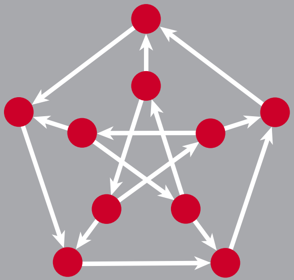


Fast Algorithms for Pseudoarboricity

Meeting on Algorithm Engineering and Experiments – ALENEX 2016

Markus Blumenstock

Institute of Computer Science, University of Mainz

January 10th, 2016

A tree...

...an unrooted tree...

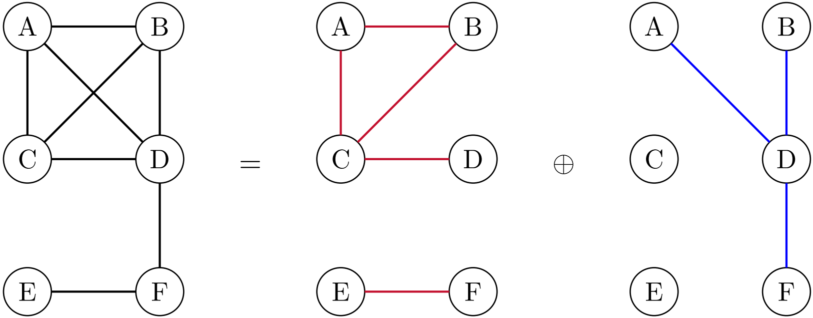
...a pseudotree!

?!

Pseudoarboricity

Definition

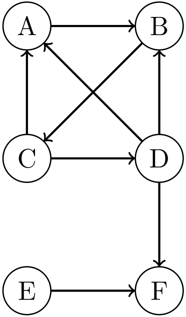
The pseudoarboricity $p(G)$ of an undirected graph G is the minimum number of pseudoforests into which the graph can be decomposed.



Pseudoarboricity and Orientations

Theorem (Frank–Gyárfás 1976 + Picard–Queyranne 1982)

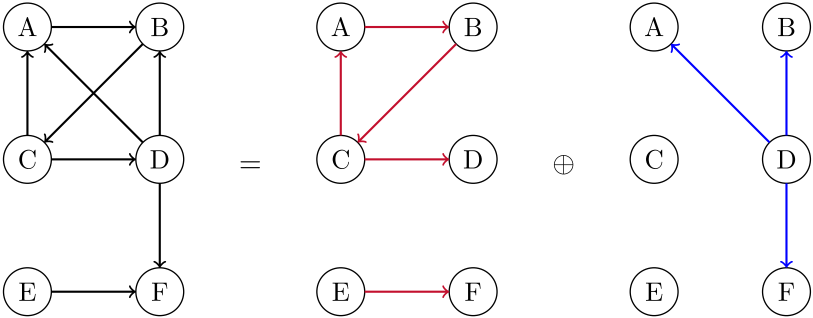
Let $\vec{G}$ be an orientation of G such that the maximum indegree is minimal. Then this indegree equals the pseudoarboricity $p(G)$.



Pseudoarboricity and Orientations

Theorem (Frank–Gyárfás 1976 + Picard–Queyranne 1982)

Let $\vec{G}$ be an orientation of G such that the maximum indegree is minimal. Then this indegree equals the pseudoarboricity $p(G)$.



The 'Re-orientation' Algorithm

- Determine the integer p in a binary search on a search interval; $I = \{0, ..., |V|\}$ always contains p

The 'Re-orientation' Algorithm

- Determine the integer p in a binary search on a search interval; $I = \{0, ..., |V|\}$ always contains p
- For a test value d in the interval, we test whether there is an orientation of the graph such that the maximum indegree is at most d.

JG|U

The ‘Re-orientation’ Algorithm

- Determine the integer p in a binary search on a search interval; $I = \{0, ..., |V|\}$ always contains p
- For a test value d in the interval, we test whether there is an orientation of the graph such that the maximum indegree is at most d.
- The test can be performed with a maximum flow algorithm, which ‘re-orients’ an arbitrary orientation to a d-orientation by reversing directed paths, if possible

JG|U

The 'Re-orientation' Algorithm

- Determine the integer p in a binary search on a search interval; $I = \{0, ..., |V|\}$ always contains p
- For a test value d in the interval, we test whether there is an orientation of the graph such that the maximum indegree is at most d.
- The test can be performed with a maximum flow algorithm, which 're-orients' an arbitrary orientation to a d-orientation by reversing directed paths, if possible
- With Dinitz's algorithm, we need $\mathcal{O}(|E|^{3/2} \log |I|)$ time on a search interval I that contains p

JG|U

Approximating p

- The algorithm can be turned into an approximation scheme which returns d satisfying $p \leq d \leq \lceil (1+\epsilon)p \rceil$ in time

$$\mathcal{O}\left(|E| \, \frac{\log|V|}{\epsilon} \, \log|I|\right)$$

  for $\epsilon > 0$ on a search interval I containing p
- The approximation is not achieved by stopping the binary search early, but by stopping Dinitz's algorithm after $2 + \log_{1+\epsilon}|V|$ phases (Kowalik 2006)

Speeding up Exact Computation – the Idea

- First compute a $(1 + \epsilon)$-approximation, then the exact algorithm can be performed in $\mathcal{O}(\log(\epsilon p))$ tests

Speeding up Exact Computation – the Idea

- First compute a $(1+\epsilon)$-approximation, then the exact algorithm can be performed in $\mathcal{O}(\log(\epsilon p))$ tests
- We set ϵ to balance the runtimes of the approximation and the exact phase

JG|U

Speeding up Exact Computation – the Idea

- First compute a $(1+\epsilon)$-approximation, then the exact algorithm can be performed in $\mathcal{O}(\log(\epsilon p))$ tests
- We set ϵ to balance the runtimes of the approximation and the exact phase
- We will use the fact that always $p \in \mathcal{O}(\sqrt{|E|})$ holds

A Selection of New Results

Bound on p	Runtime bound to compute p
–	$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log\log p}\right)$

5

A Selection of New Results

Bound on p	Runtime bound to compute p
–	$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log\log p}\right)$
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\log^* p\right)$

5

A Selection of New Results

Bound on p	Runtime bound to compute p
–	$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log\log p}\right)$
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\log^* p\right)$
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log^2\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\right)$

5

A Selection of New Results

Bound on p	Runtime bound to compute p	
$\Omega(\sqrt{\lvert E\rvert})$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\right)$	(later)
–	$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log\log p}\right)$	
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\log^* p\right)$	
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log^2\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\right)$	

5

JG|U

A Selection of New Results

Bound on p	Runtime bound to compute p	
$\Omega(\sqrt{\lvert E\rvert})$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\right)$	(later)
–	$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log\log p}\right)$	
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\log^* p\right)$	
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log^2\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\right)$	

Comparison: matroid partitioning algorithm (Westermann 1988):

$$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log p}\right)$$

5

JG|U

How to obtain $\mathcal{O}\left(|E|^{3/2} \log\log|V|\right)$ unconditionally

- Use a $(1+\epsilon)$-approximation to accelerate the exact algorithm!
- Set

$$\epsilon = \frac{(\log|V|)^2}{p}.$$

JG|U

How to obtain $\mathcal{O}\left(|E|^{3/2} \log\log|V|\right)$ unconditionally

- Use a $(1+\epsilon)$-approximation to accelerate the exact algorithm!
- Compute a 2-approximation $\tilde{p}$ of p in $\mathcal{O}(|E|)$. Set

$$\epsilon = \frac{(\log|V|)^2}{\tilde{p}} \leq \frac{(\log|V|)^2}{p}.$$

JG|U

How to obtain $\mathcal{O}\left(|E|^{3/2}\log\log|V|\right)$ unconditionally

- Use a $(1+\epsilon)$-approximation to accelerate the exact algorithm!
- Compute a 2-approximation $\tilde{p}$ of p in $\mathcal{O}(|E|)$. Set

$$\epsilon = \frac{(\log|V|)^2}{\tilde{p}} \leq \frac{(\log|V|)^2}{p}.$$

- The approximation scheme runs in time

$$\mathcal{O}\left(|E|\log|V|\frac{p}{(\log|V|)^2}\log|V|\right)$$

How to obtain $\mathcal{O}\left(|E|^{3/2}\log\log|V|\right)$ unconditionally

- Use a $(1+\epsilon)$-approximation to accelerate the exact algorithm!
- Compute a 2-approximation $\tilde{p}$ of p in $\mathcal{O}(|E|)$. Set

$$\epsilon = \frac{(\log|V|)^2}{\tilde{p}} \leq \frac{(\log|V|)^2}{p}.$$

- The approximation scheme runs in time

$$\mathcal{O}\left(|E|\cancel{\log|V|}\frac{p}{\cancel{(\log|V|)^2}}\cancel{\log|V|}\right) \subseteq \mathcal{O}(|E|^{3/2})$$

JG|U

How to obtain $\mathcal{O}\left(|E|^{3/2}\log\log|V|\right)$ unconditionally

- Use a $(1+\epsilon)$-approximation to accelerate the exact algorithm!
- Compute a 2-approximation $\tilde{p}$ of p in $\mathcal{O}(|E|)$. Set

$$\epsilon = \frac{(\log|V|)^2}{\tilde{p}} \leq \frac{(\log|V|)^2}{p}.$$

- The approximation scheme runs in time

$$\mathcal{O}\left(|E|\cancel{\log|V|}\cancel{\frac{p}{(\log|V|)^2}}\cancel{\log|V|}\right) \subseteq \mathcal{O}(|E|^{3/2})$$

- A binary search in an exact algorithm needs $\mathcal{O}(\log(\epsilon p))$ tests on the narrowed search interval – the runtime is

$$\mathcal{O}(|E|^{3/2}\log\log^{\cancel{2}}|V|).$$

Sketch: Obtaining $\mathcal{O}\left(|E|^{3/2}\log^* p\right)$ if $p \in \mathcal{O}(\sqrt{|E|}/\log|V|)$

Sketch: Obtaining $\mathcal{O}\left(|E|^{3/2} \log^* p\right)$ if $p \in \mathcal{O}(\sqrt{|E|}/ \log |V|)$

- Compute a 2-approximation of p in $\mathcal{O}(|E|)$.
- Run the approximation scheme $i = 1, ..., k$ times:

$$\epsilon_1 \simeq \frac{\log p}{p},\ \epsilon_2 \simeq \frac{\log \log p}{p},\ \epsilon_3 \simeq \frac{\log \log \log p}{p},\ ...,\ \epsilon_k \simeq \frac{\log^{\times k} p}{p}$$

Sketch: Obtaining $\mathcal{O}\left(|E|^{3/2} \log^* p\right)$ if $p \in \mathcal{O}(\sqrt{|E|}/\log|V|)$

- Compute a 2-approximation of p in $\mathcal{O}(|E|)$.
- Run the approximation scheme $i = 1, ..., k$ times:

$$\epsilon_1 \simeq \frac{\log p}{p},\ \epsilon_2 \simeq \frac{\log\log p}{p},\ \epsilon_3 \simeq \frac{\log\log\log p}{p},\ \dots,\ \epsilon_k \simeq \frac{\log^{\times k} p}{p}$$

- The initial interval size is $|I_0| \in \mathcal{O}(p)$ with the 2-approximation.

JG|U

Sketch: Obtaining $\mathcal{O}\left(|E|^{3/2}\log^* p\right)$ if $p \in \mathcal{O}(\sqrt{|E|}/\log|V|)$

- Compute a 2-approximation of p in $\mathcal{O}(|E|)$.
- Run the approximation scheme $i = 1, ..., k$ times:

$$\epsilon_1 \simeq \frac{\log p}{p},\ \epsilon_2 \simeq \frac{\log\log p}{p},\ \epsilon_3 \simeq \frac{\log\log\log p}{p},\ ...,\ \epsilon_k \simeq \frac{\log^{\times k} p}{p}$$

- The initial interval size is $|I_0| \in \mathcal{O}(p)$ with the 2-approximation.
- In every phase $i = 1, ..., k$, we reduce the interval to size

$$|I_i| \in \mathcal{O}\left(\log^{\times i} p\right).$$

JG|U

Sketch: Obtaining $\mathcal{O}\left(|E|^{3/2}\log^* p\right)$ if $p \in \mathcal{O}\left(\sqrt{|E|}/\log|V|\right)$

- The runtime of the i-th approximation phase with $\epsilon_i = \frac{\log^{\times i} p}{p}$ is $\mathcal{O}\left(|E|^{3/2}\right)$

Sketch: Obtaining $\mathcal{O}\left(|E|^{3/2}\log^* p\right)$ if $p \in \mathcal{O}\left(\sqrt{|E|}/\log|V|\right)$

- The runtime of the i-th approximation phase with $\epsilon_i = \frac{\log^{\times i} p}{p}$ is $\mathcal{O}\left(|E|^{3/2}\right)$
- Run $k = \log^*(p) - 1$ approximation phases
- Run the exact algorithm on the remaining constant-size interval in $\mathcal{O}\left(|E|^{3/2}\right)$ time

Sketch: Obtaining $\mathcal{O}\left(|E|^{3/2}\log^* p\right)$ if $p \in \mathcal{O}\left(\sqrt{|E|}/\log|V|\right)$

- The runtime of the i-th approximation phase with $\epsilon_i = \frac{\log^{\times i} p}{p}$ is $\mathcal{O}\left(|E|^{3/2}\right)$
- Run $k = \log^*(p) - 1$ approximation phases
- Run the exact algorithm on the remaining constant-size interval in $\mathcal{O}\left(|E|^{3/2}\right)$ time
- The total runtime is $\mathcal{O}\left(|E|^{3/2}\log^* p\right)$

What have we shown?

Bound on p	Runtime bound to compute p	
–	$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log\log p}\right)$	($\checkmark$)
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\log^* p\right)$	$\checkmark$

Cmp. Westermann 1988: $\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log p}\right)$

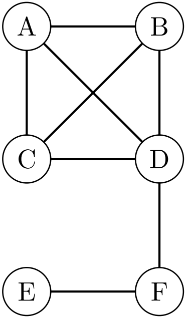
JG|U

What have we shown?

Bound on p	Runtime bound to compute p	
$\Omega(\sqrt{\lvert E\rvert})$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\right)$	up next
–	$\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log\log p}\right)$	(✓)
$\mathcal{O}\left(\frac{\sqrt{\lvert E\rvert}}{\log\lvert V\rvert}\right)$	$\mathcal{O}\left(\lvert E\rvert^{3/2}\log^{*} p\right)$	✓

Cmp. Westermann 1988: $\mathcal{O}\left(\lvert E\rvert^{3/2}\sqrt{\log p}\right)$

The Densest Subgraph Problem

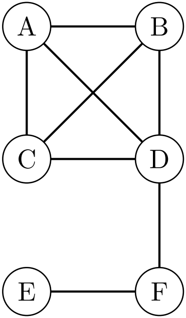


Definition

Let $G = (V, E)$ be an undirected graph. The maximum density is defined as

$$d^*(G) := \max_{H \subseteq G} \frac{|E_H|}{|V_H|}.$$

The Densest Subgraph Problem



Definition

Let $G = (V, E)$ be an undirected graph. The maximum density is defined as

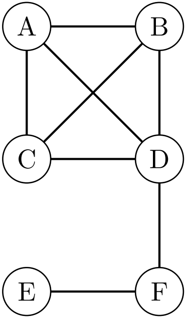
$$d^*(G) := \max_{H \subseteq G} \frac{|E_H|}{|V_H|}.$$

Observation (Khuller and Saha 2009)

A vertex v with $\deg(v) < d^*(G)$ cannot be in a densest subgraph.

JG|U

The Densest Subgraph Problem



Definition

Let $G = (V, E)$ be an undirected graph. The maximum density is defined as

$$d^*(G) := \max_{H \subseteq G} \frac{|E_H|}{|V_H|}.$$

Observation (Khuller and Saha 2009)

A vertex v with $\deg(v) < d^*(G)$ cannot be in a densest subgraph.

Theorem (Picard–Queyranne 1982)

For any undirected graph G, we have $\lceil d^*(G) \rceil = p$.

JG|U

Preprocessing

Observation (Khuller and Saha 2009)

A vertex v with $\deg(v) < d^*(G)$ *cannot be in a densest subgraph.*

JG|U

Preprocessing

Observation (Khuller and Saha 2009)

A vertex v with $\deg(v) < d^*(G)$ cannot be in a densest subgraph.

- Compute a lower bound $d \leq d^*(G)$

JG|U

Preprocessing

Observation (Khuller and Saha 2009)

A vertex v with $\deg(v) < d^*(G)$ cannot be in a densest subgraph.

- Compute a lower bound $d \leq d^*(G)$
- Remove all vertices whose degree is less than d (repeatedly)

JG|U

Preprocessing

Observation (Khuller and Saha 2009)

A vertex v with $\deg(v) < d^*(G)$ cannot be in a densest subgraph.

- Compute a lower bound $d \leq d^*(G)$
- Remove all vertices whose degree is less than d (repeatedly)
- For the resulting graph $G' = (V', E')$, we have $d^*(G') = d^*(G)$ and thus $p(G') = p(G)$ by the Picard–Queyranne theorem

JG|U

Preprocessing

Observation (Khuller and Saha 2009)

A vertex v with $\deg(v) < d^*(G)$ cannot be in a densest subgraph.

- Compute a lower bound $d \leq d^*(G)$
- Remove all vertices whose degree is less than d (repeatedly)
- For the resulting graph $G' = (V', E')$, we have $d^*(G') = d^*(G)$ and thus $p(G') = p(G)$ by the Picard–Queyranne theorem
- G' is possibly smaller (great!) and has a higher average density, i.e.

$$\frac{|E'|}{|V'|} \geq \frac{|E|}{|V|}.$$

Preprocessing – A Theoretical Guarantee

- Utilize Gabow's algorithm, which runs in $\mathcal{O}\left(|E|^{3/2} \log \frac{|V|^2}{|E|}\right)$ time

Preprocessing – A Theoretical Guarantee

- Utilize Gabow's algorithm, which runs in $\mathcal{O}\left(|E|^{3/2}\log\frac{|V|^2}{|E|}\right)$ time

JG|U

Preprocessing – A Theoretical Guarantee

- Utilize Gabow's algorithm, which runs in $\mathcal{O}\left(|E|^{3/2}\log\frac{|V|^2}{|E|}\right)$ time

Proposition (B.)

If $d^*(G) \in \Omega(\sqrt{|E|})$, then we can obtain a subgraph $G' = (V', E')$ with $|E'| \in \Theta(|V'|^2)$ and $d^*(G') = d^*(G)$ in linear time.

Preprocessing – A Theoretical Guarantee

- Utilize Gabow's algorithm, which runs in $\mathcal{O}\left(|E|^{3/2} \log \frac{|V|^2}{|E|}\right)$ time

Proposition (B.)

If $d^*(G) \in \Omega(\sqrt{|E|})$, then we can obtain a subgraph $G' = (V', E')$ with $|E'| \in \Theta(|V'|^2)$ and $d^*(G') = d^*(G)$ in linear time.

Corollary

If $d^*(G) \in \Omega(\sqrt{|E|})$, p can be determined in $\mathcal{O}(|E|^{3/2})$.

The Re-orientation Algorithm in Practice (Dinitz's algorithm)

		Without preprocessing		With preprocessing	
Graph	p	$\|E\|$	Runtime [s]	$\|E'\|$	Runtime [s]
Amazon	5	900K	23	800K	1
DBLP	57	1M	4	14K	0
YouTube	46	3M	4	417K	0
LiveJournal	194	35M	251	540K	0
Orkut	228	117M	1012	13M	33

Note: Push-relabel algorithms are slower by an order of magnitude

Open Questions

- Do other problems exist where we can speed up computation with an approximation scheme?

Open Questions

- Do other problems exist where we can speed up computation with an approximation scheme?
- How much are certain graph families, e.g. power-law graphs, reduced by preprocessing?

JG|U

Open Questions

- Do other problems exist where we can speed up computation with an approximation scheme?
- How much are certain graph families, e.g. power-law graphs, reduced by preprocessing?
- A 2-approximation of p (and d^*) can be found in $\mathcal{O}(|E|)$ time with a greedy algorithm. Is a factor smaller than 2 possible in linear time?
  (Any constant-factor approximation can be found in $\mathcal{O}(|E| \log |V| \log p)$ time)

Thank you for your attention!