Approximating the Dense Set-Cover Problem

Reuven Bar-Yehuda [1], Zehavit Kehat

Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000 Israel

Abstract

We study the Set-Cover problem, i.e. given a collection C of subsets of a finite set U, find a minimum size subset $C' \subseteq C$ such that every element in U belongs to at least one member of C. An instance (C, U) of the Set-Cover problem is k-bounded if the number of occurrences in C of any element is bounded by a constant $k \geq 2$.

We present an approximation algorithm for the k-bounded Set-Cover problem, that achieves the ratio $\frac{k}{k-(k-1)\cdot \sqrt[k]{1-\epsilon}} + o(1)$, where ϵ is defined as $|U|/\binom{|C|}{k}$. If ϵ is relatively high, we say that the problem is dense, and this ratio in this case is better than k, which is the best known constant ratio for this problem. In the case that the number of occurrences in C of any element is exactly $k = 2$ the problem is known as the Vertex-Cover problem. For dense graphs, our algorithm achieves an approximation ratio better than that of Nagamochi and Ibaraki [10], and the same approximation ratios as Karpinski and Zelikovsky [8]. In our algorithm we use a combinatorial property of the Set-Cover problem, which is based on the classical greedy algorithm for the Set-Cover problem. We use this property to define a "Greedy-Sequence", which is defined over a given instance of the Set-Cover problem and its cover.

In addition, we show evidence that the ratio we achieve for the ϵ-dense k-bounded set-cover problem is the best constant ratio one can expect. We do this by showing that finding a better constant ratio is as hard as finding a constant ratio better than k for the k-bounded set-cover problem in which the optimal cover is known to be of size at least $\frac{|C|}{k}$. (k is the best known constant ratio for this version of the k-bounded Set-Cover problem). We show a similar lower bound for the approximation ratio for the Vertex-Cover problem in ϵ-dense graphs.

Key words: Approximation Algorithm, Set-Cover, Vertex-Cover

Email address: `reuven@cs.technion.ac.il` (Reuven Bar-Yehuda).

[1] This research was supported by the fund for the promotion of research at the Technion.

1 Introduction

Given a collection $C = \{s_1, \ldots, s_n\}$ of subsets of a finite set U of size m, a sub-collection $C' \subseteq C$ is called a *set-cover* if every element in U belongs to at least one member of C'. The *Minimum Set Cover* problem is defined as follows: Given an instance (C, U), find a set-cover of minimum cardinality. Through out this paper we assume that all set cover instances (C, U) are simple, i.e. for every two elements $u_1, u_2 \in U$, there are two sets $s_1, s_2 \in C$ such that $u_1 \in s_1$, $u_1 \notin s_2$, $u_2 \in s_2$ and $u_2 \notin s_1$, otherwise one of these elements can be deleted without changing the problem. We call an instance (C, U) of the set-cover problem *k-bounded* if the number of occurrences in C of any element is bounded by a constant $k \geq 2$, and there exists an element which appears in C exactly k times. If the number of occurrences in C is the same for all elements in U we call the instance *homogeneous*. An instance is called *k-homogeneous* if it is k-bounded and also homogeneous. Note that the 2-homogeneous set-cover problem is known as the *Vertex-Cover* problem (where C and U are the vertex set and the edge set of a graph).

Given an instance (C, U) of the set-cover problem, a set-cover C' is called an *r-approximation*, if $|C'| \leq r \cdot |\mathrm{OPT}|$, where OPT is an optimal set cover for (C, U). We say that an algorithm has an approximation ratio of r, if for every instance it returns a cover which is an r-approximation.

Johnson [7] has presented the best known approximation algorithm for the set-cover problem which has a ratio of $1 + \ln m$. The best approximation ratio for the k-bounded set-cover problem is $k - \frac{(k-1)\cdot \ln\ln n}{\ln n}$ due to Halperin [5]. Bar-Yehuda and Even [1], and Hochbaum [6] presented k-approximation algorithms for the k-bounded set-cover problem. k is the best known constant ratio for the k-bounded problem.

Define $\epsilon = m/\frac{n^k}{k!}$. We call a k-bounded instance (C, U) *dense* if ϵ is relatively high (e.g., $m = \Omega(n^k)$). Formally, given a positive $\epsilon < 1$, we call a k-bounded instance of (C, U) *ϵ-dense* if $m/\frac{n^k}{k!} = \epsilon$. A graph $G = (V, E)$ is called ϵ-dense if $|E|/\frac{|V|^2}{2} = \epsilon$.

We present a $\frac{k}{k-(k-1)\cdot \sqrt[k]{1-\epsilon}}$-approximation algorithm for the ϵ-dense k-homogeneous set-cover problem, and a $\frac{k}{k-(k-1)\cdot \sqrt[k]{1-\epsilon}} + o(1)$-approximation algorithm for the k-bounded set-cover problem. As far as we know, there were no previous works for the ϵ-dense k-homogeneous and k-bounded problems. In the special case of ϵ-dense graphs, our algorithm achieves a better approximation ratio than the algorithm presented by Nagamochi and Ibaraki [10], and the same ratio as the one presented by Karpinski and Zelikovsky [8].

The parameter ϵ, which represents the density of the problem, can be defined in other ways. Given a k-bounded instance (C, U), we assume it is simple, thus the maximum number of elements m is bounded by $\binom{n}{k}$. Therefore, another definition, possibility more natural, is $\epsilon = m/\binom{n}{k}$. We discuss the former definition ($\epsilon = m/\frac{n^k}{k!}$), as did Karpinski and Zelikovsky [8] with respect to dense graphs. However, we show that when using the latter definition we get similar results (up to an additive factor of $o(1)$).

We show evidence that the ratio we achieve for the ϵ-dense k-bounded set-cover problem is the best constant ratio one can expect. We do this by showing that finding a better constant ratio is as hard as finding a constant ratio better than k for the k-bounded set-cover problem in which the optimal cover is known to be at least of size $\frac{n}{k}$. (k is the best known constant ratio for this version of the k-bounded set-cover problem.). This is also true for the case of ϵ-dense graphs, in which our ratio is the best constant ratio, unless there exists an r-approximation algorithm for the vertex-cover problem, where r is a constant less than 2. It has been recently shown by Eremeev [3], that the vertex-cover problem in all-over ϵ-dense graphs is NP-hard to approximate within a factor less than $\frac{7+\epsilon}{6+2\epsilon}$. Clearly there is still a gap between the approximation $\frac{2}{1+\epsilon}$ for this problem, and the factor which is NP-hard to obtain. We try to measure the hardness of finding better approximation ratios than the ratios $\frac{2}{1+\epsilon}$ and $\frac{2}{2-\sqrt{1-\epsilon}}$ for the vertex-cover problem in all-over ϵ-dense graphs and in ϵ-dense graphs, respectively.

The remainder of the paper is organized as follows: In Section 2 we present a property of a cover in the set cover problem, which we use to define "the greedy sequence". We use this property to construct an approximation algorithm for the k-bounded set-cover problem in Section 3. In Section 4 we analyze the performance of the algorithm when applied to ϵ-dense k-bounded set-cover problem, and we show that the ratio we achieve is the best constant ratio we can expect. Section 5 deals with the special case of the 2-homogeneous set-cover problem, which is the vertex-cover problem.

2 Greedy-Sequences

The classical algorithm for the set-cover problem is the *greedy algorithm* [7], which in each iteration selects a set that covers a maximal number of uncovered elements. Given (C, U) we define *greedy-sort(C,U)* as a list of the sets in C sorted according to the order of their selection by the greedy algorithm.

Given an instance (C, U), let k be the maximum number of occurrences of any element in C. Given a cover $C' \subseteq C$ we define the *greedy-sequence of* (C', C, U) as a vector $(n_k, ..., n_1)$. The first n_k sets in greedy-sort(C, U) are in C' and the following set, which we denote by s_{n_k+1}, is not. Let $\tilde{s}_{n_k+1}$ denote the set of the elements of s_{n_k+1} which are still uncovered when it is selected by the greedy algorithm. Consider the instance in which we need to cover only the elements in $\tilde{s}_{n_k+1}$, i.e., $(C_{k-1}, U_{k-1}) = (\{s \cap \tilde{s}_{n_k+1} | \ s \in C \setminus \{s_{n_k+1}\}\}, U \cap \tilde{s}_{n_k+1})$, which is a $(k-1)$-bounded instance. The first n_{k-1} sets in greedy-sort($\{s \cap \tilde{s}_{n_k+1} \mid s \in C \setminus \{s_{n_k+1}\}\}, U \cap \tilde{s}_{n_k+1}$) are in C' and the next set is not. The other n_i's are defined similarly, and we finally get a 2-bounded instance, (C_2, U_2), in which the first n_2 sets in greedy-sort(C_2, U_2) are in C' and the next set s_{n_2+1} is not. Therefore the collection of sets $\{s | \ s \in C_2 \text{ and } s \cap \tilde{s}_{n_2+1} \neq \phi\}$ is in C'. We denote by n_1 the size of this collection. In (C_2, U_2) every element appears in at most two sets, so each of these n_1 sets covers at most one element of $\tilde{s}_{n_2+1}$, and therefore $|\tilde{s}_{n_2+1}| = n_1$. (See Figures A.1 and A.2.)

We say that a sequence $(n_k, ..., n_1)$ is a *possible greedy-sequence of* (C, U), if there exists a set-cover $C' \subseteq C$ such that the sequence $(n_k, ..., n_1)$ is the greedy-sequence of (C', C, U).

We present a procedure called *Extract-Cover*, which receives as an input an instance (C, U) and a sequence of numbers $(n_k, ..., n_2)$, and returns a partial cover, denoted by SC, such that $(n_k, ..., n_2)$ is the prefix of a possible greedy-sequence of (C, U). Note that given a sequence $n_k, ..., n_2$ and an instance (C, U) the number n_1 is defined uniquely, therefore, $(n_k, ..., n_2)$ is sufficient when calling Procedure Extract-Cover.

Procedure Extract-Cover$(C, U, (n_k, ..., n_2))$
  $(C_k, U_k) = (C, U)$
  $SC = \phi$
  For $i = k$ downto 2
    If $n_i \geq |C_i|$
      Return "illegal greedy-sequence"
    $SC = SC \cup \{$ the first n_i sets in greedy-sort(C_i, U_i) $\}$
    $s_{n_i+1} =$ the next set in greedy-sort(C_i, U_i)
    If $i > 2$
      $(C_{i-1}, U_{i-1}) = (\{s \cap (s_{n_i+1} \setminus SC) | \ s \in C_i\}, U_i \cap (s_{n_i+1} \setminus SC))$
  $SC = SC \cup \{s | \ s \in C_2 \text{ and } s \cap (s_{n_2+1} \setminus SC) \neq \phi\}$
  Return SC

In the next section we use this procedure in our main approximation algorithm. We would like to find a lower bound for the size of the partial cover, which is returned by the procedure. The resulting partial cover is of the same size as the sum of the numbers in the input greedy-sequence. Given a homogeneous instance (C, U) of the set-cover problem, we prove that for any possible greedy-sequence of (C, U), there exists a lower bound for $\sum_{i=1}^{k} n_i$, which is a function of $\epsilon = m / \frac{n^k}{k!}$.

Lemma 1 *Let (C, U) be a homogeneous instance of the set-cover problem and let $(n_k, ..., n_1)$ be a possible greedy-sequence of (C, U). Then*

$$\sum_{i=1}^{k} n_i \geq (1 - \sqrt[k]{1-\epsilon}) \cdot n.$$

Proof. Let x_i denote the number of elements in the set s_{n_i+1}, which are still uncovered when $s_{n_{i+1}}$ is chosen by the greedy algorithm, for $2 \leq i \leq k$.. Let S_k be the first n_k sets in greedy-sort(C, U) and let U_A be the set of elements that are covered by S_k. Denote $U_B = U \setminus U_A$. Examine the instance $(C \setminus S_k, U_B)$. The number of sets in $C \setminus S_k$ is $n - n_k$ and the maximum size of a set is x_k. Also, the number of occurrences in $C \setminus S_k$ of any element is exactly k, therefore, $|U_A| \leq \binom{n}{k} - \binom{n-n_k}{k}$ and $k \cdot |U_B| = \sum_{s \in C \setminus S_k} |s| \leq x_k \cdot (n - n_k)$. Thus

$$|U| = |U_A| + |U_B| \leq \binom{n}{k} - \binom{n-n_k}{k} + \frac{1}{k} \cdot x_k \cdot (n - n_k) \tag{1}$$

By definition $|U_i| = x_{i+1}$, for $2 \leq i < k$. Let $C_k = C$, $U_k = U$ and $x_{k+1} = |U_k|$. We can bound $|U_i|$ similarly to the way we have bounded $|U|$ in (1), therefore, any possible-greedy-sequence, $(n_k, ..., n_1)$, has to satisfy the following, where $2 \leq i \leq k$:

$$\begin{aligned}
x_{i+1} = |U_i| &\leq \binom{|C_i|}{i} - \binom{|C_i| - n_i}{i} + \frac{x_i}{i} \cdot (|C_i| - n_i) && (2)\\
&\leq \binom{n - \sum_{j=i+1}^{k} n_j}{i} - \binom{n - \sum_{j=i}^{k} n_j}{i} + \frac{x_i}{i} \cdot (n - \sum_{j=i}^{k} n_j) && (3)\\
&\leq \frac{(n - \sum_{j=i+1}^{k} n_j)^i}{i!} - \frac{(n - \sum_{j=i}^{k} n_j)^i}{i!} + \frac{x_i}{i} \cdot (n - \sum_{j=i}^{k} n_j) && (4)
\end{aligned}$$

The inequality from (3) is due to $|C_i| \leq n - \sum_{j=i+1}^{k} n_j$, and due to the following claim: For any x, y, z, k such that $0 \leq z \leq x$ and $1 \leq k \leq x - z$, if $x \leq y$ then $\binom{x}{k} - \binom{x-z}{k} \leq \binom{y}{k} - \binom{y-z}{k}$. The inequality from (4) is due the following claim: For any n, z, k such that $1 \leq k \leq n - z$ and $0 \leq z \leq n$: $\binom{n}{k} - \binom{n-z}{k} \leq \frac{n^k}{k!} - \frac{(n-z)^k}{k!}$. Both claims can be easily proven by induction on k.

Therefore, we get the following constraint for any i such that $2 \leq i \leq k$:

$$i! \cdot x_{i+1} = i! \cdot |U_i| \leq (n - \sum_{j=i+1}^{k} n_j)^i - (n - \sum_{j=i}^{k} n_j)^i + (i-1)! \cdot x_i \cdot (n - \sum_{j=i}^{k} n_j) \quad (5)$$

It follows that in order to bound $\sum_{i=1}^{k} n_i$ for some possible-greedy-sequence $(n_k, ..., n_1)$, it is sufficient to solve the following optimization problem:

$$\begin{aligned} &\text{Min } \textstyle\sum_{i=1}^{k} n_i \\ &\text{s.t. constraint } \ (5) \\ &\qquad \text{and } \textstyle\sum_{i=1}^{k} n_i \leq n \\ &\qquad \text{and } n_i, x_i \in \mathbb{R} \end{aligned} \qquad (6)$$

In the appendix we show that the optimal solution for this optimization problem is $(1 - \sqrt[k]{1-\epsilon}) \cdot n$. Note that we find an optimal solution for the relaxed problem, where $n_i, x_i \in \mathbb{R}$, instead of $n_i, x_i \in \mathbb{N}$. $\square$

3 Approximating the k-Bounded Set-Cover Problem

In this section we present an approximation algorithm called Multi-Greedy. This algorithm checks all the possibilities of a prefix $(n_k, ..., n_2)$ of a greedy-sequence. For each possibility it calls Procedure Extract-Cover in order to get a partial cover, which is a subset of a cover that has such a greedy-sequence, and extends it to a possible set-cover. Then the algorithm chooses a set-cover of minimum size among all these possibilities. At least one possibility will be a prefix of a greedy-sequence of some optimal cover of the problem. In this case, Procedure Extract-Cover will return a subset of an optimal cover, whose size is equal to the sum of the numbers in the corresponding greedy-sequence vector.

Algorithm Multi-Greedy(C,U)

APPX $= C$

Let $U_k =$ the set of all the elements that belong to exactly k sets

For each possible prefix $n_k, ..., n_2$ of a greedy-sequence

$SC =$ Extract-Cover$(C, U_k, (n_k, ..., n_2))$

$Covered = \bigcup_{s \in SC} s$

Find $kSC =$ a k-approximation for $(C \setminus SC, U \setminus Covered)$

If $|SC \cup kSC| < |\mathrm{APPX}|$

APPX $= SC \cup kSC$

Return APPX

Let OPT denote some optimal cover of a given k-bounded instance, (C, U). We define the sequence $(\hat{n}_k, ..., \hat{n}_1)$ as the greedy-sequence of (OPT, C, U_k). Algorithm Multi-Greedy checks all the possible sequences, therefore, there will be a step in the algorithm when it calls Procedure Extract-Cover with the sequence $(\hat{n}_k, ..., \hat{n}_2)$. Let $\widehat{SC}$ and $\widehat{Covered}$ denote the values of the parameters SC and $Covered$ in this specific step. Note that as mentioned in Section 2, given a sequence $\hat{n}_k, ..., \hat{n}_2$ and an instance (C, U), the number $\hat{n}_1$ is defined uniquely.

Our first step in the analysis of Algorithm Multi-Greedy is to bound the size of the resulting cover APPX as a function of $\hat{n}_k, ..., \hat{n}_1$.

Lemma 2

$$|\mathrm{APPX}| \leq \frac{k}{1 + (k-1) \cdot \frac{\sum_{i=1}^{k} \hat{n}_i}{n}} \cdot |\mathrm{OPT}|$$

Proof. By definition $\widehat{SC} \subseteq \mathrm{OPT}$ and $|\widehat{SC}| = \sum_{i=1}^{k} \hat{n}_i$. The sets in $\widehat{SC}$ cover all the elements in $\widehat{Covered}$. Therefore the rest of the sets in OPT, denote them by OPT', cover the elements in $U \setminus \widehat{Covered}$. Moreover, OPT' is an optimal cover of $U \setminus \widehat{Covered}$. Algorithm Multi-Greedy applies a k-approximation algorithm on $U \setminus \widehat{Covered}$, therefore its result will have the size of at most $k \cdot |\mathrm{OPT}'|$.

Let $R = \frac{|\mathrm{APPX}|}{|\mathrm{OPT}|}$. By the algorithm, APPX is the set of minimum size among all possible covers, in particular it is smaller or equal to the possibility that includes $\widehat{SC}$, therefore we can bound the set cover APPX as follows:

$$R \cdot |\mathrm{OPT}| = |\mathrm{APPX}| \leq \sum_{i=1}^{k} \hat{n}_i + k \cdot |\mathrm{OPT}'| = \sum_{i=1}^{k} \hat{n}_i + k \cdot (|\mathrm{OPT}| - \sum_{i=1}^{k} \hat{n}_i) =$$

$$k \cdot |\mathrm{OPT}| - (k-1) \cdot \sum_{i=1}^{k} \hat{n}_i = k \cdot |\mathrm{OPT}| - (k-1) \cdot \frac{\sum_{i=1}^{k} \hat{n}_i}{n} \cdot n$$

By definition $n \geq |\mathrm{APPX}| = R \cdot |\mathrm{OPT}|$, thus,

$$R \cdot |\mathrm{OPT}| = |\mathrm{APPX}| \leq k \cdot |\mathrm{OPT}| - (k-1) \cdot \frac{\sum_{i=1}^{k} \hat{n}_i}{n} \cdot R \cdot |\mathrm{OPT}| = \\ |\mathrm{OPT}| \cdot (k - (k-1) \cdot \frac{\sum_{i=1}^{k} \hat{n}_i}{n} \cdot R)$$

and, therefore,

$$R \leq \frac{k}{1 + (k-1) \cdot \frac{\sum_{i=1}^{k} \hat{n}_i}{n}}$$

□

The following step is to find a lower bound for the expression $\sum_{i=1}^{k} \hat{n}_i$ as a function of $\epsilon = m/\frac{n^k}{k!}$ (for the same $\hat{n}_k, ..., \hat{n}_1$ as defined above).

Lemma 3 *Given a k-bounded instance (C, U)*

$$\sum_{i=1}^{k} \hat{n}_i \geq (1 - \sqrt[k]{1-\epsilon} - o(1)) \cdot n$$

Proof. Given a k-bounded instance (C, U), we consider the following devision of U into two sets: U_k is defined as the set of all the elements that belong to exactly k sets in C, and $\overline{U}_k = U \setminus U_k$. It is simple to see that $|\overline{U}_k| = o(\frac{n^k}{k!})$. Therefore, by the definition of ϵ we get

$$\epsilon \cdot \frac{n^k}{k!} = |U| = |U_k| + |\overline{U}_k| = |U_k| + o(\frac{n^k}{k!})$$
$$|U_k| = \frac{n^k}{k!} \cdot (\epsilon - o(1))$$

We denote $\epsilon_k = \epsilon - o(1)$. The proof of Lemma (1) is based on Constraint (5), which has been constructed under the assumption that the instance of the problem is homogeneous. This lemma gives a lower bound for any possible greedy-sequence of the homogeneous instance (C, U_k), in particular it is true for the sequence $\hat{n}_k, ..., \hat{n}_1$, therefore,

$$\sum_{i=1}^{k} \hat{n}_i \geq (1 - \sqrt[k]{1-\epsilon_k}) \cdot n$$

By the definition of ϵ_k

$$\sum_{i=1}^{k} \hat{n}_i \geq (1 - \sqrt[k]{1 - \epsilon + o(1)}) \cdot n = (1 - \sqrt[k]{1-\epsilon} - o(1)) \cdot n$$

□

Theorem 4 *Algorithm Multi-Greedy has an approximation ratio of* $\frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}}$ *for the k-homogeneous set-cover problem, and a ratio of* $\frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}} + o(1)$ *for the k-bounded set-cover problem.*

Proof. Given a k-bounded instance (C, U), by Lemma (2),

$$|\mathrm{APPX}| \leq \frac{k}{1 + (k-1) \cdot \frac{\sum_{i=1}^{k} \hat{n}_i}{n}} \cdot |\mathrm{OPT}|.$$

If (C, U) is homogeneous, then $U_k = U$, and $\hat{n}_k, ..., \hat{n}_1$ is the greedy-sequence of (OPT, C, U). Lemma (1) gives a lower bound for any possible greedy-sequence of (C, U), in particular for the sequence $\hat{n}_k, ..., \hat{n}_1$, thus,

$$\sum_{i=1}^{k} \hat{n}_i \geq (1 - \sqrt[k]{1-\epsilon}) \cdot n$$

and, therefore,

$$\begin{aligned} |\mathrm{APPX}| &\leq \frac{k}{1 + (k-1) \cdot \frac{(1-\sqrt[k]{1-\epsilon})\cdot n}{n}} \cdot |\mathrm{OPT}| \\ &= \frac{k}{k - (k-1) \cdot \sqrt[k]{1-\epsilon}} \cdot |\mathrm{OPT}|. \end{aligned}$$

Otherwise, by lemma (3) we get,

$$\sum_{i=1}^{k} \hat{n}_i \geq (1 - \sqrt[k]{1-\epsilon} - o(1)) \cdot n$$

and, therefore,

$$
\begin{aligned}
|\mathrm{APPX}| &\leq \frac{k}{1+(k-1)\cdot\frac{(1-\sqrt[k]{1-\epsilon}-o(1))\cdot n}{n}}\cdot|\mathrm{OPT}| \\
&= \frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}-o(1)}\cdot|\mathrm{OPT}| \\
&= \left(\frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}}+o(1)\right)\cdot|\mathrm{OPT}|.
\end{aligned}
$$

□

Corollary 5 *Algorithm Multi-Greedy has an approximation ratio of* $\frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}}$ *for the* ϵ*-dense* k*-homogeneous set-cover problem, and a ratio of* $\frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}}+o(1)$ *for the* ϵ*-dense* k*-bounded set-cover problem.*

Now we intend to bound the approximation ratio of Algorithm Multi-Greedy as a function of a possibly more natural expression, $\binom{n}{k}$, which is the maximum number of possible elements in a k-bounded instance, (C,U), of the set-cover problem.

Theorem 6 *Algorithm Multi-Greedy has an approximation ratio of*

$$
\frac{k}{k-(k-1)\cdot\sqrt[k]{1-\frac{m}{\binom{n}{k}}}}+o(1)
$$

for the k*-bounded set-cover problem.*

Proof. The approximation ratio from Theorem (4), denoted by *Ratio*, was computed for $\epsilon = m/\frac{n^k}{k!}$, therefore,

$$
\begin{aligned}
Ratio &= \frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}}+o(1) \\
&= \frac{k}{k-(k-1)\cdot\sqrt[k]{1-\frac{m}{\binom{n}{k}+o(1)}}}+o(1) \\
&= \frac{k}{k-(k-1)\cdot\sqrt[k]{1-\frac{m}{\binom{n}{k}}+o(1)}}+o(1) \\
&= \frac{k}{k-(k-1)\cdot\sqrt[k]{1-\frac{m}{\binom{n}{k}}}-o(1)}+o(1) \\
&= \frac{k}{k-(k-1)\cdot\sqrt[k]{1-\frac{m}{\binom{n}{k}}}}+o(1)
\end{aligned}
$$

□

Since there are n^{k-1} candidates for $n_k, ..., n_2$, and each candidate can be checked in polynomial time, we get the following lemma:

Lemma 7 *Algorithm Multi-Greedy can be implemented in polynomial time.*

4 Hardness Result

The best constant approximation ratio that is known today for the k-bounded set-cover problem is k (see [1] and [6]). This is the best constant approximation ratio even if we know that the size of an optimal cover of a problem is at least $\frac{n}{k}$. The following theorem tries to measure the hardness of finding an approximation for the ϵ-dense k-bounded set-cover problem.

Theorem 8 *For every fixed constant ϵ there is no polynomial time algorithm with an approximation ratio better than $\frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}}$ by a constant for the ϵ-dense k-bounded set-cover problem, unless there is a polynomial time algorithm with an approximation ratio better than k by a constant for the general k-bounded set-cover problem in which the size of the optimal cover is at least $\frac{n}{k}$.*

Proof. We call a k-hyper-graph a hyper-graph in which each edge has at most k endpoints, for a given k. In this proof we consider the vertex-cover problem in k-hyper-graphs, which is equivalent to the k-bounded set-cover problem.

We assume there exists a polynomial time algorithm A with an approximation ratio of R for the vertex-cover problem in ϵ-dense k-hyper-graphs, and we show a lower bound for R. Given a general k-hyper-graph $G(V, E)$, $(n = |V|)$, we build an ϵ-dense k-hyper-graph $G'(V', E')$, $(n' = |V'|)$. Then we apply Algorithm A on G' and get a cover $VC_A(G')$, which is by assumption an R-approximation for G'. We show that if R is less than this lower bound, then the part of $VC_A(G')$, that is a cover of G, has an approximation ratio better than k by a constant. Therefore, we get an approximation ratio better than k by a constant for the vertex-cover problem in a general k-hyper-graph, which is known to be impossible.

We build G' as follows: We add a clique of $\alpha n'$ hyper-vertices to V, and add all possible hyper-edges with k endpoints, such that their endpoints are from V and at least one endpoint is from the clique. Then, $n' = \alpha n' + n \Rightarrow n' = \frac{n}{1-\alpha}$. We denote by OPT the optimal vertex cover of G, and by OPT' the optimal vertex cover of G'.

Let us look at the total number of hyper-edges in G': This has to be at least all possible hyper-edges in a k-hyper-graph with n' hyper-vertices, possibly without the hyper-edges, which all their endpoints are from V. Therefore,

$$
\begin{aligned}
|E'| &\geq \binom{n'}{k} - \binom{n'-\alpha n'}{k} \\
&= \frac{1}{k!}\cdot n'\cdot(n'-1)\cdots(n'-k+1) \\
&\quad -\frac{1}{k!}\cdot(n'-\alpha n')\cdot(n'-\alpha n'-1)\cdots(n'-\alpha n'-k+1) \\
&\geq \frac{1}{k!}\cdot(n'-k+1)^k - \frac{1}{k!}(n'-\alpha n')^k = \frac{1}{k!}\cdot n'^k\cdot((1-\frac{k-1}{n'})^k-(1-\alpha)^k)
\end{aligned}
$$

Given ϵ, we try to find a value of α for which $|E'| = \epsilon \cdot \frac{n'^k}{k!}$. Therefore by $(1-\frac{k-1}{n'})^k - (1-\alpha)^k = \epsilon - o(1)$, we get that G' is ϵ-dense for $\alpha = 1 - \sqrt[k]{(1-\frac{k-1}{n'})^k - \epsilon + o(1)}$.

Any optimal cover of G' includes all the hyper-vertices of the clique. Also, we can assume that $VC_A(G')$ includes them, otherwise all the hyper-vertices of V' would be in this cover (except maybe a constant number of hyper-vertices). Therefore, $|\text{OPT'}| = \alpha n' + |\text{OPT}|$.

The hyper-vertices of the clique cover all the hyper-edges of the clique and the hyper-edges between the clique and V, therefore, the other hyper-vertices in the cover $VC_A(G')$ are a cover for G, which we denote by $VC_A(G)$.
By the above and by the assumption on Algorithm A:

$$|VC_A(G')| = \alpha n' + |VC_A(G)| \leq R\cdot|\text{OPT'}| = R\cdot(\alpha n' + |\text{OPT}|)$$

$$|VC_A(G)| \leq \alpha n'\cdot(R-1) + R\cdot|\text{OPT}|$$

By our assumption on G, $n \leq k\cdot|\text{OPT}|$, thus, $n' \leq \frac{k}{1-\alpha}\cdot|\text{OPT}|$. Therefore, we get,

$$|VC_A(G)| \leq \frac{\alpha k}{1-\alpha}\cdot(R-1)\cdot|\text{OPT}| + R\cdot|\text{OPT}| = |\text{OPT}|\cdot(\frac{\alpha k}{1-\alpha}\cdot(R-1)+R)$$

We assume that $\frac{\alpha k}{1-\alpha}(R-1)+R \geq k$, otherwise we get an approximation ratio which is better than k for the general k-hyper-graph G.
Then,

$$R \geq \frac{k}{1+(k-1)\cdot\alpha}$$

By choosing $\alpha = 1 - \sqrt[k]{(1-\frac{k-1}{n'})^k - \epsilon + o(1)}$ we get

$$
\begin{aligned}
R &\geq \frac{k}{k-(k-1)\cdot\sqrt[k]{(1-\frac{k-1}{n'})^k - \epsilon + o(1)}} \\
&\geq \frac{k}{k-(k-1)\cdot\sqrt[k]{1-\epsilon}} - o(1)
\end{aligned}
$$

Note that the value of α is legal only for $\epsilon \leq (1-\frac{k-1}{n'})^k$, which is true asymptotically for all $n' \geq \frac{k-1}{1-\sqrt[k]{\epsilon}}$. □

5 The Vertex-Cover Problem in ϵ-Dense Graphs

The case of the k-homogeneous set-cover problem when $k=2$ is known as the *Vertex-Cover* problem. Karpinski and Zelikovsky [8] discuss two kinds of dense graphs: For some $\epsilon > 0$, they call a graph $G(V,E)$ *all-over ϵ-dense* if every vertex in G has at least $\epsilon|V|$ neighbors, and they call a graph $G(V,E)$ *ϵ-dense* if the number of edges is relatively high, i.e. $|E| \geq \frac{1}{2}\epsilon|V|^2$. The later definition is similar to our definition of the ϵ-dense 2-homogeneous set-cover problem.

Algorithm Multi-Greedy achieves the same approximation ratios of $\frac{2}{2-\sqrt{1-\epsilon}}$ and $\frac{2}{1+\epsilon}$ as Karpinski and Zelikovsky in ϵ-dense graphs and in all-over ϵ-dense graphs, respectively. The ratio for ϵ-dense graphs is a special case of Corollary (5). It can be easily shown that the algorithm achieves the ratio of $\frac{2}{1+\epsilon}$ in the case of all-over ϵ-dense graphs.

The best approximation ratios known for the vertex-cover problem are $2 - \frac{2\log\log|V|}{2\log|V|}$ due to Monien and Speckenmeyer [9] and due to Bar-Yehuda and Even [2], and $2 - \frac{2\ln\ln|V|}{\ln|V|}\cdot(1-o(1))$ due to Halperin [5]. It has been recently shown by Eremeev [3], that the vertex-cover problem in all-over ϵ-dense graphs is NP-hard to approximate within a factor less than $\frac{7+\epsilon}{6+2\epsilon}$. Clearly there is still a gap between the approximation $\frac{2}{1+\epsilon}$ for this problem, and the factor which is NP-hard to obtain. The following theorem tries to measure the hardness of finding better approximation ratios than the ratios $\frac{2}{1+\epsilon}$ and $\frac{2}{2-\sqrt{1-\epsilon}}$ for the vertex-cover problem in all-over ϵ-dense graphs and in ϵ-dense graphs, respectively.

Theorem 9 *For every fixed constant ϵ there is no polynomial time algorithm with an approximation ratio better than $\frac{2}{1+\epsilon}$ and $\frac{2}{2-\sqrt{1-\epsilon}}$ by a constant for the vertex-cover problem in all-over ϵ-dense and in ϵ-dense graphs, unless there* *is a polynomial time algorithm for the vertex-cover problem in general graphs with an approximation ratio better than 2 by a constant.*

Proof. By Theorem (8), for every fixed constant ϵ there cannot be a polynomial time algorithm with an approximation ratio better by a constant than $\frac{2}{2-\sqrt{1-\epsilon}}$ for the vertex-cover problem in ϵ-dense graphs, unless there is a polynomial time algorithm with an approximation ratio better by a constant than 2 for the general vertex-cover problem in graphs in which the size of the optimal cover is at least $\frac{n}{2}$. By Nemhauser and Trotter (see [11] and [2]) we can assume that for any given graph $G(V, E)$, the size of the optimal vertex-cover of the graph is at least $\frac{n}{2}$. Therefore the claim is true for any graph.

The case of all-over ϵ-dense graphs can be proven similarly to the proof of Theorem (8), by choosing $\alpha = \epsilon$. □

Acknowledgments

We would especially like to thank Dror Rawitz for the helpful discussions and corrections, and Ronit Reger for her helpful suggestions. Many thanks to Hadas Heier for her help in typing the manuscript.

References

[1] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.

[2] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[3] A. Eremeev, On Some Approximation Algorithms for Dense Vertex Cover Problem . *Proceedings of SOR'99, Springer-Verlag*, pp.58-62, 2000.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[5] E. Halperin, Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs, *Proc. 11th Ann. ACM-SIAM Symp. on Discrete Algorithms, ACM-SIAM*, 2000.

[6] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

[7] D. S. Johnson, Approximation algorithms for combinatorial problems *J. Comput. System Sci.*, 9, 256-278, 1974.

[8] M. Karpinski and A. Zelikovsky, Approximating dense cases of covering problems, *Proc. DIMACS Workshop on Network Design: Connectivity and Facilities Location, Princeton,* Vol. 40 , pp. 169-178. 1998 (the slightly extended version appeared as a preprint at ICSI in Berkeley).

[9] B. Monien and E. Speckenmeyer, Ramsey numbers and an approximation algorithm for the vertex cover problem , *Acta Inf. 22,* 115-123, 1985.

[10] H. Nagamochi and T. Ibaraki, An approximation of the minimum vertex cover in a graph. *Japan J. Indust. Appl. Math.*, 16, 369-375. 1999.

[11] G. L. Nemhauser and L. E. Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

A

In the proof of Lemma (1) we presented the following optimization problem. In this section we prove that the optimal solution of this optimization problem is $(1 - \sqrt[k]{1-\epsilon}) \cdot n$:

$$\begin{aligned} &\text{Min } \textstyle\sum_{i=1}^{k} n_i \\ &\text{s.t. } (5) \text{ and } \textstyle\sum_{i=1}^{k} n_i \le n \\ &\qquad n_i, x_i \in \mathbb{R} \end{aligned} \tag{A.1}$$

Where (5) is as follows:

$$i! \cdot x_{i+1} = i! \cdot |U_i| \le (n - \sum_{j=i+1}^{k} n_j)^i - (n - \sum_{j=i}^{k} n_j)^i + (i-1)! \cdot x_i \cdot (n - \sum_{j=i}^{k} n_j)$$

for any i such that $2 \le i \le k$.

Proof. We intend to show that there exists a solution, whose value is N_{min} for the problem, that satisfies:

$$\begin{aligned} &\forall\ 1 \le i < k \ \ n_i = 0 \\ &\forall\ 2 \le i \le k \ \ x_i = 0 \\ &\ n_k = N_{min} \end{aligned} \tag{A.2}$$

Therefore by using (5) and by the definition of ϵ,

$$\begin{aligned}
\epsilon \cdot n^k = k! \cdot |U| &\leq n^k - (n - n_k)^k \\
\epsilon \cdot n^k &\leq n^k - n^k \cdot (1 - \frac{n_k}{n})^k \\
\epsilon &\leq 1 - (1 - \frac{n_k}{n})^k \\
n_k &\geq (1 - \sqrt[k]{1 - \epsilon}) \cdot n
\end{aligned}$$

Let $N' = (n'_k, ..., n'_1)$ be an optimal solution for the problem which was stated in (A.1), i.e. $n'_k + ... + n'_1 = N_{min}$. And let $(x'_k, ..., x'_2)$ be the suitable parameters for the solution N'. If N' satisfies (A.2) then we are done. Otherwise we show that we can take the optimal solution N', and build another optimal solution from it, which satisfies (A.2). We also show that after this process Constraint (5) is still satisfied.

We show the following claim:

For each $2 \leq i \leq k$
if for each $j < i$ $n_j = 0$ and $x_j = 0$
then $x_i = 0$.

In the special case of $i = 2$, by definition $x_2 = n_1$, so $x_2 = 0$ whenever $n_1 = 0$. By (5) we get for $2 < i \leq k$:

$$(i-1)! \cdot x_i \leq (n - \sum_{j=i}^{k} n_j)^{i-1} - (n - \sum_{j=i-1}^{k} n_j)^{i-1} + (i-2)! \cdot x_{i-1} \cdot (n - \sum_{j=i-1}^{k} n_j)$$

By the assumption, $n_{i-1} = 0$ and $x_{i-1} = 0$, therefore $x_i \leq 0$, which means that $x_i = 0$.

Now we intend to show how we can change N' into a suitable optimal solution for (A.1) that satisfies (A.2): Let $j < k$ be the first index such that for each $i < j$, $n'_i = 0$ and $n'_j \neq 0$. By the previous claim, for each $i \leq j$, $x'_i = 0$. We move the value of n'_j to n'_{j+1}, then after the change $n'_j = 0$ and n'_{j+1} is equal to the sum of the old values of n'_j and n'_{j+1}. This change influences only the following two constraints of (5):
Before the change, since $x'_j = 0$

$$j! \cdot x'_{j+1} \leq (n - \sum_{i=j+1}^{k} n'_i)^j - (n - \sum_{i=j}^{k} n'_i)^j \tag{A.3}$$

$$(j+1)! \cdot x'_{j+2} \leq (n - \sum_{i=j+2}^{k} n'_i)^{j+1} - (n - \sum_{i=j+1}^{k} n'_i)^{j+1} \tag{A.4}$$
$$+j! \cdot x'_{j+1} \cdot (n - \sum_{i=j+1}^{k} n_i)$$

After the change:

$$j! \cdot x'_{j+1} \leq 0$$

We decrease x'_{j+1} to 0 in order to satisfy the first constraint. Therefore, the second constraint after the changes will be as follows:

$$(j+1)! \cdot x'_{j+2} \leq (n - \sum_{i=j+2}^{k} n'_i)^{j+1} - (n - \sum_{i=j}^{k} n'_i)^{j+1} \tag{A.5}$$

Denote $z = n - \sum_{i=j+1}^{k} n'_i$, and by Δ the change in the second constraint, i.e. the difference between (A.5) and (A.4), then

$$\Delta = -(z - n'_j)^{j+1} + z^{j+1} - j! \cdot x'_{j+1} \cdot z$$

By (A.3)

$$\begin{aligned}\Delta &\geq -(z - n'_j)^{j+1} + z^{j+1} - z \cdot (z^j - (z - n'_j)^j) \\ &= -(z - n'_j)^{j+1} + z \cdot (z - n'_j)^j \\ &= n'_j \cdot (z - n'_j)^j \geq 0\end{aligned}$$

After this change we get an optimal solution such that for $i \leq j$, $n'_i = 0$, while the validity of Constraint (5) is not harmed. By induction we can move the value of n'_{j+1} to n'_{j+2} and so on, until we get an optimal solution in which for each $i < k$, $n'_i = 0$ and $n'_k = N'$. $\square$

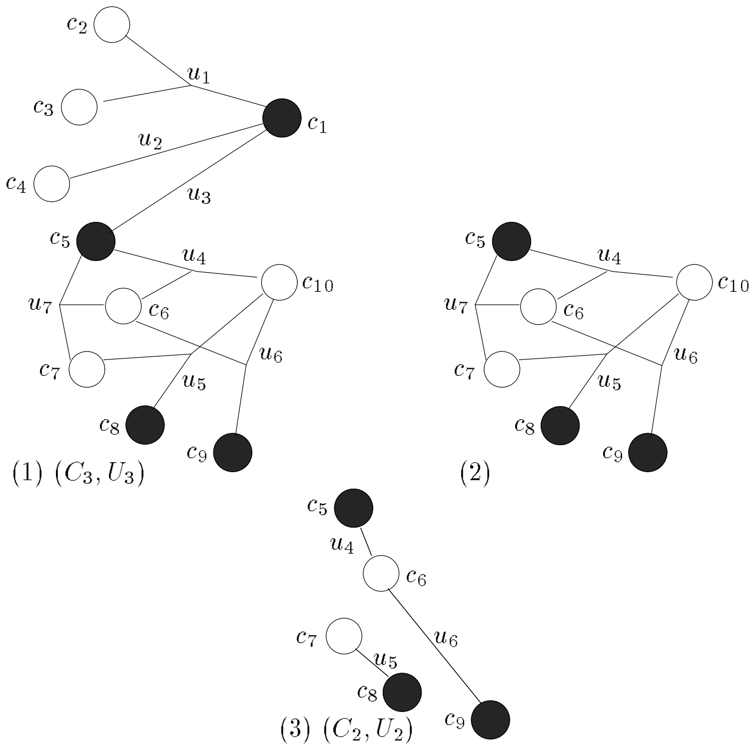

Fig. A.1. "Greedy-Sequences"- Example.

Let (C, U) be an instance of the set-cover problem, where $C = \{c_1, \cdots, c_{10}\}$, $U = \{u_1, \cdots, u_7\}$, and $k = 3$. Let $C' = \{c_1, c_5, c_8, c_9\}$ be a possible cover for the instance. The first two sets from C that the greedy algorithm may choose are c_1 and c_{10}. We can see that $c_1 \in C'$ and $c_{10} \notin C'$, and therefore, $n_3 = 1$ in the greedy-sequence of (C', C, U). Following the greedy algorithm on (C_2, U_2) we can see that the first set to be chosen is c_6. The set c_6 is not in C', thus $n_2 = 0$ and $n_1 = 2$ in the greedy-sequence of (C', C, U).

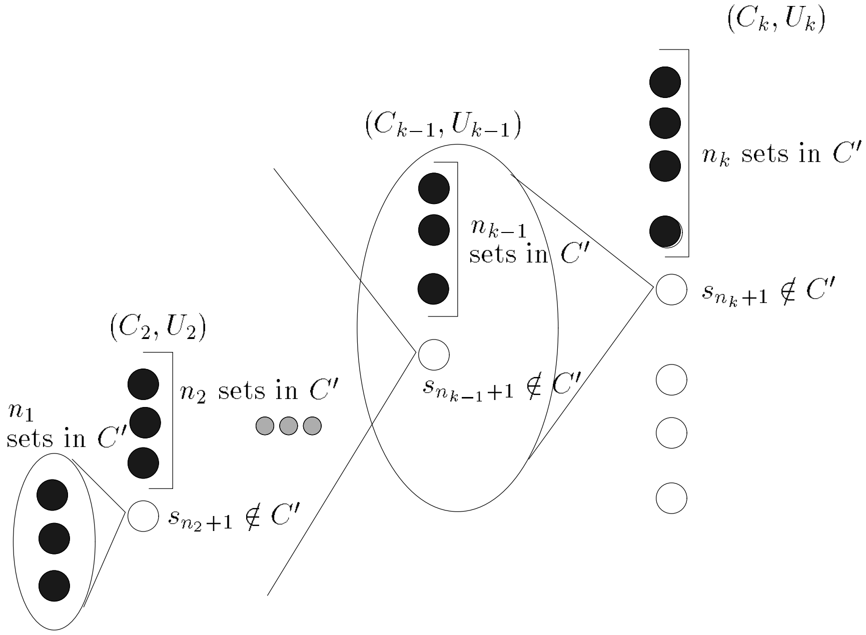

Fig. A.2. "Greedy-Sequences"- A schematic figure