# A 6/5-Approximation Algorithm for the Maximum 3-Cover Problem*

Ioannis Caragiannis[1] and Gianpiero Monaco[2]

[1] Research Academic Computer Technology Institute and Department of Computer Engineering and Informatics University of Patras, 26500 Rio, Greece
[2] Department of Computer Science, University of L'Aquila Via Vetoio, Coppito 67100, L'Aquila, Italy

**Abstract.** In the maximum cover problem, we are given a collection of sets over a ground set of elements and a positive integer $w$, and we are asked to compute a collection of at most $w$ sets whose union contains the maximum number of elements from the ground set. This is a fundamental combinatorial optimization problem with applications to resource allocation. We study the simplest APX-hard variant of the problem where all sets are of size at most 3 and we present a 6/5-approximation algorithm, improving the previously best known approximation guarantee. Our algorithm is based on the idea of first computing a large packing of disjoint sets of size 3 and then augmenting it by performing simple local improvements.

## 1 Introduction

In the maximum cover problem, we are given a collection of sets over a ground set of elements $V$ and a positive integer $w$, and we are asked to compute a collection of at most $w$ sets of maximum benefit, i.e., so that their union contains the maximum number of elements from the ground set. This is a fundamental combinatorial optimization problem with applications to the resource allocation scenario with $w$ available resources, users wishing to access one of the resources, and compatibility constraints defined as sets of users that can simultaneously access the same resource. The problem of computing an assignment of the maximum number of users to the resources so that the compatibility constraints are satisfied is equivalent to the maximum cover problem.

The natural greedy algorithm which starts with an empty solution and iteratively includes in the solution a set of maximum size that consists of elements that have not been covered before until $w$ sets are selected has approximation ratio $\frac{e}{e-1}$. In general, this result is tight due to an inapproximability result due

* This work was partially supported by the EU COST Action 293 "Graphs and Algorithms in Communication Networks" (GRAAL), by the EU IST FET Integrated Project 015964 AEOLUS, and by a "Caratheodory" research grant from the University of Patras.

to Feige [4]. An interesting special case of the problem where this inapproximability result does not hold is when the size of the sets in $\mathcal{S}$ is small. In maximum $k$-cover, $k$ denotes the maximum size of each set in $\mathcal{S}$. Without loss of generality, we assume that $\mathcal{S}$ is closed under subsets.

When applied to maximum $k$-cover, the greedy algorithm essentially belongs to the following class of iterative algorithms. An iterative algorithm for the maximum $k$-cover problem starts with an empty solution and works in phases, one phase for each $i = k, k-1, ....$ In the phase associated with $i$, the algorithm includes a maximal collection of disjoint sets each consisting of exactly $i$ elements that have not been included in previous phases. Maximality implies that any set of size $i$ (henceforth called $i$-set) intersects at least one of the selected sets. The algorithm terminates when $w$ sets have been selected in total. Intuitively, a large number of disjoint sets in the early phases is desirable in order to obtain good solutions. So, the problem that has to be solved in each phase is known as $k$-set packing. In $k$-set packing, we are given a collection of sets of size exactly $k$ over a ground set of elements $V$ and the objective is to select the maximum number of disjoint sets among the collection. This problem is known to be APX-hard for $k \geq 3$ [10] (see also [8]) while it is equivalent to maximum matching for $k = 2$ and trivial for $k = 1$. For $k \geq 3$, a well-known local search heuristic has an approximation ratio very close to $k/2$. An analysis technique for the class of iterative algorithms is presented in [2]. In that paper, an upper bound of the benefit of an iterative algorithm follows by the solution of a linear program whose constraints capture the approximation guarantee of the $i$-set packing algorithm used in phase $i$. Algorithms for $k$-set packing have also been used recently in [1] in order to improve the known approximation guarantees for the related $k$-set cover problem [3, 11].

In this work, we study the maximum 3-cover problem. This is the simplest variant of maximum cover which is still APX-hard while the maximum 2-cover problem can be solved in polynomial time. Both statements follow by similar statements for 3-set packing [10] and 2-set packing, respectively. The analysis of [2] yields an approximation ratio of 18/13-approximation for the greedy algorithm while the best result that can be obtained using the techniques of [2] is a 9/7-approximation algorithm that first computes a maximal 3-set packing of any size and then completes the solution by including the maximum possible number of elements outside the selected 3-sets into 2-sets and (if necessary) 1-sets so that exactly $w$ sets are used. In general, the upper bounds on the approximation ratio of these algorithms cannot be improved as we observe in the next section, and new algorithmic ideas are required in order to obtain better results.

Local search seems to be a promising approach since it has been proved to be efficient for the related 3-set cover and 3-set packing problems [3, 5–7, 9, 11]. For example, such an algorithm could start with any solution consisting of $w$ sets and repeatedly follow a better solution that is produced by the previous one by changing (inserting and/or removing) a constant number of 3-sets and updating the 2-sets and 1-sets. Unfortunately, such pure local search algorithms seem to be complicated to analyze. In this paper, we present a more structured algo-

rithm which combines the idea of starting with a large packing of disjoint 3-sets and then augments this solution by 2-sets and 1-sets by performing appropriate simple local improvements while this is possible. The analysis of our algorithm distinguishes between several cases depending on the structure of the covering obtained. The algorithm is proved to have an approximation ratio of 6/5 by a series of combinatorial arguments applied on the covering produced.

The rest of the paper is structured as follows. In Section 2, we discuss the limitations of iterative algorithms and outline the main ideas behind our algorithm which is presented in detail in Section 3. The statement of our main result and its proof then follow in Section 4. Due to lack of space, some proofs of intermediate lemmas have been omitted.

## 2 Limitations of Iterative Algorithms

In this section, we briefly discuss the limitations of iterative algorithms when applied to maximum 3-cover and give the intuition behind our algorithm in four different examples presented in Figure 1. In the first (upper left) example, a solution produced by the greedy algorithm is depicted. The vertical 3-sets form the optimal solution with $w = 12$. The greedy algorithm first selects 4 3-sets that intersect each optimal 3-set in exactly one element. Then, it selects 6 2-sets again intersecting each optimal 3-set in exactly one element. So, in order to complete the solution, the algorithm can select only 2 additional elements (1-sets) for a total benefit of 26. The optimal benefit is 36 and this is an example where the greedy algorithm has approximation ratio 18/13. Of course, the algorithm could

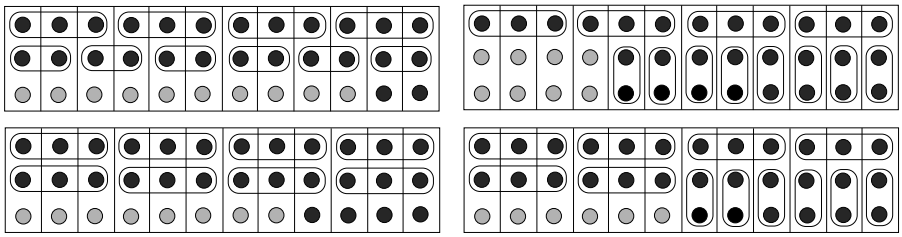

**Fig. 1.** Four examples of coverings with approximation ratios 18/13, 9/7, 9/7, and 6/5.

do better when selecting 2-sets since the maximum number of disjoint 2-sets corresponds to a maximum matching computation in the graph whose nodes correspond to the elements not covered by 3-sets and whose edges correspond to 2-sets among these elements. Since exactly one element from each optimal 3-set has been included in the 3-sets selected by the algorithm, there are 12 disjoint 2-sets among the uncovered elements, one 2-set including the two uncovered elements of each optimal 3-set. Even in this case (see the second example at the upper right part of Figure 1), at most 8 additional 2-sets can be selected for a total benefit of 28. The algorithm of this example has approximation ratio 9/7.

One attempt to improve the benefit could be to use a better 3-set packing algorithm when selecting the disjoint 3-sets. The best known such algorithm is based on local search [9] and works as follows. It uses a constant parameter $t$. Starting with an empty solution, it repeatedly searches for a neighbor solution which results from the current one by removing $p-1$ 3-sets and inserting $p$ new 3-sets for some $p \leq t$ (so that the 3-sets are disjoint). If such a neighbor solution is found, this is set as the current solution. The algorithm terminates when a locally maximum solution is reached, i.e., a solution with no neighbor solutions. The following result has been proved in [9].

**Theorem 1 (Hurkens and Schrijver [9]).** *The local search algorithm that computes a 3-set packing by performing at most $t$ local improvements at each step has approximation ratio at most $\frac{3}{2} + \frac{3}{2(2 \cdot 2^r - 3)}$ when $t = 2r - 1$, and at most $\frac{3}{2} + \frac{1}{2 \cdot 2^r - 2}$ when $t = 2r$.*

For any constant value of $t$, this algorithm may result to a locally maximum solution with 2 of the elements of each optimal 3-set covered. So, at most 4 additional elements (1-sets) can be used to complete the solution. The approximation ratio is again 9/7. See the third example in Figure 1 where the 3-sets form a locally maximum solution for the local search 3-set packing algorithm that performs at most 2 local improvements per step.

We point out that another approach could be to consider the problem as an instance of 3-set cover, attempt to minimize the number of sets required in order to cover all elements, and then, simply keep the elements in the first $w$ sets. Using the semi-local optimization algorithm of Duh and Furer [3], we would have ended up with a solution similar to the one in the second example above. Similar attempts through semi-local optimization algorithms for the partial set cover problem with sets of maximum size 3 [5] would have led to the same situation with approximation ratio 9/7.

Instead, if we could force the 3-sets to intersect half of the 3-sets in two elements and half of them in one element, the selection of 2-sets would result in at least 6 additional sets; this would give a 6/5-approximate solution (see the fourth example in Figure 1). This is what our algorithm is trying to achieve. Starting with a good 3-set packing completed with optimally selected 2-sets, it further considers improving the solution according to simple local improvement steps in order to achieve some balance between the number of 3-sets and 2-sets and avoid the situations in the second and third examples of Figure 1. Of course, the instances of the problem may be much different than those in Figure 1 and this should be taken into account in our analysis.

## 3 The Algorithm

Our algorithm receives as input an integer $w$, a set of elements $V$, a collection $\mathcal{S}$ of subsets of $V$ each containing at most 3 elements ($\mathcal{S}$ is closed under subsets), and it produces as output $w$ sets of $\mathcal{S}$ covering some of the elements of $V$. It

works in three steps. Step A computes a covering that consists of a large 3-set packing and optimally selected 2-sets (i.e., using the maximum matching computation mentioned in the second example of the previous section). Phase B performs two simple local improvement steps while this is possible. Step C simply completes the solution with 1-sets if this is necessary. A detailed description of the algorithm follows.

**A.** Compute a maximal set of disjoint 3-sets using the local search heuristic for 3-set packing with parameter $t = 2$ and complete the solution optimally with 2-sets. Denote by $\alpha_2$ and $\alpha_3$ the number of 2-sets and 3-sets computed, respectively.

**B.** Set $Stuck = FALSE$
While $\alpha_2 + \alpha_3 < w$ and $Stuck = FALSE$ do:
Try to perform a *good local change*, otherwise set $Stuck = TRUE$.

**C.** If $\alpha_2 + \alpha_3 < w$ then
Include $w - \alpha_2 - \alpha_3$ elements of $V$ that have not been included in 2-sets and 3-sets yet into 1-sets.

A *good local change* can be performed in the following two cases:

- A *replacement* means to remove one 3-set from the current solution and include another one that does not intersect with the remaining 3-sets and to complete the solution optimally with 2-sets. A replacement is performed only if it is good. A replacement is good if $\alpha_2 + \alpha_3 \leq w - 1$ at its beginning and the number of 2-sets increases after it. If the new number of 2-sets exceeds $w - \alpha_3$, the algorithm arbitrarily keeps only $w - \alpha_3$ of them and discards the rest. After performing a good replacement, the algorithm updates the numbers $\alpha_2$ and $\alpha_3$ of 2-sets and 3-sets in the current solution.
- A *removal* means to remove a 3-set from the current solution and complete optimally with 2-sets. A removal is performed only if it is good. A removal is good if $\alpha_2 + \alpha_3 \leq w - 2$ at its beginning and the number of 2-sets increases by 3. After performing a good removal, the algorithm updates the numbers $\alpha_2$ and $\alpha_3$ of 2-sets and 3-sets in the current solution.

## 4 Analysis of the Algorithm

In the analysis of the algorithm, we denote by $b_3$, $b_2$ and $b_1$ the number of 3-sets, 2-sets and 1-sets in the optimal solution, respectively. We will show that the algorithm computes a solution of benefit at least $\frac{5}{2}b_3 + \frac{7}{4}b_2 + b_1 - \frac{1}{2}$ while the optimal benefit is $3b_3 + 2b_2 + b_1$. Due to the integrality of $b_3$, $b_2$, and $b_1$, the only case in which the algorithm above may fail to produce a 6/5-approximate solution is when $b_3$ is odd and $b_2 = b_1 = 0$. In order to avoid this situation, we may run the algorithm $O(|V|^3)$ times and pick the best solution, each time guessing one of the optimal 3-sets and run our algorithm to the remaining instance. The total number of elements covered will then be at least $3 + \frac{5}{2}(b_3 - 1) + \frac{7}{4}b_2 + b_1 - \frac{1}{2} = \frac{5}{2}b_3 + \frac{7}{4}b_2 + b_1 \geq \frac{5}{6}(3b_3 + 2b_2 + b_1)$. In this way, we obtain our main result.

**Theorem 2.** *There exists a polynomial-time* $6/5$*-approximation algorithm for the maximum* 3*-cover problem.*

The proof of the lower bound on the benefit of our algorithm has been divided into four parts. In Section 4.1, we consider the two simple cases where the algorithm finishes step B having used $w$ sets (Lemma 2) or $w-1$ sets (Lemma 3). In Section 4.2, in order to handle the remaining cases where the algorithm finishes step B by having produced a covering on which no good local change is possible, we define a graph based on the structure of this covering compared to the optimal solution. Then, we distinguish between two cases which are studied in Sections 4.3 and 4.4, respectively. The results for the benefit of the algorithm in these cases are stated in Lemmas 6 and 9, respectively.

### 4.1 Some Easy Cases

We start with an important property maintained after step A of the algorithm.

**Lemma 1.** *At each step after step A of the algorithm, it always holds*

$$\alpha_2 + 3\alpha_3 \geq b_2 + 2b_3.$$

*Proof.* We first consider the case where the algorithm uses at least $b_2 + b_3$ 2-sets and 3-sets at step A, i.e., $\alpha_2 + \alpha_3 \geq b_2 + b_3$. Note that the maximum number of disjoint 3-sets in the instance of 3-set packing solved at step A is at least $b_3$. The local search 3-set packing algorithm with parameter $t = 2$ guarantees a 2-approximate solution and, hence, $2\alpha_3 \geq b_3$. The lemma then follows by summing the two inequalities.

If after step A it is $\alpha_2+\alpha_3 < b_2+b_3$, denote by $T_i$ for $i = 1, 2, 3$, the number of optimal 3-sets from which the algorithm has included $i$ elements in 3-sets. Denote by $D_i$ for $i = 1, 2$, the number of optimal 2-sets from which the algorithm has included $i$ elements in 3-sets. Since there are $T_1$ optimal 3-sets from which the algorithm does not include two elements in 3-sets and $b_2 - D_2 - D_1$ optimal 2-sets from which the algorithm has not included any elements in 3-sets, it will also use $\alpha_2 \geq T_1 + b_2 - D_2 - D_1$ 2-sets, otherwise the selection of 2-sets would not be optimal. Also, $3\alpha_3 \geq 3T_3 + 2T_2 + T_1 + 2D_2 + D_1$ since three elements in $T_3$ optimal 3-sets, two elements in $T_2$ optimal 3-sets and $D_2$ optimal 2-sets, and one element in $T_1$ optimal 3-sets and $D_1$ optimal 2-sets have been included in 3-sets by the algorithm. So, we have that

$$\begin{aligned}
\alpha_2 + 3\alpha_3 &\geq T_1 + b_2 - D_2 - D_1 + 3T_3 + 2T_2 + T_1 + 2D_2 + D_1 \\
&= 2T_1 + 2T_2 + 3T_3 + D_2 + b_2 \\
&\geq 2(T_1 + T_2 + T_3) + b_2 \\
&= b_2 + 2b_3,
\end{aligned}$$

where the second equality holds due to the maximality of the collection of 3-sets computed during step A.

So far, we have proved that the inequality holds just after step A. In order to prove that it holds at all steps of the algorithm we observe that neither removals nor replacements decrease $\alpha_2 + 3\alpha_3$. □

Next, we consider two easy cases.

**Lemma 2.** *If the algorithm finishes step B with $Stuck = FALSE$ then its benefit is at least $\frac{5}{2}b_3 + 2b_2 + \frac{3}{2}b_1$.*

*Proof.* Since the algorithm exits at step B with $Stuck = FALSE$ it must be that $\alpha_2 + \alpha_3 = w \geq b_1 + b_2 + b_3$. Using Lemma 1, we have that the benefit of the algorithm after exiting step B is

$$\begin{aligned} 2\alpha_2 + 3\alpha_3 &= \frac{3}{2}(\alpha_2 + \alpha_3) + \frac{1}{2}(\alpha_2 + 3\alpha_3) \\ &\geq \frac{3}{2}(b_1 + b_2 + b_3) + \frac{1}{2}(b_2 + 2b_3) \\ &\geq \frac{5}{2}b_3 + 2b_2 + \frac{3}{2}b_1. \end{aligned}$$

□

**Lemma 3.** *If the algorithm finishes step B with $Stuck = TRUE$ and $\alpha_2 + \alpha_3 = w - 1$, then its benefit is at least $\frac{5}{2}b_3 + 2b_2 + \frac{3}{2}b_1 - \frac{1}{2}$.*

*Proof.* The proof is the same as the proof of lemma 2 by considering the extra 1-set the algorithm will include at step C. □

### 4.2 Non-Improving Coverings

In the following we consider the case when the algorithm finishes step B with $Stuck = TRUE$ and $\alpha_2 + \alpha_3 \leq w - 2$. In this case, we say that the algorithm has computed a *non-improving covering* i.e., a collection of 3-sets and optimally selected 2-sets in which no good removal or good replacement is possible. The algorithm will then enter step C and include $w - \alpha_2 - \alpha_3$ additional single elements in the solution for a total benefit of

$$benefit = 2\alpha_2 + 3\alpha_3 + w - \alpha_2 - \alpha_3 \geq \alpha_2 + 2\alpha_3 + b_1 + b_2 + b_3. \quad (1)$$

We introduce the following notation. Given a non-improving covering, we say that an optimal 3-set is of type:

- 333 if all its elements are included in 3-sets by the algorithm.
- 332 if two of its elements are included in 3-sets and one element in a 2-set.
- 33$*$ if two of its elements are included in 3-sets and one is not covered.
- 322 if one of its elements is included in a 3-set and two elements in 2-sets.
- 32$*$ if one of its elements is included in a 3-set, one element in a 2-set and one element is not covered.
- 222 if all its elements are included in 2-sets.
- 22$*$ if two of its elements are included in 2-sets and one is not covered.

We say that an optimal 2-set is of type:

– 33 if all its elements are included in 3-sets.
– 32 if one of its elements is included in a 3-set and one element in a 2-set.
– 3∗ if one of its elements is included in a 3-set and one is not covered.
– 22 if all its elements are included in 2-sets.
– 2∗ if one of its elements is included in a 2-set, and one element is not covered.

We say that an element that does not belong in optimal 2-sets or 3-sets is of type:

– 3 if the element is included in a 3-set.
– 2 if the element is included in a 2-set.
– ∗ if the element is not covered.

We remark that we use the term optimal sets to refer to these elements as well.

Now, we will define a graph associated with a non-improving covering as follows. Given an optimal solution and the 3-sets and 2-sets in a non improving covering, we define the graph $G$ that has one node for each 3-set and 2-set of the optimal solution, one node for each element not included in 3-sets and 2-sets in the optimal solution, and an edge connecting two nodes if the algorithm has included an element from each corresponding set in the same 2-set. The graph may have self-loops (for example, when two of the elements of an optimal set of type 322 have been included in the same 2-set by the algorithm), or parallel edges. The degree of a node is the number of 2's in its type. We call nodes corresponding to optimal sets of type 33∗, 32∗, 3∗, 22∗, 2∗, or ∗ *faulty* nodes. We also call *faulty* a connected component of $G$ if it contains at least one faulty node. The next lemma states an important property of faulty connected components of $G$.

**Lemma 4.** *Each faulty connected component of $G$ contains exactly one faulty node.*

We distinguish between two cases depending on whether the graph of the non-improving covering has a faulty connected component that contains a cycle or not. In the former, we say that the non-improving covering is *cyclic*; in the latter, we say that the non-improving covering is *acyclic*.

### 4.3 The Case of Cyclic Non-Improving Covering

We first consider the case where the non-improving covering computed by the algorithm after step B is cyclic. In this case, we can show the following property of the cyclic covering. The proof is omitted.

**Lemma 5.** *If the algorithm finishes step B by computing a cyclic non-improving covering, then (a) there are no optimal sets of type* 33∗ *or* 3∗ *and (b) the number of optimal sets of type* 32∗ *is at most the number of optimal sets of type* 222*.*

We are now ready to prove the next statement using Lemmas 1 and 5 and the lower bound on the benefit of the algorithm from inequality (1).

**Lemma 6.** *If the algorithm finishes step B by computing a cyclic non-improving covering, then its benefit at the end is at least* $\frac{5}{2}b_3 + \frac{7}{4}b_2 + b_1$.

*Proof.* We use the notation $T_X$ to denote the number of optimal sets or elements out of optimal sets of type $X$. Taking into account that $T_{33*} = T_{3*} = 0$ (by Lemma 5), this definition immediately yields

$$b_2 = T_{33} + T_{32} + T_{22} + T_{2*} \tag{2}$$
$$b_3 = T_{333} + T_{332} + T_{322} + T_{32*} + T_{222} + T_{22*} \tag{3}$$

Since the three elements in optimal sets of type 222, two of the elements in optimal sets of type 322, 22∗, and 22, and one of the elements in optimal sets of type 332, 32∗, 32, 2∗, and 2 are included in 2-sets by the algorithm, we have that

$$\alpha_2 \geq \frac{1}{2}T_{332} + T_{322} + \frac{1}{2}T_{32*} + \frac{3}{2}T_{222} + T_{22*} + \frac{1}{2}T_{32} + T_{22} + \frac{1}{2}T_{2*} + \frac{1}{2}T_2 \tag{4}$$

Similarly, since the three elements in optimal sets of type 333, two of the elements in optimal sets of type 332 and 33, and one of the elements in optimal sets of type 322, 32∗, 32, and 3 are included in 3-sets by the algorithm we have that

$$\alpha_3 \geq T_{333} + \frac{2}{3}T_{332} + \frac{1}{3}T_{322} + \frac{1}{3}T_{32*} + \frac{2}{3}T_{33} + \frac{1}{3}T_{32} + \frac{1}{3}T_3 \tag{5}$$

We use inequality (1) and inequalities (2)-(5) to obtain

$$\begin{aligned} benefit &\geq 2\alpha_3 + \alpha_2 + b_3 + b_2 + b_1 \\ &\geq 3T_{333} + \frac{17}{6}T_{332} + \frac{8}{3}T_{322} + \frac{13}{6}T_{32*} + \frac{5}{2}T_{222} + 2T_{22*} \\ &\quad + \frac{7}{3}T_{33} + \frac{13}{6}T_{32} + 2T_{22} + \frac{3}{2}T_{2*} + \frac{2}{3}T_3 + \frac{1}{2}T_2 + b_1 \end{aligned} \tag{6}$$

By Lemma 5, we have $T_{32*} \leq T_{222}$ which we express as

$$0 \geq \frac{1}{12}(T_{32*} - T_{222}) \tag{7}$$

By Lemma 1 and using inequalities (2)-(5), we obtain

$$T_{333} + \frac{1}{2}T_{332} + T_{33} + \frac{1}{2}T_{32} + T_3 + \frac{1}{2}T_2 \geq \frac{1}{2}T_{32*} + \frac{1}{2}T_{222} + T_{22*} + \frac{1}{2}T_{2*}$$

and, equivalently,

$$0 \geq \frac{1}{2}(-T_{333} - \frac{1}{2}T_{332} - T_{33} - \frac{1}{2}T_{32} - T_3 - \frac{1}{2}T_2 + \frac{1}{2}T_{32*} + \frac{1}{2}T_{222} + T_{22*} + \frac{1}{2}T_{2*}) \tag{8}$$

Summing (6), (7), (8) we obtain that

$$benefit \geq \frac{5}{2}T_{333} + \frac{31}{12}T_{332} + \frac{8}{3}T_{322} + \frac{5}{2}T_{32*} + \frac{8}{3}T_{222} + \frac{5}{2}T_{22*}$$

$$
\begin{aligned}
&+\frac{11}{6}T_{33}+\frac{23}{12}T_{32}+2T_{22}+\frac{7}{4}T_{2*}+\frac{1}{6}T_3+\frac{1}{4}T_2+b_1\\
&\geq \frac{5}{2}(T_{333}+T_{332}+T_{322}+T_{32*}+T_{222}+T_{22*})\\
&+\frac{7}{4}(T_{33}+T_{32}+T_{22}+T_{2*})+b_1\\
&=\frac{5}{2}b_3+\frac{7}{4}b_2+b_1
\end{aligned}
$$

which completes the proof of the lemma. □

### 4.4 The Case of Acyclic Non-Improving Covering

The only case that remains to be considered is when the algorithm finishes step B by computing an acyclic non-improving covering. In the analysis of this case, we will use non-improving coverings of a particular type.

**Definition 1.** *An acyclic non-improving covering is called* canonical *if each faulty connected component is singleton (i.e. of type $*$, $3*$ or $33*$).*

**Lemma 7.** *For each acyclic non-improving covering there is a connected non-improving covering with the same 3-sets and the same number of 2-sets.*

*Proof.* We will construct a new covering by replacing the 2-sets corresponding to some of the edges in a faulty connected component $F$ of the corresponding graph containing a faulty node $v_0$ of type $32*$, $22*$, or $2*$ with an equal number of 2-sets so that the connected components induced by the nodes of $F$ in the graph corresponding to the new covering belong either to non-faulty connected components or to a connected component of type $33*$, $3*$, or $*$.

Since $F$ is acyclic, there is a non-faulty node of degree 1 in $F$; this node corresponds to an optimal set of type 332, 32, or 2. Let $p = \langle v_0, v_1, ..., v_t \rangle$ be the path from $v_0$ to such a node $v_t$. Let $s_0$ be the element of the optimal set corresponding to node $v_0$ which has not been included in 3-sets and 2-sets by the algorithm, and for $i = 0, 1, ..., t-1$ let $s_{2i+1}$ and $s_{2(i+1)}$ be the elements of the optimal sets $v_i$ and $v_{i+1}$, respectively, which have been included in the same 2-set by the algorithm that corresponds to the edge $(v_i, v_{i+1})$ of $p$. By replacing the $t$ 2-sets $\{s_{2i+1}, s_{2(i+1)}\}$ for $i = 0, 1, ..., t-1$ with the $t$ 2-sets $\{s_{2i}, s_{2i+1}\}$ for $i = 0, 1, ..., t-1$ (and leaving element $s_{2t}$ uncovered), we obtain a new covering whose subgraph replaces the edges of $p$ with self-loops in each of the nodes $v_0$, $v_1$, ..., $v_{t-1}$, while node $v_t$ now corresponds to an optimal set of type $33*$, $3*$, or $*$ (depending on whether it was originally of type 332, 32, or 2, respectively) and defines a singleton connected component. □

An important property of a canonical non-improving covering is given by the next lemma.

**Lemma 8.** *Any 3-set in a canonical non-improving covering produced by applying Lemma 7 to the acyclic non-improving covering computed when algorithm finishes step B intersects at least one optimal set that corresponds to a non-faulty node.*

We are now ready to prove the following lemma.

**Lemma 9.** *If the algorithm finishes step B by computing an acyclic non-improving covering, then its benefit at the end is at least* $\frac{5}{2}b_3 + \frac{7}{4}b_2 + b_1$.

*Proof.* Consider the acyclic non-improving covering computed by the algorithm and the canonical non-improving covering obtained by it with the same 3-sets and the same number of 2-sets. By Lemma 7, in order to account for the number of elements covered in the non-improving covering, it suffices to account for the number of elements covered in the canonical non-improving covering. We use the notation $T_X$ to denote the number of optimal 3-sets and 2-sets or nodes out of optimal 3-sets and 2-sets of type $X$ in the canonical non-improving covering. Taking into account that $T_{32*} = T_{22*} = T_{2*} = 0$ (by the definition of the canonical non-improving covering), this definition immediately yields

$$b_2 = T_{33} + T_{32} + T_{3*} + T_{22} \tag{9}$$

$$b_3 = T_{333} + T_{332} + T_{33*} + T_{322} + T_{222} \tag{10}$$

Since the three elements in optimal sets of type 222, two of the elements in optimal sets of type 322 and 22, and one element in optimal sets of type 332, 32, and 2 are included in 2-sets by the algorithm, we have that

$$\alpha_2 \geq \frac{1}{2}T_{332} + T_{322} + \frac{3}{2}T_{222} + \frac{1}{2}T_{32} + T_{22} + \frac{1}{2}T_2 \tag{11}$$

Since the three elements in optimal sets of type 333, two of the elements in optimal sets of type 332, 33∗, and 33, and one element in optimal sets of type 322, 32, 3∗, and 3 are included in 3-sets by the algorithm, we have that

$$\alpha_3 \geq T_{333} + \frac{2}{3}T_{332} + \frac{1}{3}T_{322} + \frac{2}{3}T_{33*} + \frac{2}{3}T_{33} + \frac{1}{3}T_{32} + \frac{1}{3}T_{3*} + \frac{1}{3}T_3 \tag{12}$$

We use inequality (1) and inequalities (9)-(12)to obtain

$$\begin{aligned} benefit &\geq 2\alpha_3 + \alpha_2 + b_3 + b_2 + b_1 \\ &\geq 3T_{333} + \frac{17}{6}T_{332} + \frac{7}{3}T_{33*} + \frac{8}{3}T_{322} + \frac{5}{2}T_{222} + \frac{7}{3}T_{33} \\ &\quad + \frac{13}{6}T_{32} + \frac{5}{3}T_{3*} + 2T_{22} + \frac{2}{3}T_3 + \frac{1}{2}T_2 + b_1 \end{aligned} \tag{13}$$

By Lemma 8, each 3-set of the canonical non-improving covering has at least one of its elements in optimal sets of type 333, 332, 322, 33, 32, 3, and at most two of its elements in optimal sets of type 33∗ and 3∗. So,

$$\alpha_3 \leq 3T_{333} + 2T_{332} + 2T_{33} + T_{322} + T_{32} + T_3 \tag{14}$$

and

$$2\alpha_3 \geq T_{3*} + 2T_{33*}. \tag{15}$$

By multiplying all the terms in inequalities (14) and (15) with $-\frac{1}{6}$ and $\frac{1}{12}$, respectively, and summing them, we obtain

$$0 \geq \frac{1}{12}(T_{3*} + 2T_{33*} - 6T_{333} - 4T_{332} - 4T_{33} - 2T_{322} - 2T_{32} - 2T_3) \qquad (16)$$

Summing (13) and (16), we obtain

$$\begin{aligned} benefit &\geq \frac{5}{2}T_{333} + \frac{5}{2}T_{332} + \frac{5}{2}T_{33*} + \frac{5}{2}T_{322} + \frac{5}{2}T_{222} + 2T_{33} + 2T_{32} + \frac{7}{4}T_{3*} \\ &\quad +2T_{22} + \frac{1}{2}T_3 + \frac{1}{2}T_2 + b_1 \\ &\geq \frac{5}{2}(T_{333} + T_{332} + T_{33*} + T_{322} + T_{222}) + \frac{7}{4}(T_{33} + T_{32} + T_{3*} + T_{22}) + b_1 \\ &= \frac{5}{2}b_3 + \frac{7}{4}b_2 + b_1 \end{aligned}$$

which completes the proof of the lemma. □

## References


1. S. Athanassopoulos, I. Caragiannis, and C. Kaklamanis. Analysis of approximation algorithms for $k$-set cover using factor-revealing linear programs. In *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory (FCT '07)*, LNCS 4639, Springer, pp. 52-63, 2007.
2. I. Caragiannis. Wavelength management in WDM rings to maximize the number of connections. In *Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS '07)*, LNCS 4393, Springer, pp. 61-72, 2007.
3. R. Duh and M. Fürer. Approximation of $k$-set cover by semi local optimization. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC '97)*, pp. 256-264, 1997.
4. U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4), pp. 634-652, 1998.
5. R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1), pp. 55-84, 2004.
6. M. M. Halldórsson. Approximating discrete collections via local improvements. In *Proceedings of the 6th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA '95)*, pp. 160-169, 1995.
7. M. M. Halldórsson. Approximating $k$-set cover and complementary graph coloring. In *Proceedings of the 5th Conference on Integer Programming and Combinatorial Optimization (IPCO '96)*, LNCS 1084, Springer, pp. 118-131, 1996.
8. E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15(1), pp. 20-39, 2006.
9. C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every $t$ of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1), pp. 68-72, 1989.
10. V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37, pp. 27-35, 1991.
11. A. Levin. Approximating the unweighted $k$-set cover problem: greedy meets local search. In *Proceedings of the 4th International Workshop on Approximation and Online Algorithms (WAOA '06)*, LNCS 4368, Springer, pp. 290-310, 2006.