
DIP2	30	30	10	8	**30**	10	9	3	**30**	10	27	33	10	12	31	1	16
EMA1	13	13	10	2	**13**	10	12	9	**13**	10	18	**13**	10	24	**13**	1	32
EMA2	14	14	10	2	**14**	10	3	48	**14**	10	4	15	10	4	15	7	9
EU20	21	21	10	2	**21**	10	14	15	22	10	4	25	10	4	23	3	6
HCBI	23	23	10	4	**23**	7	26	3	**23**	6	28	**23**	10	32	**23**	1	28
HEPT	15	15	10	2	**15**	10	19	4	16	10	11	16	10	20	17	2	28
IMDB	14	14	10	2	**14**	10	17	18	15	8	25	16	10	4	16	3	29
IN20	43	43	10	2	**43**	10	5	11	44	10	4	44	10	8	**43**	1	8
INDO	43	43	10	2	**43**	8	39	21	44	10	20	44	10	8	45	4	14
IT20	45	45	10	2	**45**	10	30	15	46	10	6	47	10	4	46	3	29
ITDK	26	26	10	2	**26**	10	7	5	28	10	4	29	10	4	28	4	17
P2PG	11	11	10	3	14*	10	14	132	15	9	20	14	10	12	15	6	19
P2P	9	9	10	4	**9**	8	1125	1266	10	7	22	10	10	4	11	6	28
ROA1	865	865	9	15	987	9	8	2	987	9	12	1047	10	16	988	1	32
ROA2	794	794	8	12	803	8	22	5	803	6	13	873	10	32	832	1	36
ROA3	1064	1064	10	8	1079	3	18	9	1079	1	16	1166	10	24	1128	1	24
SK20	40	40	10	5	**40**	10	18	18	41	10	21	41	10	4	41	7	12
SOCE	15	15	10	4	**15**	10	19	11	16	10	9	16	10	4	16	8	17
SOCL	20	20	10	2	**20**	10	7	12	**20**	10	8	**20**	10	28	22	7	19
SOC1	16	16	10	2	**16**	10	3	5	**16**	10	4	**16**	10	8	17	8	11
SOC2	13	13	10	2	**13**	10	7	17	**13**	10	4	**13**	10	20	14	7	18
SOC3	12	12	10	6	**12**	10	13	13	13	10	6	**12**	10	20	13	2	32
SOC4	13	13	10	3	**13**	10	7	17	14	10	4	**13**	10	20	14	6	18
TRUS	14	14	10	2	**14**	10	3	2	**14**	10	4	15	10	4	15	8	19
WEB	32	32	10	2	**32**	10	52	63	34	10	4	38	10	4	34	6	20
WIK1	11	11	10	2	**11**	10	9	10	12	10	4	12	10	8	12	10	12
WIK2	7	7	10	2	**7**	10	64	134	8	10	8	8	10	4	8	4	22

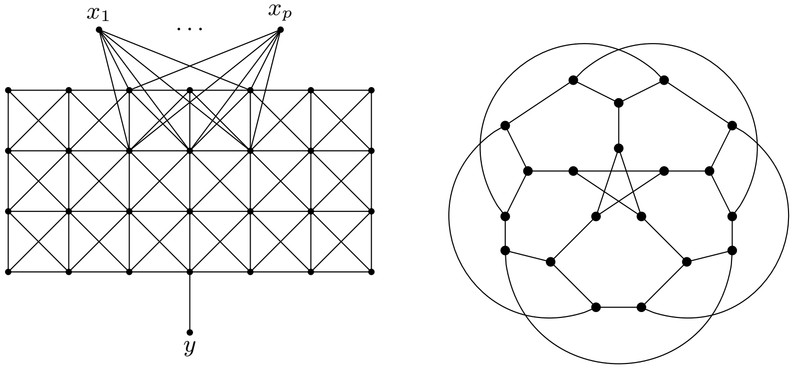

Fig. 1. A bad network ($k = 4$) for the double sweep algorithm (left) and for the fringe algorithm (right)

L is incredibly low with respect to the number of vertices in the network (apart from the p2p network in which case the number of searches is 1125). It is also worth observing that in the case of 22 networks, our algorithm is the only one able to compute the tight upper bound.

Note that, as one could expect, the maximum size of the fringe is most of the times greater than the average number of performed breadth-first searches: however, it is worth noting that this value is quite small with respect to the number of vertices in the network.

As we already said, our fringe algorithm fails to prove the tightness of the lower bound only in 7 cases. In the case of the ca-HepTh, however, if we execute the algorithm 20 times (instead of 10), the correct upper bound is computed and the tightness of the lower bound can be proved. The same holds in the case of the dblp20080824-MAX network. In the case of the p2p-Gnutella31 network, by running the algorithm 200 times, we have been able to lower the upper bound down to 12: however, this bound is not tight. In the case of the cit-Patents network and of the three road networks, finally, we have not been able to obtain better results, even by increasing the number of algorithm executions.

One might object that in some cases the number of breadth-first searches performed by our fringe algorithm is higher than the number of searches performed by other algorithms. For instance, in the case of the ca-HepPh network, the value S_U corresponding to fub is 50, while the value corresponding to rtub is 25. We then executed this latter algorithm with a much greater number of breadth-first searches (that is, 400). In this case, the bound returned by rtub improved to 14, but it did not reach the lower bound 13 (which is tight).

We have also tried to modify our algorithm by introducing a different selection of the vertex u, that is, by choosing as u the highest degree vertex. It turns out, however, that this modification almost always returns bounds which are higher than the bounds obtained by fub.

Finally, going back to the impressive performance of the double sweep algorithm, we would like to point out that it is not difficult to design a network for which the behavior of this algorithm is not very good (see the left part of Figure 1). To this aim consider a unitary grid with k rows and $1+3k/2$ columns. Each vertex of the grid is connected to all vertices whose Euclidean distance is at most $\sqrt{2}$. Moreover, there are p additional vertices $x_1, x_2, \ldots, x_p$ which are connected to the neighbors of the middle point of the upper row of the grid and one additional vertex y which is connected to the middle point of the lower row of the grid. If p is sufficiently large with respect to k, then the `dslb` algorithm will choose one of these points as the vertex r. As a consequence, the vertex y will be chosen as the vertex a of the algorithm. The breadth-first search starting from y will have height equal to $k+1$: this will be the value reported by the algorithm. It is easy to see that the diameter of the network is instead $3k/2$.

Note that, in the case of the graph shown in the left part of Figure 1 with $p = 100000$, the fringe algorithm, for every experiment, computes an upper bound which is equal to the diameter of the graph (that, is 6) by executing (as expected) 7 BFSes on the average. However, it is easy to find combinations of the values of p and k for which the computed upper bound is not equal to the diameter: for instance, if $k = 8$ and $p = 100000$, the best answer we obtained was 15. In the case of the fringe algorithm (and of the other upper bound methods), moreover, we can define a family of graphs for which the probability of computing the correct value is 0: these graphs have the property that no BFS tree has the diameter equal to the diameter of the graph (an example of such graphs is shown in the right part of Figure 1 [1]). Note that in the case of these latter graphs the lower bound computed by the double sweep method is tight with probability 1, since the height of each BFS tree is equal to the diameter of the graph.

5 Conclusion

In this paper, we have proposed an algorithm which uses few BFSes to find an upper bound which matches, for almost all the graphs in our dataset, the lower bound computed by the double sweep method to find a lower bound for the diameter. By using this algorithm and, in few cases, the textbook algorithm we have been able to prove that, in all networks, the diameter is equal to the lower bound found by the double sweep algorithm. The lesson learned is that this latter method can be used to find the diameter of real-world graphs in many more cases than expected, and our fringe algorithm can quickly validate this finding for most of them.

It is an interesting open question to understand why, in the case of some real-world graphs, the fringe method fails to validate the lower bound computed by the double sweep algorithm: to this aim, it might be useful to investigate the similarities between these real-world graphs and the synthetic graphs introduced at the end of the previous section. It would also be interesting to explore the behavior of both the double sweep and the fringe algorithms when applied to synthetic networks produced by well-known models, such as the Erdös-Rényi or the preferential attachment models.

Finally, as already specified in the introduction, our study indicates two further lines of research: the former is related to the weighted graphs, while the latter is related to the question of finding a theoretical $o(n^2/\text{polylog}\,(n))$-time algorithm that can check if a given value D is the diameter of the input graph G.

Acknowledgments. The first author would like to thank Michel Habib for introducing him to the double sweep algorithm. We all wish to thank Maurizio Davini for his effort to provide us with powerful machines to run the experiments. We are grateful to Andrea Clementi, Francesco Pasquale, and Riccardo Silvestri for suggesting us the counter-example for the double sweep algorithm. Finally, we thank the anonymous referees for their comments.

References

1. Alegre, I., Fiol, M., Yebra, J.: Some large graphs with given degree and diameter. J. Graph Theory 10, 219–224 (1986)
2. Boldi, P., Vigna, S.: Webgraph (2001), http://webgraph.dsi.unimi.it/
3. Boldi, P., Vigna, S.: The WebGraph framework I: Compression techniques. In: Proc. of the 13th International World Wide Web Conference, pp. 595–601 (2004)
4. Brandes, U., Erlebach, T.: Network Analysis: Methodological Foundations. Springer, Heidelberg (2005)
5. Complexnetworks Team: Complex networks and real-world graphs (2008), http://complexnetworks.fr/
6. Corneil, D.G., Dragan, F.E., Habib, M., Paul, C.: Diameter determination on restricted graph families. Discrete Appl. Math. 113(2-3), 143–166 (2001)
7. Faloutsos, C.: Graph mining: Patterns, generators and tools. In: Combinatorial Pattern Matching, p. 274 (2009)
8. Handler, G.: Minimax location of a facility in an undirected tree graph. Transportation Science 7(287–293) (1973)
9. IMDB: The internet movie database (1990), http://www.imdb.com/
10. Leskovec, J.: Stanford Network Analysis Package (SNAP) Website (2009), http://snap.stanford.edu
11. Library, S.L.D.: Citeseer Website (1997), http://citeseer.ist.psu.edu/citeseer.html
12. Magnien, C., Latapy, M., Habib, M.: Fast computation of empirically tight bounds for the diameter of massive graphs. J. Exp. Algorithmics 13 (2009)
13. Massa, P., Souren, K.: Trustlet Website (2007), http://www.trustlet.org
14. Mehlhorn, K., Meyer, U.: External-memory breadth-first search with sublinear i/o. In: Proceedings of the 10th Annual European Symposium on Algorithms, pp. 723–735 (2002)
15. Sommer, C.: Christian sommer's homepage (2009), http://www.sommer.jp/graphs/
16. Zwick, U.: Exact and approximate distances in graphs - a survey. In: Meyer auf der Heide, F. (ed.) ESA 2001. LNCS, vol. 2161, pp. 33–48. Springer, Heidelberg (2001)