Constructing Large k-cores in Low Degeneracy Graphs

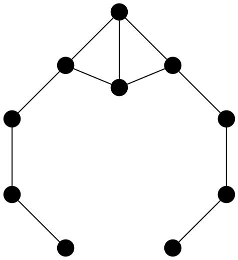
Fedor Fomin[1] Danil Sagunov[2] **Kirill Simonov**[1]

[1]University of Bergen

[2]St. Petersburg Department of Steklov Mathematical Institute

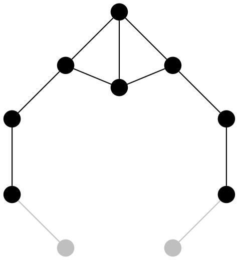
Bergen, 24th January 2020

Social network unraveling



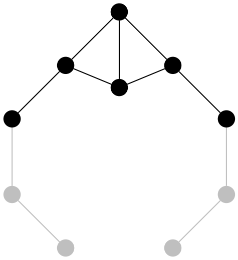
- A user is active if at least k of their connections are active

Social network unraveling



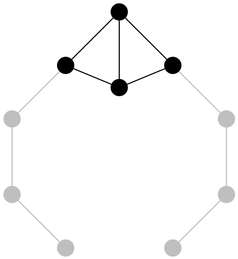
- A user is active if at least k of their connections are active

Social network unraveling



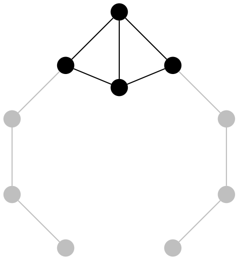
- A user is active if at least k of their connections are active

Social network unraveling



- A user is active if at least k of their connections are active

Social network unraveling



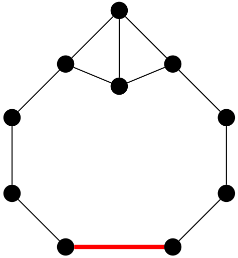
- A user is active if at least k of their connections are active
- The k-core is the maximal induced subgraph where degree of each vertex is at least k

Strengthening a network

- Want to prevent the unraveling

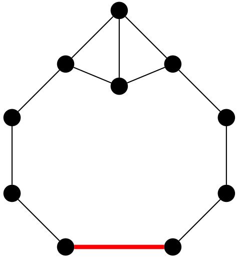
Strengthening a network

- Want to prevent the unraveling
- Create connections to obtain a large *k*-core



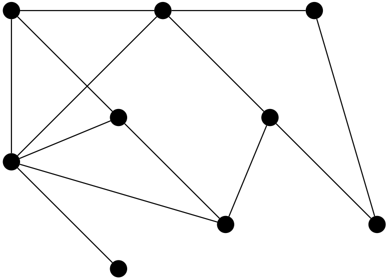
Strengthening a network

- Want to prevent the unraveling
- Create connections to obtain a large k-core

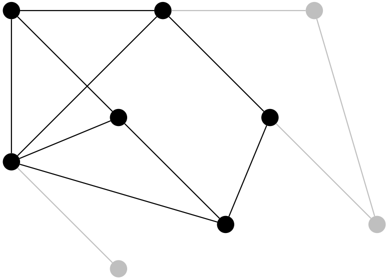


- EDGE k-CORE : Can we add at most b edges so that the k-core size is at least p?

Example

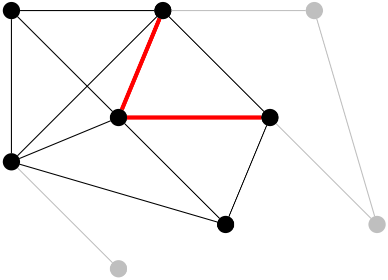


Example



- Fix the vertex set of the 3-core H of size at least $p = 6$

Example



- Fix the vertex set of the 3-core H of size at least $p = 6$
- Add at most $b = 2$ edges inside H so that degrees are $\geq k = 3$

Previous work on EDGE k-CORE

[Chitnis and Talmon 2018]

- NP-complete when $k = 3$, even for 2-degenerate graphs

Previous work on Edge k-Core

[Chitnis and Talmon 2018]

- NP-complete when $k = 3$, even for 2-degenerate graphs
- W[1]-hard parameterized by $k + b + p$ (no $f(k + b + p) \cdot n^{\mathcal{O}(1)}$ algorithm)

Previous work on Edge k-Core

[Chitnis and Talmon 2018]

- NP-complete when $k = 3$, even for 2-degenerate graphs
- W[1]-hard parameterized by $k + b + p$ (no $f(k + b + p) \cdot n^{\mathcal{O}(1)}$ algorithm)
- Solvable in time $(k + \mathrm{tw})^{\mathcal{O}(\mathrm{tw}+b)} \cdot n^{\mathcal{O}(1)}$ when treewidth is tw

Previous work on EDGE k-CORE

[Chitnis and Talmon 2018]

- NP-complete when $k = 3$, even for 2-degenerate graphs
- W[1]-hard parameterized by $k + b + p$ (no $f(k + b + p) \cdot n^{\mathcal{O}(1)}$ algorithm)
- Solvable in time $(k + \mathrm{tw})^{\mathcal{O}(\mathrm{tw}+b)} \cdot n^{\mathcal{O}(1)}$ when treewidth is tw
- [Zhou et al. 2019] APX-hard to maximize p

Our results on Edge k-Core

- $\mathcal{O}(k \cdot |V(G)|^2)$ when G is a forest

Our results on Edge k-Core

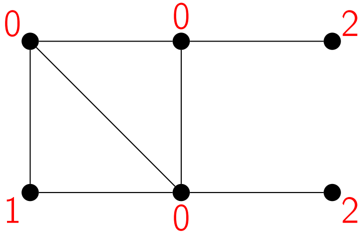
- $\mathcal{O}(k \cdot |V(G)|^2)$ when G is a forest
- FPT parameterized by $\mathrm{tw} + k$

Our results on EDGE k-CORE

- $\mathcal{O}(k \cdot |V(G)|^2)$ when G is a forest
- FPT parameterized by $\mathrm{tw} + k$
  Compared to $\mathrm{tw} + k + b$ by Chitnis and Talmon
- FPT parameterized by vertex cover, k is arbitrary
  There is no poly kernel parameterized by $\mathrm{vc} + k + b + p$, unless $\mathrm{PH} = \Sigma_p^3$

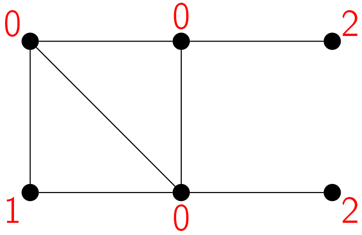
Deficiency

- $\mathrm{df}_G(v) = \max\{0, k - \deg_G(v)\}$, the *deficiency* of v in G: How many edges do we lack in each vertex



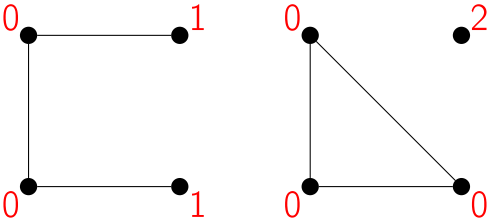
Deficiency

- $\text{df}_G(v) = \max\{0, k - \deg_G(v)\}$, the *deficiency* of v in G: How many edges do we lack in each vertex

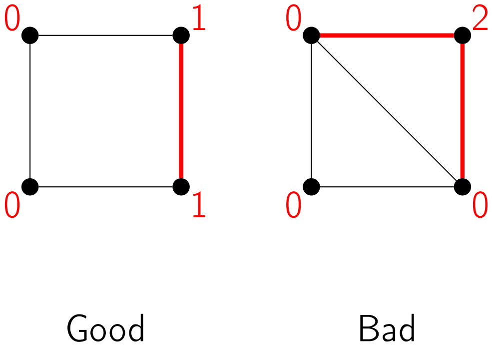


- The *total* deficiency $\text{df}(G) = \sum_{v \in V(G)} \text{df}_G(v)$
  We need at least $\lceil \text{df}(G)/2 \rceil$ edges

Good and bad edges

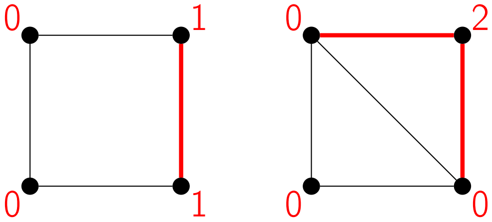


Good and bad edges



- A good edge lowers deficiency by 2, a bad by 1

Good and bad edges



Good Bad

- A good edge lowers deficiency by 2, a bad by 1
- Nice when G could be completed *optimally*, using exactly $\lceil \mathrm{df}(G)/2 \rceil$ edges

Forests

Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$edges, and in the case $k \geq 4$ these edges form a connected subgraph.

Forests

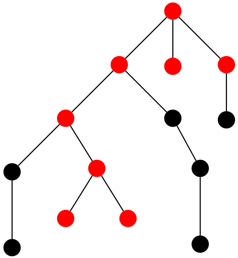
Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$edges, and in the case $k \geq 4$ these edges form a connected subgraph.

- Allows to keep track of deficiency only
- Dynamic programming in time $\mathcal{O}(k \cdot |V(T)|^2)$

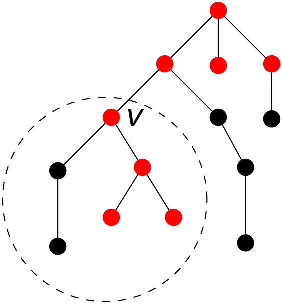
Dynamic Programming

- Find $H \subset V(T)$ s.t.
  $|H| \geq p$, $\mathrm{df}(G[H]) \leq 2b$



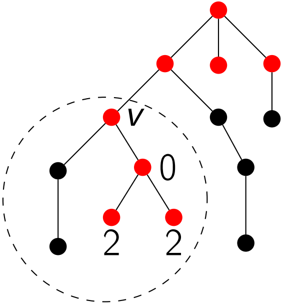
Dynamic Programming

- Find $H \subset V(T)$ s.t. $|H| \geq p$, $\mathrm{df}(G[H]) \leq 2b$
- DP on subtrees



Dynamic Programming

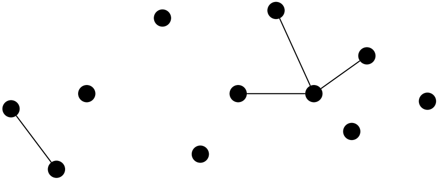
- Find $H \subset V(T)$ s.t. $|H| \geq p$, $\mathrm{df}(G[H]) \leq 2b$
- DP on subtrees
- Store
    - how many vertices taken inside,
    - their total deficiency,
    - whether the root is taken and how many neighbors of the root are taken inside



Proof of the Theorem: Warmup, $k = 1$

Theorem

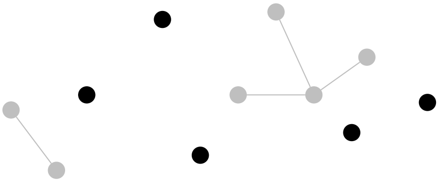
For any k, any forest T on $\geq k + 1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$edges, and in the case $k \geq 4$ these edges form a connected subgraph.



Proof of the Theorem: Warmup, $k = 1$

Theorem

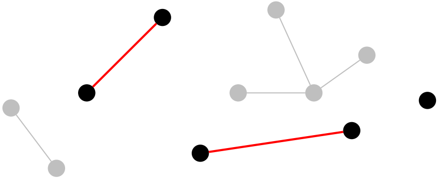
For any k, any forest T on $\geq k + 1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$ edges, and in the case $k \geq 4$ these edges form a connected subgraph.



Proof of the Theorem: Warmup, $k = 1$

Theorem

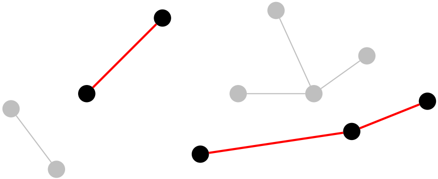
For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$ edges, and in the case $k \geq 4$ these edges form a connected subgraph.



Proof of the Theorem: Warmup, $k = 1$

Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$ edges, and in the case $k \geq 4$ these edges form a connected subgraph.

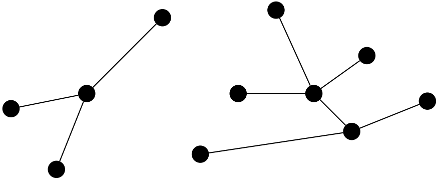


Proof of the Theorem

Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$ edges, and in the case $k \geq 4$ these edges form a connected subgraph.

- Enough to consider trees

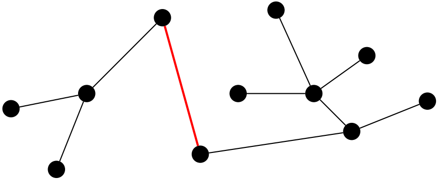


Proof of the Theorem

Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$edges, and in the case $k \geq 4$ these edges form a connected subgraph.

- Enough to consider trees

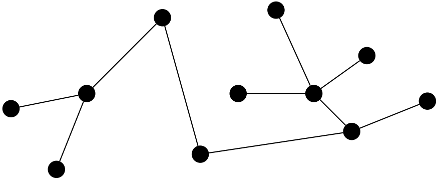


Proof of the Theorem

Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$ edges, and in the case $k \geq 4$ these edges form a connected subgraph.

- Enough to consider trees



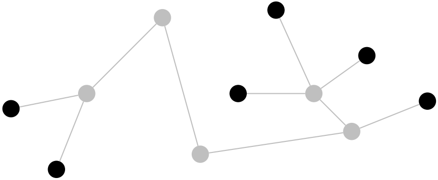
- $k = 2$: Arbitrary matching on the leaves

Proof of the Theorem

Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$edges, and in the case $k \geq 4$ these edges form a connected subgraph.

- Enough to consider trees



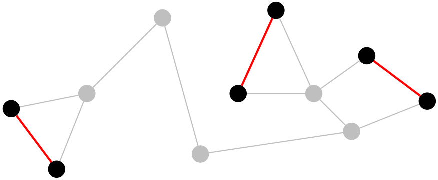
- $k = 2$: Arbitrary matching on the leaves

Proof of the Theorem

Theorem

For any k, any forest T on $\geq k+1$ vertices can be completed to a graph of minimum degree k by adding at most $\lceil \mathrm{df}(T)/2 \rceil$ edges, and in the case $k \geq 4$ these edges form a connected subgraph.

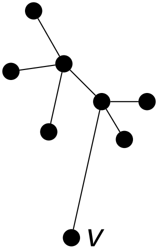
- Enough to consider trees



- $k = 2$: Arbitrary matching on the leaves

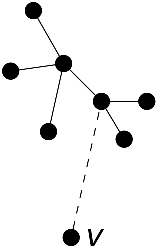
Proof of the Theorem

- Proof by induction on $|T|$
- Base case: $|T| = k + 1$, complete to a clique



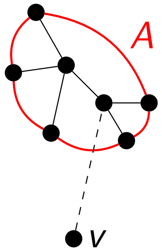
Proof of the Theorem

- Proof by induction on $|T|$
- Base case: $|T| = k + 1$, complete to a clique
- Step: Take away a leaf v



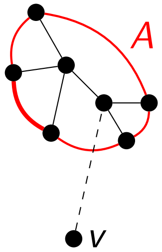
Proof of the Theorem

- Proof by induction on $|T|$
- Base case: $|T| = k + 1$, complete to a clique
- Step: Take away a leaf v
- A is the graph of added edges



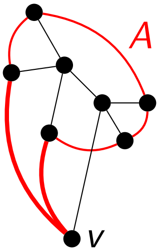
Proof of the Theorem

- Proof by induction on $|T|$
- Base case: $|T| = k + 1$, complete to a clique
- Step: Take away a leaf v
- A is the graph of added edges
- Find a large matching in A, roughly $k/2$

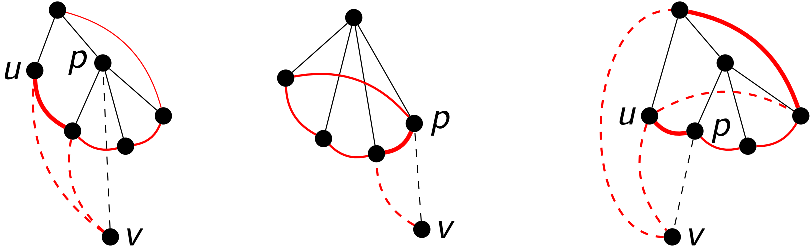


Proof of the Theorem

- Proof by induction on $|T|$
- Base case: $|T| = k + 1$, complete to a clique
- Step: Take away a leaf v
- A is the graph of added edges
- Find a large matching in A, roughly $k/2$
- Reroute to v



Cases for rerouting



Theorem (Henning and Yeo, 2018)

For any integer $t \geq 3$, any connected graph G with $|V(G)| = n$, $|E(G)| = m$ and $\Delta(G) \leq t$, contains a matching of size at least

$$\left(\frac{t-1}{t(t^2-3)}\right) n + \left(\frac{t^2-t-2}{t(t^2-3)}\right) m - \frac{t-1}{t(t^2-3)},$$

if t is odd, or at least

$$\frac{n}{t(t+1)} + \frac{m}{t+1} - \frac{1}{t}, \text{ if } t \text{ is even.}$$

Large deficiency is optimal

Lemma

For any integer $k \geq 2$, any graph G with $\mathrm{df}(G) \geq 3k^3$ can be completed to a graph of minimum degree k optimally.

Large deficiency is optimal

Lemma

For any integer $k \geq 2$, any graph G with $\mathrm{df}(G) \geq 3k^3$ can be completed to a graph of minimum degree k optimally.

- Connect arbitrarily two vertices with non-zero deficiency
- When a vertex v is left, can replace (u, w) by (u, v) and (w, v) if u and w are not neighbors of v
- $\deg(v) \leq k$, so there are many enough edges among the added

Treewidth algorithm

- If $b \leq 3k^3$, run the $(k + \mathrm{tw})^{\mathcal{O}(\mathrm{tw}+b)} \cdot n^{\mathcal{O}(1)}$ algorithm

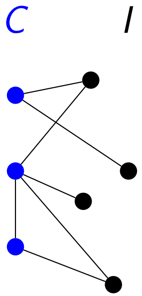
Treewidth algorithm

- If $b \leq 3k^3$, run the $(k + \mathrm{tw})^{\mathcal{O}(\mathrm{tw}+b)} \cdot n^{\mathcal{O}(1)}$ algorithm
- If $b > 3k^3$, use DP on tree decomposition to find large enough H with the smallest deficiency d

Treewidth algorithm

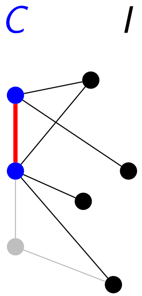
- If $b \leq 3k^3$, run the $(k + \mathrm{tw})^{\mathcal{O}(\mathrm{tw}+b)} \cdot n^{\mathcal{O}(1)}$ algorithm
- If $b > 3k^3$, use DP on tree decomposition to find large enough H with the smallest deficiency d
- If $b \geq \lceil d/2 \rceil$, we report YES by Lemma
- Otherwise report NO since the smallest deficiency is too large

EDGE k-CORE parameterized by vc



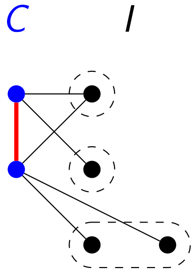
EDGE k-CORE parameterized by vc

- Fix vertices and edges of C



EDGE k-CORE parameterized by vc

- Fix vertices and edges of C
- I is partitioned into classes by edges to C



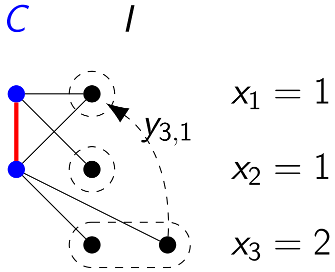
$x_1 = 1$

$x_2 = 1$

$x_3 = 2$

EDGE k-CORE parameterized by vc

- Fix vertices and edges of C
- I is partitioned into classes by edges to C
- ILP, introduce variables $y_{d,d'}$ for the number of vertices going from d to d' after adding edges to C



- Edges from I to C are fixed, additionally fix the bad edges
- For each deficiency $d \in [k - |C|, k]$ we have a variable for the number of corresponding vertices
- A modification of the Erdős-Gallai theorem verifies whether there exists a graph with these degrees

Theorem (Erdős and Gallai, 1960)

A sequence of non-negative integers $d_1 \geq d_2 \geq \ldots \geq d_n$ is graphic if and only if $\sum_{i=1}^{n} d_i$ is even and for each $t \in [n]$ holds

$$\sum_{i=1}^{t} d_i \leq t \cdot (t-1) + \sum_{j=t+1}^{n} \min\{d_j, t\}.$$

Open quiestions

- Could EDGE k-CORE be solved efficiently on other graph classes?

Open quiestions

- Could Edge k-Core be solved efficiently on other graph classes?
- Is there other meaningful ways to enforce a large k-core?

Open quiestions

- Could Edge k-Core be solved efficiently on other graph classes?
- Is there other meaningful ways to enforce a large k-core?

Thanks for attention!