On Computing the Diameter of Real-World Directed (Weighted) Graphs

Pierluigi Crescenzi Roberto Grossi
Leonardo Lanzi Andrea Marino

University of Florence and University of Pisa, Italy

11th International Symposium on Experimental Algorithms
June 7-9, 2012
Bordeaux, France

Definition

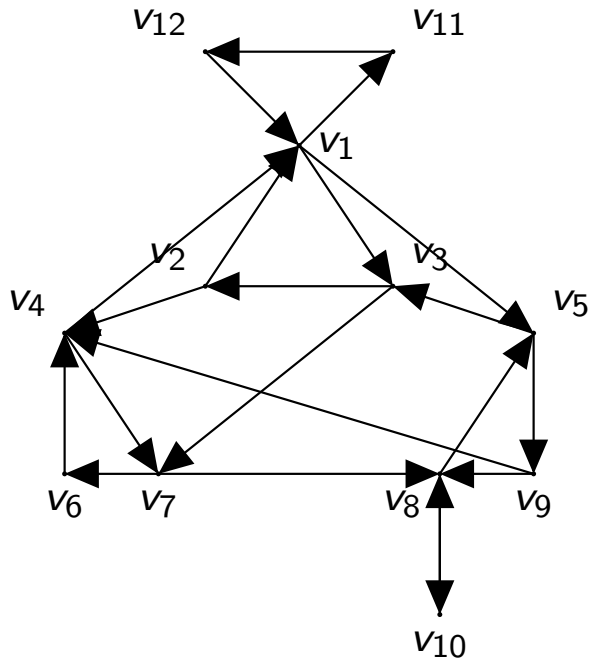
- (Un)directed graph $G = (V, E)$ strongly connected.
- The distance $d(u, v)$ is the number of edges along shortest path from u to v.
- The diameter D of a graph is the length of the longest shortest path, $D = \max_{u,v \in V} d(u, v)$

Motivations

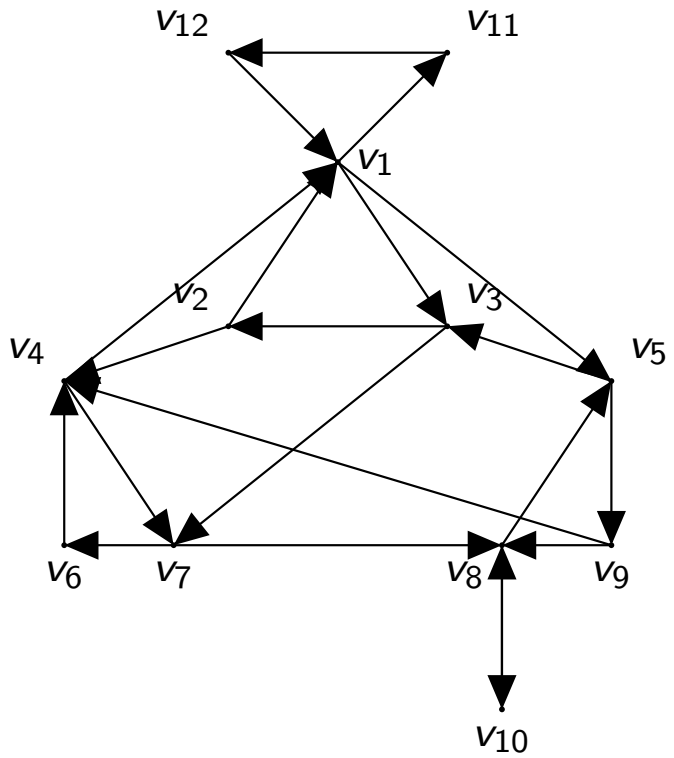
- Social networks
  - how quickly information reaches every individual
- Web graphs
  - how quickly, in terms of mouse clicks, any page can be reached
- Biological networks
  - how many reactions have to be performed in order to produce any metabolite from any other metabolite
- Communication networks
  - completion time of broadcast protocols based on flooding

Definition

- Forward Eccentricity of u: in how many hops u can reach any node?
  - $\mathrm{ecc_F}(u) = \max_{v \in V} d(u, v)$
- Backward Eccentricity of u: in how many hops u can be reached starting from any node?
  - $\mathrm{ecc_B}(u) = \max_{v \in V} d(v, u)$
- Diameter: maximum $\mathrm{ecc_F}$ or $\mathrm{ecc_B}$



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	ecc_F
v_1	0	2	1	3	1	3	2	3	2	4	1	2	4
v_2	1	0	2	1	2	3	2	3	3	4	2	3	4
v_3	2	1	0	2	3	2	1	2	4	3	3	4	4
v_4	1	3	2	0	2	2	1	2	3	3	2	3	3
v_5	3	2	1	2	0	3	2	2	1	3	4	5	5
v_6	2	4	3	1	3	0	2	3	4	4	3	4	4
v_7	3	4	3	2	2	1	0	1	3	2	4	5	5
v_8	4	3	2	3	1	4	3	0	2	1	5	6	6
v_9	2	4	3	1	2	3	2	1	0	2	3	4	4
v_{10}	5	4	3	4	2	5	4	1	3	0	6	7	7
v_{11}	2	4	3	5	3	5	4	5	4	6	0	1	6
v_{12}	1	3	2	4	2	4	3	4	3	5	2	0	5
ecc_B	5	4	3	5	3	5	4	5	4	6	6	7	

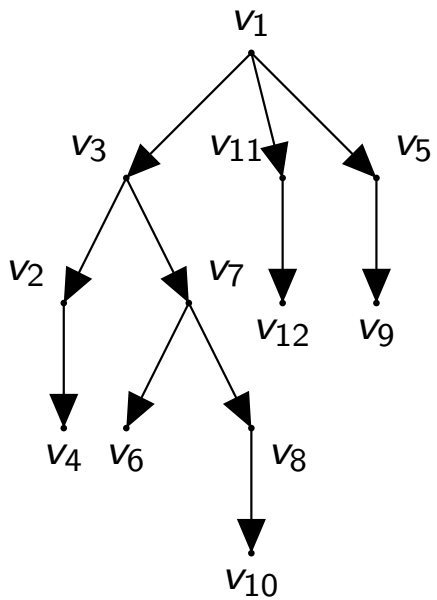


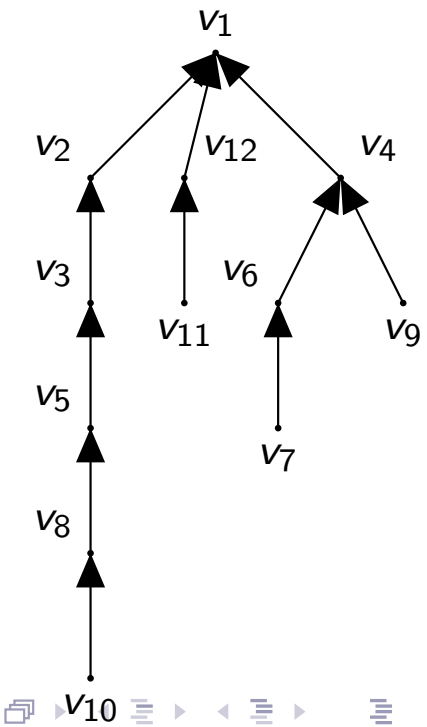
Forward BFS Tree [O(m) time]

For any i, the forward fringe, $F_i^F(u)$ (which nodes are at distance i from u)

Backward BFS Tree [O(m) time]

For any i, the backward fringe, $F_i^B(u)$ (from which nodes, u is at distance i).





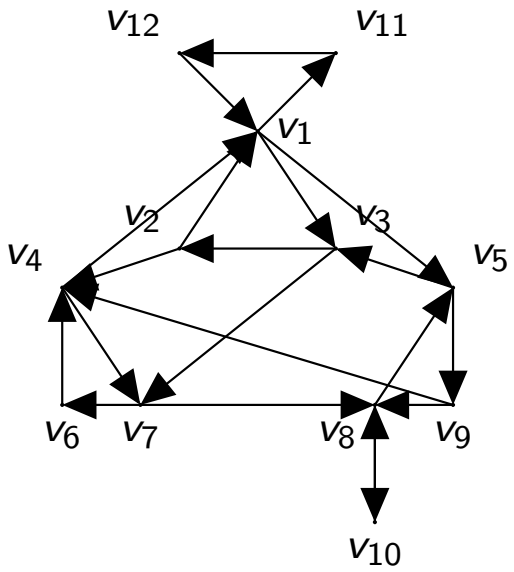
i	$F_i^F(v_1)$	$F_i^B(v_1)$
1	v_3, v_5, v_{11}	v_2, v_4, v_{12}
2	v_2, v_7, v_9, v_{12}	v_3, v_6, v_9, v_{11}
3	v_4, v_6, v_8	v_5, v_7
4	v_{10}	v_8
5		v_{10}

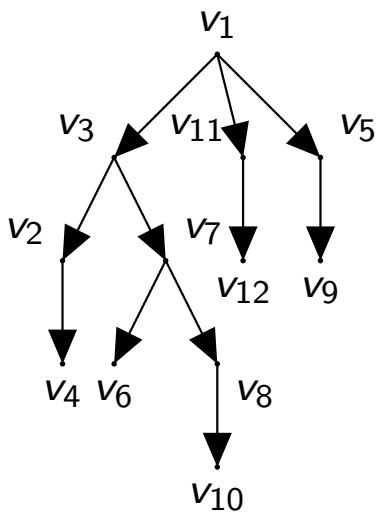
The known approaches to compute the diameter

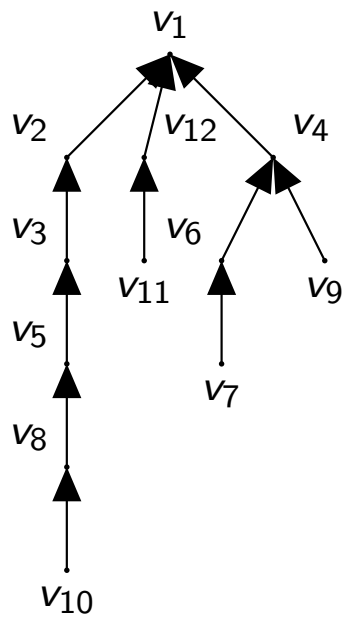
- Textbook Algorithm ($n = |V|$, $m = |E|$). Too expensive.
  - Perform n FBFS and return maximum $\mathrm{ecc}_{\mathrm{F}}$.
    - Each FBFS takes $O(m)$ time.
- Several other approaches (see [Zwick, 2001]) that solves all pairs shortest path. Still too expensive.
  - $O(n^{(3+\omega)/2} \log n)$ where ω is the exponent of the matrix multiplication.
- Empirically finding lower bound L and upper bound U
  - That is, $L \leq D \leq U$
  - D found, when $L = U$

Lower bound and upper bound

By using Single source (FBFS) and Single target (BBFS) Shortest Path







Lower Bound The maximum between the forward, $\mathrm{ecc_F}$, and the backward eccentricity, $\mathrm{ecc_B}$, of a node.
In the example 5: at least a pair is at distance 5.

Upper Bound The forward eccentricity plus the backward eccentricity $\mathrm{ecc_B}$ of a node.
In the example 9: every node can reach another node going to v_1 in ≤ 5 steps and going to the destination in ≤ 4 steps.

$$x \in F_i^B(u) \text{ and } y \in F_j^F(u) \Longrightarrow d(x,y) \leq i+j$$

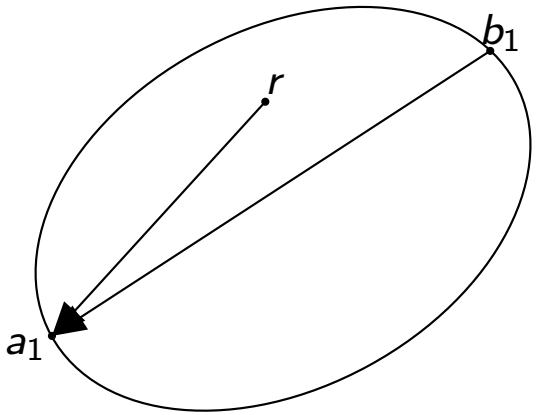
$i+j$ is the length of a path from x to y passing through u.

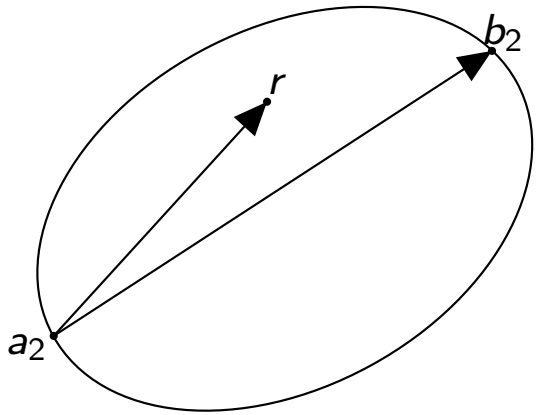
- Very often: $L < D < U$ (see SNAP experiments)
  In the example diameter is 7: $d(v_{10}, v_{12}) = 7$.

A good lower bound: directed double sweep

2-*d*SWEEP

1. Run a forward BFS from a random node r: let a_1 be the farthest node.
2. Run a backward BFS from a_1: let b_1 be the farthest node.
3. Run a backward BFS from r: let a_2 be the farthest node.
4. Run a forward BFS from a_2: let b_2 be the farthest node.
5. If $\mathrm{ecc}_B(a_1) > \mathrm{ecc}_F(a_2)$, then return the length of the path from b_1 to a_1. Otherwise return the length of the path from a_2 to b_2.





Return the maximum between $d(a_2, b_2)$ and $d(b_1, a_1)$.

In the undirected case, $d(a_2, b_2) = d(b_1, a_1)$ and the algorithm is called 2-SWEEP.

First time used in directed graph by Broder et al. to study *Graph structure in the web*.

Lower bound: experiments

(snap.stanford.edu and webgraph.dsi.unimi.it dataset)

Network name	D	Numb. of runs (out of 10) in which $LB = D$	Worst LB found
Wiki-Vote	9	10	9
p2p-Gnutella08	19	9	**18**
p2p-Gnutella09	19	9	**18**
p2p-Gnutella06	19	10	19
p2p-Gnutella05	22	9	**21**
p2p-Gnutella04	25	7	**22**
p2p-Gnutella25	21	8	**20**
p2p-Gnutella24	28	10	28
p2p-Gnutella30	23	2	**22**
p2p-Gnutella31	30	9	**29**
s.s.Slashdot081106	15	10	15
s.s.Slashdot090216	15	10	15
s.s.Slashdot090221	15	10	15
soc-Epinions1	16	9	**15**
Email-EuAll	10	10	10
soc-sign-epinions	16	10	16
web-NotreDame	93	10	93
Slashdot0811	12	10	12
Slashdot0902	13	3	**12**
WikiTalk	10	9	9
web-Stanford	210	10	210
web-BerkStan	679	10	679
web-Google	51	10	51

Network name	D	Numb. of runs (out of 10) in which $LB = D$	Worst LB found
wordassociation-2011	10	9	**9**
enron	10	10	10
uk-2007-05@100000	7	10	7
cnr-2000	81	10	81
uk-2007-05@1000000	40	10	40
in-2004	56	10	56
amazon-2008	47	10	47
eu-2005	82	10	82
indochina-2004	235	10	235
uk-2002	218	10	218
arabic-2005	133	10	133
uk-2005	166	10	166
it-2004	873	10	873

Upper bound: directed iterative fringe upper bound

Recall that

The *textbook* algorithm runs a FBFS for any node and return the maximum $\mathrm{ecc_F}$ found (or a BBFS for any node and return the maximum $\mathrm{ecc_B}$ found).

DiFUB is a particular case in which we:

- specify the order in which the BFSes have to be executed
- refine a lower bound,
  - that is the maximum $\mathrm{ecc_F}$ or $\mathrm{ecc_B}$ found until that moment.
- upper bound the eccentricities of the remaining nodes.
- stop when the remaining nodes cannot have eccentricity higher than our lower bound.

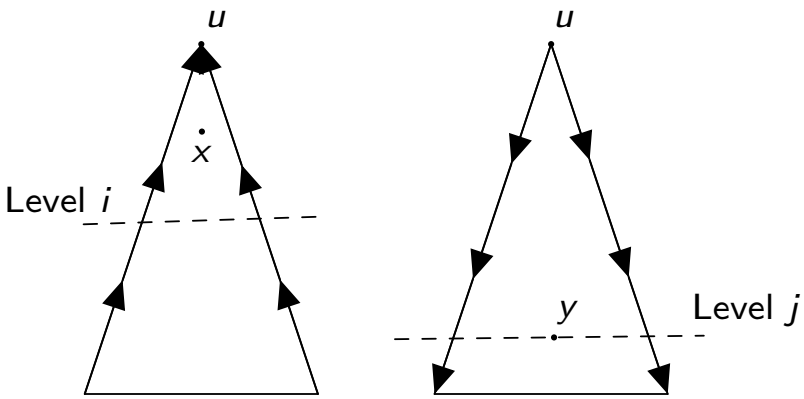
Good order by analyzing how nodes are placed in the FBFS or BBFS tree of a central node u.

- u is the middle node of the path whose length has been returned by 2-dSWEEP

The main observation

Theorem

For any integer i with $1 < i \leq \mathrm{ecc}_B(u)$, for any integer k with $1 \leq k < i$, and for any node $x \in F^B_{i-k}(u)$ such that $\mathrm{ecc}_F(x) > 2(i-1)$, there exists $y \in F^F_j(u)$, for some $j \geq i$, such that $d(x,y) = \mathrm{ecc}_F(x)$.



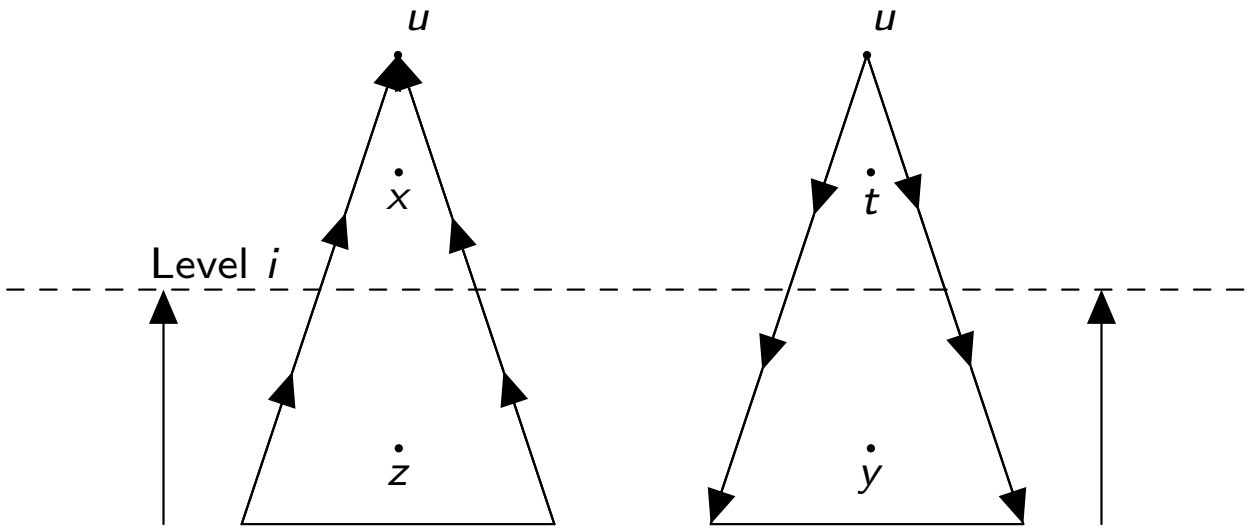
If the forward eccentricity of x is $> 2(i-1)$, the node y, such that $d(x,y) > 2(i-1)$, is below in the FBFS tree of u.

Analogously

Theorem

For any integer i with $1 < i \leq \mathrm{ecc}_F(u)$, for any integer k with $1 \leq k < i$, and for any node $x \in F^F_{i-k}(u)$ such that $\mathrm{ecc}_B(x) > 2(i-1)$, there exists $y \in F^B_j(u)$, for some $j \geq i$, such that $d(y,x) = \mathrm{ecc}_B(x)$.

1. All the nodes x above the level i in $\textsc{bbfs}(u)$ having $\mathrm{ecc}_{\mathrm{F}}$ greater than $2(i-1)$ have a corresponding node y, whose $\mathrm{ecc}_{\mathrm{B}}$ is greater or equal to $\mathrm{ecc}_{\mathrm{F}}(x)$, below or on the level i in $\textsc{fbfs}(u)$.
2. All the nodes t above the level i in $\textsc{fbfs}(u)$ having $\mathrm{ecc}_{\mathrm{B}}$ greater than $2(i-1)$ have a corresponding node z, whose $\mathrm{ecc}_{\mathrm{F}}$ is greater or equal to $\mathrm{ecc}_{\mathrm{B}}(t)$, below or on the level i in $\textsc{bbfs}(u)$.



- At level i we have computed all the $\mathrm{ecc}_{\mathrm{B}}$ of nodes y and the $\mathrm{ecc}_{\mathrm{F}}$ of nodes z. The maximum is our lower bound l_i.
- If l_i is already bigger than $2(i-1)$, l_i is the diameter.
  - No node to be examined can have $\mathrm{ecc}_{\mathrm{F}}$ or $\mathrm{ecc}_{\mathrm{B}}$ bigger than l_i.

$$B_j^F(u) = \begin{cases} \max_{x \in F_j^F(u)} \mathrm{ecc}_B(x) & \text{if } j \leq \mathrm{ecc}_F(u), \\ 0 & \text{otherwise} \end{cases}$$

$$B_j^B(u) = \begin{cases} \max_{x \in F_j^B(u)} \mathrm{ecc}_F(x) & \text{if } j \leq \mathrm{ecc}_B(u), \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 1: D*i*FUB

Input: A strongly connected di-graph G, a node u, a lower bound l for the diameter
Output: The diameter D

$i \leftarrow \max\{\mathrm{ecc}_F(u), ecc_B(u)\}$;
$lb \leftarrow \max\{\mathrm{ecc}_F(u), ecc_B(u), l\}$;
$ub \leftarrow 2i$;
while $ub - lb > 0$ **do**
    if $\max\{lb, B_i^B(u), B_i^F(u)\} > 2(i-1)$ **then**
        return $\max\{lb, B_i^B(u), B_i^F(u)\}$;
    else
        $lb \leftarrow \max\{lb, B_i^B(u), B_i^F(u)\}$;
        $ub \leftarrow 2(i-1)$;
    end
    $i \leftarrow i - 1$;
end
return lb;

Upper bound: experiments (snap.stanford.edu dataset)

Network name	n	m	Avg. Visits	Visits worst run
Wiki-Vote	1300	39456	17	17
p2p-Gnutella08	2068	9313	45.9	64
p2p-Gnutella09	2624	10776	202.1	230
p2p-Gnutella06	3226	13589	236.6	279
p2p-Gnutella05	3234	13453	60.4	94
p2p-Gnutella04	4317	18742	36.7	38
p2p-Gnutella25	5153	17695	85.1	161
p2p-Gnutella24	6352	22928	13	13
p2p-Gnutella30	8490	31706	255.4	516
p2p-Gnutella31	14149	50916	208.7	255
s.s.Slashdot081106	26996	337351	22.3	25
s.s.Slashdot090216	27222	342747	21.5	26
s.s.Slashdot090221	27382	346652	22.8	26
soc-Epinions1	32223	443506	6.1	7
Email-EuAll	34203	151930	6	6
soc-sign-epinions	41441	693737	6	6
web-NotreDame	53968	304685	7	7
Slashdot0811	70355	888662	40	40
Slashdot0902	71307	912381	32.9	40
WikiTalk	111881	1477893	13.6	19
web-Stanford	150532	1576314	6	6
web-BerkStan	334857	4523232	7	7
web-Google	434818	3419124	9.4	10

Upper bound: experiments (webgraph.dsi.unimi.it dataset)

Network name	n	m	Avg. Visits	Visits worst run
wordassociation-2011	4845	61567	412.5	423
enron	8271	147353	19	22
uk-2007-05@100000	53856	1683102	14	14
cnr-2000	112023	1646332	17	17
uk-2007-05@1000000	480913	22057738	6	6
in-2004	593687	7827263	14	14
amazon-2008	627646	4706251	136.3	598
eu-2005	752725	17933415	6	6
indochina-2004	3806327	98815195	8	8
uk-2002	12090163	232137936	6	6
arabic-2005	15177163	473619298	58	58
uk-2005	25711307	704151756	170	170
it-2004	29855421	938694394	87	87

The number of visits seems to be constant.
$\Rightarrow$ For any graph with more than 10000 nodes, DiFUB performs $0.001\%n$ visits in stead of n.

These experiments are just for directed graphs. The effectiveness in the case of the undirected graphs has been already shown: iFUB has been used to compute the diameter of Facebook (721.1M nodes, 68.7G edges, D 41) with just 17 BFSes.

Conclusions and Future Works

DiFUB can be generalized to weighted graphs, using Dijkstra algorithm instead of BFS and sorting the nodes according to their distance from u. It works well in general, but not for Road Networks.

Still to understand:

- why 2-Sweep both in the directed and in the undirected version, is so effective in finding tight lower bounds.
  - Which is the topological underlying property that can lead us to these results? Might be related to some sort of chordality measure? Why real world graphs exhibit this property?
- why the DiFUB method works so well in general.
  - Might be related to eccentricities distribution?
- Clever (external memory) and parallel implementation of BFS.

Work in progress: computing the radius (minimum eccentricity).

Thanks

Code and networks are available at `amici.dsi.unifi.it/lasagne/`