Diameter computations

Michel Habib
habib@irif.fr
http://www.irif.fr/~habib

7 novembre 2016

Schedule of this course

Diameter computations

Schedule of this course

Diameter computations

Computing diameter using fewest BFS possible

Schedule of this course

Schedule of this course

Schedule of this course

Schedule of this course

Joint work with :
D. Corneil (Toronto), C. Paul (Montpellier), F. Dragan (Kent), V. Chepoi (Marseille), B. Estrellon (Marseille), Y. Vaxes (Marseille), Y. Xiang (Kent), C. Magnien (Paris), M. Latapy (Paris), P. Crescenzi (Firenze), R. Grossi (Pisa), A. Marino (Pisa), J. Dusart (Paris), R. Charpey (Paris), M. Borassi (Firence)
and discussion with many others . . .

Basics Definitions

Definitions :

Let G be an undirected graph :

- $exc(x) = max_{y \in G}\{distance(x, y)\}$ excentricity
- $diam(G) = max_{x \in G}\{exc(x)\}$ diameter
- $radius(G) = min_{x \in G}\{exc(x)\}$
- $x \in V$ is a center of G, if $exc(x) = radius(G)$

Basics Definitions

Definitions :

Let G be an undirected graph :

- $exc(x) = max_{y \in G}\{distance(x, y)\}$ excentricity
- $diam(G) = max_{x \in G}\{exc(x)\}$ diameter
- $radius(G) = min_{x \in G}\{exc(x)\}$
- $x \in V$ is a center of G, if $exc(x) = radius(G)$

First remarks of the definitions

distance computed in $\#$ edges
If x and y belong to different connected components $d(x, y) = \infty$.
diameter : Max Max Min
radius : Min Max Min

Trivial bounds

For any graph G :
$radius(G) \leq diam(G) \leq 2radius(G)$ and $\forall e \in G$,
$diam(G) \leq diam(G - e)$

Trivial bounds

For any graph G :
$radius(G) \leq diam(G) \leq 2radius(G)$ and $\forall e \in G$,
$diam(G) \leq diam(G - e)$

These bounds are tight

Trivial bounds

For any graph G :
$radius(G) \leq diam(G) \leq 2radius(G)$ and $\forall e \in G$,
$diam(G) \leq diam(G - e)$

These bounds are tight

- If G is a path of length 2K, then $diam(G) = 2k = 2radius(G)$, and G admits a unique center, i.e. the middle of the path.

Trivial bounds

For any graph G :
$radius(G) \leq diam(G) \leq 2radius(G)$ and $\forall e \in G$,
$diam(G) \leq diam(G - e)$

These bounds are tight

- If G is a path of length 2K, then $diam(G) = 2k = 2radius(G)$, and G admits a unique center, i.e. the middle of the path.
- If $radius(G) = diam(G)$, then $Center(G) = V$. All vertices are centers (as for example in a cycle).

If $2.radius(G) = diam(G)$, then *roughly* G has a tree shape (at least it works for trees).
But there is no nice characterization of this class of graphs.

Diameter

Applications

1. A graph parameter which measures the quality of services of a network, in terms of worst cases, when all have a unitary cost. Find critical edges e s.t. $diam(G-e) > diam(G)$

Diameter

Applications

1. A graph parameter which measures the quality of services of a network, in terms of worst cases, when all have a unitary cost. Find critical edges e s.t. $diam(G-e) > diam(G)$
2. Many distributed algorithms can be analyzed with this parameter (when a flooding technique is used to spread information over the network or to construct routing tables).

Diameter

Applications

1. A graph parameter which measures the quality of services of a network, in terms of worst cases, when all have a unitary cost. Find critical edges e s.t. $diam(G - e) > diam(G)$
2. Many distributed algorithms can be analyzed with this parameter (when a flooding technique is used to spread information over the network or to construct routing tables).
3. Verify the small world hypothesis in some large social networks, using J. Kleinberg's definition of small world graphs.

Diameter

Applications

1. A graph parameter which measures the quality of services of a network, in terms of worst cases, when all have a unitary cost. Find critical edges e s.t. $diam(G - e) > diam(G)$
2. Many distributed algorithms can be analyzed with this parameter (when a flooding technique is used to spread information over the network or to construct routing tables).
3. Verify the small world hypothesis in some large social networks, using J. Kleinberg's definition of small world graphs.
4. Compute the diameter of the Internet graph, or some Web graphs, i.e. massive data.

1. Examples of diameter searches based on the algorithms presented in this course :
`http://gang.inria.fr/road/`

1. Examples of diameter searches based on the algorithms presented in this course :
`http://gang.inria.fr/road/`
2. OpenStreetMap (OSM) : 80 millions of nodes, average degree 3

1. Examples of diameter searches based on the algorithms presented in this course : `http://gang.inria.fr/road/`
2. OpenStreetMap (OSM) : 80 millions of nodes, average degree 3
3. Roadmaps graphs a special domain of research interest Quasi-planar graph (bridges on the roads)

1. Examples of diameter searches based on the algorithms presented in this course :
   `http://gang.inria.fr/road/`
2. OpenStreetMap (OSM) : 80 millions of nodes, average degree 3
3. Roadmaps graphs a special domain of research interest
   Quasi-planar graph (bridges on the roads)
4. Never forget that computer science has an important experimental part.

1. Examples of diameter searches based on the algorithms presented in this course : `http://gang.inria.fr/road/`
2. OpenStreetMap (OSM) : 80 millions of nodes, average degree 3
3. Roadmaps graphs a special domain of research interest Quasi-planar graph (bridges on the roads)
4. Never forget that computer science has an important experimental part.
5. Many algorithmic ideas come out some experiment.

Frequently Asked Questions (FAQ)

Usual questions on diameter, centers and radius :

- What is the best Program (resp. algorithm) available ?

Frequently Asked Questions (FAQ)

Usual questions on diameter, centers and radius :

- What is the best Program (resp. algorithm) available ?
- What is the complexity of diameter, center and radius computations ?

Frequently Asked Questions (FAQ)

Usual questions on diameter, centers and radius :

- What is the best Program (resp. algorithm) available ?
- What is the complexity of diameter, center and radius computations ?
- How to compute or approximate the diameter of huge graphs ?

Frequently Asked Questions (FAQ)

Usual questions on diameter, centers and radius :

- What is the best Program (resp. algorithm) available ?
- What is the complexity of diameter, center and radius computations ?
- How to compute or approximate the diameter of huge graphs ?
- Find a center (or all centers) in a network, (in order to install serveurs).

Some notes

1. I was asked first this problem in 1980 by France Telecom for the phone network (FT granted a PhD).

Some notes

1. I was asked first this problem in 1980 by France Telecom for the phone network (FT granted a PhD).
2. Marc Lesk obtained his PhD in 1984 with the title :
   Couplages maximaux et diamètres de graphes.
   Maximum matchings and diameter computations

Some notes

1. I was asked first this problem in 1980 by France Telecom for the phone network (FT granted a PhD).
2. Marc Lesk obtained his PhD in 1984 with the title :
   Couplages maximaux et diamètres de graphes.
   Maximum matchings and diameter computations
3. But, with very little practical results for diameter computations.

- Our aim is to design an algorithm or heuristic to compute the diameter of very large graphs

- Our aim is to design an algorithm or heuristic to compute the diameter of very large graphs
- Any algorithm that computes all distances between all pairs of vertices, complexity $O(n^3)$ or $O(nm)$. As for example with $|V|$ successive Breadth First Searches in $O(n(n+m))$.

- Our aim is to design an algorithm or heuristic to compute the diameter of very large graphs
- Any algorithm that computes all distances between all pairs of vertices, complexity $O(n^3)$ or $O(nm)$. As for example with $|V|$ successive Breadth First Searches in $O(n(n+m))$.
- Best known complexity for an exact algorithm is $O(\frac{n^3}{log^3 n})$, in fact computing all shortest paths.

- Our aim is to design an algorithm or heuristic to compute the diameter of very large graphs
- Any algorithm that computes all distances between all pairs of vertices, complexity $O(n^3)$ or $O(nm)$. As for example with $|V|$ successive Breadth First Searches in $O(n(n+m))$.
- Best known complexity for an exact algorithm is $O(\frac{n^3}{log^3 n})$, in fact computing all shortest paths.
- But also with at most $O(Diam(G))$ matrix multiplications.

- Clemence Magnien and M. Latapy asked me again (2006) this question about diameter.

- Clemence Magnien and M. Latapy asked me again (2006) this question about diameter.
- But in the meantime, I met Derek Corneil and Feodor Dragan, we proved some theorems about diameter and chordals graphs but above all I had learned many properties of graph searches from Derek Corneil.

- Clemence Magnien and M. Latapy asked me again (2006) this question about diameter.
- But in the meantime, I met Derek Corneil and Feodor Dragan, we proved some theorems about diameter and chordals graphs but above all I had learned many properties of graph searches from Derek Corneil.
- I answered to Olivier Gascuel's usual question, how to compute diameter of phylogenetic trees, using the following algorithm.

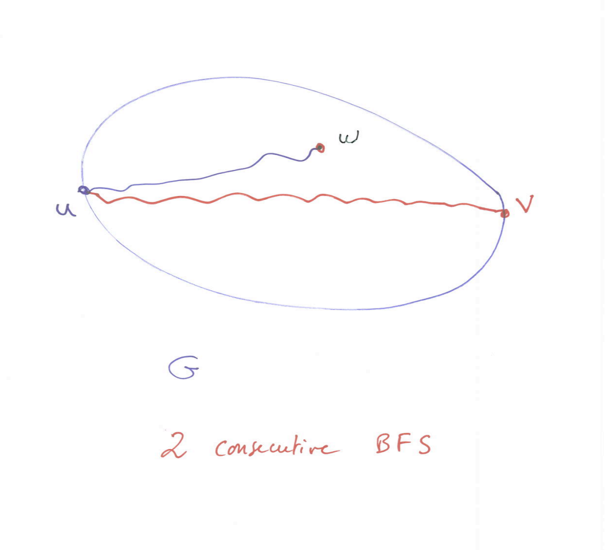
1. Let us consider the procedure called : 2 consecutive BFS [1]

Data: A graph $G = (V, E)$
Result: u, v two vertices
Choose a vertex $w \in V$
$u \leftarrow BFS(w)$
$v \leftarrow BFS(u)$

Where BFS stands for Breadth First Search.
Therefore it is a linear procedure

1. Proposed the first time by Handler 1973

Intuition behind the procedure



- Handler's classical result 73
  If G is a tree, $diam(G) = d(u, v)$
  Easy using Jordan's theorem.

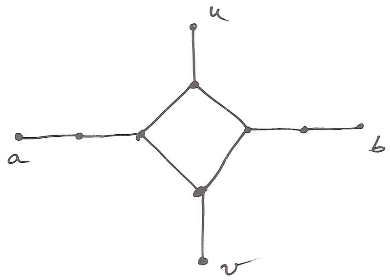
- Boris Aronov, Prosenjit Bose, Erik D. Demaine, Joachim Gudmundsson, John Iacono, Stefan Langerman, and Michiel Smid, *Data structures for halfplane proximity queries and incremental Voronoi diagrams*, LATIN 2006 : Theoretical informatics, Lecture Notes in Comput. Sci., vol. 3887, Springer, Berlin, 2006, pp. 80–92.
- Stephen Alstrup, Thore Husfeldt, and Theis Rauhe, *Marked ancestor problems*, IEEE Symposium on Foundations of Computer Science, 1998, pp. 534–544.
- Camille Jordan, *Sur les assemblages de lignes*, Journal für reine und angewandte Mathematik **70** (1869), 185–190.

First theorem

Camille Jordan 1869 :

A tree admits one or two centers depending on the parity of its diameter and furthermore all chains of maximum length starting at any vertex contain this (resp. these) centers.
And $radius(G) = \lceil \frac{diam(G)}{2} \rceil$

Unfortunately it is not an algorithm !



Certificates for the diameter

To give a certificate $diam(G) = k$, it is enough to provide :

- two vertices x, y s.t. $d(x, y) = k$ ($diam(G) \geq k$).

Certificates for the diameter

To give a certificate $diam(G) = k$, it is enough to provide :

- two vertices x, y s.t. $d(x, y) = k$ ($diam(G) \geq k$).
- a subgraph $H \subset G$ with $diam(H) = k$ ($diam(G) \leq k$).
  H may belong to a class of graphs on which diameter computations can be done in linear time, for example trees.

Experimental results : M.H., M.Latapy, C. Magnien 2008

Randomized BFS procedure

Data: A graph $G = (V, E)$
Result: u, v two vertices
Repeat α times :
Randomly Choose a vertex $w \in V$
$u \leftarrow BFS(w)$
$v \leftarrow BFS(u)$
Select the vertices u_0, v_0 s.t. $distance(u_0, v_0)$ is maximal.

1. This procedure gives a vertex u_0 such that :
   $exc(u_0) \leq diam(G)$ i.e. a lower bound of the diameter.

1. This procedure gives a vertex u_0 such that : $exc(u_0) \leq diam(G)$ i.e. a lower bound of the diameter.
2. Use a spanning tree as a subgraph on the same vertex set to obtain an upper bound by computing its exact diameter in linear time (using the trivial bound $diam(G) \leq diam(G - e)$).

1. This procedure gives a vertex u_0 such that : $exc(u_0) \leq diam(G)$ i.e. a lower bound of the diameter.
2. Use a spanning tree as a subgraph on the same vertex set to obtain an upper bound by computing its exact diameter in linear time (using the trivial bound $diam(G) \leq diam(G - e)$).
3. Spanning trees given by the BFS.

- The Program and some Data on Web graphs or P-2-P networks can be found

- The Program and some Data on Web graphs or P-2-P networks can be found
- `http://www-rp.lip6.fr/~magnien/Diameter`

- The Program and some Data on Web graphs or P-2-P networks can be found
- `http://www-rp.lip6.fr/~magnien/Diameter`
- 2 millions of vertices, diameter 32 within 1

- The Program and some Data on Web graphs or P-2-P networks can be found
- `http://www-rp.lip6.fr/~magnien/Diameter`
- 2 millions of vertices, diameter 32 within 1
- Further experimentations by Crescenzi, Grossi, Marino (in ESA 2010)
  which confirm the excellence of the lower bound using BFS ! ! ! !

- Since α is a constant (≤ 1000), this method is still in linear time and works extremely well on huge graphs (Web graphs, Internet ...)

- Since α is a constant (≤ 1000), this method is still in linear time and works extremely well on huge graphs (Web graphs, Internet ...)
- How can we explain the success of such a method ?

- Since α is a constant (≤ 1000), this method is still in linear time and works extremely well on huge graphs (Web graphs, Internet . . .)
- How can we explain the success of such a method ?
- Due to the many counterexamples for the 2 consecutive BFS procedure. An explanation is necessary !

2 kind of explanations

The method is good or the data used was good.

2 kind of explanations

The method is good or the data used was good.

Partial answer

The method also works on several models of random graphs.
So let us try to prove the first fact

2 kind of explanations

The method is good or the data used was good.

Partial answer

The method also works on several models of random graphs.
So let us try to prove the first fact

Restriction

First we are going to focus our study on the 2 consecutive BFS.

Chordal graphs

1. A graph is chordal if it has no chordless cycle of length ≥ 4 .

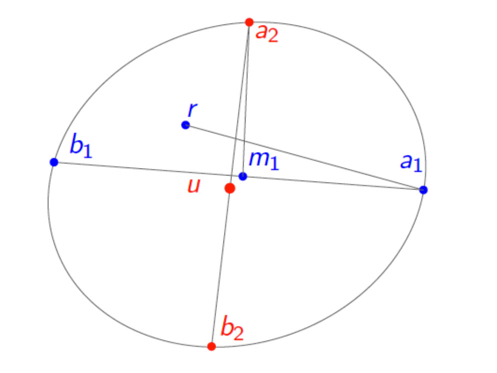
Chordal graphs

1. A graph is chordal if it has no chordless cycle of length ≥ 4 .
2. If G is a chordal graph, Corneil, Dragan, H., Paul 2001, using a variant called 2 consecutive LexBFS
   $d(u, v) \leq diam(G) \leq d(u, v) + 1$

Chordal graphs

1. A graph is chordal if it has no chordless cycle of length ≥ 4 .
2. If G is a chordal graph, Corneil, Dragan, H., Paul 2001, using a variant called 2 consecutive LexBFS
   $d(u,v) \leq diam(G) \leq d(u,v) + 1$
3. Generalized by Corneil, Dragan, Kohler 2003 using 2 consecutive BFS :
   $d(u,v) \leq diam(G) \leq d(u,v) + 1$

The 4-sweep : Crescenzi, Grossi, MH, Lanzi, Marino 2011



$$Diam = max\{ecc(a_1), ecc(a_2)\} \text{ and } Rad = min\{ecc(r), ecc(m_1)\}$$

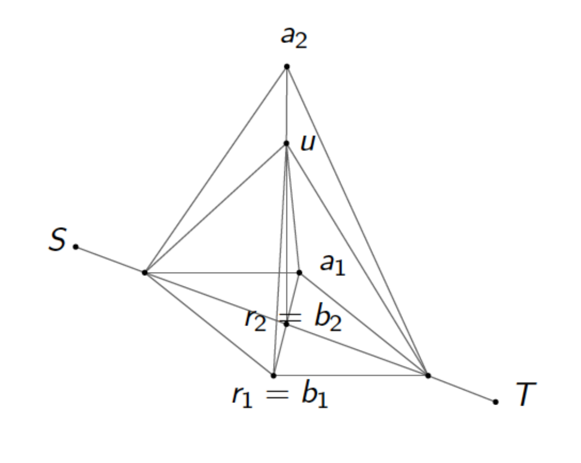
Intuition behind the 4-sweep heuristics

- Chepoi and Dragan has proved that for chordal graphs that a center is at distance at most one of the middle vertex (m_1 in the picture).

Intuition behind the 4-sweep heuristics

- Chepoi and Dragan has proved that for chordal graphs that a center is at distance at most one of the middle vertex (m_1 in the picture).
- Roughly, we have the same results with 4-sweep than with 1000 2-sweep.

It is still not al algorithm ! !



An exact algorithm !

Compute the excentricity of the leaves of a BFS rooted in m_1 with a stop condition.
Complexity is $O(nm)$ in the worst case, but often linear in practice.

Simple Lemma

If for some $x \in Level(i)$ of the tree, we have $ecc(x) > 2(i-1)$ then we can stop the exploration.

Simple Lemma

If for some $x \in Level(i)$ of the tree, we have $ecc(x) > 2(i-1)$ then we can stop the exploration.

Proof

Let us consider $y \in L(j)$ with $j < i$. $\forall z \in \cup_{1 \leq k \leq i-1} L(k)$
$dist(z, y) \leq 2(i-1)$
Therefore $ecc(y) \leq ecc(x)$ or the extreme vertices from y belong to lower layers and have already been considered.

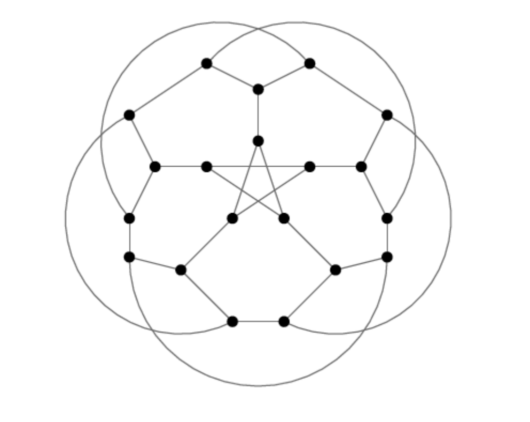
iFub an exact $O(mn)$ algorithm

Algorithm 1: iFUB (iterative Fringe Upper Bound)

Input: G, u, lower bound l
Output: A value M such that $D - M \leq k$.

$i \leftarrow \mathrm{ecc}(u)$; $lb \leftarrow \max\{\mathrm{ecc}(u), l\}$; $ub \leftarrow 2\mathrm{ecc}(u)$;
while $ub \neq lb$ **do**
    if $\max\{B_1(u), \ldots, B_i(u)\} > 2(i-1)$ **then**
        return $\max\{B_1(u), \ldots, B_i(u)\}$;
    else
        $lb \leftarrow \max\{B_1(u), \ldots, B_i(u)\}$;
        $ub \leftarrow 2(i-1)$;
    end
    $i \leftarrow i - 1$;
end
return lb;

Bad example



Results :

v	# of graphs in which v BFSes done on the average					
		Number n of vertices				
	Total	$\leq 10^3$	$\leq 10^4$	$\leq 10^5$	$\leq 10^6$	$> 10^6$
$v = 5$	29	2	8	9	10	0
$5 < v \leq 100$	123	17	44	43	11	8
$100 < v \leq 1000$	21	1	3	10	4	3
$1000 < v \leq 10^4$	18	0	4	12	1	1
$10^4 < v \leq 10^5$	8	0	0	3	3	2

- The 200th graph: Facebook network
  - 721.1M nodes and 68.7G edges
  - After 17 BFSes...

Diametre Facebook = 41 !, Average distance 4.74, Backstrom, Boldi, Rosa, Uganden, Vigna 2011

Comments

- Boldi and his group had to parallelize our algorithm and a BFS on the giant connected component of Facebook would take several hours. But only 17 BFS's were needed.

Comments

- Boldi and his group had to parallelize our algorithm and a BFS on the giant connected component of Facebook would take several hours. But only 17 BFS's were needed.
- The 4-sweep method alway gives a lower bound of the diameter not too far from the optimal,
  the hard part is to obtain an upper bound with iFUB

Comments

- Boldi and his group had to parallelize our algorithm and a BFS on the giant connected component of Facebook would take several hours. But only 17 BFS's were needed.
- The 4-sweep method alway gives a lower bound of the diameter not too far from the optimal,
  the hard part is to obtain an upper bound with iFUB
- The worst examples are roadmap graphs with big treewidth and big grids.

Stanford Large Network Dataset Collection
http ://snap.stanford.edu/data/

- A very practical database for having large graphs to play with.

Stanford Large Network Dataset Collection
http ://snap.stanford.edu/data/

- A very practical database for having large graphs to play with.
- Graphs are described that way : number of vertices, number of edges (arcs), diameter.

Graph	diam SNAP	diam 4-Sweep
soc-Epinions1	14	15
soc-pokec-relationships	11	14
soc-Slashdot0811	10	12
soc-Slashdot0902	11	13
com-lj.ungraph	17	21
com-youtube.ungraph	20	24
com-DBLP	21	23
com-amazon	44	47
email-Enron	11	13
wikiTalk	9	11
cit-HepPh	12	14
cit-HepTh	13	15
CA-CondMat	14	15
CA-HepTh	17	18
web-Google	21	24

Graph	diam SNAP	diam 4-Sweep
amazon0302	32	38
amazon0312	18	20
amazon0505	20	22
amazon0601	21	25
p2p-Gnutella04	9	10
p2p-Gnutella24	10	11
p2p-Gnutella25	10	11
p2p-Gnutella30	10	11
roadNet-CA	849	865
roadNet-TX	1054	1064
Gowalla-edges	14	16
BrightKite-edges	16	18

How can I certify my results ?

- How can I beat the value of Stanford database ?

How can I certify my results ?

- How can I beat the value of Stanford database ?
- Then some * explains in a little footnote that the SNAP value is heuristically obtained by 1000 random BFS

How can I certify my results ?

- How can I beat the value of Stanford database ?
- Then some * explains in a little footnote that the SNAP value is heuristically obtained by 1000 random BFS
- I like the idea that 4 searches totally dependant are better that 1000 independant searches

How can I certify my results ?

- How can I beat the value of Stanford database ?
- Then some * explains in a little footnote that the SNAP value is heuristically obtained by 1000 random BFS
- I like the idea that 4 searches totally dependant are better that 1000 independant searches
- See the example of a long path.

How can I certify my results ?

- How can I beat the value of Stanford database ?
- Then some * explains in a little footnote that the SNAP value is heuristically obtained by 1000 random BFS
- I like the idea that 4 searches totally dependant are better that 1000 independant searches
- See the example of a long path.
- The last vertex of a BFS is not at all a random vertex (NP-complete to decide : Charbit, MH, Mamcarz 2014 to appear in DMTCS).

How can I certify my results ?

- By certifying the longest path $[x, y]$ (as hard as computing a BFS ?)

How can I certify my results ?

- By certifying the longest path $[x, y]$ (as hard as computing a BFS ?)
- Using another BFS programmed by others starting at x.

How can I certify my results ?

- By certifying the longest path $[x, y]$ (as hard as computing a BFS ?)
- Using another BFS programmed by others starting at x.
- Certifying that the computed BFS ordering is a legal BFS ordering, using the 4-point condition. Which can be checked in linear time for BFS and DFS.

Graph Name	$\frac{Vertices}{Edges}$	Diameter iFUB	Diam. FourSweep
CA-HepTh	0.190	18	18
CA-GrQc	0.181	17	17
CA-CondMat	0.124	15	15
CA-AstroPh	0.047	14	14
roadNet-CA	0.355	865	865
roadNet-PA	0.353	794	780
roadNet-TX	0.359	1064	1064
email-Enron	0.1	13	13
email-EuAll	0.631	14	14
com-amazon	0.361	47	47
Amazon0302	0.212	38	38
Amazon0312	0.125	20	20
Amazon0505	0.122	22	22
Amazon0601	0.119	25	25
Gowalla_edges	0.207	25	16
Brightkite_edges	0.272	18	18
soc-Epinions1	0.149	15	15

FIGURE: 4-Sweep Results

Easy extensions

1. To weighted graphs by replacing BFS with Dijkstra's algorithm

Easy extensions

1. To weighted graphs by replacing BFS with Dijkstra's algorithm
2. To directed graphs

A method symmetric for computing radius and diameter

M. Borassi, P. Crescenzi, R. Grossi, M.H., W. Kosters, A. Marino and F. Takes, 2014

- A mixture with our approach and that of W. Kosters and F. Takes in which a lower bound of the eccentricity of every vertex is maintained at each BFS.

A method symmetric for computing radius and diameter

M. Borassi, P. Crescenzi, R. Grossi, M.H., W. Kosters, A. Marino and F. Takes, 2014

- A mixture with our approach and that of W. Kosters and F. Takes in which a lower bound of the eccentricity of every vertex is maintained at each BFS.
- It generalizes the 4-sweep to k-sweep.

A method symmetric for computing radius and diameter

M. Borassi, P. Crescenzi, R. Grossi, M.H., W. Kosters, A. Marino and F. Takes, 2014

- A mixture with our approach and that of W. Kosters and F. Takes in which a lower bound of the eccentricity of every vertex is maintained at each BFS.
- It generalizes the 4-sweep to k-sweep.
- we generalize to maintain k values in each vertex.

A method with no name yet

- Given a random vertex v_1 and setting $i = 1$, repeat k times the following :
  1. Perform a BFS from v_i and choose the vertex v_{i+1} as the vertex x maximizing $\sum_{j=1}^{i} d(v_j, x)$.
  2. Increment i.
- The maximum eccentricity found, i.e. $\max_{i=1,\ldots,k} exc(v_i)$, is a lower bound for the diameter.
- Compute the eccentricity of w, the vertex minimizing $\sum_{i=1}^{k} d(w, v_i)$.
- The minimum eccentricity found, i.e. $\min\{\min_{i=1,\ldots,k} exc(v_i), exc(w)\}$, is an upper bound for the radius.

Halting conditions

To compute the exact values of radius and diameter, we use the next lemmas.

Lemma 1

Let $Diam(G)$ be the diameter, let x and y be diametral vertices (that is, $d(x,y) = Diam(G)$), and let $v_1, \ldots, v_k$ be k other vertices. Then, $Diam(G) \leq \frac{2}{k} \sum_{i=1}^{k} d(x, v_i)$ or $Diam(G) \leq \frac{2}{k} \sum_{i=1}^{k} d(v_i, y)$.

Halting conditions

To compute the exact values of radius and diameter, we use the next lemmas.

Lemma 1

Let $Diam(G)$ be the diameter, let x and y be diametral vertices (that is, $d(x, y) = Diam(G)$), and let $v_1, \dots, v_k$ be k other vertices. Then, $Diam(G) \leq \frac{2}{k} \sum_{i=1}^{k} d(x, v_i)$ or $Diam(G) \leq \frac{2}{k} \sum_{i=1}^{k} d(v_i, y)$.

proof

$kDiam(G) = \sum_{i=1}^{k} d(x, y) \geq \sum_{i=1}^{k} [d(x, v_i) + d(v_i, y)] = \sum_{i=1}^{k} d(x, v_i) + \sum_{i=1}^{k} d(v_i, y)$. □

Lemma 2

Let $x \in V$ be a center and let $v_1, \ldots, v_k$ be k other vertices. Then $Radius(G) \geq 1/k \sum_{i=1}^{k} d(x, v_i)$

Lemma 2

Let $x \in V$ be a center and let $v_1, \dots, v_k$ be k other vertices. Then $Radius(G) \geq 1/k \sum_{i=1}^{k} d(x, v_i)$

proof

Let $y \in V$ such that : $Radius(G) = d(x, y)$
Then $kRadius(G) = \sum_{i=1}^{k} d(x, y) \geq \sum_{i=1}^{k} [d(x, v_i) + d(v_i, y)] = \sum_{i=1}^{k} d(x, v_i) + \sum_{i=1}^{k} d(v_i, y)$. □

- If during the algorithm we maintain two variables Macsofar and Minsofar (being respectively the maximum and the minimum computed eccentricity)

- If during the algorithm we maintain two variables Macsofar and Minsofar (being respectively the maximum and the minimum computed eccentricity)
- We only compute eccentricity of vertices x such that $Maxsofar \leq \frac{2}{k} \sum_{i=1}^{k} d(x, v_i)$

- If during the algorithm we maintain two variables Macsofar and Minsofar (being respectively the maximum and the minimum computed eccentricity)
- We only compute eccentricity of vertices x such that $Maxsofar \leq \frac{2}{k} \sum_{i=1}^{k} d(x, v_i)$
- To find centers we only compute eccentricity of vertices x such that : $1/k \sum_{i=1}^{k} d(x, y) \leq Minsofar$

- This method generalizes the 4-sweep and seems to better handle the cases where 1000 BFS was needed to find the exact value in the previous method.

- This method generalizes the 4-sweep and seems to better handle the cases where 1000 BFS was needed to find the exact value in the previous method.
- For the same examples it never goes further 10-100 BFS.

- This method generalizes the 4-sweep and seems to better handle the cases where 1000 BFS was needed to find the exact value in the previous method.
- For the same examples it never goes further 10-100 BFS.
- Strangely replacing Sum by Max as suggested by some experts does not change the behavior of the algorithm.

Real Applications

With this method we were able to disprove conjectures inspired from S. Milgram about the 6 degrees of separation

1. Kevin Bacon games on the actors graph

Real Applications

With this method we were able to disprove conjectures inspired from S. Milgram about the 6 degrees of separation

1. Kevin Bacon games on the actors graph
2. Diameter of Wikipedia (the Wiki Game)

Kevin Bacon

His name was used for a popular TV game in US, The Six Degrees of Kevin Bacon, in which the goal is to connect an actor to Kevin Bacon in less than 6 edges.

Actors graph 2014

- The 2014 graph has 1.797.446 in the biggest connected component, a few more if we consider the whole graph. The number of undirected edges in the biggest connected component is 72.880.156.

Actors graph 2014

- The 2014 graph has 1.797.446 in the biggest connected component, a few more if we consider the whole graph. The number of undirected edges in the biggest connected component is 72.880.156.
- An actor with Bacon number 8 is Shemise Evans, and the path can be found at http ://oracleofbacon.org/ by writing Shemise Evans in the box. Even if their graph does not coincide exactly with our graph, this is a shortest path in both of them :

Shemise Evans → Casual Friday (2008) → Deniz Buga
Deniz Buga → Walking While Sleeping (2009)→ Onur Karaoglu
Onur Karaoglu→ Kardesler (2004)→ Fatih Genckal
Fatih Genckal → Hasat (2012) → Mehmet Ünal
Mehmet Ünal→ Kayip özgürlük (2011)→ Aydin Orak
Aydin Orak → The Blue Man (2014)→Alex Dawe
Alex Dawe→ Taken 2 (2012)→ Rade Serbedzija
Rade Serbedzija→ X-Men : First Class (2011) → Kevin Bacon

Graphe de Twitter 2011

Graphe orienté de 500 millions de sommets
2,5 Milliard d'arêtes
Diamètre 150 de la comp. fortement connexe géante, calculé fin 2015.

Radius versus diameter

- Let D, R be respectively two potential values for $diam(G)$ and $radius(G)$.

Radius versus diameter

- Let D, R be respectively two potential values for $diam(G)$ and $radius(G)$.
- To certify these values we need to prove :

Radius versus diameter

- Let D, R be respectively two potential values for $diam(G)$ and $radius(G)$.
- To certify these values we need to prove :
- $\forall x \in V(G)$, $\forall y \in V(G)$, we have $d(x, y) \leq D$.

Radius versus diameter

- Let D, R be respectively two potential values for $diam(G)$ and $radius(G)$.
- To certify these values we need to prove :
- $\forall x \in V(G)$, $\forall y \in V(G)$, we have $d(x, y) \leq D$.
- $\forall x \in V(G)$, $\exists y \in V(G)$ such that $d(x, y) \geq R$.

Radius versus diameter

- Let D, R be respectively two potential values for $diam(G)$ and $radius(G)$.
- To certify these values we need to prove :
- $\forall x \in V(G), \forall y \in V(G)$, we have $d(x, y) \leq D$.
- $\forall x \in V(G), \exists y \in V(G)$ such that $d(x, y) \geq R$.
- Not exactly the same quantifiers !

Relationships between diameter and δ-hyperbolicity

δ-Hyperbolic metric spaces have been defined by M. Gromov in 1987 via a simple 4-point condition :
for any four points u, v, w, x, the two larger of the distance sums $d(u,v) + d(w,x), d(u,w) + d(v,x), d(u,x) + d(v,w)$ differ by at most 2δ.

Theorem Chepoi, Dragan, Estellon, M.H., Vaxes 2008

If u is the last vertex of a 2-sweep then :
$exc(u) \geq diam(G)$-2.$\delta(G)$ and
$radius(G) \leq \lceil (d(u,v)+1)/2 \rceil + 3\delta(G)$
Furthermore the set of all centers $C(G)$ of G is contained in the ball of radius $5\delta(G)+1$ centered at a middle vertex m of any shortest path connecting u and v in G.

Theorem Chepoi, Dragan, Estellon, M.H., Vaxes 2008

If u is the last vertex of a 2-sweep then :
$exc(u) \geq diam(G)$-$2.\delta(G)$ and
$radius(G) \leq \lceil (d(u,v)+1)/2 \rceil + 3\delta(G)$
Furthermore the set of all centers $C(G)$ of G is contained in the ball of radius $5\delta(G)+1$ centered at a middle vertex m of any shortest path connecting u and v in G.

Consequences

The 2-sweep (resp 4-sweep) method failure is bounded by the δ-hyperbolicity of the graph.

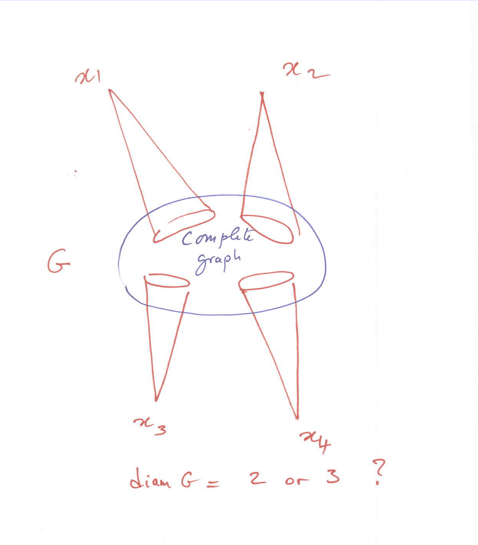
Nice

Because many real networks have small δ-hyperbolicity.

The difficulty of the certificate

δ-hyperbolicity and treewidth (existence of big grids as subgraphs) must play a role.

Chordal graphs and split graphs



Disjoint sets problem

Disjoint sets problem

A finite set X, $\mathcal{F}$ a collection $\{S_1, \ldots, S_k\}$ of subsets of X.
$\exists i, j \in [1, k]$ such that $S_i \cap S_j = \emptyset$?

Disjoint sets problem

Disjoint sets problem

A finite set X, $\mathcal{F}$ a collection $\{S_1, \ldots, S_k\}$ of subsets of X.
$\exists i, j \in [1, k]$ such that $S_i \cap S_j = \emptyset$?

Linearity

Can this problem be solved in linear time ?
Size of the problem : $|X| + k + \sum_{i=1}^{i=k} |S_i|$
size of the incidence bipartite graph

SETH : Strong Exponential Time Hypothesis

SETH

There is no algorithm for solving the k-SAT problem with n variables in $O((2-\epsilon)^n)$ where ϵ does not depend on k.

Let us consider an instance I of $k-SAT$ with 2n boolean variables $x_1, \ldots, x_{2n}$, and a set $\mathcal{C}$ of m clauses $C_1, \ldots C_m$, we build an instance of Disjoint-set problem as follows :

- The gound set X is the set of clauses + 2 extras vertices a, b.

Let us consider an instance I of $k - SAT$ with 2n boolean variables $x_1, \ldots, x_{2n}$, and a set $\mathcal{C}$ of m clauses $C_1, \ldots C_m$, we build an instance of Disjoint-set problem as follows :

- The gound set X is the set of clauses + 2 extras vertices a, b.
- We consider now A, B the sets of all truth assignments of $x_1, \ldots, x_n$, and $x_{n+1}, \ldots x_{2n}$, respectively.

Let us consider an instance I of $k-SAT$ with 2n boolean variables $x_1, \ldots, x_{2n}$, and a set $\mathcal{C}$ of m clauses $C_1, \ldots C_m$, we build an instance of Disjoint-set problem as follows :

- The gound set X is the set of clauses + 2 extras vertices a, b.
- We consider now A, B the sets of all truth assignments of $x_1, \ldots, x_n$, and $x_{n+1}, \ldots x_{2n}$, respectively.
- For each truth t assigment in A (resp. in B) we define $S_t = \{C \in \mathcal{C}$ such that t does not satisfy $C\} \cup \{a\}$ (resp. $\cup\{b\}$).

- The sets $S's$ defined with A (resp. B) always intersect because of a (resp. b).

- The sets $S's$ defined with A (resp. B) always intersect because of a (resp. b).
- If there exists S_u, S_v that do not intersect. Necessarily u is a truth assignment in A and v in B (or the converse, but they cannot be on the same set of variables because of the dummy vertices a, b).
  This means that for each clause C_i of I, if $C_i \notin S_u$, then the truth v assignment satisfies C_i.
  Similarly if $C_i \notin S_v$, then the truth u assignment satisfies C_i.
  But $S_u \cap S_v = \emptyset$ means that for every clause C_i either : $C_i \notin S_u$ or $C_i \notin S_v$.

- The sets $S's$ defined with A (resp. B) always intersect because of a (resp. b).
- If there exists S_u, S_v that do not intersect. Necessarily u is a truth assignment in A and v in B (or the converse, but they cannot be on the same set of variables because of the dummy vertices a, b).
  This means that for each clause C_i of I, if $C_i \notin S_u$, then the truth v assignment satisfies C_i.
  Similarly if $C_i \notin S_v$, then the truth u assignment satisfies C_i.
  But $S_u \cap S_v = \emptyset$ means that for every clause C_i either : $C_i \notin S_u$ or $C_i \notin S_v$.
- Therefore :
  I is satisfiable iff there exist 2 disjoint sets S_u, S_v.

Complexity issues

- Size of the $k - SAT$ instance is bounded by :
  $K = 2n + m + km$

Complexity issues

- Size of the $k - SAT$ instance is bounded by :
  $K = 2n + m + km$
- Size of the Disjoint set instance :
  $N = 2^{n+1} + m + 2$ vertices
  and at most $M = m2^{n+1}$ edges.

Complexity issues

- Size of the $k - SAT$ instance is bounded by :
  $K = 2n + m + km$
- Size of the Disjoint set instance :
  $N = 2^{n+1} + m + 2$ vertices
  and at most $M = m2^{n+1}$ edges.
- To compute this instance we need to evaluate the m, k-clauses for each half-truth assigment.
  Can be done in $O(K)$, so in the whole : $O(2^{n+1}K)$.

Complexity issues

- Size of the $k - SAT$ instance is bounded by :
  $K = 2n + m + km$
- Size of the Disjoint set instance :
  $N = 2^{n+1} + m + 2$ vertices
  and at most $M = m2^{n+1}$ edges.
- To compute this instance we need to evaluate the m, k-clauses for each half-truth assigment.
  Can be done in $O(K)$, so in the whole : $O(2^{n+1}K)$.
- If there exists an algorithm for the Disjoint set problem in less than $O(NM^{1-\epsilon})$
  it would imply an algorithm for $k - SAT$ in less than $O((2-\epsilon)^{2n})$ contradiction the SETH.

Consequences

Practically there is no hope to design a linear time algorithm for :

1. Disjoint set problem

Consequences

Practically there is no hope to design a linear time algorithm for :

1. Disjoint set problem
2. Diameter computations for chordal graphs and split graphs

Consequences

Practically there is no hope to design a linear time algorithm for :

1. Disjoint set problem
2. Diameter computations for chordal graphs and split graphs
3. And many other related problems ... such as betweenness centrality

Consequences

Practically there is no hope to design a linear time algorithm for :

1. Disjoint set problem
2. Diameter computations for chordal graphs and split graphs
3. And many other related problems . . . such as betweenness centrality
4. but not all $O(mn)$ problems as for example transitive closure, existence of a triangle . . .

Research Problem

- Since sparse graphs are not available for the above reduction.

Research Problem

- Since sparse graphs are not available for the above reduction.
- Can we compute in linear time the diameter of planar graphs ?

Research Problem

- Since sparse graphs are not available for the above reduction.
- Can we compute in linear time the diameter of planar graphs ?
- This class contains all grids !

Research Problem

- Since sparse graphs are not available for the above reduction.
- Can we compute in linear time the diameter of planar graphs ?
- This class contains all grids !
- Hot subject

BFS versus LL

- Level Layered search visits the vertices according to their distance to the starting vertex, with no extra condition on each level. It differs from BFS, since for BFS via the queue data structure the visiting ordering of Level(i+1) is forced by the visiting ordering of Level(i).

BFS versus LL

- Level Layered search visits the vertices according to their distance to the starting vertex, with no extra condition on each level. It differs from BFS, since for BFS via the queue data structure the visiting ordering of Level(i+1) is forced by the visiting ordering of Level(i).
- The end vertex problem is polynomial for LL.

BFS versus LL

- Level Layered search visits the vertices according to their distance to the starting vertex, with no extra condition on each level. It differs from BFS, since for BFS via the queue data structure the visiting ordering of Level(i+1) is forced by the visiting ordering of Level(i).
- The end vertex problem is polynomial for LL.
- Many authors make no difference between BFS and LL (even Cormen, Leiserson and Rivest in their book : Introduction to algorithms).

BFS versus LL

- Level Layered search visits the vertices according to their distance to the starting vertex, with no extra condition on each level. It differs from BFS, since for BFS via the queue data structure the visiting ordering of Level(i+1) is forced by the visiting ordering of Level(i).
- The end vertex problem is polynomial for LL.
- Many authors make no difference between BFS and LL (even Cormen, Leiserson and Rivest in their book : Introduction to algorithms).
- LL^+ ends at a vertex with minimum degree from the previous layer.

1. To handle huge graphs we already have : graph searches.

1. To handle huge graphs we already have : graph searches.
2. But BFS is not so easy to program in a distributed environment.

1. To handle huge graphs we already have : graph searches.
2. But BFS is not so easy to program in a distributed environment.
3. For example, using Map - Reduce operations as popularized by Google.

1. To handle huge graphs we already have : graph searches.
2. But BFS is not so easy to program in a distributed environment.
3. For example, using Map - Reduce operations as popularized by Google.
4. Hot topic to find good way to handle huge graphs in a distributed system.

1. To handle huge graphs we already have : graph searches.
2. But BFS is not so easy to program in a distributed environment.
3. For example, using Map - Reduce operations as popularized by Google.
4. Hot topic to find good way to handle huge graphs in a distributed system.
5. In 2010 Google proposes a language named Pregel.
   Another one Giraf for the Hadoop platform (available free)

- Some hope : Layered search is not so bad.

- Some hope : Layered search is not so bad.
- We have some theoretical results on LL

- Some hope : Layered search is not so bad.
- We have some theoretical results on LL
- We do not know if BFS is really needed ?

Theoretical aspects

- D. Corneil, F. Dragan, M. Habib, C. Paul, *Diameter determination on restricted families of graphs*, **Discrete Applied Mathematic**, Vol 113(2-3) : 143-166 (2001)
- V. Chepoi, F. Dragan, B. Estellon, M. Habib, Y. Vaxes, *Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs*, **ACM Symposium on Computational Geometry** 2008 : 59-68.
- V. Chepoi, F. Dragan, B. Estellon, M. Habib, Y. Vaxes, *Notes on diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs*, TGCT08 Paris, **Electronic Notes in Discrete Mathematics** 31(2008)231-234.
- V. Chepoi, F. Dragan, B. Estrellon, M. Habib, Y. Vaxes et Y. Xiang *Additive Spanners and Distance and Routing Labeling Schemes for Hyperbolic Graphs*, **Algorithmica** 62-(3-4) (2012) 713-732.

Computational aspects

- C. Magnien, M. Latapy, M. Habib, *Fast computation of empirically tight bounds for the diameter of massive graphs*, **Journal of Experimental Algorithmics**, 13 (2008).
- P. Crescenzi, R. Grossi, M. Habib, L. Lanzi and A. Marino, *On Computing the Diameter of Real-World Undirected graphs*, **Theor. Comput. Sci.** 514 : 84-95 (2013).
- M. Borassi, P. Crescenzi, R. Grossi, M. Habib, W. Kosters, A. Marino and F. Takes, *Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs : With an application to the six degrees of separation games*, **Theor. Comput. Sci.** 586 : 59-80 (2015)