Radius and diameter computations in huge graphs and some extensions,

Michel Habib
habib@liafa.univ-paris-diderot.fr
http://www.liafa.univ-paris-diderot.fr/~habib

Ecole ResCom, Furiani
May 2014
Radius and diameter computations in huge graphs and some extensions,

I am playing with graph algorithms since ....
I am playing with graph algorithms since ....

Graphs or networks:
I am playing with graph algorithms since ....

Graphs or networks:

One of the main problems I am looking at:
What can you learn about the structure of a given graph using a series of consecutive graph searches?
I am playing with graph algorithms since ....

Graphs or networks:

One of the main problems I am looking at:
What can you learn about the structure of a given graph using a series of consecutive graph searches?

From now on:
Graphs are undirected and supposed to be finite and connected.
Schedule of the talk

Graph searches
Radius and diameter computations in huge graphs and some extensions,

Schedule of the talk

Graph searches

Diameter computations
Radius and diameter computations in huge graphs and some extensions,

Schedule of the talk

Graph searches

Diameter computations

Computing diameter using fewest BFS possible
Schedule of the talk

Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database
Schedule of the talk

Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database

Recents results
Schedule of the talk

Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database

Recents results

Huge graphs
Schedule of the talk

- Graph searches
- Diameter computations
- Computing diameter using fewest BFS possible
- The Stanford Database
- Recents results
- Huge graphs
- Consequences and perspectives
Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database

Recents results

Huge graphs

Consequences and perspectives
Graph searches are very well known and used in many situations:

1. "Fil d’ariane" in the Greek mythology.
Graph searches are very well known and used in many situations:

1. "Fil d’ariane” in the Greek mythology.
2. Euler (1735) for solving the famous walk problem in Kœnigsberg city
Graph searches are very well known and used in many situations:

1. "Fil d’ariane" in the Greek mythology.
2. Euler (1735) for solving the famous walk problem in Køenisberg city.
3. Euler’s theorem proved by Hierholzer in 1873.
Graph searches are very well known and used in many situations:

1. "Fil d’ariane" in the Greek mythology.
2. Euler (1735) for solving the famous walk problem in Kœnisberg city.
3. Euler’s theorem proved by Hierholzer in 1873.
4. Tremaux (1882) and Tarry (1895) using DFS to solve maze problems.
Graph searches are very well known and used in many situations:

1. "Fil d’ariane" in the Greek mythology.
2. Euler (1735) for solving the famous walk problem in Køenisberg city.
3. Euler’s theorem proved by Hierholzer in 1873.
4. Tremaux (1882) and Tarry (1895) using DFS to solve maze problems.
5. Fleury, proposed a nice algorithm to compute an Euler Tour, cited in E. Lucas, Récréations mathématiques, Paris, 1891.
Graph searches are very well known and used in many situations:

1. "Fil d’ariane” in the Greek mythology.
2. Euler (1735) for solving the famous walk problem in Kœnisberg city
3. Euler’s theorem proved by Hierholzer in 1873.
4. Tremaux (1882) and Tarry (1895) using DFS to solve maze problems
5. Fleury, proposed a nice algorithm to compute an Euler Tour, cited in E. Lucas, Récréations mathématiques, Paris, 1891.
6. Computer scientists from 1950, in particular in the 70’s, Tarjan for new applications of DFS....
Graph searches are very well known and used in many situations:

1. "Fil d’ariane" in the Greek mythology.
2. Euler (1735) for solving the famous walk problem in Kœnigsberg city.
3. Euler’s theorem proved by Hierholzer in 1873.
4. Tremaux (1882) and Tarry (1895) using DFS to solve maze problems.
5. Fleury, proposed a nice algorithm to compute an Euler Tour, cited in E. Lucas, Récréations mathématiques, Paris, 1891.
6. Computer scientists from 1950, in particular in the 70’s, Tarjan for new applications of DFS....
7. 4 points characterizations Corneil, Krueger (2008), and the definition of LDFS a new interesting basic search.
LE PROBLÈME DES LABYRINTHES;
PAR M. G. TARRY.

Tout labyrinthe peut être parcouru en une seule course, en passant deux fois en sens contraire par chacune des allées, sans qu'il soit nécessaire d'en connaître le plan.

Pour résoudre ce problème, il suffit d'observer cette règle unique :

*Ne reprendre l'allée initiale qui a conduit à un carrefour pour la première fois que lorsqu'on ne peut pas faire autrement.*

Nous ferons d'abord quelques remarques. A un moment quelconque, avant d'arriver à un car-
« Pour trouver la sortie d’un labyrinthe, récita en effet Guillaume, il n’y a qu’un moyen. À chaque nœud nouveau, autrement dit jamais visité avant, le parcours d’arrivée sera marqué de trois signes. Si, à cause de signes précédents sur l’un des chemins du nœud, on voit que ce nœud a déjà été visité, on placera un seul signe sur le parcours d’arrivée. Si tous les passages ont été déjà marqués, alors il faudra reprendre la même voie, en revenant en arrière. Mais si un ou deux passages du nœud sont encore sans signes, on en choisira un quelconque, pour y apposer deux signes. Quand on s’achemine par un passage qui porte un seul signe, on en apposera deux autres, de façon que ce passage en porte trois dorénavant. Toutes les parties du labyrinthe devraient avoir été parcourues si, en arrivant à un nœud, on ne prend jamais le passage avec trois signes, sauf si d’autres passages sont encore sans signes.

— Comment le savez-vous ? Vous êtes expert en labyrinthes ?
— Non, je récita un extrait d’un texte antique que j’ai lu autrefois.
— Et selon cette règle, on sort ?
— Presque jamais, que je sache. Mais nous tenterons quand même. Et puis dans les prochains jours j’aurai des verres et j’aurai le temps de mieux me pencher sur les livres. Il se peut que là où le parcours des cartouches nous embrouille, celui des livres nous donne une règle.
**Some definitions**

**Graph Search**

The graph is **supposed to be connected** so as the set of visited vertices. After choosing an initial vertex, a search of a connected graph visits each of the vertices and edges of the graph such that a new vertex is visited only if it is adjacent to some previously visited vertex.

At any point there may be several vertices that may possibly be visited next. To choose the next vertex we need a tie-break rule. The breadth-first search (BFS) and depth-first search (DFS) algorithms are the traditional strategies for determining the next vertex to visit.
Variations

Graph Traversal

The set of visited vertices is not supposed to be connected (used for computing connected components for example)
Variations

Graph Traversal
The set of visited vertices is not supposed to be connected (used for computing connected components for example)

Graph Searching for cops and robbers games on a graph
The name Graph searching is also used in this context, with a slightly different meaning. Relationships with width graph parameters such as treewidth.
Our main question

Main Problem
What kind of knowledge can we learn about the structure of a given graph via graph searching (i.e. with one or a series of successive graph searches)?

Goals
Our main question

Main Problem
What kind of knowledge can we learn about the structure of a given graph via graph searching (i.e. with one or a series of successive graph searches)?

Goals
▶ Building bottom up graph algorithms from well-known graph searches
Our main question

Main Problem
What kind of knowledge can we learn about the structure of a given graph via graph searching (i.e. with one or a series of successive graph searches)?

Goals
- Building bottom up graph algorithms from well-known graph searches
- Develop basic theoretic tools for the structural analysis of graphs
Our main question

Main Problem
What kind of knowledge can we learn about the structure of a given graph via graph searching (i.e. with one or a series of successive graph searches) ?

Goals
- Building bottom up graph algorithms from well-known graph searches
- Develop basic theoretic tools for the structural analysis of graphs
- Applications on huge graphs :
  No need to store sophisticated data structures, just some labels on each vertex,
We can play with:

1. Find new uses of already known searches or describe new interesting searches designed for special purpose
We can play with:

1. Find new uses of already known searches or describe new interesting searches designed for special purpose

2. Seminal paper:
Basic graph searches

- Generic search, BFS, DFS
- LBFS, LDFS
- But also MNS, MCS
Generic Search

Invariant
At each step, an edge between a visited vertex and a unvisited one is selected
Generic Search

Invariant
At each step, an edge between a visited vertex and a unvisited one is selected
Generic Search

Invariant
At each step, an edge between a visited vertex and a unvisited one is selected.
Generic Search

Invariant

At each step, an edge between a visited vertex and a unvisited one is selected.
Generic Search

Invariant
At each step, an edge between a visited vertex and a unvisited one is selected.
**Invariant**

At each step, an edge between a visited vertex and a unvisited one is selected.
Generic Search

Invariant
At each step, an edge between a visited vertex and a unvisited one is selected
**Generic search**

\[ S \leftarrow \{s\} \]

\[ \text{for } i \leftarrow 1 \text{ to } n \text{ do} \]

\[ \quad \text{Pick an unnumbered vertex } v \text{ of } S \]

\[ \quad \sigma(i) \leftarrow v \]

\[ \quad \text{foreach unnumbered vertex } w \in N(v) \text{ do} \]

\[ \quad \quad \text{if } w \notin S \text{ then} \]

\[ \quad \quad \quad \text{Add } w \text{ to } S \]

\[ \quad \text{end} \]

\[ \text{end} \]
Generic question?

Let $a$, $b$ et $c$ be 3 vertices such that $ab \notin E$ et $ac \in E$.

Under which condition could we visit first $a$ then $b$ and last $c$?
Property (Generic)

For an ordering \( \sigma \) on \( V \), if \( a <_\sigma b <_\sigma c \) and \( ac \in E \) and \( ab \notin E \), then it must exist a vertex \( d \) such that \( d <_\sigma b \) et \( db \in E \).
Property (Generic)
For an ordering $\sigma$ on $V$, if $a <_{\sigma} b <_{\sigma} c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $d <_{\sigma} b$ et $db \in E$.

Theorem
For a graph $G = (V, E)$, an ordering $\sigma$ sur $V$ is a generic search of $G$ iff $\sigma$ satisfies property (Generic).
Most of the searches that we will study are refinement of this generic search
i.e. we just add new rules to follow for the choice of the next vertex to be visited
Graph searches mainly differ by the management of the tie-break set
Parcours en largeur (BFS)

**Données:** Un graphe $G = (V, E)$ et un sommet source $s$

**Résultat:** Un ordre total $\sigma$ de $V$

Initialiser la file $S$ à $s$

**pour** $i \leftarrow 1$ à $n$ **faire**

Extraitre le sommet $v$ de la tête de la file $S$

$\sigma(i) \leftarrow v$

**pour chaque sommet non-numéroté $w \in N(v)$ faire**

si $w$ n’est pas dans $S$ alors

Ajouter $w$ en fin de la file $S$

fin

fin

fin
Property (BFS)

For an ordering $\sigma$ on $V$, if $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $d <_\sigma a$ et $db \in E$
Property (BFS)

For an ordering $\sigma$ on $V$, if $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $d <_\sigma a$ et $db \in E$

Theorem

For a graph $G = (V, E)$, an ordering $\sigma$ sur $V$ is a BFS of $G$ iff $\sigma$ satisfies property (BFS).
Applications of BFS

1. Distance computations (unit length), diameter and centers
2. BFS provides a useful layered structure of the graph
3. Using BFS to search an augmenting path provides a polynomial implementation of Ford-Fulkerson maximum flow algorithm.
Recent results

- LDFS Lexicographic Depth First Search with application to hamiltonicity
Recent results

- LDFS Lexicographic Depth First Search with application to hamiltonicity
- Some other Lexicographic Searches : LexUP, LexDown
Recent results

- LDFS Lexicographic Depth First Search with application to hamiltonicity
- Some other Lexicographic Searches: LexUP, LexDown
- Recognition of cocomparability graphs using a series of LBFS
Recent results

- LDFS Lexicographic Depth First Search with application to hamiltonicity
- Some other Lexicographic Searches: LexUP, LexDown
- Recognition of cocomparability graphs using a series of LBFS
- Many questions about fixed points
Graph searches

**Diameter computations**

Computing diameter using fewest BFS possible

The Stanford Database

Recents results

Huge graphs

Consequences and perspectives
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 (Firenze) 
and discussion with many others . . .
Basics Definitions

Definitions:

Let $G$ be an undirected graph:

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

Definitions:
Let $G$ be an undirected graph:

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

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 $G$:

$\text{radius}(G) \leq \text{diam}(G) \leq 2\text{radius}(G)$ and $\forall e \in G$, $\text{diam}(G) \leq \text{diam}(G - e)$
Trivial bounds
For any graph $G$:

$\text{radius}(G) \leq \text{diam}(G) \leq 2\text{radius}(G)$ and $\forall e \in G$, 

$\text{diam}(G) \leq \text{diam}(G - e)$

These bounds are tight
Trivial bounds

For any graph $G$:

$\text{radius}(G) \leq \text{diam}(G) \leq 2\text{radius}(G)$ and $\forall e \in G$, 
$\text{diam}(G) \leq \text{diam}(G - e)$

These bounds are tight

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

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

These bounds are tight

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

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.
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 $O\left(\frac{n^3}{\log^2 n}\right)$, 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 $O\left(\frac{n^3}{\log^2 n}\right)$, in fact computing all shortest paths.

But also with at most $O(Diam(G))$ matrix multiplications.
Graph searches

Diameter computations

**Computing diameter using fewest BFS possible**

The Stanford Database

Recents results

Huge graphs

Consequences and perspectives
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 chordal 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 \( w \in V \)

\( u \leftarrow \text{BFS}(w) \)

\( v \leftarrow \text{BFS}(u) \)

*Where BFS stands for Breadth First Search.*

Therefore it is a linear procedure

---

1. Proposed the first time by Handler 1973
Radius and diameter computations in huge graphs and some extensions,

Computing diameter using fewest BFS possible

Intuition behind the procedure
Handler's classical result 73
If $G$ is a tree, $diam(G) = d(u, v)$
Easy using Jordan's theorem.
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 $\text{radius}(G) = \lceil \frac{\text{diam}(G)}{2} \rceil$
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.
Randomized BFS procedure

**Data:** A graph $G = (V, E)$

**Result:** $u, v$ two vertices

**Repeat $\alpha$ 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:
\[ \text{exc}(u_0) \leq \text{diam}(G) \] i.e. a lower bound of the diameter.
1. This procedure gives a vertex $u_0$ such that:
   \[ \text{exc}(u_0) \leq \text{diam}(G) \] i.e. a lower bound of the diameter.

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

2. Use a spanning tree as a partial subgraph to obtain an upper bound by computing its exact diameter in linear time.

3. Spanning trees given by the BFS.
The Program and some Data on Web graphs or P-2-P networks can be found...
Sample text with LaTeX: \text{Radius and diameter computations in huge graphs and some extensions,}

\text{Computing diameter using fewest BFS possible}

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

Further experimentations by Crescenzi, Grossi, Marino (in \textit{ESA 2010}) which confirm the excellence of the lower bound using BFS.
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
Radius and diameter computations in huge graphs and some extensions,

Computing diameter using fewest BFS possible

▶ 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 experimentation by Crescenzi, Grossi, Marino (in ESA 2010)
which confirm the excellence of the lower bound using BFS!!!!
Since $\alpha$ is a constant ($\leq 1000$), this method is still in linear time and works extremely well on huge graphs (Web graphs, Internet . . .)
Since $\alpha$ is a constant ($\leq 1000$), 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 $\alpha$ is a constant ($\leq 1000$), 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.
Radius and diameter computations in huge graphs and some extensions,
Computing diameter using fewest BFS possible

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 $\geq 4$. 
Chordal graphs

1. A graph is chordal if it has no chordless cycle of length $\geq 4$.
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 \( \geq 4 \).

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

\[ Diam = \max\{\text{ecc}(a_1), \text{ecc}(a_2)\} \quad \text{and} \quad Rad = \min\{\text{ecc}(r), \text{ecc}(m_1)\} \]
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 eccentricity 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 $x \in \text{Level}(i)$ of the tree, we have $\text{ecc}(x) > 2(i - 1)$ 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 $j < i$. $\forall z \in \bigcup_{1 \leq k \leq i-1} L(k)$
$dist(z, y) \leq 2(i - 1)$

Therefore $ecc(y) \leq ecc(x)$ or the extreme vertices from $y$ belong to lower layers and have already been considered.
Radius and diameter computations in huge graphs and some extensions,
Computing diameter using fewest BFS possible

**Algorithm 1: iFUB (iterative Fringe Upper Bound)**

**Input:** $G$, $u$, lower bound $l$

**Output:** A value $M$ such that $D - M \leq k$.

```plaintext
i ← \text{ecc}(u); \ lb ← \max\{\text{ecc}(u), l\}; \ ub ← 2\text{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
        $lb ← \max\{B_1(u), \ldots, B_i(u)\}$;
        $ub ← 2(i - 1)$;
    end
    $i ← i - 1$;
end
return $lb$;
```
Bad example
Results:

<table>
<thead>
<tr>
<th>$\nu$</th>
<th># of graphs in which $\nu$ BFSes done on the average</th>
<th>Number $n$ of vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>$\leq 10^3$</td>
</tr>
<tr>
<td>$\nu = 5$</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>$5 &lt; \nu \leq 100$</td>
<td>123</td>
<td>17</td>
</tr>
<tr>
<td>$100 &lt; \nu \leq 1000$</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>$1000 &lt; \nu \leq 10^4$</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>$10^4 &lt; \nu \leq 10^5$</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Diameter Facebook = 41! Backstrom, Boldi, Rosa, Ugandden, 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 always gives a lower bound of the diameter not too far from the optimal, the hard part is to obtain an upper bound with iFUB.
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.
Radius and diameter computations in huge graphs and some extensions,
The Stanford Database

Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database

Recents results

Huge graphs

Consequences and perspectives
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.
Radius and diameter computations in huge graphs and some extensions,

<table>
<thead>
<tr>
<th>Graph</th>
<th>diam SNAP</th>
<th>diam 4-Sweep</th>
</tr>
</thead>
<tbody>
<tr>
<td>soc-Epinions1</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>soc-pokec-relationships</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>soc-Slashdot0811</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>soc-Slashdot0902</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>com-lj.ungraph</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>com-youtube.ungraph</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>com-DBLP</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>com-amazon</td>
<td>44</td>
<td>47</td>
</tr>
<tr>
<td>email-Enron</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>wikiTalk</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>cit-HepPh</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>cit-HepTh</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>CA-CondMat</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>CA-HepTh</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>web-Google</td>
<td>21</td>
<td>24</td>
</tr>
</tbody>
</table>
Radius and diameter computations in huge graphs and some extensions, The Stanford Database

<table>
<thead>
<tr>
<th>Graph</th>
<th>diam SNAP</th>
<th>diam 4-Sweep</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon0302</td>
<td>32</td>
<td>38</td>
</tr>
<tr>
<td>amazon0312</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>amazon0505</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>amazon0601</td>
<td>21</td>
<td>25</td>
</tr>
<tr>
<td>p2p-Gnutella04</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>p2p-Gnutella24</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>p2p-Gnutella25</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>p2p-Gnutella30</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>roadNet-CA</td>
<td>849</td>
<td>865</td>
</tr>
<tr>
<td>roadNet-TX</td>
<td>1054</td>
<td>1064</td>
</tr>
<tr>
<td>Gowalla-edges</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>BrightKite-edges</td>
<td>16</td>
<td>18</td>
</tr>
</tbody>
</table>
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 than 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 than 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.
Radius and diameter computations in huge graphs and some extensions, The Stanford Database

<table>
<thead>
<tr>
<th>Graphe</th>
<th>Vertices/Edges</th>
<th>Diameter iFUB</th>
<th>Diam. FourSweep</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA-HepTh</td>
<td>0.190</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>CA-GrQc</td>
<td>0.181</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>CA-CondMat</td>
<td>0.124</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>CA-AstroPh</td>
<td>0.047</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>roadNet-CA</td>
<td>0.355</td>
<td>865</td>
<td>865</td>
</tr>
<tr>
<td>roadNet-PA</td>
<td>0.353</td>
<td>794</td>
<td>780</td>
</tr>
<tr>
<td>roadNet-TX</td>
<td>0.359</td>
<td>1064</td>
<td>1064</td>
</tr>
<tr>
<td>email-Enron</td>
<td>0.1</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>email-EuAll</td>
<td>0.631</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>com-amazon</td>
<td>0.361</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>Amazon0302</td>
<td>0.212</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>Amazon0312</td>
<td>0.125</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Amazon0505</td>
<td>0.122</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Amazon0601</td>
<td>0.119</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Gowalla_edges</td>
<td>0.207</td>
<td>25</td>
<td>16</td>
</tr>
<tr>
<td>Brightkite_edges</td>
<td>0.272</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>soc-Epinions1</td>
<td>0.149</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

**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
Radius and diameter computations in huge graphs and some extensions,

Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database

Recents results

Huge graphs

Consequences and perspectives
A method symmetric for computing radius and diameter


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


- 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


- 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. $\max_{i=1, \ldots, k} \operatorname{exc}(v_i)$, 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. $\min\{\min_{i=1, \ldots, k} \operatorname{exc}(v_i), \operatorname{exc}(w)\}$, is an upper bound for the radius.
Replacing Sum by Max does not change. 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 $Diam(G) \leq \frac{2}{k} \sum_{i=1}^{k} d(v_i, y)$. 
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 $Diam(G) \leq \frac{2}{k} \sum_{i=1}^{k} d(v_i, y)$.

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

$$\text{Radius}(G) \geq \frac{1}{k} \sum_{i=1}^{k} d(x, v_i)$$
Lemma 2
Let $x \in V$ be a center and let $v_1, \ldots, v_k$ be $k$ other vertices. Then
\[ \text{Radius}(G) \geq \frac{1}{k} \sum_{i=1}^{k} d(x, v_i) \]

proof
Let $y \in V$ such that : $\text{Radius}(G) = d(x, y)$
Then
\[ k \text{Radius}(G) = \sum_{i=1}^{k} d(x, y) \geq \sum_{i=1}^{k} [d(x, v_i) + d(v_i, y)] = \sum_{i=1}^{k} d(x, v_i) + \sum_{i=1}^{k} d(v_i, y). \]
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.
Radius and diameter computations in huge graphs and some extensions,

Recents results

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.
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
Relationships between diameter and $\delta$-hyperbolicity

$\delta$-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 $2\delta$. 
Theorem Chepoi, Dragan, Estellon, M.H., Vaxes 2008

If $u$ is the last vertex of a 2-sweep then:

\[ \text{exc}(u) \geq \text{diam}(G) - 2 \cdot \delta(G) \text{ and} \]
\[ \text{radius}(G) \leq \lceil \frac{(d(u, v) + 1)}{2} \rceil + 3 \delta(G) \]

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:

$\text{exc}(u) \geq \text{diam}(G) - 2.\delta(G)$ and

$\text{radius}(G) \leq \lceil (d(u, v) + 1)/2 \rceil + 3\delta(G)$

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 $\delta$-hyperbolicity of the graph.
Nice

Because many real networks have small $\delta$-hyperbolicity.
The difficulty of the certificate

$\delta$-hyperbolicity and treewidth (existence of big grids as subgraphs) must play a role.
Radius and diameter computations in huge graphs and some extensions,

Huge graphs

Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database

Recents results

Huge graphs

Consequences and perspectives
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.
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?
## Restricted Families of Graphs

<table>
<thead>
<tr>
<th>GRAPH CLASS</th>
<th>LL</th>
<th>LL+</th>
<th>BFS</th>
<th>LBFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>chordal graphs</td>
<td>≥ (D - 2) &lt;br&gt; [2] Fig. 4</td>
<td>≥ (D - 2) &lt;br&gt; [2] Fig. 5</td>
<td>≥ (D - 1) &lt;br&gt; [*] Fig. 2</td>
<td>≥ (D - 1) &lt;br&gt; [6] Fig. 6</td>
</tr>
<tr>
<td>AT-free graphs</td>
<td>≥ (D - 2) &lt;br&gt; [*] Fig. 3</td>
<td>≥ (D - 1) &lt;br&gt; [*] Fig. 7</td>
<td>≥ (D - 2) &lt;br&gt; [*] Fig. 3</td>
<td>≥ (D - 1) &lt;br&gt; [3] Fig. 7</td>
</tr>
<tr>
<td>{AT,claw}-free graphs</td>
<td>≥ (D - 1) &lt;br&gt; [*] Fig. 2</td>
<td>= (D) &lt;br&gt; [*]</td>
<td>≥ (D - 1) &lt;br&gt; [*] Fig. 2</td>
<td>= (D) &lt;br&gt; [*]</td>
</tr>
<tr>
<td>interval graphs</td>
<td>≥ (D - 1) &lt;br&gt; [*] Fig. 2</td>
<td>= (D) &lt;br&gt; [*]</td>
<td>≥ (D - 1) &lt;br&gt; [*] Fig. 2</td>
<td>= (D) &lt;br&gt; [6]</td>
</tr>
<tr>
<td>hole-free graphs</td>
<td>≥ (D - 2) &lt;br&gt; [*] Fig. 8</td>
<td>≥ (D - 2) &lt;br&gt; [*] Fig. 8</td>
<td>≥ (D - 2) &lt;br&gt; [*] Fig. 8</td>
<td>≥ (D - 2) &lt;br&gt; [*] Fig. 8</td>
</tr>
</tbody>
</table>

- No induced cycles of length >3
- No asteroidal triples
- No asteroidal triples and
- The intersection graph of intervals of a line
- No induced cycles of length >4

**Figure 7:** LBFS: \(u|cda|bv\)

**Figure 8:** LBFS: \(u|ghifdl|aby\)

asteroidal triple \(a,b,c\)
Graph searches

Diameter computations

Computing diameter using fewest BFS possible

The Stanford Database

Recents results

Huge graphs

Consequences and perspectives
We can really compute the exact value of the diameter for big graphs.
Answers to Frequently Asked Questions

- We can really compute the exact value of the diameter for big graphs.
- Conjecture: the computation of \( \text{radius}(G) \) and "\( \text{diameter}(G) \) within one" have the same complexity. "Within one" because of the case of split graphs.
For practical algorithms the rules are not exactly the same that for classical algorithms in which only worst case complexity matters.
For practical algorithms the rules are not exactly the same that for classical algorithms in which only worst case complexity matters.

But we can have fun!
"While theoretical work on models of computation and methods for analyzing algorithms has had enormous payoffs, we are not done. In many situations, simple algorithms do well. We don't understand why! Developing means for predicting the performance of algorithms and heuristics on real data and on real computers is a grand challenge in algorithms."

Challenges for Theory of Computing by:
CONDON, EDELSBRUNNER, EMERSON, FORTNOW, HABER, KARP, LEIVANT, LIPTON, LYNCH, PARBERRY, PAPADIMITRIOU, RABIN, ROSENBERG, ROYER, SAVAGE, SELMAN, SMITH, TARDOS, AND VITTER,
Report for an NSF-sponsored workshop on research in theoretical computer science.
So we need to understand

- Why the 4-sweep method works so well (analogy with Quicksort)?
So we need to understand

- Why the 4-sweep method works so well (analogy with Quicksort)?
So we need to understand

- Why the 4-sweep method works so well (analogy with Quicksort)?
- For which graphs one can avoid to use $O(n)$ BFS’s to compute the diameter?
General method used so far

1. Find a linear time algorithm proved for a wide class of graphs containing trees, chordal graphs, \ldots
   (Most of these algorithms are based on graph searches)
General method used so far

1. Find a linear time algorithm proved for a wide class of graphs containing trees, chordal graphs, . . .
   (Most of these algorithms are based on graph searches)
2. So the study of graph classes could be useful for applications!
General method used so far

1. Find a linear time algorithm proved for a wide class of graphs containing trees, chordal graphs, \ldots (Most of these algorithms are based on graph searches)
2. So the study of graph classes could be useful for applications!
3. Make an heuristic out of this algorithm applicable on arbitrary graphs
General method used so far

1. Find a linear time algorithm proved for a wide class of graphs containing trees, chordal graphs, ... (Most of these algorithms are based on graph searches)
2. So the study of graph classes could be useful for applications!
3. Make an heuristic out of this algorithm applicable on arbitrary graphs
4. Provide certificates for partial solutions
Hints for future work

- Uses a series of graph searches for preprocessing when faced to hard combinatorial problems (already used in biological applications of interval graphs and for quadratic integer programming) Consecutive Ones property (C1P) (find a good ordering of the columns of the matrix such that in each row the ones are consecutive) A kind of filtering process!
Hints for future work

- Uses a series of graph searches for preprocessing when faced to hard combinatorial problems (already used in biological applications of interval graphs and for quadratic integer programming) Consecutive Ones property (C1P) (find a good ordering of the columns of the matrix such that in each row the ones are consecutive) A kind of filtering process!
- Develop approximation algorithms even for polynomial problems but applied on huge data on which only linear time algorithms can be processed.
Computation of $\delta$-hyperbolicity of graphs. A Gromov’s parameter which measures the distance to a tree in a metric way. Polynomial to compute but not linearly.
Computation of $\delta$-hyperbolicity of graphs. A Gromov’s parameter which measures the distance to a tree in a metric way. Polynomial to compute but not linearly.

Develop similar heuristics for computing Betweenness Centrality or other centrality parameters used in biology or social networks analysis.
Computation of $\delta$-hyperbolicity of graphs. A Gromov’s parameter which measures the distance to a tree in a metric way. Polynomial to compute but not linearly.

Develop similar heuristics for computing Betweenness Centrality or other centrality parameters used in biology or social networks analysis.

Community detection in networks (using LexDFS ?)
Theoretical aspects

Computational aspects

Many thanks for your attention!!