Approximation and Hardness Results for the Max k-Uncut Problem

Peng Zhang[1], Chenchen Wu[2], Dachuan Xu[3(✉)], and Xinghe Zhang[4]

[1] School of Computer Science and Technology, Shandong University, Jinan 250101, China
algzhang@sdu.edu.cn

[2] College of Science, Tianjin University of Technology, Tianjin 300384, China

[3] Department of Information and Operations Research, College of Applied Sciences, Beijing University of Technology, Beijing 100124, China
xudc@bjut.edu.cn

[4] Shandong Experimental High School (East Campus), Jinan 250109, China

Abstract. In this paper, we propose the Max k-Uncut problem. Given an n-vertex undirected graph $G = (V, E)$ with nonnegative weights $\{w_e \mid e \in E\}$ defined on edges, and a positive integer k, the Max k-Uncut problem asks to find a partition $\{V_1, V_2, \cdots, V_k\}$ of V such that the total weight of edges that are *not* cut is maximized. This problem is just the complement of the classic Min k-Cut problem. We get this problem from the study of complex networks. For Max k-Uncut, we present a randomized $(1 - \frac{k}{n})^2$-approximation algorithm, a greedy $(1 - \frac{2(k-1)}{n})$-approximation algorithm, and an $\Omega(\frac{1}{2}\alpha)$-approximation algorithm by reducing it to Densest k-Subgraph, where α is the approximation ratio for the Densest k-Subgraph problem. More importantly, we show that Max k-Uncut and Densest k-Subgraph are in fact equivalent in approximability up to a factor of 2. We also prove a weak approximation hardness result for Max k-Uncut under the assumption P $\neq$ NP.

1 Introduction

In this paper, we investigate the Max k-Uncut problem, which is obtained from the study of the *homophyly* law [8, Chap. 4] of large scale networks. Being one of the basic laws governing the structures of large scale networks, the homophyly law states that edges in a network tend to connect nodes with the same or similar attributes, just as an old proverb says, "birds of a feather flock together". For example, in a paper citation network, papers are more likely to cite papers with which they have the same or similar keywords.

While it is common to list keywords in a paper by its authors, in a paper citation network there are still many papers whose keywords are not explicitly given. Consequently, it is natural to predict keywords for these papers using

P. Zhang—Part of the author's work was done while he was visiting at the University of California - Riverside, USA, and Beijing University of Technology, China.

T-H.H. Chan et al. (Eds.): COCOA 2016, LNCS 10043, pp. 49–61, 2016.
DOI: 10.1007/978-3-319-48749-6_4

the homophyly law. Inspired by this observation, Zhang (the first author of this paper) and Li [21] proposed the Maximum Happy Edges (MHE) problem. In the MHE problem, we are given an undirected graph $G = (V, E)$ and a color set $C = \{1, 2, \cdots, k\}$. Only part of vertices are given colors in C. An edge is *happy* if its two endpoints share the same color. The goal of MHE is to color all the uncolored vertices such that the number of happy edges is maximized. Here, vertices correspond to papers, edges correspond to citations (neglecting directions), and colors correspond to keywords.

A natural variant of MHE is that in the input graph all vertices are uncolored and the problem just asks to color them in k colors such that the number (or total weight) of happy edges is maximized. This suggests the Max k-Uncut problem we investigate in this paper.

Definition 1. The Max k-Uncut **Problem**

(Instance). *We are given an undirected graph $G = (V, E)$ with nonnegative edge weights $\{w_e \mid e \in E\}$, and a positive integer k.*

(Goal). *The problem asks to find a partition $\{V_1, V_2, \cdots, V_k\}$ of V (i.e., to find a k-coloring of vertices) such that the total weight of happy edges is maximized.*

In the definition of Max k-Uncut, by *k-coloring* we mean a coloring scheme that uses *exactly* k colors, which results in a *k-partition* $\{V_1, V_2, \cdots, V_k\}$, where V_i is the set of vertices whose color is i. In the paper we will interchangeably use k-coloring and k-partition. Note that the requirement of exactly k colors is necessary, otherwise we can color all vertices in one color and all edges are happy.

In the Max k-Uncut problem, if $k = 1$ or $k = n$, the problem becomes trivial. The optimum would be respectively the number of all edges and 0 in these two cases. So, throughout the paper we always assume $2 \leq k \leq n - 1$ for the Max k-Uncut problem.

Note that Max k-Uncut is *not* a special case of MHE. In MHE, if all vertices are un-colored, then the problem becomes trivial: Just color all vertices in one color, then all edges will become happy. In contrast, if all vertices in Max k-Uncut are uncolored, we cannot color them in one color. In Max k-Uncut, we must figure out a k-coloring.

Two problems that are closely related to Max k-Uncut have already appeared in literature. Choudhurya *et al.* [7] proposed the capacitated Max k-Uncut problem. Given an undirected graph G and k integers $s_1, s_2, \cdots, s_k$, this problem is to partition $V(G)$ into k subsets of sizes $s_1, s_2, \cdots, s_k$ respectively, such that the total weight of happy edges is maximized. Very recently, Wu *et al.* [18] studied the balanced Max 3-Uncut problem, in which an input graph is partitioned into 3 equal-sized parts so that the total weight of happy edges is maximized.

Notations and Terms. Some common notations and terms are listed here. Given a graph G, let n be the number of its vertices. Given an optimization problem, let OPT denote the value of its optimal solution. By r-clique for some integer r, we mean a clique (i.e., a complete subgraph) that contains exactly r vertices.

1.1 Related Work

To the best of our knowledge, the general Max k-Uncut problem is new and has not been studied in literature. Though it is new, Max k-Uncut has rich connection to the classic and existing problems.

Max k-Uncut is just the complement of the classic Min k-Cut problem. The Min k-Cut problem asks for a k-partition such that the total weight of cut edges is minimized. The Min k-Cut problem is strongly NP-hard [12], so is the Max k-Uncut problem. The best approximation ratio for Min k-Cut is 2 [17]. When k is a constant, the Min k-Cut problem can be optimally solved in polynomial time [12]. Obviously, Max k-Uncut with constant k is also polynomial time solvable. In a word, Max k-Uncut is strongly NP-hard (when k is given in the input), and is polynomial time solvable when k is a constant.

Previously we have pointed out two closely related variants of Max k-Uncut, i.e., the capacitated Max k-Uncut problem and the balanced Max 3-Uncut problem. Using the heuristic of local search, Choudhurya *et al.* [7] gave a $\frac{1}{d(k-1)+1}$-approximation algorithm for capacitated Max k-Uncut, where d is the ratio of the largest size and the smallest size in the partition. This ratio is somewhat poor and cannot extend to the Max k-Uncut problem studied in this paper. Using the semidefinite programming technique, Wu *et al.* [18] gave a 0.3456-approximation algorithm for the balanced Max 3-Uncut problem.

The cut problems are classic and rich. They play an important role in the study of approximation algorithms and operations research. In literature, the "uncut" problems are also been studied. Besides Max k-Uncut, three examples are Min Uncut [1], Multiway Uncut [15,20], and the complement of Min Bisection [19]. Min Uncut is the complement of the classic Max Cut problem. Agarwal *et al.* [1] gave an $O(\sqrt{\log n})$-approximation algorithm for Min Uncut, where n is the number of vertices in the input graph. Multiway Uncut is the complement of the classic Multiway Cut problem [5,6]. Langberg *et al.* [15] proposed the Multiway Uncut problem. The current best approximation ratio for Multiway Uncut is $\frac{1}{2} + \frac{\sqrt{2}}{4} f(k) \geq 0.8535$ [20], where $f(k) \geq 1$ is a function of k. Ye and Zhang [19] gave a 0.602-approximation algorithm for the complement of the Min Bisection problem.

Due to the close relation of Max k-Uncut to Multiway Uncut, we have to say more about Multiway Uncut. Given a graph $G = (V, E)$ with edge weights and a terminal set $\{s_1, s_2, \cdots, s_k\} \subseteq V$, the Multiway Uncut problem asks a partition of V that separates the k terminals from each other and maximizes the total weight of happy edges. (Multiway Uncut is a special case of MHE.) Both Max k-Uncut and Multiway Uncut ask for a k-partition. The only difference is that in Max k-Uncut, there is no terminal, and in Multiway Uncut, there are terminals.

Another closely related problem is Max k-Cut, which is the problem to find a k-partition such that the total weight of cut edges is maximized. When $k = 2$, Max k-Cut (namely, Max Cut) is already NP-hard. The current best approximation ratio for Max Cut is 0.87856, given by Goemans and Williamson [11] using the semidefinite programming technique. Frieze and Jerrum [10] extended Goemans-Williamson's technique to the Max k-Cut problem, obtained

the approximation ratio $\alpha_k = 1 - \frac{1}{k} + (1 + \epsilon(k))\frac{2 \ln k}{k^2}$ for Max k-Cut, where $\epsilon(k)$ is a function of k which tends to zero as $k \to \infty$. When $k = 3, 4, 5$, α_k is no less than 0.800217, 0.850304, and 0.874243, respectively.

1.2 Our Results

In this paper, we give three approximation algorithms for the Max k-Uncut problem and prove a (weak) approximation hardness result of Max k-Uncut. These three algorithms share the same idea, which is simple but powerful: To find a k-partition with many happy edges, one may just find a dense subgraph as large as possible. The subgraph is used as one part of the k-partition. The larger and denser the subgraph is, the more happy edges we will get. Along this line, we finally find that Max k-Uncut is in fact equivalent to the Densest k-Subgraph problem in approximability (up to a factor of 2). Note that Densest k-Subgraph is one of the current hot topics in approximation algorithms. This may be our most important find in this paper.

The first algorithm is a randomized algorithm (Algorithm 2.1) whose approximation ratio is $(1 - \frac{k}{n})^2$. This algorithm can be derandomized in polynomial time. The second algorithm is a greedy algorithm (Algorithm 2.2) whose approximation ratio is $1 - \frac{2(k-1)}{n}$. While the ratios of these two algorithms are very close, they are still incomparable. Specifically, when $k < \sqrt{2n}$, the ratio $1 - \frac{2(k-1)}{n}$ is better than the ratio $(1 - \frac{k}{n})^2$. Otherwise (when $k > \sqrt{2n}$), the latter is better than the former.

The ratio $\rho = \max\{(1 - \frac{k}{n})^2, 1 - \frac{2(k-1)}{n}\}$ for Max k-Uncut we obtain so far is already good when k is not too large. For example, if $k \leq n/2$, then $\rho \geq 1/4$. However, when k approaches $n-1$, ρ becomes worse and worse, and equals to $\frac{1}{n^2}$ finally. This observation suggests that the most difficult case of approximating Max k-Uncut should be the case when k is close to n, say, $k = n - O(\log n)$. And in this case (i.e., when k is large), we may make use of the connection to Densest k-Subgraph.

Therefore, in the third algorithm (Algorithm 2.3), we reduce Max k-Uncut to Densest $\bar{k}$-Subgraph (for some suitable $\bar{k}$) by exploring the structure of optimal solutions to Max k-Uncut. It is convenient to define the Densest k-Subgraph problem here.

Definition 2. The Densest k-Subgraph **Problem**

(Instance). *We are given an undirected graph $G = (V, E)$ with nonnegative edge weights $\{w_e \mid e \in E\}$, and a positive integer k.*

(Goal). *The problem asks to find a k-vertex subgraph G' such that the total weight of edges in $E(G')$ is maximized.*

The reduction used in Algorithm 2.3 is nontrivial. Let α be the approximation ratio for Densest k-Subgraph. Then Algorithm 2.3 approximates Max k-Uncut within $\frac{1}{2}\alpha$ in polynomial time. The current best value of α is $\Omega(1/n^{\frac{1}{4}+\epsilon})$ [4].

Consequently, Algorithm 2.3 repairs the deficiencies of Algorithms 2.1 and 2.2. Now, the approximation ratio we obtain for Max k-Uncut is $\max\{\rho, \frac{1}{2}\alpha\}$.

Surprisingly and interestingly, our technique in the analysis of Algorithm 2.3 also implies that if Max k-Uncut can be approximated within a factor of β, then Densest k-Subgraph can be approximated within $\frac{1}{2}\beta$. Therefore, Max k-Uncut and Densest k-Subgraph are equivalent in approximability up to a factor of 2. This reveals the strong connection between Max k-Uncut and Densest k-Subgraph, and may open a new viewpoint in tackling the Densest k-Subgraph problem, since this problem is known as a notorious hard problem in approximation algorithms. (There is a wide gap between its best approximation factor and its best hardness factor).

Next, we prove an approximation hardness result for Max k-Uncut: For any small constant $\epsilon > 0$, Max k-Uncut cannot be approximated within $1 - \frac{1}{2n^\epsilon}$ in polynomial time, where n is the number of vertices in the input graph. This is proved via a gap-preserving reduction from the hardness result of the Max Clique problem [3,13]. As a result, the hardness $1 - \frac{1}{2n^\epsilon}$ for any small constant $\epsilon > 0$ implies that Max k-Uncut does not admit FPTAS.

Honestly speaking, this hardness result is weak since Max k-Uncut is indeed strongly NP-hard, and the strong NP-hardness already rules out FPTAS. However, we make twofold contribution in proving the approximation hardness of Max k-Uncut. First, we give an explicit expression of the approximation hardness factor of Max k-Uncut, instead of just speaking that it is strongly NP-hard. Second, we prove a technical lemma (Lemma 2), which gives an upper bound of the number of happy edges that can be produced by any k-partition on a graph with no $(r+1)$-clique. The technical lemma is of independent interest and may find more applications in related problems. In fact, the upper bound is obtained by a special k-partition which consists of $k-1$ singletons and one subset of size $n-(k-1)$. This again hints the connection of Max k-Uncut to Densest k-Subgraph and Max Clique.

2 Approximation Algorithms

2.1 A Randomized Algorithm

A straightforward idea for Max k-Uncut is to color vertices randomly. However, if we color *every* vertex randomly, we may not get an approximation algorithm with good ratio. (We can prove that an algorithm of this type has approximation ratio $\frac{1}{k} - \frac{k-1}{n(n-1)}$. The details are omitted here.)

In graphs with only unit weight on edges, to maximize the total weight of happy edges is equivalent to leave as many as possible edges uncut. So, a clever randomized strategy is to randomly color $k-1$ vertices only, making the remaining vertices as many as possible. Intuitively, these many vertices would induce many happy edges. Algorithm $\mathcal{R}$ below is a randomized algorithm for Max k-Uncut of this idea.

Let W_{tot} be the total weight of edges in graph G.

Algorithm 2.1. (Algorithm $\mathcal{R}$ for Max k-Uncut)

1 Pick randomly $k-1$ vertices from V, and color them respectively in colors 1 to $k-1$.
2 Color all the remaining vertices in color k.

Theorem 1. *Algorithm $\mathcal{R}$ is a randomized $\left(1-\frac{k}{n}\right)^2$-approximation algorithm for the* Max k-Uncut *problem.*

Proof. First note that Algorithm $\mathcal{R}$ runs in polynomial time. Let V_i be the set of vertices of color i. Take any edge $e=(u,v)$. Then, e is happy (uncut) if and only if both u and v are not chosen in the first $k-1$ random choices (step 1). This means that

$$\Pr[\text{edge } e \text{ is happy}] = \frac{\binom{n-2}{k-1}}{\binom{n}{k-1}} = \frac{(n-k+1)(n-k)}{n(n-1)} > \frac{(n-k)^2}{n^2} = \left(1-\frac{k}{n}\right)^2.$$

Let SOL be the solution value obtained by Algorithm $\mathcal{R}$. Therefore, we have

$$\mathrm{E}[SOL] = \sum_{e\in E} w_e \cdot \Pr[\text{edge } e \text{ is happy}] \geq \left(1-\frac{k}{n}\right)^2 W_{tot}.$$

On the other hand, the optimum OPT is obviously at most W_{tot}. So, the approximation ratio of Algorithm $\mathcal{R}$ is at least $\left(1-\frac{k}{n}\right)^2$. □

Algorithm $\mathcal{R}$ can be derandomized by the conditional expectation method in polynomial time. This is sketched as follows in rounds. In the first round we determine the first vertex to be removed. We remove each vertex v_i $(1 \leq i \leq n)$ from G to obtain G_i. That is, $\forall 1 \leq i \leq n$, $G_i = G \setminus v_i$. For each G_i, we compute the expected solution value a_i of Algorithm $\mathcal{R}$ for the Max$(k-1)$-Uncut problem. We find the largest expected value in this round, say a_j. Then, v_j is the first vertex we pick and is colored in color 1. The next round begins from G with v_j removed. Repeating the above procedure for $k-1$ rounds, we obtain a solution whose value is at least as better as the expected value of Algorithm $\mathcal{R}$.

2.2 A Greedy Algorithm

The idea in Sect. 2.1 can be restated as finding a subgraph of size $n-k+1$ as dense as possible, where by dense subgraph we mean a subgraph whose total weight of edges is as much as possible. This leads to a greedy algorithm for Max k-Uncut, shown as Algorithm $\mathcal{G}$ below. For the sake of description, we define the weighted degree $d_w(v)$ of a vertex v as the sum of weights of edges incident to v. By definition, the weight of vertex v is equal to the capacity of the cut $(\{v\}, V \setminus \{v\})$. Obviously, when each edge in the graph has unit weight, the weighted degree of a vertex is simply its degree.

Algorithm 2.2. (Algorithm $\mathcal{G}$ for Max k-Uncut)

1 Pick vertices from V with the first $k-1$ smallest weighted degrees, and color them in colors 1 to $k-1$, respectively.
2 Color all the remaining vertices in color k.

Theorem 2. *Algorithm $\mathcal{G}$ is a $\left(1-\frac{2(k-1)}{n}\right)$-approximation algorithm for the* Max k-Uncut *problem.*

Proof. Algorithm $\mathcal{G}$ obviously runs in polynomial time. Let $v_1, \cdots, v_{k-1}$ be the vertices picked in the first step of Algorithm $\mathcal{G}$. By the algorithm, only edges incident to vertices in $\{v_1, \cdots, v_{k-1}\}$ would be unhappy. So, the total weight of unhappy edges is at most

$$\sum_{i=1}^{k-1} d_w(v_i) \leq \frac{k-1}{n} \sum_{v} d_w(v) = \frac{2(k-1)}{n} W_{tot}.$$

Therefore, the total weight of happy edges is at least

$$W_{tot} - \frac{2(k-1)}{n} W_{tot}.$$

Since OPT $\leq W_{tot}$, this means the approximation ratio of Algorithm $\mathcal{G}$ is at least $1-\frac{2(k-1)}{n}$. □

The approximation ratios $(1-\frac{k}{n})^2$ and $1-\frac{2(k-1)}{n}$ behave well when k is not too large. For example, $(1-\frac{k}{n})^2 \geq \frac{1}{4}$ when $k \leq \frac{n}{2}$. However, when k is large enough, say, $k = n - O(\log n)$, the approximation ratio $\max\{(1-\frac{k}{n})^2, 1-\frac{2(k-1)}{n}\}$ we obtained so far becomes bad. To remedy this deficiency, we design another approximation algorithm for Max k-Uncut, that is, Algorithm $\mathcal{T}$ in Sect. 2.3. Actually, our subsequent study on Max k-Uncut in this paper makes us realize that the hard core of Max k-Uncut just lies in the case when k is large.

2.3 Reduces to Densest k-Subgraph

In this section, we reduce Max k-Uncut to Densest $\bar{k}$-Subgraph for some suitable $\bar{k}$. For clarity, when the instance of Densest k-Subgraph is given as, e.g., $(G, w, \bar{k})$, we call it the instance of the Densest $\bar{k}$-Subgraph problem. The reader should be aware of that Densest k-Subgraph and Densest $\bar{k}$-Subgraph are the same problem. This usage also happens to the Max k-Uncut problem.

Given a vertex subset S of an edge-weighted graph G, let $w(S)$ denote the total weight of happy edges induced by S. Given a k-partition $\mathcal{P} = \{V_1, V_2, \cdots, V_k\}$ of $V(G)$, let $w(\mathcal{P})$ denote the total weight of happy edges induced by $\mathcal{P}$, i.e., $w(\mathcal{P}) = \sum_i w(V_i)$.

First we prove a technical lemma.

Lemma 1. *Let $\mathcal{P} = \{V_1, V_2, \cdots, V_k\}$ be a k-partition of graph G with weights defined on edges. Then in polynomial time (in terms of $|V(G)|$) we can construct a k-partition $\mathcal{P}' = \{V_1', V_2', \cdots, V_k'\}$ which satisfies*

(i) $|V_1'| = \cdots = |V_{k-1}'| = 1$, $|V_k'| = n - k + 1$, *and*
(ii) $w(\mathcal{P}') \geq \frac{1}{2} w(\mathcal{P})$.

Proof. We renumber the vertex subsets in $\mathcal{P}$ according to the non-decreasing order of their $w(\cdot)$ values, and rewrite $\mathcal{P}$ as $\{R_1, R_2, \cdots, R_a, S_1, S_2, \cdots, S_b\}$, where we assume that in $\mathcal{P}$ there are a singletons $R_1, \cdots, R_a$, and b non-singletons $S_1, \cdots, S_b$ (that is, each S_i has size at least two). So, we have $a+b = k$,

$$w(R_1) \leq \cdots \leq w(R_a) \leq w(S_1) \leq \cdots \leq w(S_b),$$

and

$$w(\mathcal{P}) = w(S_1) + \cdots + w(S_b).$$

Note that a may be zero.

If $b = 1$, then the theorem is proved by just letting $\mathcal{P}' = \mathcal{P}$. So, in the following we assume that $b \geq 2$. We shall convert $S_1, \cdots, S_{b-1}, S_b$ to $S_1', \cdots, S_{b-1}', S_b'$ such that the first $b-1$ S_i''s are singletons. This is done as follows.

We pick the unique $\ell \in \left[1, \lceil \frac{b-1}{2} \rceil\right]$ such that

$$|S_1| + |S_2| + \cdots + |S_\ell| \geq b - 1$$

and

$$|S_1| + |S_2| + \cdots + |S_{\ell-1}| < b - 1. \tag{1}$$

Note that it may be the case that $\ell = 1$, and in this case we do not need the condition (1). Also note that since $b \geq 2$ and $\forall 1 \leq i \leq b$, $|S_i| \geq 2$, ℓ must be at most $\lceil \frac{b-1}{2} \rceil$.

Initially S_b' is empty. We merge all vertices in $S_{\ell+1}, \cdots, S_b$ into S_b'. Then we pick arbitrarily $b-1$ vertices from $S_1, \cdots, S_\ell$ to make $b-1$ singletons S_1', S_2', $\cdots$, S_{b-1}'. If there are still remaining vertices in $S_1, \cdots, S_\ell$ (in case that $|S_1| + |S_2| + \cdots + |S_\ell| > b-1$), we then move all of them to S_b'. This finishes the construction of S_1', $\cdots$, S_{b-1}', S_b'.

Since $\ell \leq \lceil \frac{b-1}{2} \rceil \leq \frac{1}{2} b$, the number of subsets $S_{\ell+1}, \cdots, S_b$ is at least half of b. In the above construction, all the happy edges in these subsets are kept in S_b'. Since $S_1, \cdots, S_b$ are in the non-decreasing order of the total weights of happy edges they contain, we know that

$$w(S_b') \geq \frac{1}{2}(w(S_1) + \cdots + w(S_b))$$

The desired k-partition $\mathcal{P}'$ is just $\{R_1, \cdots, R_a, S_1', \cdots, S_b'\}$. □

Algorithm $\mathcal{T}$ is the algorithm reducing Max k-Uncut to Densest $\bar{k}$-Subgraph. Since the subgraph G' found in step 2 contains $\bar{k}$ vertices, there are exactly $n - \bar{k} = k-1$ vertices in $V(G) \setminus V(G')$. So, in step 4 we can color them in colors $1, \cdots, k-1$, respectively.

Algorithm 2.3. (Algorithm $\mathcal{T}$ for Max k-Uncut)

Input: An instance (G, w, k) of Max k-Uncut.
Output: A k-partition of $V(G)$.
1 $\bar{k} \leftarrow n - (k-1)$.
2 Find a subgraph G' of G by an approximation algorithm for Densest $\bar{k}$-Subgraph on instance $(G, w, \bar{k})$.
3 Color all vertices in $V(G')$ in color k.
4 Color all vertices in $V(G) \setminus V(G')$ in colors $1, \cdots, k-1$, respectively.

Theorem 3. *Let α be the approximation ratio of* Densest k-Subgraph. *Then Algorithm $\mathcal{T}$ is a $\frac{\alpha}{2}$-approximation algorithm for the* Max k-Uncut *problem.*

By [4], α can be $\Omega(1/n^{1/4+\epsilon})$ for every small constant $\epsilon > 0$. (The ratio in [4] is for unweighted Densest k-Subgraph. Using the technique in [9], this ratio can be extended to weighted Densest k-Subgraph.) This means that Max k-Uncut can be approximated within $\frac{1}{2}\alpha = \Omega(1/n^{1/4+\epsilon})$ in polynomial time.

Proof (of Theorem 3). Let $\mathrm{OPT}_{\mathrm{M}k\mathrm{U}}$ be the optimal value of Max k-Uncut on instance (G, w, k). Let $\mathcal{P}^* = \{V_1, V_2, \cdots, V_k\}$ be the corresponding optimal solution. By Lemma 1, we can build a k-partition $\mathcal{P}' = \{V_1', V_2', \cdots, V_k'\}$ from $\mathcal{P}^*$ such that $V_1', \cdots, V_{k-1}'$ are singletons. This means that $|V_k'| = n - (k-1)$. So, V_k' is a feasible solution to Densest $\bar{k}$-Subgraph on instance $(G, w, \bar{k})$ satisfying

$$w(V_k') = w(\mathcal{P}') \geq \frac{1}{2} w(\mathcal{P}^*) = \frac{1}{2} \mathrm{OPT}_{\mathrm{M}k\mathrm{U}}, \tag{2}$$

where the inequality is by Lemma 1.

Note that G' is the subgraph found in step 2 by the approximation algorithm for Densest $\bar{k}$-Subgraph. There are $k-1$ vertices in $V(G) \setminus V(G')$. Then, step 4 builds $k-1$ singletons using the vertices in $V(G) \setminus V(G')$. These singletons, together with $V(G')$, constitute a k-partition, denoted by $\mathcal{P}$, which is a feasible solution to Max k-Uncut. We have

$$w(\mathcal{P}) = w(V(G')) \geq \alpha \cdot \mathrm{OPT}_{\mathrm{D}\bar{k}\mathrm{S}} \geq \alpha \cdot w(V_k') \underset{(2)}{\geq} \frac{\alpha}{2} \cdot \mathrm{OPT}_{\mathrm{M}k\mathrm{U}},$$

where the first inequality holds since G' is an α-approximate solution to the Densest $\bar{k}$-Subgraph instance $(G, w, \bar{k})$, and the second inequality holds since V_k' is a feasible solution to $(G, w, \bar{k})$. The theorem is proved. □

Note that the running time of Algorithm $\mathcal{T}$ does not depend on the construction time of the k-partition in Lemma 1. Lemma 1 is only used in the analysis of Algorithm $\mathcal{T}$. (This construction time is useful in the following Algorithm $\mathcal{C}$.)

Interestingly and somewhat surprisingly, Lemma 1 also implies the converse of Theorem 3: Densest k-Subgraph reduces to Max k-Uncut. This is shown in Algorithm $\mathcal{C}$ and Theorem 4.

Algorithm 2.4. (Algorithm $\mathcal{C}$ for Densest k-Subgraph)

Input: An instance (G, w, k) of Densest k-Subgraph.
Output: A vertex subset $V' \subseteq V(G)$ containing exactly k vertices.

1 $\bar{k} \leftarrow n - k + 1$.
2 Find a $\bar{k}$-partition $\mathcal{P} = \{V_1, V_2, \cdots, V_{\bar{k}}\}$ of $V(G)$ by an approximation algorithm for Max $\bar{k}$-Uncut on instance $(G, w, \bar{k})$.
3 Convert $\mathcal{P}$ to a $\bar{k}$-partition $\mathcal{P}' = \{V'_1, V'_2, \cdots, V'_{\bar{k}}\}$ by Lemma 1, where $|V'_{\bar{k}}| = n - (\bar{k} - 1) = k$.
4 **return** $V' \leftarrow V'_{\bar{k}}$.

Theorem 4. *If* Max k-Uncut *can be approximated within a factor of* α*, then* Densest k-Subgraph *can be approximated within a factor of* $\alpha/2$.

Proof. We design Algorithm $\mathcal{C}$ as the approximation algorithm for Densest k-Subgraph. Step 2 calls the supposed α-approximation algorithm for Max $\bar{k}$-Uncut. By Lemma 1, step 3 can be finished in polynomial time. Therefore, the overall running time of Algorithm $\mathcal{C}$ is polynomial.

Let V^* be an optimal solution to the Densest k-Subgraph instance (G, w, k), whose value is denoted by $\mathrm{OPT}_{\mathrm{D}k\mathrm{S}}$. Note that in Algorithm $\mathcal{C}$ we have $\bar{k} = n - k + 1$. By viewing each vertex in $V(G) \setminus V^*$ as a singleton, we can build a $\bar{k}$-partition $\mathcal{P}^\circ = \{V_1^\circ, V_2^\circ, \cdots, V_{\bar{k}}^\circ\}$, where $V_{\bar{k}}^\circ = V^*$. Obviously we have $w(\mathcal{P}^\circ) = \mathrm{OPT}_{\mathrm{D}k\mathrm{S}}$. A crucial observation is that $\mathcal{P}^\circ$ is a feasible solution to the Max $\bar{k}$-Uncut instance $(G, w, \bar{k})$. This helps us get the connection

$$\mathrm{OPT}_{\mathrm{M}\bar{k}\mathrm{U}} \geq \mathrm{OPT}_{\mathrm{D}k\mathrm{S}}, \tag{3}$$

where $\mathrm{OPT}_{\mathrm{M}\bar{k}\mathrm{U}}$ is the optimal value of the instance $(G, w, \bar{k})$ of Max $\bar{k}$-Uncut.

For the two $\bar{k}$-partitions $\mathcal{P}$ and $\mathcal{P}'$, we have $w(\mathcal{P}') \geq \frac{1}{2} w(\mathcal{P})$ by Lemma 1. Since Max $\bar{k}$-Uncut can be approximated within α, we have $w(\mathcal{P}) \geq \alpha \cdot \mathrm{OPT}_{\mathrm{M}\bar{k}\mathrm{U}}$. These facts, together with (3), conclude the theorem. □

Theorems 3 and 4 show that Max k-Uncut and Densest k-Subgraph are in fact equivalent in approximability up to a factor of two.

3 Approximation Hardness

3.1 Ruling Out Constant Factor Approximation

The approximability equivalence (up to a factor of two) of Max k-Uncut and Densest k-Subgraph naturally suggests that the approximation hardness results of Densest k-Subgraph may extend to Max k-Uncut. In particular, the following conditional hardness result holds.

Corollary 1. *If* Densest k-Subgraph *cannot be approximated within any constant factor, then so do* Max k-Uncut.

Under some appropriate complexity assumptions, people indeed proved that Densest k-Subgraph cannot be approximated within any constant factor. Raghavendra and Steurer [16] proved that assuming that the Unique Games with Small Set Expansion conjecture is true, it is NP-hard to approximate the Densest k-Subgraph problem within any constant factor. Alon *et al.* [2] also ruled out constant approximation factor for Densest k-Subgraph, under an average case hardness assumption. For the exact meaning of these complexity assumptions, we refer the reader to [2,16].

Khot [14] proved that assuming NP $\not\subseteq \cap_{\epsilon>0}\text{BPTIME}(2^{n^\epsilon})$, Densest k-Subgraph has no PTAS. However, this result cannot be extended to Max k-Uncut directly, since the approximability equivalence of Max k-Uncut and Densest k-Subgraph proved above omits a constant factor 2.

3.2 An Explicit Hardness Factor

The approximation hardness results for Densest k-Subgraph mentioned above all use stronger complexity assumptions than the general assumption P $\neq$ NP. In the following, we shall prove an approximation hardness result for Max k-Uncut, assuming that P $\neq$ NP. The proved hardness factor is $1-\frac{1}{2n^\epsilon}$, where $\epsilon > 0$ is an arbitrarily small constant, and n is the vertex number of the input graph. This result implies that, if P $\neq$ NP, Max k-Uncut does not admit FPTAS.

The hardness result $1-\frac{1}{2n^\epsilon}$ for Max k-Uncut is rather weak since Max k-Uncut is strongly NP-hard, and this (the strong NP-hardness) already rules out FPTAS. However, we make twofold contribution in proving such a result. First, we give an explicit expression of the approximation hardness factor of Max k-Uncut, instead of just speaking that it is strongly NP-hard. Second, we prove a technical lemma (Lemma 2), which gives an upper bound of the number of happy edges that can be produced by any k-partition on a graph with no $(r+1)$-clique. The technical lemma is of independent interest and may find more applications in related problems.

Lemma 2. *Any k-partition on an n-vertex undirected graph with no $(r+1)$-clique can produce at most $\frac{1}{2}(1-\frac{1}{r})u^2$ happy edges, where $u = n-(k-1)$.*

Håstad [13] proved the following remarkable approximation hardness result for the Max Clique problem: For any $\epsilon > 0$, unless P = NP, there is no polynomial time algorithm that approximates Max Clique within a factor of $n^{1/2-\epsilon}$, where n is the vertex number of the input graph. By this result and Lemma 2, we can prove that

Theorem 5. *For any $\epsilon > 0$, unless P = NP, there is no polynomial time algorithm that approximates* Max k-Uncut *within a factor of $1-\frac{n^{1/2-\epsilon}-1}{n^{1/2}-1}$, where n is the vertex number of the input graph. The hardness factor is $\leq 1-\frac{1}{2n^\epsilon}$ for sufficiently large n.*

The proof of Lemma 2 is rather complicated. Due to space limitation, the proofs of Lemma 2 and Theorem 5 are omitted here and will be given in the journal version of the paper.

Håstad [13] also proved that assuming ZPP $\neq$ NP, Max Clique cannot be approximated within $n^{1-\epsilon}$ for any small constant $\epsilon > 0$. However, for technical reasons, this stronger hardness factor cannot improve the result of Theorem 5 accordingly.

A corollary of Theorem 5 is that Max k-Uncut has no FPTAS, if P $\neq$ NP.

Corollary 2. Max k-Uncut *does not admit FPTAS, if P* $\neq$ *NP.*

Proof. Suppose for contradiction that there is an FPTAS for Max k-Uncut which for any small $\epsilon' > 0$, gets a $(1-\epsilon')$-approximation to Max k-Uncut instance I, with running time poly$(\frac{1}{\epsilon'}, |I|)$, where $|I|$ denotes the length of instance I, and poly() denotes some polynomial. Given any small constant $\epsilon > 0$, if we set $\epsilon' = \frac{1}{2n^{\epsilon}}$ and run the FPTAS, where n is the number of vertices in the input graph, then we can get a $(1-\frac{1}{2n^{\epsilon}})$-approximation to instance I in time poly$(2n^{\epsilon}, |I|)$ = poly$(|I|)$, contradicting Theorem 5. □

Acknowledgements. We thank Marek Chrobak for the initial helpful discussion on this topic. Peng Zhang is supported by the National Natural Science Foundation of China (61672323), the State Scholarship Fund of China, the Natural Science Foundation of Shandong Province (ZR2013FM030 and ZR2015FM008), and the Fundamental Research Funds of Shandong University (2015JC006). Chenchen Wu is supported by the National Natural Science Foundation of China (11501412). Dachuan Xu is supported by the National Natural Science Foundation of China (11371001 and 11531014) and Collaborative Innovation Center on Beijing Society-Building and Social Governance.

References

1. Agarwal, A., Charikar, M., Makarychev, K., Makarychev, Y.: $O(\sqrt{\log n})$ approximation algorithms for min uncut, min 2CNF deletion, and directed cut problems. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 573–581 (2005)
2. Alon, N., Arora, S., Manokaran, R., Moshkovitz, D., Weinstein, O.: Inapproximability of densest κ-subgraph from average case hardness. Manuscript (2011)
3. Bellare, M., Goldreich, O., Sudan, M.: Free bits, PCPs and non-approximability – towards tight results. SIAM J. Comput. **27**(3), 804–915 (1998)
4. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC), pp. 201–210 (2010)
5. Buchbinder, N., Naor, J., Schwartz, R.: Simplex partitioning via exponential clocks and the multiway cut problem. In: Proceedings of the Annual ACM Symposium on Theory of Computing (STOC), pp. 535–544 (2013)
6. Calinescu, G., Karloff, H., Rabani, Y.: An improved approximation algorithm for multiway cut. J. Comput. Syst. Sci. **60**(3), 564–574 (2000)
7. Choudhurya, S., Gaurb, D.R., Krishnamurtic, R.: An approximation algorithm for max k-uncut with capacity constraints. Optimization **61**(2), 143–150 (2012)
8. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press, Cambridge (2010)

9. Feige, U., Kortsarz, G., Peleg, D.: The dense k-subgraph problem. Algorithmica **29**, 410–421 (2001)
10. Frieze, A., Jerrum, M.: Improved approximation algorithms for max k-cut and max bisection. Algorithmica **18**, 67–81 (1997)
11. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
12. Goldschmidt, O., Hochbaum, D.: A polynomial algorithm for the k-cut problem for fixed k. Math. Oper. Res. **19**(1), 24–37 (1994)
13. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**, 105–142 (1999)
14. Khot, S.: Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In: Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 136–145 (2004)
15. Langberg, M., Rabani, Y., Swamy, C.: Approximation algorithms for graph homomorphism problems. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX/RANDOM -2006. LNCS, vol. 4110, pp. 176–187. Springer, Heidelberg (2006). doi:10.1007/11830924_18
16. Raghavendra, P., Steurer, D.: Graph expansion and the unique games conjecture. In: Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC), pp. 755–764 (2010)
17. Saran, H., Vazirani, V.: Finding k-cuts within twice the optimal. SIAM J. Comput. **24**, 101–108 (1995)
18. Wu, C., Xu, D., Du, D., Xu, W.: A complex semidefinite programming rounding approximation algorithm for the balanced max-3-uncut problem. In: Cai, Z., Zelikovsky, A., Bourgeois, A. (eds.) COCOON 2014. LNCS, vol. 8591, pp. 324–335. Springer, Heidelberg (2014). doi:10.1007/978-3-319-08783-2_28
19. Ye, Y., Zhang, J.: Approximation of dense-$n/2$-subgraph and the complement of min-bisection. J. Global Optim. **25**(1), 55–73 (2003)
20. Zhang, P., Jiang, T., Li, A.: Improved approximation algorithms for the maximum happy vertices and edges problems. In: Xu, D., Du, D., Du, D. (eds.) COCOON 2015. LNCS, vol. 9198, pp. 159–170. Springer, Heidelberg (2015). doi:10.1007/978-3-319-21398-9_13
21. Zhang, P., Li, A.: Algorithmic aspects of homophyly of networks. Theoret. Comput. Sci. **593**, 117–131 (2015)