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

Kshipra Bhawalkar[1], Jon Kleinberg[2], Kevin Lewi[1], Tim Roughgarden[1], and Aneesh Sharma[3]

[1] Stanford University, Stanford, CA, USA
[2] Cornell University, Ithaca, NY, USA
[3] Twitter, Inc.

Abstract. We consider a model of user engagement in social networks, where each player incurs a cost to remain engaged but derives a benefit proportional to the number of engaged neighbors. The natural equilibrium of this model corresponds to the *k-core* of the social network — the maximal induced subgraph with minimum degree at least k.
We study the problem of "anchoring" a small number of vertices to maximize the size of the corresponding anchored k-core — the maximal induced subgraph in which every non-anchored vertex has degree at least k. This problem corresponds to preventing "unraveling" — a cascade of iterated withdrawals. We provide polynomial-time algorithms for general graphs with $k = 2$, and for bounded-treewidth graphs with arbitrary k. We prove strong inapproximability results for general graphs and $k \geq 3$.

1 Introduction

A defining property of social networks — where nodes represent individuals, and edges represent friendships — is that the behavior of an individual is influenced by that of his or her friends. In particular, they often exhibit positive "network effects", where the utility of an individual is increasing in the number of friends that behave in a certain way. For example, empirical work has determined that individuals are more likely to contribute useful content to a social network if their friends do [4]. Increasingly, empirical studies suggest that the influence of interactions in social network extends to behavior outside of these networks, as well [9]. An obvious question, studied from a system-building perspective in [10], is how to design or modify social networks to maximize the participation and engagement of its users.

For concreteness, consider scenarios where each individual of a social network has two strategies, to "engage" or to "drop out". Being engaged could mean contributing to a public good (like network content), signing up for a new social network feature, adopting one technology instead of another, and so on. We assume that a player is more likely to be engaged if many friends are. For this Introduction, we focus on our most basic model. We first describe our model via a process of cascading withdrawals, and then formulate it using a simultaneous-move game.

Assume that all individuals are initially engaged, and for a parameter k, a node remains engaged if and only if at least k friends are engaged. For example, engagement could represent active participation in the social network, which is worthwhile to an individual if and only if at least k friends are also actively participating. Or, dropping out could represent the abandonment of an incumbent product in favor of a newly arrived competitor; when the number of one's friends using the old product falls below k, one switches to the new product.

In this basic model, it is clear that all individuals with less than k friends will drop out. These initial withdrawals can be contagious, spreading to individuals with many more than k friends. See Figure 1 for an example of this phenomenon. In general, when such iterated withdrawals die out, the remaining engaged individuals correspond to a well-known concept in graph theory — the *k-core* of the original social network, which by definition is the (unique) maximal induced subgraph with minimum degree at least k. Alternatively, the k-core is the (unique) result of iteratively deleting nodes that have degree less than k, in any order.

Schelling [17, P.214] describes this type of "unraveling" in typically picturesque language, by contrasting the cycle with the line (with $k = 2$). He imagines people sitting with reading lamps, each of whom can get additional partial illumination from the lamps of their neighbor(s):

> *In some cases the arrangement matters. If everybody needs 100 watts to read by and a neighbor's bulb is equivalent to half one's own, and everybody has a 60-watt bulb, everybody can read as long as he and both his neighbors have their lights on. Arranged in a circle, everybody will keep his light on if everybody else does (and nobody will if his neighbors do not); arranged in a line, the people at the ends cannot read anyway and the whole thing unravels.*

A Game-Theoretic Formulation. The k-core can be seen as the maximal equilibrium in a natural game-theoretic model; it has been studied previously in this guise in the social sciences literature [5,6,16]. Concretely, imagine that each node in a social network G is considering whether to remain engaged in a social activity. We suppose that each node v in G incurs an (integer) cost of $k > 0$ for the effort it must spend to remain engaged. Node v also obtains a benefit of 1 from each neighbor w who is engaged; this reflects the idea that the benefit from participation in the activity comes from interaction with neighbors in the social network.

If each node makes its decision simultaneously, we can model the situation as a simultaneous-move game in which the nodes are the players, and v's possible strategies are to remain engaged or to drop out. For a choice of strategies σ by each player, let S_σ be the set of players who choose to remain engaged. The payoff of v is 0 if it drops out, and otherwise it is v's degree in the induced subgraph $G[S_\sigma]$ minus k. Note that we can talk about sets of engaged nodes and strategy profiles interchangeably.

There is a natural structure to the set of pure Nash equilibria in this game: σ is an equilibrium if and only if $G[S_\sigma]$ has minimum degree k (so that no engaged player wants to drop out), and no node in $V - S_\sigma$ has k or more neighbors in S_σ (so that no player who has dropped out wants to remain engaged). There will generally be multiple equilibria — for example, if G has minimum degree at least k, then $S_\sigma = \emptyset$ and $S_\sigma = V$ define two of possibly many equilibria. There is always a unique maximal equilibrium σ^*, in the sense that S_{σ^*} contains S_σ for all other equilibria σ. This maximal equilibrium is easily seen to correspond to the k-core of G — the unique maximal set S^* of minimum internal degree at least k in G.

Chwe [5,6] and Saaskilahti [16] argue that it is reasonable to assume that this maximal equilibrium will be selected in an actual play of the game, since it optimizes the welfare of all the players simultaneously (as well as the provider of the service, whose goal is to attract a large audience). That is, all incentives are aligned to coordinate on this equilibrium.

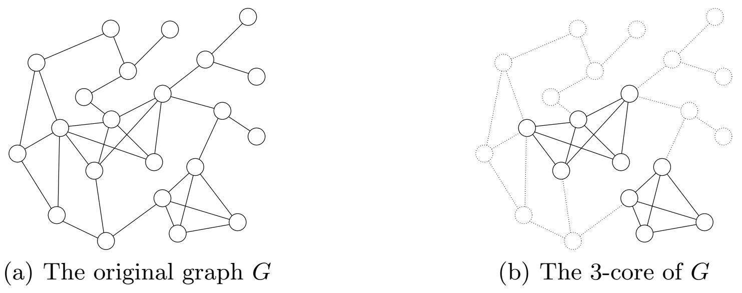
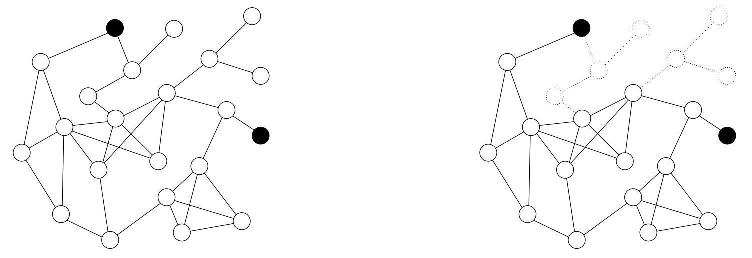

(a) The original graph G (b) The 3-core of G

Fig. 1. k-core for $k = 3$ on an example graph G

The Anchored *k*-Core Problem. The unraveling described in Schelling's line example is often undesirable, and could represent the end of a social network, a product, or a public good. When and how can such unraveling be prevented? For instance, in Schelling's example, the solution is clear: giving the two readers at the ends an extra lamp yields persistent illumination for all.

We formalize this problem as the *anchored k-core* problem. In the most basic version of the problem, the input is an undirected graph and two parameters $k, b \in \{1, 2, \ldots, n\}$, where b denotes a budget. Solutions correspond to subsets of at most b vertices, which are said to be *anchored.* Anchored vertices remain engaged no matter what their friends do — for example, due to external incentives like rewards for participation, or rebates for using a product. The anchored k-core, corresponding to the final subgraph of engaged individuals, is computed like the k-core, except that anchored vertices are never deleted. That is, unanchored vertices with degree less than k are deleted iteratively, in any order. In Schelling's line example, anchoring the two endpoints causes the anchored 2-core to be equal to the entire network. Another example, with $b = 2$ and $k = 3$, is displayed in Figure 2.


(a) An assignment of 2 anchors to G

(b) The vertices saved in an anchored 3-core (with budget 2) of G

Fig. 2. Anchored 3-core with budget 2 on G

Summarizing, we have seen that cascades of withdrawals can cause an unraveling of engagement, but that such cascades can sometimes be prevented by anchoring (i.e., rendering non-strategic) a small number of individuals. The goal of this paper is to study systematically the optimization problem of anchoring a given number of individuals to maximize the amount of engagement in a social network — to maximize the size of the resulting anchored k-core. Solving this problem identifies the individuals whose participation is most crucial to the overall health of a community.

Our Results. We first study general graphs, where we identify a "phase transition" in the computational complexity of the anchored k-core problem with respect to the parameter k. (The problem is interesting for all $k \geq 2$.) First, we prove that the anchored 2-core problem is solvable in polynomial (even near-linear) time. Second, we prove that the anchored k-core problem admits no non-trivial polynomial-time approximation algorithm for every $k \geq 3$. Precisely, we prove that it is NP-hard to distinguish between instances in which $\Omega(n)$ vertices are in the optimal anchored k-core, and those in which the optimal anchored k-core has size only $O(b)$. This inapproximability result holds even for a natural resource augmentation version of the problem. We also prove, for every $k \geq 3$, that the problem is $W[2]$-hard with respect to the budget parameter b.

Our negative results motivate studying the anchored k-core problem in restricted classes of graphs, and here we provide positive results. For arbitrary k, we show that the anchored k-core problem can be solved exactly in polynomial time in graphs with bounded treewidth. Our polynomial-time algorithm extends to many natural variations of the problem: for directed graphs, for non-uniform anchoring costs, for vertex-specific values of k, and others.

Further Related Work. Our mathematical model of user engagement in social networks appears to be new, although it is related to a number of previous works in the social sciences literature. Saaskilahti's model [16] is the closest to ours — the payoff structure in [16] includes ours as a special case, though only a few special network topologies are considered (the complete graph, the cycle, and the star). Earlier economic models that capture positive network effects of participation but consider only the complete graph are given in Arthur [1] and Katz and Shapiro [11]. Blume [2], Ellison [8], and Morris [14] analyze economic models with general network topolgies, but these works focus on competing behaviors rather than on positive network effects, resulting in models different from ours.

The papers cited above focus on equilibrium analysis and do not consider algorithms for optimizing an equilibrium, as we do here. The problem of identifying influential nodes of a social network in order to incite cascades, introduced by Kempe et al. [12] and studied further in [13,15], shares some of the spirit of the optimization problem studied in the present work.

2 General Graphs

In this section, we investigate the anchored k-core problem on general graphs. We will see that the problem can be solved exactly for $k = 2$ but becomes intractable for $k \geq 3$. Also, we'll show that one can't (under suitable complexity assumptions) substantially improve upon the brute-force algorithm that tries all $\binom{n}{b}$ subsets of placements for anchors unless FPT = W[2].

We first make the following observation. Let G be a graph for which we would like to compute the anchored k-core with budget b. We construct a new graph from G, which we will call $\mathsf{RemoveCore}(G)$, in the following manner: Compute the set of vertices of the k-core, C_k, and remove all edges between pairs of vertices $u, v \in C_k$. We also imagine an anchor placed at each vertex $v^* \in C_k$ (without actually subtracting from the budget).

Proposition 1. *An assignment of anchors has size z in* $\mathsf{RemoveCore}(G)$ *if and only if it has size z in G.*

Intuitively, each $v^* \in C_k$ would remain in the graph without the assistance of any anchors. So, we can think of these vertices as already being anchored, and the structure within C_k no longer affects the anchored k-core of G. Thus, it is enough to devise a solution for the graph $\mathsf{RemoveCore}(G)$ in order to obtain a solution for the anchored k-core problem on G.

2.1 Anchored 2-Core

The $\mathsf{RemoveCore}$ procedure greatly simplifies the structure of the input graph G for the case of $k = 2$.

Proposition 2. *For every input graph G, with $k = 2$,* RemoveCore(G) *is a forest, where each tree in the forest contains at most one member of the k-core.*

For each tree in the graph, we will call a tree *rooted* if there exists a member of the k-core in the tree, and *non-rooted* otherwise. Let $\mathcal{R}$ and $\mathcal{S}$ be the set of rooted and non-rooted trees, respectively.

An Efficient Optimal Algorithm. We now show how to solve the anchored 2-core problem for any budget b. We describe this algorithm intuitively and present it more explicitly as Algorithm 2.1 below. First, find two vertices $v_1, v_2 \in \mathcal{R}$ such that placing an anchor at v_1 maximizes the number of vertices saved across all placements of a single anchor in $\mathcal{R}$, and v_2 does the same assuming that v_1 has already been placed. Next, find $v_3, v_4 \in \mathcal{S}$ such that placing anchors at v_3 and v_4 simultaneously maximizes the number of vertices saved across all placements of two anchors in $\mathcal{S}$. Let $c_{\mathcal{R}}(v_1)$, $c_{\mathcal{R}}(v_2)$, and $c_{\mathcal{S}}(v_3, v_4)$ denote the number of vertices saved by placing v_1, v_2, and v_3 and v_4 (together), respectively. If $c_{\mathcal{R}}(v_1)+c_{\mathcal{R}}(v_2) > c_{\mathcal{S}}(v_3, v_4)$, or $b = 1$, then place an anchor at v_1 and decrease b by 1. Otherwise, place anchors at v_3 and v_4 and decrease b by 2. After the anchor placement, re-run RemoveCore on the graph (now with the saved vertices as part of the k-core) and repeat the process of determining $\{v_1, v_2, v_3, v_4\}$ until $b = 0$.

Algorithm 2.1 An efficient, exact algorithm for anchored 2-core

```
$G \leftarrow$ RemoveCore($G$)        // $G$ is now a forest
$S \leftarrow \emptyset$
while $b > 0$ do
    Partition the trees of $G$ into sets $\mathcal{R}$ and $\mathcal{S}$
    $v_1 \leftarrow$ a vertex furthest from root of trees in $\mathcal{R}$
    $v_2 \leftarrow$ a vertex to be furthest from root of trees in $\mathcal{R}$ after $v_1$ is anchored
    $(v_3, v_4) \leftarrow$ a pair of vertices on the endpoints of a longest path across trees in $\mathcal{S}$
    if $c_{\mathcal{R}}(v_1) + c_{\mathcal{R}}(v_2) > c_{\mathcal{S}}(v_3, v_4)$ or $b = 1$ then
        $S \leftarrow S \cup \{v_1\}$, $b \leftarrow b - 1$        // Place an anchor on $v_1$
    else
        $S \leftarrow S \cup \{v_3, v_4\}$, $b \leftarrow b - 2$        // Place an anchor on $v_3$ and $v_4$
    $G \leftarrow$ RemoveCore($G$)        // $G$ modified due to anchoring vertices
```

Theorem 1. *Algorithm 2.1 yields an anchored 2-core of maximum size.*

A Faster Implementation. The above algorithm runs in time $O(n^2)$, since both the RemoveCore step and finding the maximum path across all trees takes time $O(n)$, and this must be repeated for each anchor placed. However, there is an implementation of the algorithm that runs in time $O(m + n \log n)$ through the use of priority queues, and is detailed in the full version.

Corollary 1. *There is an $O(m + n \log n)$ time exact algorithm for the anchored 2-core problem.*

2.2 Inapproximability of $k \geq 3$

The natural next step is to determine the complexity of the anchored k-core problem for $k \geq 3$. Note that every solution to the anchored k-core problem has objective function value in the range $[b, n]$. In this section we show that for $k \geq 3$, it is NP-hard to approximate the optimal anchored k-core within a factor of $O(n^{1-\epsilon})$.

A Preliminary Construction. Our reduction is from the Set Cover problem. Fix an instance I of set cover with n sets $S_1, \ldots, S_n$ and m elements $\{e_1, \ldots, e_m\} = \bigcup_{i=1}^{n} S_i$. We first give the construction only for instances of set cover such that for all i, $|S_i| \leq k-1$. Then, we show how to lift this restriction while still obtaining the same results.

We now define a corresponding anchored k-core instance. Let H be an arbitrarily large graph where every vertex has degree k except for a single vertex with degree $k-1$ — call this vertex $t(H)$. Now, consider the graph consisting of a set of nodes $\{v_1, \ldots, v_n\}$ associated with sets $S_1, \ldots, S_n$ and a set $B = \{H_1, \ldots, H_m\}$ consisting of m disjoint copies H_j of H, where each copy of H is associated with an element e_j. There is an edge between v_i and $t(H_j)$ if and only if $e_j \in S_i$.

Lemma 1. *For instances of set cover with maximum set size at most $k-1$, there is a set cover of size z if and only if there exists an assignment in the corresponding anchored k-core instance using only z anchors such that all vertices in B are saved.*

Proof. Notice that the H_j's are designed such that if there exists some i such that v_i is adjacent to $t(H_j)$, then all vertices in H_j will be saved. Thus, if there is a set cover $\mathcal{C}$ of size z, then one can place the z anchors at v_i for all i such that $S_i \in \mathcal{C}$ and hence save all vertices in B. For the converse, we see that it is enough to restrict attention to assignments with anchors placed on v_i's. Since we are assuming that $|S_i| < k$ for all sets, each v_i will not be saved unless anchored. Thus, we must anchor some vertex adjacent to $t(H_j)$ for each copy H_j, which corresponds precisely to a set cover of size z.

Now, define $\mathsf{tree}(d, y)$ to be a perfect d-ary tree (each node has exactly d children) with exactly y leaves, if $y \geq d$, and a single root node with y leaves when $y < d$. To lift the restriction on the maximum set size, we replace each instance of v_i with $\mathsf{tree}(k-1, |S_i|)$, and if $y_1, \ldots, y_{|S_i|}$ are the leaves, then for each $e_\ell \in S_i$, we contract the pairs of vertices (y_ℓ, e_ℓ).

Lemma 2. *For every instance of Set Cover, there is a set cover of size z if and only if there exists an assignment for the corresponding anchored k-core instance using only z anchors such that all vertices in B are saved.*

The Reduction from Set Cover. At this point, we have already shown that obtaining an optimal solution for the anchored k-core problem for $k \geq 3$ is NP-hard. Now, there exists a way to arrange the edges between each copy of H

such that if there exists some vertex of B which is not saved, then *the majority* of the vertices will not be saved, either. Since this construction is somewhat complicated, we defer the details to the full version. From here on, we refer to the full construction as the graph $G(c, I)$, where I is an instance of Set Cover and c is an arbitrarily large constant.

Recall the decision problem for (unweighted) Set Cover: Given a collection of sets $\mathcal{C}$ which contain elements from a universe of size m, does there exist a set cover of size at most ℓ? We are able to show that $G(c, I)$ has the following two properties:

1. If I is a yes-instance, then there exists an assignment of ℓ anchors such that at least $km^{c+1}\ell$ vertices are saved.
2. If I is a no-instance, then no assignment of ℓ anchors can save more than $km\ell$ anchors.

Since c can be arbitrarily large, we can ensure that $km^{c+1}\ell = \Omega(n)$, where n is the number of vertices in the anchored k-core instance. We can therefore conclude with the following theorem and corollary.

Theorem 2. *It is NP-hard to distinguish between instances of anchored k-core where the optimal solution has value $\Omega(n)$ versus when the optimal solution has value $O(b)$.*

Corollary 2. *It is NP-hard to approximate the anchored k-core problem on general graphs within an $O(n^{1-\epsilon})$ factor.*

Resource Augmentation Extensions. Suppose we are interested in comparing the performance of an algorithm given a budget of $O(b \cdot \alpha)$ anchors against the optimal assignment given only b anchors, for some $\alpha > 1$. In the full version, we augment the original reduction above to yield the following result.

Corollary 3. *For $\alpha = o(\log n)$, unless P = NP, there does not exist a polynomial time algorithm which, given $O(b \cdot \alpha)$ anchors, finds a solution within an $O(n^{1-\epsilon})$ multiplicative factor of the optimal solution with b anchors.*

W[2]-hardness with Respect to Budget. We can also show that the anchored k-core problem is not in FPT with respect to the budget parameter b. We establish this result via a reduction from the Dominating Set problem, and the proof can be found in the full version. Recall that the decision version of Dominating Set is as follows: given a budget ℓ, determine if there a subset S of vertices such that $|S| \leq \ell$ and each vertex in $V \setminus S$ is adjacent to a vertex in S. As shown in [7], this problem is W[2]-hard. In the full version, we establish that if the anchored k-core problem can be solved in time $O(f(b) \cdot \mathsf{poly}(n))$ for any $k \geq 3$, then Dominating Set is in FPT with respect to ℓ, and hence FPT = W[2].

Theorem 3. *For every $k \geq 3$, the anchored k-core problem is W[2]-hard with respect to the parameter b.*

3 Graphs with Bounded Treewidth

Although we see that the anchored k-core problem is hopelessly inapproximable on general graphs, we next give polynomial time exact algorithms for graphs with bounded treewidth. The treewidth of a graph is defined as the minimum width over all tree decompositions of the graph, where the width of a tree decomposition is one more than the size of the largest node in the tree decomposition (see [3] for a tutorial and survey on treewidth). In this section, we present an algorithm that runs in time $O(f(k,w) \cdot b^2) \cdot \mathsf{poly}(n)$, where $f(k,w) = (3(k+1)^2)^w$, using $w-1$ as the graph's treewidth. To distinguish the vertices of a tree decomposition from the vertices of the original graph, we will call the elements of a tree decomposition nodes, and the elements of the original graph will remain as vertices. We will use the concept of nice tree decompositions for graphs, defined in [3] — the idea that a tree decomposition can be converted into another tree decomposition (a "nice" one) of the same treewidth and $O(n)$ nodes, but with the special property that each node comes in one of four types:

- Leaf Node: Only one vertex is associated with this node
- Introduce Node: The node has a single child, and if X is the set of vertices associated with this node and Y is the child, then $X = Y \cup \{v\}$ for some v.
- Forget Node: The node has a single child, and if X is the set of vertices associated with this node and Y is the child, then $X = Y \setminus \{v\}$ for some v.
- Join Node: The node has two children, and if X is the set of vertices associated with this node, Y and Z are its children, then $X = Y = Z$.

We show how to solve a generalization of the anchored k-core problem. Let $\mathsf{threshold}(v)$ represent the *threshold* of v, which is the minimum number of neighbors that v requires in order to remain in the k-core (assuming v is not anchored). Traditionally, in k-core, we use $\mathsf{threshold}(v) = k$ for all $v \in V(G)$. Since we are now considering a situation where the threshold function varies across vertices, we will instead use k to denote the maximum threshold across all vertices.

For a fixed assignment of anchors, a vertex is either *anchored*, *not saved*, or *indirectly saved* (not anchored, but saved). We show that the categorization of vertices into these three types is enough to capture the complexity of the problem on graphs with bounded treewidth. Define a *fixture* f of a tree decomposition T to be an assignment of these three types to a subset of $G[r(T)]$, the vertices of G that are associated with the root node of T. We say that an assignment A of anchors to vertices *satisfies* a fixture f if under the assignment A, the type of each vertex designated by the fixture agrees with the type induced by the assignment A. Define a threshold alteration m (which we will simply call an *alteration*) to be a setting of the thresholds of some subset $S \subseteq V(G)$ so that for each $v \in S$, m reduces the threshold of v by some integer in the interval $[0, \mathsf{threshold}(v)]$. We use the notation $m(T)$ to denote the tree obtained by lowering the thresholds of all vertices as prescribed by m.

3.1 The Algorithm

Algorithm 3.1 $\mathsf{Solve}(T)$: The main subroutine used in the exact algorithm for graphs with bounded treewidth.

```
$(\mathsf{Solutions}_{T_1}, \mathsf{Solutions}_{T_2}) \leftarrow (\mathsf{Solve}(T_1), \mathsf{Solve}(T_2))$
**for all** fixtures $f$, alterations $m$, and budgets $b$ **do**
  **if** $r(T)$ is a leaf node **then**
    $\mathsf{Solutions}_T[f][m][b] \leftarrow$ the result dicated by the fixture $f$
  **if** $r(T)$ is a forget node **then**
    $S \leftarrow \{\text{fixtures } f' \text{ of } T_1 : \forall v \in G[r(T)], f(v) = f'(v)\}$
    $\mathsf{Solutions}_T[f][m][b] \leftarrow \max_{f' \in S} \mathsf{Solutions}_{T_1}[f'][m][b]$
  **if** $r(T)$ is an introduce node **then**
    Set $f'$ to be such that $\forall v \in G[rT_1)], f'(v) = f(v)$
    Let $v$ be the sole element of $G[r(T)] \setminus G[r(T_1)]$
    Set $m'$ so that $\forall u \in N(v), m'(u) = m(u) - 1$ and $\forall u \notin N(v), m'(u) = m(u)$
    **if** $f(v)$ is anchored **then**
      $\mathsf{Solutions}_T[f][m][b] \leftarrow \mathsf{Solutions}_{T_1}[f'][m'][b-1]$
    **if** $f(v)$ is not saved **then**
      $\mathsf{Solutions}_T[f][m][b] \leftarrow \mathsf{Solutions}_{T_1}[f'][m][b]$
    **if** $f(v)$ is indirectly saved **then**
      $\mathsf{Solutions}_T[f][m][b] \leftarrow \mathsf{Solutions}_{T_1}[f'][m'][b]$
  **if** $r(T)$ is a join node **then**
    $\hat{b} \leftarrow b - |\{v : f(v) \text{ is anchored}\}|$
    $t \leftarrow \max_{i \in [0,\hat{b}], \hat{m} \in [0,k]^w} (\mathsf{Solutions}_{T_1}[f][\hat{m}][i] + \mathsf{Solutions}_{T_2}[f][m - \hat{m}][\hat{b} - i])$
    $\mathsf{Solutions}_T[f][m][b] \leftarrow t$
**return** $\mathsf{Solutions}_T$
```

We first define a subroutine: $\mathsf{Solve}(T)$ for a tree decomposition T, also outlined in Algorithm 3.1. The output of $\mathsf{Solve}(T)$ will be, for all fixtures f and all alterations m (within $r(T)$), and $\hat{b} \in [1, b]$, the table $\mathsf{Solutions}(m(T), f, \hat{b})$. Each entry of this table of solutions, $\mathsf{Solutions}_T[f][m][b]$, will describe an optimal assignment of b anchors which satisfies the fixture f on the graph $G[T]$, the vertices of G that are associated with the nodes of T, under alteration m. Note that if no such assignment that satisfies the stated restrictions can exist, then the output of the entry is $\perp$. (This could occur if, for example, the fixture f requires 3 nodes to be anchored yet $b < 3$.)

Thus, $\mathsf{Solve}(T)$ will output at most $(3(k+1))^w \cdot b$ solutions. The 3^w term is due to the fact that there are 3^w possible fixtures, and the $(k+1)^w$ term comes from the fact that each vertex has threshold at most k, and so there are at most $k+1$ choices for lowering the thresholds of each vertex in the root node (within the range $[0, k]$). We now show that given the outputs of $\mathsf{Solve}(T_i)$ for each child subtree T_i, we can compute $\mathsf{Solve}(T)$. This will be done through a case analysis on the node type of the root node, denoted by $r(T)$.

If $r(T)$ is a leaf node, then $\mathsf{Solutions}(m(T), f, b)$ is trivial to determine for all alterations m, fixtures f, and budgets b, as there is only one vertex associated with $r(T)$ and there are no other nodes in T (and so, an anchor is placed on this vertex if it is anchored under f and $b \geq 1$). Otherwise, let T_1 and T_2 denote the child subtrees of $r(T)$. If $r(T)$ is a forget node, then let v be the vertex that is associated with the child node but not in $G[r(T)]$. To find $\mathsf{Solutions}(m(T), f, b)$ for all f, m, and b, we simply compute the maximum solution over possible choices of the fixture type of v and the threshold alteration induced by the partial fixture of the other vertices in $G[r(T)]$.

If $r(T)$ is an introduce node, then let v be the vertex in $G[r(T)]$ that is not associated with the child node. For fixtures f such that v is anchored, we simply subtract one from the budget for T_1 and retrieve the optimal solution there under the induced partial fixture. This is obtained from the output of $\mathsf{Solutions}(m'(T), f, b)$, where $m'(u) = m(u) - 1$ if $u \in N(v)$ and $m'(u) = m(u)$ otherwise. If v is indirectly saved, we do not change the budget but obtain the optimal solution under the induced partial fixture for $\mathsf{Solutions}(m'(T), f, b)$. If v is not saved, we use the optimal solution on T_1 under the induced fixture f. Finally, if $r(T)$ is a join node, then let $i \in [1, b]$. Also, for a fixture f, let S be the set of all vertices in the root node that are indirectly saved. Note that $|S| \leq w$. We iterate over all of the at most $(k + 1)^w$ possibilities of dividing up the thresholds for each $v \in R$ between T_1 and T_2. We then iterate over all pairs of solutions $\mathsf{OPT}(m(T_1), f, i)$ and $\mathsf{OPT}(m(T_2), f, b - i)$ and take the maximum over such i.

This approach is repeated in a bottom-up manner until we have covered the entire tree. Finally, we simply take the output of $\mathsf{Solve}(T)$ (the original tree decomposition T of the entire graph) and find the optimal solution corresponding to the tree on b anchors by taking the maximum value over all fixtures. The formal proof of correctness is deferred to the full version.

Generalizations. This dynamic programming approach allows for several generalizations to the anchored k-core problem on graphs with bounded treewidth, all of which can be solved exactly and run in time $O(f(k, w) \cdot \mathsf{poly}(n, b))$. For example, one can assign weights to vertices. Also, as we have already seen, the vertex thresholds can be non-uniform. Furthermore, the edges of the graph can be directed, and each arc $a = (u, v)$ can have a weight $w(a)$ such that ensuring that u is in the graph contributes a value of $w(a)$ to the threshold of v.

4 Concluding Remarks

There remain several attractive open problems related to the anchored k-core problem, especially on restricted classes of graphs. Is there a PTAS for planar graphs? What can be said about the problem on random graphs? Can the running time of our polynomial-time algorithm for bounded treewidth graphs be improved? Is there a linear-time algorithm for the anchored 2-core problem in general graphs?

References

1. W. Brian Arthur. Competing technologies, increasing returns, and lock-in by historical events. *The Economic Journal*, 99(394):116–131, March 1989.

2. Larry Blume. The statistical mechanics of strategic interaction. *Games and Economic Behavior*, 5:387–424, 1993.
3. Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, pages 255–269, 2008.
4. Moira Burke, Cameron Marlow, and Thomas Lento. Feed me: motivating newcomer contribution in social network sites. In *CHI*, 2009.
5. Michael Suk-Young Chwe. Structure and strategy in collective action. *American Journal of Sociology*, 105(1):128–156, July 1999.
6. Michael Suk-Young Chwe. Communication and coordination in social networks. *Review of Economic Studies*, 67:1–16, 2000.
7. Rodney G. Downey, Michael R. Fellows, Catherine McCartin, and Frances Rosamond. Parameterized approximation of dominating set problems. *Information Processing Letters*, 109:68–70, 2008.
8. Glenn Ellison. Learning, local interaction, and coordination. *Econometrica*, 61:1047–1071, 1993.
9. Nicole B. Ellison, Charles Steinfield, and Cliff Lampe. The Benefits of Facebook Friends: Social Capital and College Students' Use of Online Social Network Sites. *Journal of Computer-Mediated Communication*, 2007.
10. Rosta Farzan, Laura A. Dabbish, Robert E. Kraut, and Tom Postmes. Increasing commitment to online communities by designing for social presence. In *CSCW*, 2011.
11. Michael L. Katz and Carl Shapiro. Network externalities, competition, and compatibility. *American Economic Review*, 75(3):424–440, June 1985.
12. David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.
13. David Kempe, Jon Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, pages 1127–1138, 2005.
14. Stephen Morris. Contagion. *Review of Economic Studies*, 67:57–78, 2000.
15. Elchanan Mossel and Sébastien Roch. On the submodularity of influence in social networks. In *STOC*, pages 128–134, 2007.
16. Pekka Saaskilahti. Monopoly pricing of social goods. Technical report, University Library of Munich, Germany, 2007.
17. Thomas C Schelling. *Micromotives and Macrobehavior*. Norton, 1978.