

# Preventing Unraveling in Social Networks: The Anchored $k$ -Core Problem

Kshipra Bhawalkar<sup>1</sup>   Jon Kleinberg<sup>2</sup>   **Kevin Lewi**<sup>1</sup>  
Tim Roughgarden<sup>1</sup>   Aneesh Sharma<sup>3</sup>

<sup>1</sup>Stanford University

<sup>2</sup>Cornell University

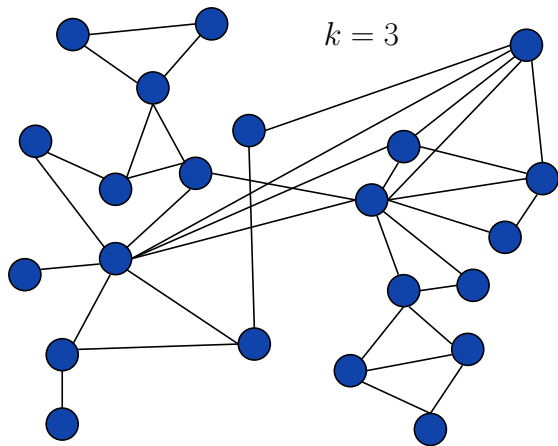
<sup>3</sup>Twitter, Inc.

ICALP 2012

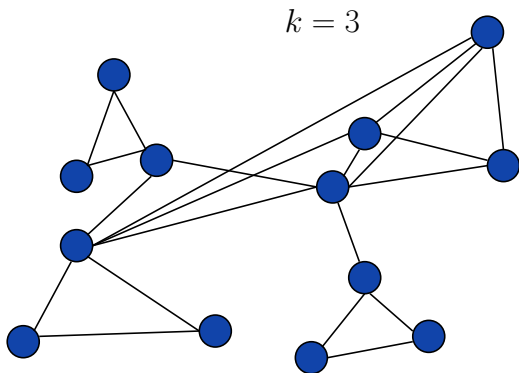
- 1 The Problem
- 2 Easy Cases
- 3 Hardness of Approximation
- 4 Graphs with Bounded Treewidth
- 5 Open Problems

- The  $k$ -core of a graph = the maximal induced subgraph where all vertices have degree at least  $k$ .
- Computing the  $k$ -core of a graph is easy
- for example, suppose  $k = 3$ , and  $G$  looks like this:

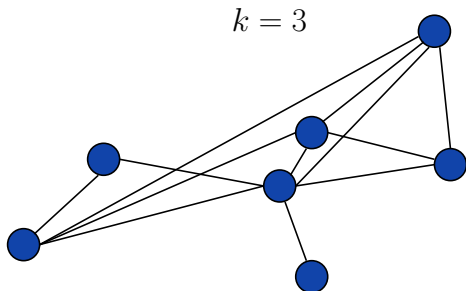
# An Example



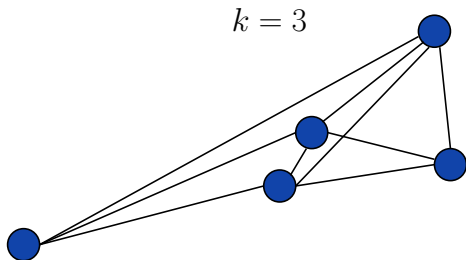
# An Example



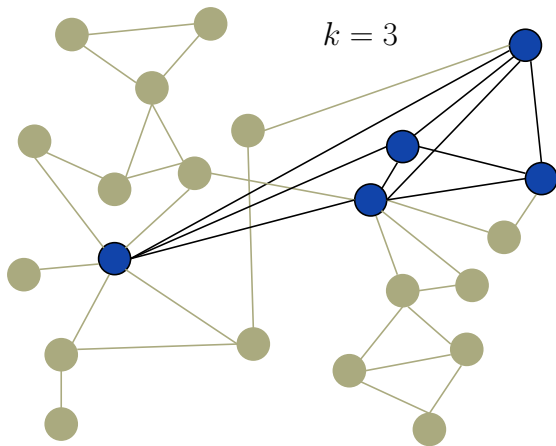
# An Example



# An Example



# An Example





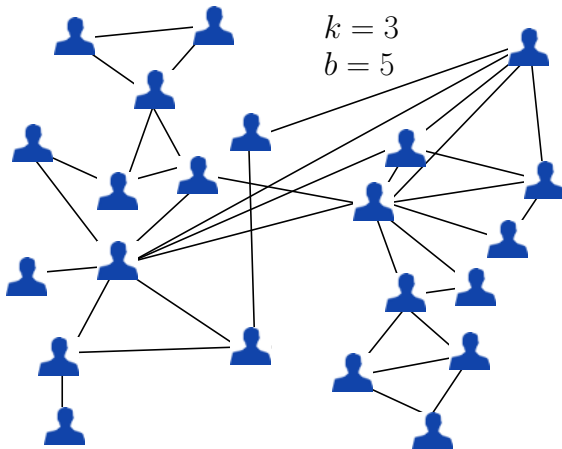


- Suppose I know that people will pay attention to the talk as long as they have at least 3 other friends who are also paying attention

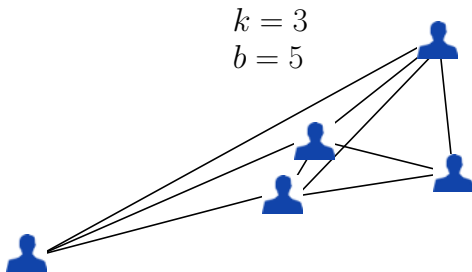
- Suppose I know that people will pay attention to the talk as long as they have at least 3 other friends who are also paying attention
- My goal is to maximize the number of people paying attention

- Suppose I know that people will pay attention to the talk as long as they have at least 3 other friends who are also paying attention
- My goal is to maximize the number of people paying attention
- Also, assume that I have \$50 to spare.

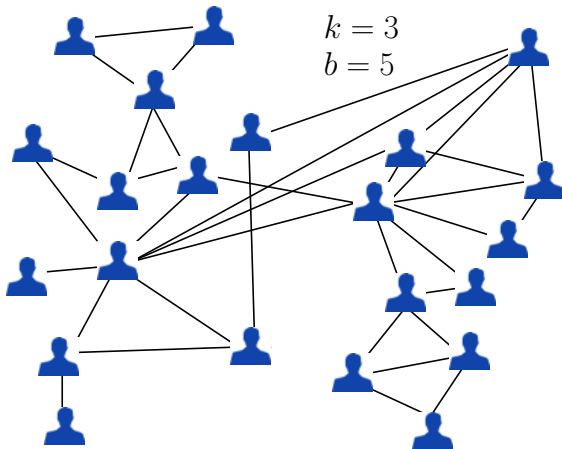
# Another Example



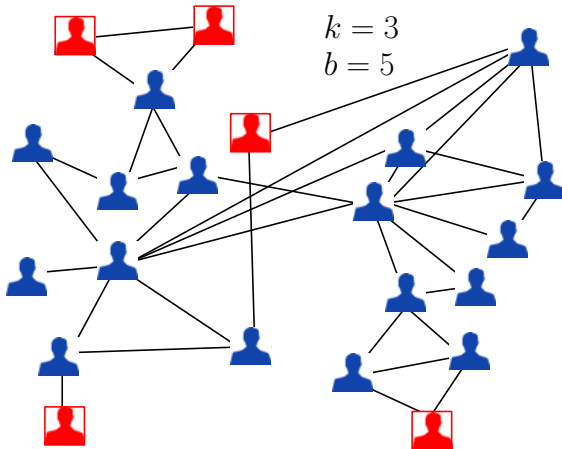
# Another Example



# Another Example

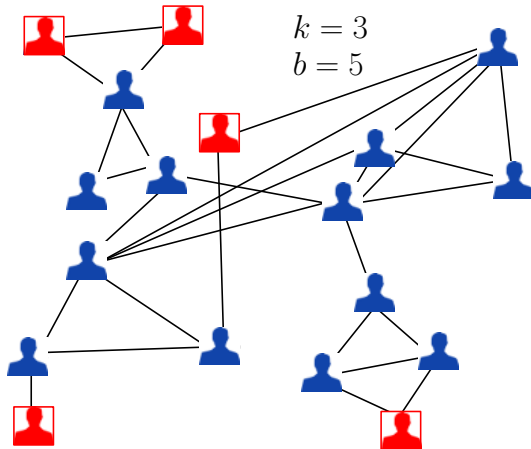


# Another Example

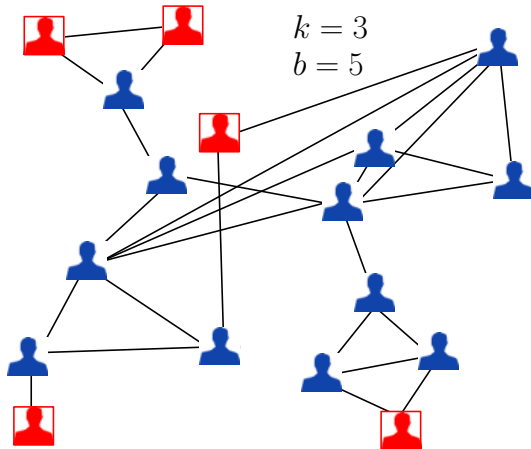




# Another Example



# Another Example



# Anchored $k$ -Core with budget $b$

- the maximum induced subgraph in which every vertex either has degree at least  $k$ , or is anchored
- Goal: Find a set  $S$ ,  $|S| = b$ , such that placing anchors on the nodes of  $S$  maximizes the number of vertices that remain.
- Fact: there is a simple algorithm that runs in time  $O(n^b)$
- Fact:  $k = 0$  and  $k = 1$  are trivial

- What happens for  $k = 2$ ?

- What happens for  $k = 2$ ?
- Observation: Every vertex that lies on a cycle already belongs to the 2-core.

- What happens for  $k = 2$ ?
- Observation: Every vertex that lies on a cycle already belongs to the 2-core.
- Algorithm: take longest paths (runs in time  $O(n^2)$ )

- What happens for  $k = 2$ ?
- Observation: Every vertex that lies on a cycle already belongs to the 2-core.
- Algorithm: take longest paths (runs in time  $O(n^2)$ )
- Can be more clever (runs in time  $O(m + n \log n)$ )

- What happens for  $k = 3$ ?



- What happens for  $k = 3$ ?
- Cannot simplify graph into a forest in this case

- What happens for  $k = 3$ ?
- Cannot simplify graph into a forest in this case
- In fact, we show that for  $k \geq 3$ , the problem is difficult.

- Fact:  $b \leq \text{solution quality} \leq n$
- So, any algorithm obtains an  $n/b$  approximation...

- Fact:  $b \leq \text{solution quality} \leq n$
- So, any algorithm obtains an  $n/b$  approximation...
- Theorem: It is NP-hard to obtain an  $O(n^{1-\epsilon})$  approximation for  $k \geq 3$ .

# Reduction from Set Cover

- Recall Set Cover:  $n$  sets,  $m$  elements
- covering  $< m$  elements is just like covering 0 elements
- this property is ensured by having lots of cycles in the graph
- it's too difficult to figure out how to cover all  $m$  elements (because set cover is NP-hard)

- The treewidth of a graph is a measure of how closely a graph resembles a tree
- Trees have treewidth 1, and cliques on  $n$  vertices have treewidth  $n - 1$

- For graphs of treewidth  $w$ , we have an  $O((3(k+1)^2)^w \cdot b^2 \cdot \text{poly}(n))$ -time exact algorithm
- given graph  $G$ , decomposes it into a tree
- runs a dynamic programming procedure on the resulting tree

# Generalizations

- weighted budget costs
- weighted rewards
- directed edges
- variable thresholds
- variations on the objective function



# Summary and Open Problems

- In general, anchored  $k$ -core is very difficult, but as soon as there is some structure to the graph, it becomes easier
- PTAS for graphs which exclude fixed minors (planar, etc.)?
- Can it be analyzed for random graphs?

## Resource Augmentation:

- compare us against the optimal solution using  $b$  anchors
- But, allow ourselves to use  $2b$  anchors

## Resource Augmentation:

- compare us against the optimal solution using  $b$  anchors
- But, allow ourselves to use  $b \cdot \log n$  anchors

## Resource Augmentation:

- compare us against the optimal solution using  $b$  anchors
- But, allow ourselves to use  $b \cdot \text{polylog} n$  anchors

## Resource Augmentation:

- compare us against the optimal solution using  $b$  anchors
- But, allow ourselves to use  $b \cdot \text{polylog}n$  anchors
- Still NP-hard to get an  $O(n^{1-\epsilon})$  approximation!

# Parameterization by the budget

- Is anyone here a fan of FPT?
- Recall that  $O(n^b)$  is easy to obtain. Can we get  $O(f(b) \cdot \text{poly}(n))$ ?

# Parameterization by the budget

- Is anyone here a fan of FPT?
- Recall that  $O(n^b)$  is easy to obtain. Can we get  $O(f(b) \cdot \text{poly}(n))$ ?
- **No**. The problem is  $W[2]$ -hard with respect to  $b$  (via Dominating Set).