Tutorial on Graph Searching
Part2: LDFS and Cocomparability Graphs

Derek Corneil[1] and others to be named

[1]Computer Science, University of Toronto

Charles University
Oct. 1, 2013

Overview

CHARACTERIZATIONS OF VARIOUS GRAPH SEARCHES

- LDFS and Maximal Neighbourhood Search

COCOMPARABILITY GRAPHS AND LDFS

- Minimum path cover problem
- Other cocomparability problems

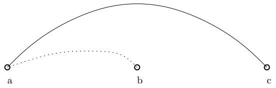
HEURISTIC APPLICATIONS OF BFS

OPEN QUESTIONS

Vertex Ordering Characterizations of Various Graph Searches - Joint work with Richard Krueger.

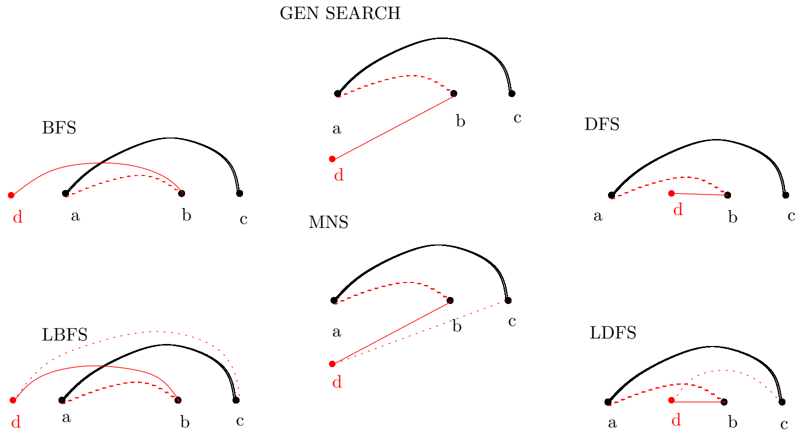
Yesterday we saw:

THEOREM (Golumbic; Dragan, Nicolai + Brandstadt):
An ordering σ is an LBFS ordering iff for all $a <_\sigma b <_\sigma c$ where $ac \in E$, $ab \notin E$, there exists $d <_\sigma a$ such that $db \in E, dc \notin E$.



and asked if there are such characterizations for other graph searches.

CHARACTERIZATIONS OF GRAPH SEARCHES
Joint work with Richard Krueger



$d < a$ $\quad$ $d < b$ $\quad$ $a < d < b$

Maximal Neighbourhood Search

- At each stage choose a vertex whose neighbourhood of previously visited vertices is maximal by set inclusion.
- MNS contains LBFS, LDFS and MCS.
- MNS was first studied by Shier in his work on generating all PEOs of a chordal graph.

  THEOREM [Shier]:

  For any MNS consider the step when a vertex is chosen as one that has a maximal neighbourhood in the set of visited vertices. Replace that step with:

  - Let $\{C_i\}$ be the connected components of $G[V']$.
  - Choose a vertex v in any C_i such that of all the vertices in C_i, v has a maximal neighbourhood in the set of visited vertices.

  Then the resulting $MNS^\star$ can generate all PEOs of G.

MNS Cont.d

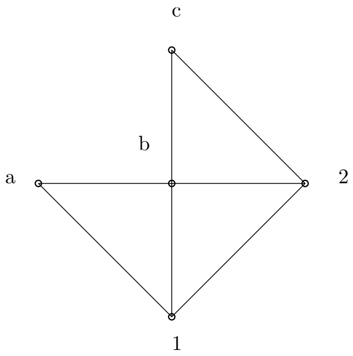
This 4-vertex condition was first discovered by Rose, Tarjan and Lueker where they showed that G is chordal iff any search satisfying the 4-vertex condition is a PEO!

There is no evidence that they realized that the 4-vertex condition characterized MNS.

Lexicographic Depth First Search (LDFS)

Roughly speaking, LDFS is a DFS where ties are broken to favour vertices with recently visited neighbours.

COMPLETE the LDFS

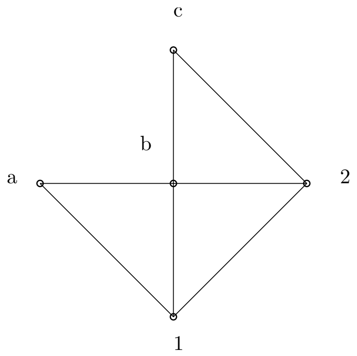


$\tau = 1\ 2\ ?$

Lexicographic Depth First Search (LDFS)

Roughly speaking, LDFS is a DFS where ties are broken to favour vertices with recently visited neighbours.

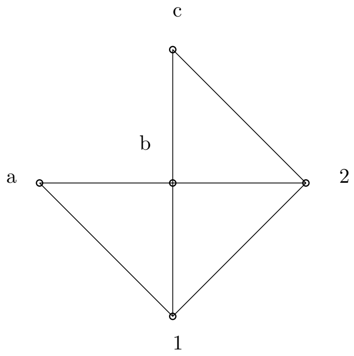
COMPLETE the LDFS



$\tau = 1\ 2\ b\ ?$

Lexicographic Depth First Search (LDFS)

COMPLETE the LDFS



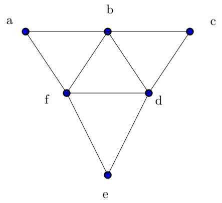
$\tau = 1\ 2\ b\ c\ a$

Lexicographic Depth First Search (LDFS):

Input: Graph $G(V, E)$
Output: Vector σ where $\sigma(i)$ is the i'th vertex chosen

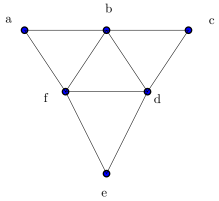
```
1  $V' \leftarrow V$ {$V'$ is the set of unchosen vertices}
2  Step 1: $D$ is a set of vertices, each with a label consisting of a string of decreasing positive integers; initially $D$ contains all vertices in $V'$, each with label $\epsilon$
3  for $i = 1$ to $n$ do
4      Step 2: $v$ chosen from $S \leftarrow$ the set of vertices of $D$ with lexicographically largest label
5      $\sigma(i) \leftarrow v$
6      $V' \leftarrow V' - \{v\}$
7      Step 3: $D \leftarrow D - \{v\}$, and prepend label $i$ to the label of all vertices in $N(v) \cap V'$
8  end for
```

Example of LDFS



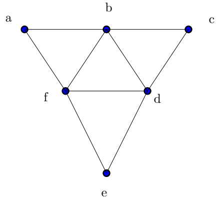
v	a	b	c	d	e	f
e	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

Example of LDFS



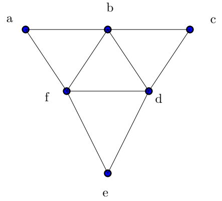
v	a	b	c	d	e	f
e	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
f	ϵ	ϵ	ϵ	1		1

Example of LDFS



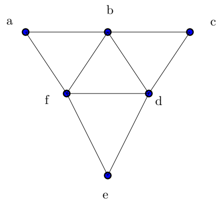
v	a	b	c	d	e	f
e	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
f	ϵ	ϵ	ϵ	1		1
d	2	2	ϵ	21		

Example of LDFS



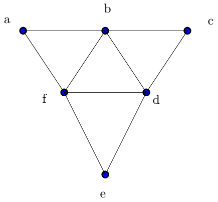
v	a	b	c	d	e	f
e	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
f	ϵ	ϵ	ϵ	1		1
d	2	2	ϵ	21		
b	2	32	3			

Example of LDFS



v	a	b	c	d	e	f
e	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
f	ϵ	ϵ	ϵ	1		1
d	2	2	ϵ	21		
b	2	32	3			
c	42		43			

Example of LDFS



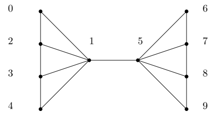
v	a	b	c	d	e	f
e	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
f	ϵ	ϵ	ϵ	1		1
d	2	2	ϵ	21		
b	2	32	3			
c	42		43			
a	42					

Implementations of LDFS

- Note that there is a partition refinement implementation but the refinement is not done in situ. Hence the apparent need to sort.
- The running time of this algorithm is $O(min\{n^2, n + mlogn\})$.
- The current fastest implementation of LDFS is $O(min\{n^2, n + mloglogn\})$ by Jerry Spinrad and an anonymous referee of our paper. They used van Emde Boas trees.

Are there any applications of LDFS? First Attempts

- What about LDFS$^+$ to find a Hamiltonian Path in an Interval Graph, starting with an I ORDER?

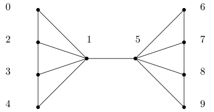


$LBFS : 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$LDFS^+ : 9\ 8\ 5\ 7\ 6\ 1\ 4\ 3\ 2\ 0$

Are there any applications of LDFS? First Attempts

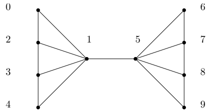
- What about LDFS$^+$ to find a Hamiltonian Path in an Interval Graph, starting with an I ORDER?



$LBFS : 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$LDFS^+ : 9\ 8\ 5\ 7\ 6\ 1\ 4\ 3\ 2\ 0$

- Complete failure - BUT, DFS$^+$ seems to work!



$LBFS : 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$DFS^+ : 9\ 8\ 7\ 6\ 5\ 1\ 4\ 3\ 2\ 0$

- Michel Habib and I convinced ourselves that this algorithm is correct and then turned our attention to cocomparability graphs.
- M.Sc. student, Barnaby Dalton, then looked at the Hamiltonian Path Problem for interval graphs.
- DEFINITION: Minimum Path Cover (MPC): As few paths as possible such that every vertex of G is in exactly one such path - this problem is NP-complete for arbitrary given graphs.
- Note that this is a generalization of the Hamiltonian Path Problem and that DFS$^+$ fails for the MPC problem - Right Most Neighbour is needed instead.
- The MPC problem had been solved for cocomparability graphs but only via the equivalent Bump Number problem for posets.
- It has been an open question for over 20 years whether there is a "direct" graph algorithm.

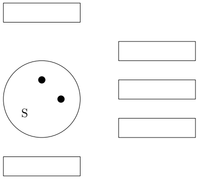
Are there any applications of LDFS?

- DEFINITION: $G(V, E)$ is a cocomparability graph if its complement $\overline{G}$ has a transitive orientation of its edges (i.e., if $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$).
- Notice the close tie between comparability graphs and partially ordered sets (posets).
- Unit Interval Graphs $\subset$ Interval Graphs $\subset$ cocomparability graphs $\subset$ AT-free Graphs.
- OBSERVATION: G is cocomparability iff there is an ordering of V such that for all $x < y < z, xz \in E$ implies $xy \in E$ OR $yz \in E$ OR both (COCOMP order).
  Note how this generalizes the interval order condition.

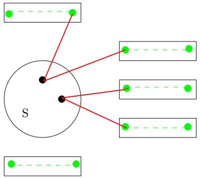
MPC algorithm for interval graphs

1. Let π be an arbitrary I ORDER
2. Let τ be the Right Most Neighbour (RMN) sweep of π [S. Rao Arikati + C. Pandu Rangan; P. Damaschke]
3. If τ is not a Hamiltonian Path, then from τ construct a separator S that certifies that τ is a Minimum Path Cover [B. Dalton]. In particular, S achieves the "scattering number".

$$\#\mathit{of\ paths\ in}\ \tau = \#\ \mathit{of\ connected\ components}\ (G \setminus S) - |S|$$

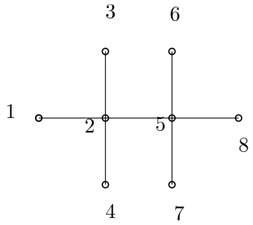


$|S| = 2; \#cc = 5$



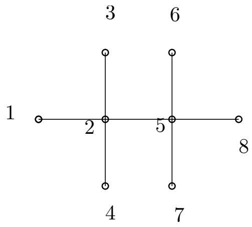
$mpc\# \geq 5 - 2(= 3)$

Example

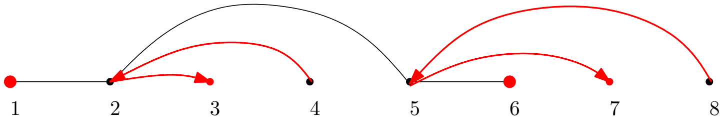


$\pi = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$

Example

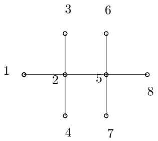


$\pi = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$

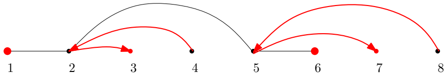


$\tau = 8\ 5\ 7\ ||\ 6\ ||4\ 2\ 3\ ||\ 1$

Example



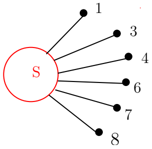
$\pi = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$



$\tau = 8\ 5\ 7\ ||\ 6\ ||4\ 2\ 3\ ||\ 1$

$S = \{2,\ 5\}$

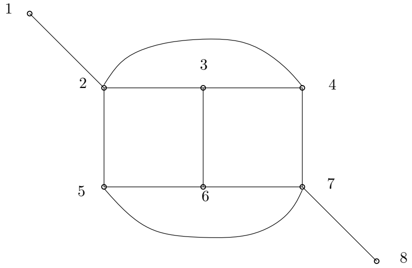
$4 = 6 - 2$



Attempt to solve the problem for cocomparability graphs

1. Let π be an arbitrary COCOMP order
2. Do as many LBFS$^+$s as necessary to end up with "good" sweep σ (Note: an LBFS$^+$ of a COCOMP order is still a COCOMP order)
3. Let τ be the Right Most Neighbour (RMN) sweep of σ. Possibly more than one RMN sweeps.
4. If τ is not a Hamiltonian Path, then from τ construct a separator S that certifies that τ is a Minimum Path Cover. (Such an S exists, by previous work of Habib et al.)

BUT!



For no LBFS COCOMP order of this graph can an RMN produce a Hamiltonian path!

For example:
$\sigma = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$
$\tau = 8\ 7\ 6\ 5\ 2\ 4\ 3\ ||\ 1$

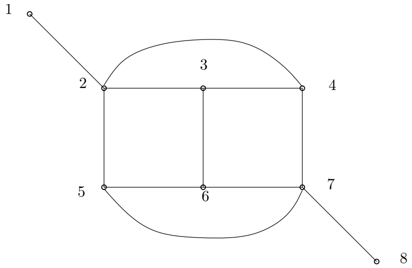
Could LDFS$^+$ work?

Simplest possible algorithm

1. Let π be an arbitrary COCOMP order
2. Let σ be LDFS$^+(\pi)$
3. Let τ be RMN(σ)
4. If τ is not a Hamiltonian Path, then from τ, use Dalton's algorithm to construct a separator S that certifies τ

Note that we're not trying to construct the MPC via LDFS but rather we're treating LDFS$^+$ as a preprocessing step in the hope that it would capture the "interval structure of cocomparability graphs", at least from the perspective of the MPC problem.

Old example



π = 1 2 3 4 5 6 7 8 (arbitrary cocomp order)
σ = 8 7 6 5 2 3 4 1 (LDFS$^+$ of π)
τ = 1 2 4 3 6 5 7 8 (RMN of σ) !

Could this algorithm possibly work?

Consider:

- Why doesn't the interval graph algorithm require an LDFS$^+$ preprocessing step?
  Every I-ORDER is automatically both an LBFS and an LDFS.

Could this algorithm possibly work?

Consider:

- Why doesn't the interval graph algorithm require an LDFS^{+} preprocessing step?
  Every I-ORDER is automatically both an LBFS and an LDFS.
- Is an LDFS^{+} of a COCOMP order a COCOMP order?
  YES! - and so is an RMN of a COCOMP order.

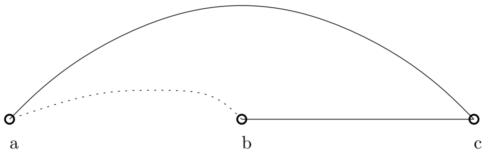
Could this algorithm possibly work?

Consider:

- Why doesn't the interval graph algorithm require an LDFS$^+$ preprocessing step?
  Every I-ORDER is automatically both an LBFS and an LDFS.
- Is an LDFS$^+$ of a COCOMP order a COCOMP order?
  YES! - and so is an RMN of a COCOMP order.
- Does an LDFS COCOMP order have any special properties?
  Yes, the "C_4 property".

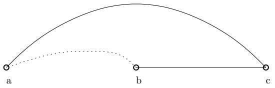
The "C_4 property"

Since an I-ORDER is a PEO, an I-ORDER cannot have:

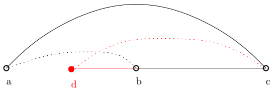


The "C_4 property"

Since an I-ORDER is a PEO, an I-ORDER cannot have:



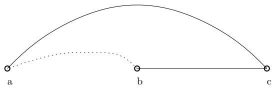
What about an LDFS COCOMP order - is there a similar result?



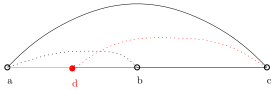
LDFS: $\exists d, a < d < b \mid bd \in E \wedge dc \notin E$

The "C_4 property"

Since an I-ORDER is a PEO, an I-ORDER cannot have:



What about an LDFS COCOMP order - is there a similar result?



$$\text{LDFS: } \exists d, a < d < b \mid bd \in E \wedge dc \notin E$$
$$\text{COCOMP: } ad \in E$$

Note the C_4 on $\{a, b, c, d\}$

MPC cocomp algorithm

1. Let π be an arbitrary COCOMP order
2. Let σ be LDFS$^+(\pi)$
3. Let τ be RMN(σ)
4. If τ is not a Hamiltonian Path, then from τ, use Dalton's algorithm to construct a separator S that certifies τ

The algorithm works.

It also gives us a scattering set for G. (i.e., a set S that maximizes $|\#cc\{G \setminus S\} - |S|\,|$)

Notes about this algorithm

- McConnell and Spinrad have a linear time algorithm that given a cocomparability graph produces a cocomp order. If the graph is not cocomparability, it produces an ordering that is not a cocomp order.

Notes about this algorithm

- McConnell and Spinrad have a linear time algorithm that given a cocomparability graph produces a cocomp order. If the graph is not cocomparability, it produces an ordering that is not a cocomp order.
- The currently known fastest algorithm to determine if a given order is a cocomp order requires $O(MM)$ time.

Notes about this algorithm

- McConnell and Spinrad have a linear time algorithm that given a cocomparability graph produces a cocomp order. If the graph is not cocomparability, it produces an ordering that is not a cocomp order.
- The currently known fastest algorithm to determine if a given order is a cocomp order requires $O(MM)$ time.
- Thus the importance of having a certifying step, so that the algorithm will either conclude that the produced path cover is of minimum cardinality or that the order is not a cocomp order.

Notes about this algorithm

- McConnell and Spinrad have a linear time algorithm that given a cocomparability graph produces a cocomp order. If the graph is not cocomparability, it produces an ordering that is not a cocomp order.
- The currently known fastest algorithm to determine if a given order is a cocomp order requires $O(MM)$ time.
- Thus the importance of having a certifying step, so that the algorithm will either conclude that the produced path cover is of minimum cardinality or that the order is not a cocomp order.
- It's very surprising that with an LDFS cocomp order, the interval algorithm works perfectly on cocomparability graphs. In fact, the certification proof is easier than for interval graphs since we knew that the RMN sweep is a cocomp order.

Further comments

- What about the timing? Except for the LDFS$^+$ step, the running time is linear; the LDFS$^+$ algorithm requires a *loglogn* factor over linear time.

Further comments

- What about the timing? Except for the LDFS$^+$ step, the running time is linear; the LDFS$^+$ algorithm requires a *loglogn* factor over linear time.
- BUT recently Lalla Mouatadid and Ekki Koehler found a linear time algorithm that given a cocomp order can produce different types of LDFS cocomp orders including LDFS$^+$.

Further comments

- What about the timing? Except for the LDFS$^+$ step, the running time is linear; the LDFS$^+$ algorithm requires a *loglogn* factor over linear time.
- BUT recently Lalla Mouatadid and Ekki Koehler found a linear time algorithm that given a cocomp order can produce different types of LDFS cocomp orders including LDFS$^+$.
- All the algorithm needs is an arbitrary LDFS cocomp order. Thus the running time of the algorithm is now linear.

Are there other problems where LDFS plays such a role on cocomparability graphs?

Longest Path: Ionnidou, Mertzios and Nikolopoulos produced an $O(n^4)$ algorithm to find a longest path in an interval graph. George and I showed that by using that algorithm on a LDFS cocomp order, there is an $O(n^4)$ algorithm to find a longest path in a cocomparability graph. I & N found a very complicated $O(n^7)$ algorithm for the longest path problem on cocomparability graphs without using LDFS.

Are there other problems where LDFS plays such a role on cocomparability graphs?

- Hamiltonian Cycle:
  THEOREM: A cocomparability graph G has an HC iff for all $v \in G$ $G \setminus \{v\}$ has an HP. Furthermore, if $G \setminus \{v\}$ doesn't have an HP, then its scattering set certificate together with v is a toughness certificate of no HC.

  Jeremie Dusart, Ekki Koehler and I are guardedly optimistic that we can modify Keil's HC algorithm for interval graphs to get a linear time algorithm for cocomparability graphs, again using an LDFS cocomp order.

Are there other problems where LDFS plays such a role on cocomparability graphs?

- Hamiltonian Cycle:
  THEOREM: A cocomparability graph G has an HC iff for all $v \in G$ $G \setminus \{v\}$ has an HP. Furthermore, if $G \setminus \{v\}$ doesn't have an HP, then its scattering set certificate together with v is a toughness certificate of no HC.

  Jeremie Dusart, Ekki Koehler and I are guardedly optimistic that we can modify Keil's HC algorithm for interval graphs to get a linear time algorithm for cocomparability graphs, again using an LDFS cocomp order.
- Maximum (weighted) Independent Set: To be discussed in Thursday's talk.

HEURISTIC APPLICATION OF BFS

- The diameter of social networks has long been of interest; "Six degrees of separation".
- Using BFS there is a brute-force O(nm) algorithm to compute the diameter of a graph.
- P. Crescenzi, R. Grossi, M.H., L. Lanzi and A. Marino have recently developed an algorithm, based on BFS that has worst case running time of O(nm), but for real-world networks runs in linear time - approximately 200 real-world graphs were tested.
- The algorithm exploits the expected structure of real-world networks that the radius is half the diameter, and that there are not a lot of vertices of maximum eccentricity. They use a 4-sweep BFS algorithm that hopefully finds a vertex u of eccentricity close to the radius; the algorithm then works on u's BFS tree, and may require more BFS sweeps. It guarantees that the value returned is within given constant $k, k \geq 0$ of the diameter.

HEURISTIC APPLICATION OF BFS Cont.d

- Crescenzi et al. wrote: “The main contribution of this paper consists of showing that BFS can indeed be an extremely powerful tool in order to compute the exact value of the diameter, whenever it is used in a more clever way.”

HEURISTIC APPLICATION OF BFS Cont.d

- Crescenzi et al. wrote: “The main contribution of this paper consists of showing that BFS can indeed be an extremely powerful tool in order to compute the exact value of the diameter, whenever it is used in a more clever way.”
- Recently, L. Backstrom, P. Boldi, M. Rosa, J. Ugander and S. Vigna implemented a highly parallel version of this algorithm and studied the full Facebook graph (approx. 721 million vertices and 69 billion edges), as well as various country (or pair of countries) subgraphs.

HEURISTIC APPLICATION OF BFS Cont.d

- Crescenzi et al. wrote: “The main contribution of this paper consists of showing that BFS can indeed be an extremely powerful tool in order to compute the exact value of the diameter, whenever it is used in a more clever way.”
- Recently, L. Backstrom, P. Boldi, M. Rosa, J. Ugander and S. Vigna implemented a highly parallel version of this algorithm and studied the full Facebook graph (approx. 721 million vertices and 69 billion edges), as well as various country (or pair of countries) subgraphs.
- They found that the average distance in the graph is 4.74 (i.e. 3.74 “degrees of separation”), and the diameter of the “giant component” (approx. 98% of the vertices) is 41. The total number of BFS sweeps performed was 17.

Another application of the BFS diameter algorithm

- Recently Michel and a student looked at the Stanford testbed of large graphs and ran the BFS diameter algorithm on these graphs.
- They found that for most (all?) the diameter found by their algorithm (and guaranteed) was higher than the diameter estimate provided.
- They asked how the Stanford team had estimated the diameter and were told that 1,000 vertices were chosen at random and the diameter estimate was the maximum of the eccentricities of these vertices.
- To compute the exact diameter, the BFS diameter algorithm used on average around 10 BFSs.

Open Questions

1. Can the results on cocomparability graphs be extended to AT-free graphs?

Open Questions

1. Can the results on cocomparability graphs be extended to AT-free graphs?
2. (Simple) linear time implementation of LDFS (even for other restricted families of graphs)?

Open Questions

1. Can the results on cocomparability graphs be extended to AT-free graphs?
2. (Simple) linear time implementation of LDFS (even for other restricted families of graphs)?
3. More applications of LDFS and LBFS?

Open Questions

1. Can the results on cocomparability graphs be extended to AT-free graphs?
2. (Simple) linear time implementation of LDFS (even for other restricted families of graphs)?
3. More applications of LDFS and LBFS?
4. Multi-sweep (possibly hybrid) LDFS algorithms with LBFS etc. ?

Open Questions

1. Can the results on cocomparability graphs be extended to AT-free graphs?
2. (Simple) linear time implementation of LDFS (even for other restricted families of graphs)?
3. More applications of LDFS and LBFS?
4. Multi-sweep (possibly hybrid) LDFS algorithms with LBFS etc. ?
5. Other heuristic uses of graph searches?

Open Questions

1. Can the results on cocomparability graphs be extended to AT-free graphs?
2. (Simple) linear time implementation of LDFS (even for other restricted families of graphs)?
3. More applications of LDFS and LBFS?
4. Multi-sweep (possibly hybrid) LDFS algorithms with LBFS etc. ?
5. Other heuristic uses of graph searches?
6. What role does LDFS play with respect to posets? Discussed on Thursday.

Thank you for your attention