The Ordered Covering Problem

Uriel Feige* Yael Hitron †

November 8, 2016

Abstract

We introduce the *Ordered Covering* (OC) problem. The input is a finite set of n elements X, a color function $c : X \to \{0, 1\}$ and a collection $\mathcal{S}$ of subsets of X. A solution consists of an ordered tuple $T = (S_1, ..., S_\ell)$ of sets from $\mathcal{S}$ which covers X, and a coloring $g : \{S_i\}_{i=1}^{\ell} \to \{0, 1\}$ such that $\forall x \in X$, the first set covering x in the tuple, namely S_j with $j = \min\{i : x \in S_i\}$, has color $g(S_j) = c(x)$. The minimization version is to find a solution using the minimum number of sets. Variants of OC include OC(α_0, α_1) in which each element of color $i \in \{0, 1\}$ appears in at most α_i sets of $\mathcal{S}$, and k-OC in which the first set of the solution S_1 is required to have color 0, and there are at most $k - 1$ alternations of colors in the solution. Our main results are as follows:

1. There is a polynomial time approximation algorithm for Min-OC(2,2) with approximation ratio 2. (This is best possible unless Vertex Cover can be approximated within a ratio better than 2.) Moreover. Min-OC(2,2) can be solved optimally in polynomial time if the underlying instance is bipartite.
2. For every $\epsilon > 0$, Min-OC is hard to approximate within a factor better than $2^{\ln^{1-\epsilon} n}$, assuming $NP \not\subseteq DTIME(n^{poly \log n})$.
3. For every $\alpha_0, \alpha_1 \geq 2$, there is a polynomial time approximation algorithm for Min-3-OC(α_0, α_1) with approximation $\alpha_1(\alpha_0 - 1)$. Unless the unique games conjecture is false, this is best possible .

*Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: uriel.feige@weizmann.ac.il.

†Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: yael.hitron@weizmann.ac.il.

1 Introduction

We introduce the *Ordered Covering* problem. As a motivating example, consider an image that one wishes to print. Each pixel in the $\sqrt{n}$ by $\sqrt{n}$ image will correspond to an element in our setting, and it has a given color (for simplicity, assume only one of two colors, such as black and white). The printing technology involves a given collection of templates, where each template corresponds to a set of pixels. For example, every rectangle in the $\sqrt{n}$ by $\sqrt{n}$ grid might be a template. Using a template, one can simultaneously color all pixels within the template in any desired color. (For example, a template can be a mask hiding all other pixels, and then one sprays the color through the openings in the template. Or the templates can correspond to a prefabricated collection of drawings on stones used in a process such as *chromolitography.*) Using a sequence of templates, each pixel remains colored by the color given to it by the last template that included it. The goal is to use the shortest sequence of templates in order to produce the desired image. The OC problem captures such problems (when viewing the sequence of templates in reverse order).

1.1 Definitions

The Ordered Covering Problem (OC) The input to OC is a finite set of n elements X, a color function $c : X \to \{0, 1\}$ and a collection $\mathcal{S}$ of m subsets of X. A solution consists of an ordered tuple $T = (S_1, ..., S_\ell)$ of sets from $\mathcal{S}$, and a coloring $g : \{S_i\}_{i=1}^{\ell} \to \{0, 1\}$ such that T covers X and $\forall x \in X$, the first set covering x in the tuple, namely S_j with $j = \min\{i : x \in S_i\}$, has color $g(S_j) = c(x)$. In the minimization version, the goal is to find a solution using the minimum number of sets.

k–Ordered Covering (k–OC) In the k–OC problem, the input is the same as to OC. A solution is a solution to OC in which the first set of the solution S_1 is required to have color 0, and the number of alternations of colors is at most $k - 1$, namely $|\{i : g(S_i) \neq g(S_{i+1})\}| \leq k - 1$. In this variation, a solution can be viewed as k monochromatic layers of sets with alternating colors, where the first layer is of color 0. In the minimization version, the goal is to find a solution using the minimum number of sets.

The OC(α_0, α_1) problem In this variation of OC, each element of color $i \in \{0, 1\}$ appears in at most α_i sets of $\mathcal{S}$. The input to OC(2,2) can be viewed as a graph, where the elements are the edges and sets are the vertices (we allow self loops).

1.2 Observations

1.2.1 *Set Cover* and *Vertex Cover* are special cases of OC

Consider Min–OC restricted to instances where all elements are of the same color. In this case, there is no importance to the order of the sets in the solution. Thus, we can view the *Set Cover* problem as Min–OC using only one color.

An input to OC can be viewed as a hypergraph with a given 0/1 coloring of its hyperedges. The sets can be viewed as vertices, elements as hyperedges, and the hyperedge corresponding to element x is composed of those vertices that represent sets that contain x. When there is only one color, OC becomes the problem of finding a set of vertices hitting all hyperedges. In particular, *minimum Vertex Cover* is a special case of Min–OC(2,2). Moreover, if we restrict the coloring function to one color, every solution has no alternation between different colors. Therefore, *Set Cover* and *Vertex Cover* are special cases of Min–k–OC for every k.

1.2.2 Feasibility

Here we consider the problem of deciding whether a given instance of Min–OC has a feasible solution, without limiting the number of sets used in the solution tuple.

Definition 1. A set $S \in \mathcal{S}$ is called *monochromatic* if it contains only elements of the same color, meaning that $S \subseteq c^{-1}(i)$ for some $i \in \{0,1\}$. Similarly, in the graph setting, we call a vertex *monochromatic* if all edges touching it are of the same color. In situations in which a partial solution is given and some elements are already covered, a set S is called monochromatic if it contains at least one uncovered element, and all the uncovered elements that it contains have the same color.

Proposition 1. *Deciding if an input to OC has a feasible solution can be done in polynomial time.*

Proof. We describe a procedure that greedily builds a solution if one exists. On input $(X, \mathcal{S}, c)$, while X is not empty and there exists a monochromatic set S_i of color c_i in $\mathcal{S}$, add S_i with color $g(S_i) = c_i$ to the end of the tuple, update $X, \mathcal{S}$ and repeat.

If the procedure manages to cover all elements, it constructs a legal solution and the instance is feasible. Likewise, if the instance is feasible, say by a solution $(S_1, ...S_k)$ and coloring g, the set $S_j \setminus \{S_i\}_{i=1}^{j-1}$ is monochromatic with color $g(S_j)$. Consequently, regardless of the sets chosen by the greedy procedure, as long as there remain uncovered elements, a monochromatic set remains available for the greedy procedure (the set S_j with smallest j that still contains an uncovered element). □

For any k, using a similar procedure we can also check feasibility for k–OC in polynomial time.

If we view instances of OC as hypergraphs as described in Observation 1.2.1, the following characterization of feasible instances holds.

Proposition 2. *A hypergraph G with coloring c (as input to OC) has a feasible solution if and only if every vertex induced sub-hypergraph of G contains a monochromatic vertex.*

Proof. The first vertex in every solution tuple must be monochromatic. If a vertex induced sub-hypergraph G' contains no monochromatic vertex, (G', c) has no solution, and neither does (G, c).

If every vertex induced sub-hypergraph of G contains a monochromatic vertex, the instance is feasible as implied by the proof of Proposition 1. □

1.2.3 *Colored Cover* – omitting the order constraints

It is instructive to consider a problem similar to OC in some respects, but different in the sense that the order of the sets in the solution does not matter.

Definition 2. In the *Colored Cover (CC)* problem, the input is the same as to OC. A solution is a collection of sets $\mathcal{T} \subseteq \mathcal{S}$ and colors $g : T \to \{0,1\}$ such that each element x is contained in a set $T \in \mathcal{T}$ colored $g(T) = c(x)$.

Similarly to OC(2,2), we define CC(2,2).

Definition 3. *The CC(2,2) problem is CC restricted to instances where each element is contained in at most two sets.*

In contrast to Proposition 1 we have:

Proposition 3. *Deciding whether an instance of CC has a feasible solution is NP-hard.*

Proof. A hypergraph is said to be *2-Colorable* if there exists a red-blue coloring of the vertices, with no monochromatic hyperedge. Deciding whether a hypergraph is 2-colorable is NP-hard [23]. Given a hypergraph $G(V, E)$ we construct an instance of CC. Define the elements to be two copies of the hyperedges, one with color 1 and one with color 0. The sets are defined by the vertices, where each vertex set contains both copies of every hyperedge the vertex is contained in. Given a solution to the CC instance, by coloring all vertices with color 0 in blue and all vertices with color 1 in red we obtain a 2-coloring for G. On the other hand given a 2-coloring of G, including all blue vertices with color 0 and all other vertices with color 1 forms a solution to the instance of CC. Thus, deciding feasibility for CC is NP-hard □

Proposition 4. *Deciding whether an instance of CC(2,2) has a feasible solution can be done in polynomial time.*

Proof. Checking feasibility of an input to $CC(2, 2)$ can be viewed as a *2–SAT* problem. The variables are the sets, and the elements and coloring function define the following constraints. Suppose that element x is contained in sets S_i, S_j. If x has color 1 introduce a constraint $S_i \vee S_j$, and if x has color 0 introduce the constraint $\bar{S}_i \vee \bar{S}_j$. Since *2–SAT* is in P, deciding feasibility of CC(2,2) can be done in polynomial time. □

1.2.4 Min–k–OC for $k = 1, 2$ is equivalent to the Set Cover problem

- For $k = 1$, solutions are allowed only to use the color 0, and hence feasible instances consist only of elements of color 0. Thus, this problem can be viewed as the *Set Cover* problem.
- For $k = 2$, a solution contains at most one alternation between different colors. Thus, a solution can be viewed as two layers of sets, the first with color 0 and the second with color 1. For a feasible instance and a legal solution, the first layer covers all elements of color 0 using monochromatic sets. The second layer forms a cover to the elements of color 1, and can use all remaining sets in $\mathcal{S}$. Hence, the 2–OC problem can be translated into two independent instances of *Set Cover*.

We can conclude that the first case in which k–OC differs from *Set Cover* is when $k \geq 3$.

1.3 Our results

Min-OC(2,2)

Theorem 5. *There exists a polynomial time approximation algorithm for Min–OC(2,2) achieving approximation ratio* 2.

Note that since *Vertex Cover* is a special case of Min–OC(2,2), Theorem 5 is best possible (up to low order terms) unless the known approximation ratio for *Vertex Cover* can be improved.

On bipartite graphs Min-OC(2,2) can be solved optimally.

Theorem 6. *There exists a polynomial time algorithm for Min-OC(2,2) on bipartite graphs.*

Min-OC

One can design relatively simple polynomial time algorithms that approximate Min-OC within a factor of $o(m)$, where m is the number of sets. It is more challenging to obtain approximation ratios of $o(n)$, where n is the number of elements. This is achieved in the following theorem.

Theorem 7. *There exist polynomial time approximation algorithms for the following problems:*

- *For Min-3-OC achieving approximation ratio of* $\sqrt{n} \cdot \ln n$.
- *For Min-OC achieving approximation ratio of* $\frac{2n \ln \ln n}{\ln n}$.

Regarding the hardness of approximating Min-OC, the following holds.

Theorem 8. *Unless* $NP \subseteq DTIME(n^{polylogn})$, *Min–OC cannot be approximated by a factor of* $2^{\log^{1-\epsilon} n}$ *for any* $0 < \epsilon < 1$.

Min-3-OC

As shown in Theorem 7, Min-3-OC can be approximated by a factor of $\sqrt{n} \cdot \ln n$. Regarding hardness of approximation, we show a connection between the hardness of approximating the *Densest k subgraph (DkS)* problem, and the hardness of approximating Min–3–OC.

Theorem 9. $\forall \alpha > 0$, $\exists \beta > 0$ *s.t if* DkS *is hard to approximate within a factor of* n^{α}, *Min–3–OC is hard to approximate better than* N^{β} *where* n *is the number of vertices in the instance of* DkS, *and* N *is the number of element in the Min–3–OC instance.*

For the Min-3-OC(α_0, α_1) variation, we achieve the following.

Theorem 10. *For any* $\alpha_0, \alpha_1 \geq 2$, *there exists a polynomial time approximation algorithm for Min–3–OC*(α_0, α_1) *achieving an approximation ratio of* $\alpha_1(\alpha_0 - 1)$.

Moreover, under the Unique Games Conjecture this approximation ratio is best possible (up to low order terms) for every constant α_1, α_0.

1.4 Related work

We are not aware of previous work on the OC problem or on problems equivalent to OC. OC involves both an ordering aspect and a covering aspect, where the cover needs to satisfy color constraints. Such a combination does not seem to appear in previously studied problems, but problems combining some of these aspects in other ways include *Graph Coloring*, *Min Sum Set Cover* [10], *Black-White Pebbling* [4], *Perfectly Orderable Graph* (which include *Chordal Graphs*) [3, 25], the game *Lights Out* [11], and a variety of other problems that can be found in the Appendix of [12]. We did not find any of these problems directly relevant to our work.

Two well studied NP-Hard problems which can be viewed as a special case of OC and OC(2,2) are *Set Cover* and *Vertex Cover*. In the *Set Cover* problem, given a finite set of n elements X and collection $\mathcal{S}$ of subsets of X the goal is to find a subset of $\mathcal{S}$ of minimum size which covers X. Approximation algorithms achieving a ratio of $\ln n$ were obtained in [22] [24]. On the other hand, unless $NP = P$, *Set Cover* cannot be approximated by a factor of $\big(1 - o(1)\big) \cdot \ln n$ [8, 6]. As *Set Cover* is a spacial case of Min–OC and Min–k–OC (see 1.2.1), the same hardness of approximation result also holds for these problems.

Given a k-uniform hypergraph, the *Ek-Vertex-Cover* problem is to find the smallest subset of vertices that intersects every hyperedge. This problem is equivalent to the *Set Cover* problem where the edges can be viewed as elements, the vertices as sets, and each element is contained in exactly k sets. Note that the classic *Vertex Cover* problem, is *Ek-Vertex Cover* with $k = 2$. A k-approximation for *Ek-Vertex-Cover* can be obtained using maximal matching. The best known algorithms for *Ek-Vertex-Cover* (as well as for *Vertex Cover*) achieve an approximation ratio of $(1 - o(1))k$ [13, 17]. Regarding hardness results, *Ek-Vertex-Cover* is NP-hard to approximate within

a factor of $(k-1-\epsilon)$ for every constants $\epsilon > 0$ and $k \geq 3$ [5]. For $k = 2$, *Vertex Cover* is NP-hard to approximate within a factor of 1.3606 [7].

The Unique Games Conjecture, made by Khot [20], refers to the NP–hardness of approximating a certain type of game, known as a unique game. Assuming the Unique Game Conjecture, for every $k \geq 2$ (including *Vertex Cover*) for every $\epsilon > 0$, *Ek-Vertex-Cover* is hard to approximate by a factor better than $k - \epsilon$ [19]. We will deduce hardness result for Min–3–OC(α_0, α_1) from the hardness of approximating *Ek-Vertex-Cover*. *Vertex Cover* is a special case of Min–OC(2,2), and therefore the hardness of approximation bounds holds for Min–OC(2,2) as well.

In order to show hardness results, we shall also consider the *Min-Rep* and *DkS* problems described below.

Min-Rep. The *Min Rep* problem (see [21]), can be viewed as a minimization version of *Label Cover*. The input is a bipartite graph $G = (A, B, E)$, $|A| = |B| = n$, and partitions $A_1 ... A_l$ of A and $B_1 ... B_l$ of B into l clusters of size n/l. The "superedges" between clusters are defined as $H = \{(A_i, B_j) : \exists (a, b) \in E \text{ s.t } a \in A_i, b \in B_j\}$. We say $(a, b) \in E$ *covers* the superedge (A_i, B_j) if $a \in A_i$ and $b \in B_j$. The goal is to choose $A' \subseteq A$ and $B' \subseteq B$ such that the pairs $\{(a, b) \in E : a \in A', b \in B'\}$ cover all the superedges of H while minimizing $|A'| + |B'|$.

Via a standard reduction (can be found in [21]), a ρ approximation ratio for *Min-Rep* is translated to $8\rho^2$ approximation ratio for *Label Cover*. This implies that if *Label Cover* is hard to approximate better than ρ, *Min–Rep* is hard to approximate within a factor better than $\frac{1}{2\sqrt{2}}\sqrt{\rho}$.

Thus, we can deduce hardness results for *Min-Rep* from hardness results for *Label Cover*. For example, unless $NP \subseteq DTIME(n^{polylog(n)})$ *Label Cover* has no approximation algorithm achieving a ratio better than $2^{\log^{1-\epsilon} n}$, for any $0 < \epsilon < 1$ [1, 14]. Under the weaker assumption that $P \neq NP$ *Label Cover* is hard to approximate better than $\ln^t n$ for any constant t [26]. The same hardness results also holds for *Min-Rep*.

***The densest k subgraph*(DkS).** Given a graph G on n vertices and a parameter k, the goal is to find a subgraph of G induced on k vertices, which contains the largest number of edges. This problem generalizes the *max-clique* problem and therefore is NP-hard. The best approximation ratio known, approximates *DkS* within a ratio of $n^{1/4+\epsilon}$ for every $\epsilon > 0$ [2]. Regarding hardness results, under the assumption that refuting 3SAT is hard on average on a natural distribution, *DkS* is hard to approximate better than some constant c [9]. Under the assumption that NP $\not\subseteq \cap_{\epsilon>0}$ BPTIME(2^{n^ϵ}) *DkS* has no polynomial time approximation scheme (PTAS) [18].

1.5 Discussion of our results

Recall that set cover is a special case of min-OC and that vertex cover is a special case of min-OC(2,2). It is instructive to compare the approximation ratios achieved for min-OC (and its special cases) with those known for set cover (and its special cases). In this respect, theorems 5 and 6 show that min-OC(2,2) behaves quite similarly to vertex cover. In contrast, Theorem 8 shows that min-OC behaves very differently than set cover. A better understanding of what aspects make min-OC have different approximation ratios than set cover is provided by considering the special case of Min-3-OC. Theorem 9 indicates that the approximation ratios expressed as a function of n are very different than those for set cover, whereas Theorem 10 shows that like set cover, if the number of sets in which an element can appear is small, one gets improved approximation ratios (and moreover, in both cases the unique games conjecture is an obstacle for further improvement in the approximation ratio).

2 Proofs

2.1 Approximation algorithm for Min-OC(2,2)

Theorem 5. *There exists a polynomial time approximation algorithm for Min–OC(2,2) achieving approximation ratio* 2*.*

In order to prove Theorem 5, first we show a 2–approximation algorithm for the unordered version Min–CC(2,2) defined in Section 1.2.3. Then show a gap preserving reduction from Min–OC(2,2) to Min-CC(2,2) achieving the same ratio.

2 approximation algorithm for Min–CC(2,2)

We describe an approximation algorithm for Min–CC(2,2) based on Linear Programming. An instance of CC(2,2) can be viewed as a graph where the elements are viewed as edges and sets as vertices. For each vertex $v \in V$ introduce two variables v^0, v^1 with the intended meaning that $v^i = 1$ if v is chosen to be included in the solution with color $g(v) = i$, and $v^i = 0$ otherwise. With this interpretation, one can see that Min–CC(2,2) can be formulated as the following Integer Program.

$$\text{minimize} \sum_{v \in V} (v^0 + v^1)$$

subject to

$$v^0 + v^1 \leq 1 \quad , v \in V \tag{1}$$

$$u^i + v^i \geq 1 \quad , \{u, v\} \in E,\ c(\{u, v\}) = i \tag{2}$$

$$v^i \in \{0, 1\} \quad , v \in V,\ i \in \{0, 1\} \tag{3}$$

To obtain a Linear Program relaxation, replace constraints (3) by non-negativity constraints.

Constructing a solution

First check whether the instance is feasible (this can be done in polynomial time by Proposition 4 in Section 1.2.3). Next, given a fractional solution $\{v^{i*}\}$ to the LP, build a solution as follows. Denote by

$$S^1 = \{v : v^{1*} > \frac{1}{2}\} \quad S^0 = \{v : v^{0*} > \frac{1}{2}\} \quad S^{\frac{1}{2}} = \{v : v^{0*} = \frac{1}{2} \text{ or } v^{1*} = \frac{1}{2}\}$$

Since $\{v^{i*}\}$ satisfies constraint (1), the sets $S^1, S^0, S^{\frac{1}{2}}$ are disjoint. Due to constraint (2), for every edge that is not covered by $S^1 \cup S^0$ both its endpoints are in $S^{\frac{1}{2}}$.

Since (G, c) is feasible so is the subgraph induced by $S^{\frac{1}{2}}$ and coloring c. As stated in Proposition 4, a solution T, g' for the subgraph induced by $S^{\frac{1}{2}}$ and coloring c can be obtained in polynomial time. Combining all vertices of S^1 with color 1, all vertices of S^0 with color 0 and T with colors g' we get a legal solution for (G, c).

Regarding the approximation ratio, given a feasible input let OPT be the size of a minimal solution. By the definition of the sets $S^{\frac{1}{2}}, S^0$ and S^1 it holds that

$$val(Algo) = |T| + |S^0| + |S^1| \leq |S^{\frac{1}{2}}| + |S^0| + |S^1| \leq 2 \cdot val(LP) \leq 2 \cdot OPT$$

Proof of Theorem 5

Proof. Recall that an input to OC(2,2) can be viewed as a graph where the elements are the edges and sets are the vertices. Given a graph and coloring (G, c) first check whether the instance is feasible. As Proposition 1 Section 1.2.2 states, this can be done in polynomial time. If (G, c) is feasible as an input to OC(2,2), by ignoring the order of the vertices in the solution it is also feasible as an input to CC(2,2). Let S and coloring g be a solution to (G, c) as an instance of CC(2,2). Since (S, g) is a solution to CC(2,2), for every edge $\{u, v\} \in E$ of color $c(\{u, v\}) = i$ either $v \in S$ with $g(v) = i$ or $u \in S$ with $g(u) = i$.

Define a directed graph $G' = (S, E')$ in the following way. For every edge $\{u, v\} \in E$ with both endpoints in S and color $c(\{u, v\}) = i$, if $g(v) = g(u) = i$ omit the edge. Else, w.lo.g $g(u) = i,\ g(v) = \bar{i}$ and we add a directed edge (v, u) to E'.

Lemma 11. *G' is a directed acyclic graph (DAG).*

Proof. Assume by contradiction that G' contains a cycle. If the cycle edges have alternating colors, the induced subgraph on the vertices of the cycle has no monochromatic vertices. By the characterization of feasible instances presented in Proposition 2 of Section 1.2.2, (G, c) is not feasible. Otherwise, the cycle contains two consecutive edges with the same color i, denote $v \to u \to w$. Due to the definition of the edges directions in G', $g(u) = g(w) = i$, and the edge $\{u, w\}$ should have been omitted. □

A DAG graph has a topological ordering of the vertices, such that every edge is directed from later to earlier vertex. In order to construct a solution to the OC(2,2) problem, order of the vertices of S by the topological order into a tuple $T = (v_k, ..., v_1)$, and keep the same color function g.

Claim 12. *T, g is a legal solution for (G, c) as an input to OC(2,2).*

Proof. Given an edge $\{u, v\} \in E$ with color i, since S and coloring g is a solution to CC(2,2), either $v \in S$ with $g(v) = i$ or $u \in S$ with $g(u) = i$. consider the following cases.

- In case only one of the vertices u, v is in S, only one of them appears in T and with color i.
- If both vertices are in S with color $g(u) = g(v) = i$, both of them are in T with color i.
- In case both of the vertices are in S with different colors, w.lo.g $g(u) = i$ and $g(v) = \bar{i}$. We add the directed edge (v, u) to G' and therefore u appears before v in T.

Hence in all cases the edge $\{u, v\}$ is covered correctly. □

Let OPT be the size of a minimal solution for (G, c). Using the approximation algorithm described in 1.2.3, we obtain a solution to (G, c) as an instance to CC(2,2) denoted by (S, g). By the procedure described above we obtain a solution as an instance to OC(2,2) denoted by (T, g). Every solution to OC(2,2) is a solution to CC(2,2) by ignoring the order of the vertices. Thus, the value of the solution obtained is $|T| = |S| \leq 2 \cdot OPT$ and we achieve approximation ratio 2. □

2.2 Min-OC(2,2) on bipartite graphs

Theorem 6. *There exists a polynomial time algorithm for Min-OC(2,2) on bipartite graphs.*

In order to prove Theorem 6 we will use a Lemma regarding totally unimodular matrices and Linear Programming.

Definition 4. *A matrix is totally unimodular if each square submatrix of it has a determinant that is either* 0,1, *or* -1.

Lemma 13. *For an LP* $Ax \leq b,\ x \geq 0$, *if* A *is totally unimodular and* A *and* b *are integral, then all vertices of the polytope are integer. It particular, the optimal solution with respect to any linear objective function is integer.*

Proof of theorem 6

Proof. As shown in Section 2.1 solving the Min-OC(2,2) problem can be reduced to solving Min-CC(2,2). Let $(G(V,E),c)$ be an input to Min-CC(2,2) where G is bipartite, $V = V_1 \cup V_2$.

Recall that the Min-CC(2,2) problem has the following Linear Program relaxation:
For each vertex $v \in V$ introduce two variables v^0, v^1 with the intended meaning that $v^j = 1$ if v is chosen to be included in the solution with color $g(v) = j$, and $v^j = 0$ otherwise.

The LP:

$$\begin{aligned} \text{minimize } & \sum_{v \in V} (v^0 + v^1) \\ \text{subject to} & \\ & v^0 + v^1 \leq 1 \qquad , v \in V \\ & -u^i - v^i \leq -1, \{u,v\} \in E,\ c(\{u,v\}) = i \\ & v^i \geq 0 \qquad\qquad , v \in V,\ i \in \{0,1\} \end{aligned}$$

The constraint matrix contain $|V| + |E|$ rows and $2 \cdot |V|$ columns: $V_1^0, V_1^1, V_2^0, V_2^1$.

Claim 14. *The constraint matrix of the LP is totally unimodular*

Proof. By induction on the size of the submatrix.
Every submatrix of size 1 contains either $0, 1$ or -1. Consider a square submatrix B of size $n > 1$. If B has a 0 column or row, then its determinant is 0. If there is a row or column with exactly one ± 1 entry, remove the row and column on which the ± 1 lies. The determinant changes only by a multiplicative factor of ± 1.

Otherwise, Every row has exactly two 1 or two -1 entries.

- Every row corresponding to a vertex $v_i \in V_i$ in B has exactly one 1 in column v_i^0 and one 1 in column v_i^1.
- Every row corresponding to an edge (v_1, v_2) of color i, has exactly one -1 on column v_1^i and one -1 on column v_2^i.

Summing the columns corresponding to V_1^0 and V_2^1 we obtain a column of the form

$$(1, 1 \ldots 1, -1, -1 \ldots, -1)^T$$

By summing the columns corresponding to V_1^1 and V_2^0 we get the same vector, and therefore the determinant of B is 0.

□

Solving a Linear Program can be done in polynomial time. By Lemma 13, the LP has an integer solution that is optimal, and therefore it is also an optimal solution to the Integer Program i.e. the Min-CC(2,2) problem. □

2.3 Approximation algorithms for Min-OC

2.3.1 $o(m)$ approximation algorithm for Min-OC

We begin by describing a simple polynomial time approximation algorithm that approximates Min-OC within a factor of $\frac{m}{\log m}$, where m is the number of sets.

As stated in Section 1.2.2, given an instance of OC, constructing a solution if one exists can be done in polynomial time. Denote the polynomial time procedure computing this task on elements Y, sets $\mathcal{R}$, and color function c by $P(Y, \mathcal{R}, c)$. Given an input $(X, \mathcal{S}, c)$ to OC with parameters $|\mathcal{S}| = m$ and $|X| = n$, partition the sets in $\mathcal{S}$ into $\log m$ "supersets" denoted by $T_1, ..., T_{\log m}$, where each superset is a collection $\frac{m}{\log m}$ sets from $\mathcal{S}$.

- Starting from $i = 1$ to $\log m$, for each subset $\mathcal{T} \subseteq \{T_1, .., T_{\log m}\}$ of size $|\mathcal{T}| = i$, invoke $P(X, \bigcup_{T_j \in \mathcal{T}} T_j, c)$.
- If P returns a solution $\rightarrow$ return it and halt.
- If P failed on all subsets of $\{T_1, ..T_{\log m}\}$, the input is not feasible $\rightarrow$ return false.

Let $(S_1, ...S_\ell)$ and color function g be an optimal solution to $(X, \mathcal{S}, c)$, and let $T_{i_1}, T_{i_2}, ...T_{i_k}$ be the supersets containing $S_1, .., S_\ell$. Thus, $P(X, \bigcup_{j=1}^{k} T_{i_j}, c)$ constructs a solution, meaning that the solution obtained by the algorithm uses at most k supersets, and is of size at most $k \cdot \frac{m}{\log m} \leq \ell \cdot \frac{m}{\log m}$.

Denote the running time of the procedure P by $T(n, m)$. The running time of the algorithm is bounded by:

$$\sum_{i=1}^{\log m} \binom{\log m}{i} \cdot T(n, m) \leq m \cdot T(n, m) = poly(n, m)$$

For any k, by replacing P with the polynomial time procedure which decides feasibility of k–OC introduced in Section 1.2.2, we obtain an approximation algorithm for Min-k–OC with the same ratio.

2.3.2 $o(n)$ approximation algorithm

Theorem 7. *There exist polynomial time approximation algorithms for the following problems:*

- *For Min-3-OC achieving approximation ratio of $\sqrt{n} \cdot \ln n$.*
- *For Min-OC achieving approximation ratio of $\frac{2n \ln \ln n}{\ln n}$.*

In order to prove Theorem 7, we give an approximation algorithm for $Min - k - OC$, for $k \geq 3$.

Lemma 15. *For every $k \geq 3$, there exists a polynomial time approximation algorithm for Min-k-OC achieving approximation ratio of $n^{1-\frac{1}{k-1}} \cdot \ln n$, where n is the number of elements.*

Proof of Theorem 7

Proof. For the Min-3-OC problem, the approximation algorithm and ratio follows directly from Lemma 15 for $k = 3$.

Regarding the Min-OC problem, we now use Lemma 15 in order to describe an approximation algorithm for Min-OC. Given an input to the OC problem with n elements, consider the following procedure.

- For $k = 3$ to $\frac{\ln n}{2 \ln \ln n}$ do:
  - Check if the input is feasible as an instance of Min-k-OC. As shown in Observation 1.2.2 this can be done in polynomial time.
  - If the input is feasible, obtain a solution to Min-k-OC using the approximation algorithm described in Lemma 15.
- If the instance is not feasible as an input to Min-k-OC for all $3 \leq k \leq \frac{\ln n}{2 \ln \ln n}$, obtain a solution using the greedy procedure which checks feasibility for OC described in Section 1.2.2. The procedure returns a solution of size at most n.
- Return the minimal size solution obtained.

Recall that in every solution to Min–k–OC the first set is of color 0. In order to allow the first set to be of color 1, we switch the colors of the elements between 0 and 1 and repeat the procedure. In the resulting solution, switch the colors of the sets composing the solution between 1 and 0. Return the solution with minimal size between the two solutions obtained by the procedure.

Let *opt* be the size of an optimal solution. If $opt \geq \frac{\ln n}{2 \ln \ln n}$, the solution obtained by the procedure is of size at most $n \leq opt \cdot \frac{2n \ln \ln n}{\ln n}$.

If $opt < \frac{\ln n}{2 \ln \ln n}$, let k be the number of layers used in an optimal solution.
Note that $k \leq opt < \frac{\ln n}{2 \ln \ln n}$ and therefore by Lemma 15 the size of the solution obtained by the procedure is at most

$$opt \cdot n^{1-\frac{1}{k-1}} \cdot \ln n \leq opt \cdot \frac{n \ln n}{n^{\frac{2 \ln \ln n}{\ln n}}} = opt \cdot \frac{n}{\ln n} < opt \cdot \frac{2n \ln \ln n}{\ln n}$$

□

Proof of Lemma 15 Let $k \geq 3$. Given an input $(X, \mathcal{S}, c)$ to Min-k-OC with n elements and m sets, let X_i denote the elements in X with color i, and $\mathcal{S}_i$ denote all monochromatic sets with color i for $i \in \{0, 1\}$.

First we build a table T of size $m \times k$ where the rows indicate the sets in $\mathcal{S}$, and the columns indicate the k layers. The intention is that the cell in row S and column i will contain a value that forms an upper bound on the "cost" of using the set S in layer i. This cost is the number of sets that suffice in layers prior to i if one is to construct a legal solution to $(X, \mathcal{S}, c)$ that includes the set S in layer i, and ∞ if there is no such legal solution.

Constructing the table T Denote the ith column of T by T_i and the cell in row S and column i by $T_i(S)$. We construct the table T in an inductive manner using a procedure approximating the *Minimum Weighted Set Cover* problem. The *Minimum Weighted Set Cover* problem has a polynomial time approximation algorithm achieving ratio of $\ln n$ where n is the number of elements [22] [24]. Denote the procedure which obtains a solution with such an approximation ratio on elements Y, sets $\mathcal{T}$ and cost function c by $ApproxSetCover(Y, \mathcal{T}, c)$.

- The first layer consists of only monochromatic sets of color 0, therefore we set the values:

$$T_1(S) = \begin{cases} 1 & S \in \mathcal{S}_0 \\ \infty & \text{else} \end{cases}$$

- For odd $1 < i \leq k$ and $S \in \mathcal{S}$, in order to use S in the ith layer we need to cover the elements $S \cap X_1$ in the first $i-1$ layers. We use the column T_{i-1} as a weight function and set the values:
$$T_i(S) = 1 + |ApproxSetCover(S \cap X_1, \mathcal{S}, T_{i-1})|$$

- In the same manner for even $1 < i < k$ and $S \in \mathcal{S}$ set values:
$$T_i(S) = 1 + |ApproxSetCover(S \cap X_0, \mathcal{S}, T_{i-1})|$$

For every $1 \leq i \leq k$ and $S \in \mathcal{S}$, when computing $T_i(S)$ we save which sets are used by the procedure *ApproxSetCover*. This can later be used in order to decide which sets to include in the first $i-1$ layers, if one is to use S in layer i.

Let $L_1^*, L_2^* ... L_k^*$ be the layers of an optimal solution of size *opt*. For each set S in the solution, we give an upper bound on the values $T_i(S)$.

Claim 16. *For every $1 \leq i \leq k$, and for each set $S \in L_j^*$ s.t $j \leq i$ and $j \bmod 2 = i \bmod 2$ it holds that $T_i(S) \leq (opt \ln n)^{i-1}$.*

Proof. We prove the claim by induction on i. For $i = 1$, every set $S \in L_1^*$ is monochromatic with color 0 and therefore $T_1(S) = 1$. For $1 < i \leq k$, without loss of generality we assume i is odd and the sets in the ith layer are of color 0 (the same holds for even i replacing 0 and 1). Let $S \in L_j^*$ s.t $j \leq i$ and $j \bmod 2 = i \bmod 2$, meaning that S is colored 0. Since $L_1^* .. L_k^*$ is a legal solution of size *opt*, there exists $r \leq opt - 1$ sets $Q_1, .. Q_r$ colored 1 in layers $L_1^*, ..., L_{j-1}^*$ which cover $S \cap X_1$. By the induction step, $T_{i-1}(Q_j) \leq (opt \ln n)^{i-2}$ for every Q_j. Thus there exists a solution for the *Weighted Set Cover* problem on elements $S \cap X_1$, sets $\mathcal{S}$ and weight T_{i-1} of size at most $(opt-1)(opt \ln n)^{i-2}$. Hence, by the approximation ratio of *ApproxSetCover* we conclude

$$T_i(S) = 1 + |ApproxSetCover(S \cap X_1, \mathcal{S}, T_{i-1})| \leq 1 + \ln n (opt - 1)(opt \ln n)^{i-2} \leq (opt \ln n)^{i-1}$$

□

Constructing a solution Next we use the table T in order to obtain a solution for $(X, \mathcal{S}, c)$. Let $a = k \bmod 2$.

- We cover all elements of color a in the $(k-1)$th layer denoted by L_{k-1}. We use the column T_{k-1} as a weight function and set L_{k-1} to be:
$$L_{k-1} \leftarrow ApproxSetCover(X_a, \mathcal{S}, T_{k-1})$$

- Next we construct layers $L_1, .. L_{k-2}$ in an inductive manner. Starting from $i = k-1$ to 2, for each set $S \in L_i$, add the sets used by the procedure *ApproxSetCover* when computing $T_i(S)$ to L_{i-1} (when constructing T we saved in each cell which sets are used).

- We construct the kth layer L_k to be a cover for the elements of color $\bar{a}$:
$$L_k \leftarrow ApproxSetCover(X_{\bar{a}}, \mathcal{S}, 1)$$

Running time The table T contains $k \cdot m$ cells. Computing each cell requires at most the time needed for the procedure *ApproxSetCover* which is polynomial in the input size.

Given T, constructing layers L_{k-1} and L_k requires the time needed for the procedure *ApproxSetCover*. Constructing Layers $L_1, .., L_{k-2}$ requires at most $(k-2) \cdot m^2$ accesses to the table. Hence, the running time is polynomial in m, n and k.

Claim 17. *The algorithm obtains a legal solution for $(X, \mathcal{S}, c)$ as an input to $k - OC$.*

Proof. First note that all elements are covered, the elements of color a are covered in layer L_{k-1}, and the elements of color $\bar{a}$ are covered in layer L_k.

Next we claim that for every element x, the first set which covers x is of color $c(x)$. Let x be an element of color b and let L_j be the first layer containing a set S which covers x. If $j = 1$, S is monochromatic of color 0, and $b = 0$. If $j > 1$, assume by contradiction S is colored $\bar{b}$. Hence $x \in S \cap X_b$ and therefore x is covered in the $(j-1)$th layer, by a set from $ApproxSetCover(S \cap X_b, \mathcal{S}, T_{j-1})$ in contradiction to the minimality of j. □

Regarding the size of the solution, first we bound the number of sets used in layers $L_1, ..., L_{k-1}$.

Claim 18. *Layers $L_1, L_2, ..L_{k-1}$ contain at most $|ApproxSetCover(X_a, \mathcal{S}, T_{k-1})|$ sets.*

Proof. By the definition of the table T, and the layers $L_1, .., L_{k-1}$ it holds that:

$$\begin{aligned}
|ApproxSetCover(X_a, \mathcal{S}, T_{k-1})| &= \sum_{S \in L_{k-1}} T_{k-1}(S) = \\
&= \sum_{S \in L_{k-1}} (1 + |ApproxSetCover(S \cap X_{\bar{a}}, \mathcal{S}, T_{k-2})|) \geq \\
&\geq |L_{k-1}| + \sum_{S \in L_{k-2}} T_{k-2}(S) = \\
&= |L_{k-1}| + \sum_{S \in L_{k-2}} (1 + |ApproxSetCover(S \cap X_a, \mathcal{S}, T_{k-3})|) \geq \\
&\geq |L_{k-1}| + |L_{k-2}| + ...|L_2| + \sum_{S \in L_1} T_1(S) = \sum_{\ell=1}^{k-1} |L_\ell|
\end{aligned}$$

We use inequalities instead of equalities because in every layer the same set may be used in *ApproxSetCover* by different sets from the previous layer. □

Next we bound the size of the solution compared to the optimal size.

Claim 19. *The algorithm obtains a solution of size at most $n^{1-\frac{1}{k-1}} \cdot \ln n$ times the size of an optimal solution.*

Proof. Let OPT be an optimal solution of size *opt*, and let $L_1, L_2..., L_k$ be the solution of size *alg* obtained by the algorithm. We can assume the instance contains elements of both colors (if the input contains elements with only one color the problem becomes the classic *Set Cover* problem). Thus, there exist $r \leq opt - 1$ sets $Q_1, ..Q_r$ in OPT colored a which form a cover to X_a. By Claim 16 for each Q_i it hold that $T_{k-1}(Q_i) \leq (opt \ln n)^{k-2}$. Thus there exists a weighted set cover to X_a, with weight function T_{k-1} of size at most $(opt - 1)(opt \ln n)^{k-2}$. By the approximation ratio of *ApproxSetCover*, and Claim 18 the number of sets in $L_1, ..L_{k-1}$ is bounded by:

$$|ApproxSetCover(X_a, \mathcal{S}, T_{k-1})| = \sum_{S \in L_{k-1}} T_{k-1}(S) \leq \ln n \cdot (opt - 1) \cdot (opt \ln n)^{k-2}$$

Regarding the kth layer, the sets colored $\bar{a}$ in the optimal solution form a cover to $X_{\bar{a}}$. Thus the number of sets used in layer L_k is:

$$|L_k| = |ApproxSetCover(X_{\bar{a}}, \mathcal{S}, 1)| \leq \ln n \cdot opt$$

If $opt \leq \frac{n^{1/(k-1)}}{\ln n}$ then

$$alg \leq (opt-1)opt^{k-2}\ln^{k-1} n + \ln n \cdot opt \leq opt(opt^{k-2}\ln^{k-1} n) \leq opt \cdot (\frac{n^{\frac{1}{k-1}}}{\ln n})^{k-2}\ln^{k-1} n = opt \cdot n^{1-\frac{1}{k-1}}\ln n$$

If $opt \geq \frac{n^{1/(k-1)}}{\ln n}$, we can assume each set in the solution covers at least one element for the first time and therefore

$$alg \leq n \leq n^{1-\frac{1}{k-1}}\ln n \cdot opt$$

□

Hence we can conclude there exist a polynomial time approximation algorithm for Min-k-OC which obtains a legal solution with approximation ratio of at most $n^{1-\frac{1}{k-1}} \cdot \ln n$.

2.4 Hardness of approximating Min-OC

Theorem 8. *Unless $NP \subseteq DTIME(n^{polylogn})$, Min–OC cannot be approximated by a factor of $2^{\log^{1-\epsilon} n}$ for any $0 < \epsilon < 1$.*

In order to show hardness results for Min–OC we use a reduction from the *Min–Rep* problem discussed in 1.4. We will need the following Lemma.

Lemma 20. *If* Min-Rep *is hard to approximate within a factor $f(n)$, it is hard to approximate Min–OC within a factor better than $O(f(N^{\frac{1}{2}}))$, where n is the number vertices in the* Min-Rep *instance and N is the number of elements in the input to Min–OC.*

Moreover, it is hard to distinguish between instances of OC with a solution of size q with 3 alternations between different colors (namely, 4 layers), and instances in which every solution (with unlimited number of alternations) uses at least $q \cdot f(N^{\frac{1}{2}})$ sets.

Proof of Theorem 8

Proof. Unless $NP \subseteq DTIME(n^{polylog(n)})$ *Label Cover* has no approximation algorithm achieving a ratio better than $2^{\log^{1-\epsilon} n}$, for any $0 < \epsilon < 1$ [1, 14]. As discussed in 1.4, via a standard reduction (can be found in [21]) the same hardness results apply also for *Min–Rep*. By Lemma 20 under the same assumption, for any $\epsilon' > 0$ Min–OC is hard to approximate by a factor better than $O(2^{\log^{1-\epsilon'} N^{\frac{1}{2}}})$. Given $\epsilon > 0$, for a small enough ϵ' we achieve the required $2^{\log^{1-\epsilon} N}$ ratio also for Min–OC. □

Proof of Lemma 20 Given an input $G(A, B, E)$, $\{A_i\}_{i=1}^{l}$, $\{B_j\}_{j=1}^{l}$, H, with parameters $|A| = |B| = n$, $|E| = m$, $|H| = h$ to *Min-Rep* construct an instance of OC with parameters $|X| = |\mathcal{S}| = 2 + 2 \cdot \frac{n^2}{h} + m$.

- **The elements:** Define all superedges in H, and all edges in E to be elements with color 1. For every vertex $v \in A \cup B$, add k elements $v_1 ... v_k$ of color 0 (k is a parameter to be determined later) denote the set of these elements by V. In addition we introduce two new elements, x_0 of color 0, and x_1 of color 1.

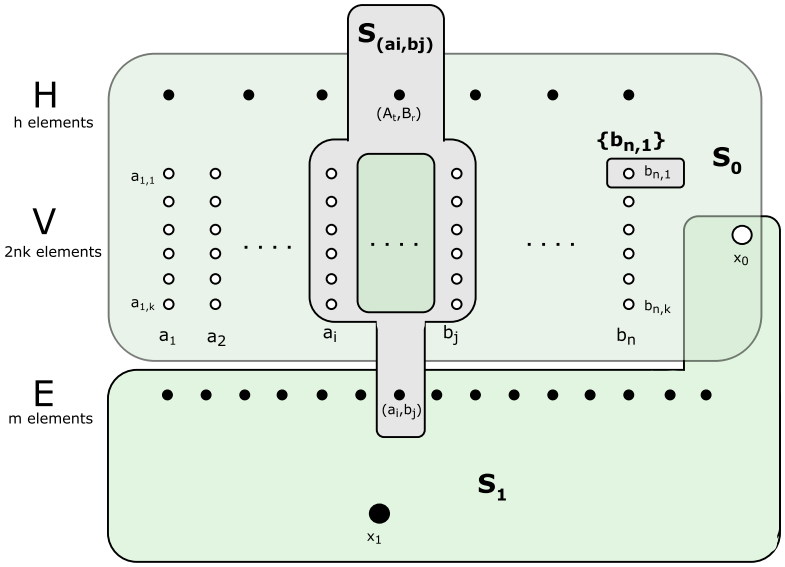

Figure 1: The elements colors in the figure are black for elements colored 1, and white for color 0. For each copy of a vertex, add a singleton illustrated as $\{b_{n,1}\}$. For each edge add a set containing the edge, corresponding vertices and superedge, illustrated as $S_{(a_i,b_j)}$. The set S_1 covers x_0, x_1 and all elements of E. S_0 contain x_1 and all elements of H and V.

- **The sets:** Add a singleton $\{v_i\}$ for every $v_i \in V$. For each $(a,b) \in E$ introduce a set $S_{(a,b)} = \{a_1..a_k, b_1...b_k, (a,b), (A_i,B_j) : a \in A_i, b \in B_j\}$. Define a set $S_0 = H \cup V \cup \{x_0\}$ and a set $S_1 = E \cup \{x_1\} \cup \{x_0\}$.

See Figure 1 for an illustration of the construction.

Observe that the instance we created is feasible. Taking all the singletons with color 0, then all sets of the form S_e with color 1 and at the end S_0 with color 0 and S_1 with color 1 defines a legal solution.

Claim 21. *If the* Min–Rep *instance G has a solution of size q, the OC instance $(X,\mathcal{S},c)$ has a solution of size $2+h+k\cdot q$.*

Proof. Let A', B' be a solution of size q for G. Build a solution T, g as follows. First add all sets $\{v_i\}$, s.t $v \in A' \cup B'$, $i = 1...k$ at the beginning of T with color $g(\{v_i\}) = 0$ ($k \cdot q$ sets). Next we add all sets $S_{(a,b)}$ which correspond to edges between A' and B' with color $g(S_{(a,b)}) = 1$. If there is more than one edge (a,b) covering the same superedge, pick only one set $S_{(a,b)}$ arbitrarily (h sets). At the end of the tuple add S_0 and S_1. Since A', B' covers all superedges, all elements of H are covered by the sets $S_{(a,b)}$. One can check that all other elements are also covered with the right color. □

Claim 22. *If every solution for the* Min–Rep *instance G contains at least $\rho \cdot q$ vertices, every solution to the OC instance $(X,\mathcal{S},c)$ is of size at least $2+h+k\cdot\rho\cdot q$.*

Proof. Let T, g be a solution for $(X,\mathcal{S},c)$. Observe that x_1 is included only in S_1, and x_0 is included only in the sets S_0 and S_1. Therefore T must include S_0 with color 0 before S_1 with color 1. In order to cover the elements of H, we need to use sets of the form S_e. The vertices touching these edges form a solution to the *Min Rep* instance and therefore the sets S_e covering H contain at

least $\rho \cdot q \cdot k$ elements from V denote them by V'. Each element x_e appears only in the sets S_1 and S_e, hence a set S_e can appear before S_1 only with color 1, meaning that the elements V' must be covered at the beginning of the tuple by singletons.

In total, every solution contain at least $2 + h + \rho \cdot q \cdot k$ sets. □

Analysis of parameters Assuming it is hard to distinguish between instances of *Min Rep* with a solution of size q and instances where every solution is of size at least $\rho(n) \cdot q$, it is hard to distinguish between instances of Min–OC with solution of size $2 + h + k \cdot q$ and instances in which every solution is of size at least $2 + h + k \cdot \rho(n) \cdot q$. Note that $q \leq 2n$ and $h \leq q^2$. Choosing $k = \frac{h}{q}$ we get an inapproximability result of $\Omega(\rho(n))$, for instances of size $N = 2 + 2 \cdot \frac{h}{q} \cdot n + m = O(n^2)$. Hence if the *Min Rep* problem is hard to approximate within a factor of $f(n)$, Min–OC is hard to approximate better than $f(O(\sqrt{N}))$. Since f is sublinear (or in the worse case linear), Min–OC is hard to approximate better than $O(f(\sqrt{N}))$

2.5 On the hardness of Min-3-OC

Theorem 9. *$\forall \alpha > 0$, $\exists \beta > 0$ s.t if* DkS *is hard to approximate within a factor of n^α, Min–3–OC is hard to approximate better than N^β where n is the number of vertices in the instance of* DkS*, and N is the number of element in the Min–3–OC instance.*

In order to prove Theorem 9 we show a reduction from the gap version of *DkS* to Min–3–OC. Given a graph G and value q, the *ρ–DkS–gap problem* is to distinguish between two cases:
Yes: There exist q edges in $G(E)$ that touch only k vertices.
No: Every q/ρ edges touch at least k vertices.

Lemma 23. *There exists a polynomial time random reduction that, given parameters n, $1 \leq k \leq n$, $k \leq q \leq k^2$, takes an instance of ρ–*DkS*–gap problem $G(V, E)$ with n vertices to an instance of 3–OC, with parameters $|X| = \frac{q}{k} \cdot n + \frac{q}{\ln(n)}$, $|\mathcal{S}| = \frac{q}{k} \cdot n + |E|$ in which*
If G is a YES instance $\rightarrow$ there exist a solution to the 3–OC instance using at most $1 + 2q$ sets.
If G is a NO instance $\rightarrow$ every solution to the 3–OC instance consists of at least $q \cdot (\frac{1}{2 \ln n} + \sqrt{\frac{\rho}{\ln n}})$ sets.

We now prove Theorem 9 using Lemma 23.

Proof of Theorem 9

Proof. Via the reduction stated in Lemma 23, starting from a graph with n vertices, we construct an instance of 3–OC with $N = O(n^2)$ elements ($q \leq k^2$). Assuming *DkS* is hard to approximate within a factor of $\rho(n)$, we conclude that Min–3–OC is hard to approximate within a factor of $\approx \frac{1}{2}\sqrt{\frac{\rho(n)}{\ln(n)}} = \frac{1}{2}\sqrt{\frac{\rho(O(N^{1/2}))}{\ln(O(N^{1/2}))}}$.

For $\alpha > 0$, if *DkS* is hard to approximate by a factor better than n^α, Min–3–OC is hard to approximate better than $\frac{d}{\ln^{1/2} N} \cdot N^{\frac{\alpha}{4}} \geq N^\beta$, where d is some constant and $\beta < \alpha/4$ can be chosen arbitrarily close to $\alpha/4$ (when N is sufficiently large).

For example, for parameters $|V| = n, k = n^{\frac{1}{2}}, \rho = n^{\frac{1}{4}}, q = n^{\frac{3}{4}}$ which seem to be a barrier for obtaining an approximation ratio of $n^{\frac{1}{4}-\epsilon}$ for *DkS* [2], by Lemma 23 we achieve a gap of $\frac{N^{\frac{1}{10}}}{c \ln(N)}$ for some constant c, for Min–3–OC. □

Proof of Lemma 23 In order to prove Lemma 23, first we reduce *ρ–DkS–gap problem* to an intermediate problem called *dense k vs sparse k' subgraph* , then we show a random reduction from this problem to 3–OC.

The *dense k vs sparse k' subgraph* is a gap problem, with parameters k, ρ' $k' > k$. Given a graph G, and value q, distinguish between two cases:
Yes: There exists q edges in $G(E)$ touching only k vertices.
No: Every q/ρ' edges touch at least k' vertices.
We can assume $q >> \rho'$, otherwise the problem is in P.

Claim 24. *The* ρ–DkS–gap problem *can be reduced to the* dense k vs sparse k' subgraph *problem with parameters* ρ', k, k' *s.t* $\rho'/\rho \leq (k/k')^2, k' > k$.

Proof. Given a graph G and value q.

- If G is a YES instance of *ρ–DkS–gap problem*, G is also a YES instance of the *dense k vs sparse k' subgraph* .
- If G is not a NO instance of *dense k vs sparse k' subgraph* , there exists k' vertices, touching at least q/ρ' edges. We can deduce using averaging arguments there exists k vertices in G touching at least $\frac{k(k-1)}{k'(k'-1)} \cdot \frac{q}{\rho'} \approx (\frac{k}{k'})^2 \cdot \frac{q}{\rho'} \geq \frac{q}{\rho}$ edges, i.e. G is not a NO instance of the *ρ–DkS–gap problem.*

□

Given a value q and a graph $G(V, E)$ with parameters $|V| = n$, $|E| = m$, we view them as an instance of *dense k vs sparse k' subgraph* , with parameters $\rho' = 2 \ln n$, and $k' \approx \sqrt{\frac{\rho}{\ln n}} \cdot k$ (k' is chosen to be the largest value for which Claim 24 holds). Next we randomly reduce *dense k vs sparse k' subgraph* with the above parameters to 3–OC. Construct an instance $(X, \mathcal{S}, c)$ in the following way.
The Elements: For each $v \in V$ we introduce $t = \frac{q}{k}$ elements $v_1 ... v_t$ of color 0. In addition introduce $\frac{q}{2 \ln n}$ elements $B = \{u_1 ... u_{\frac{q}{2 \ln n}}\}$ of color 1.
The sets: Include X as a set. For every $v \in V$, $i \in [t]$ add a singleton $\{v_i\}$. For each $(v, w) \in E$ introduce a random set $S_{(v,w)} = \{v_1 ... v_t, w_1 ... w_t, u_i\}$, where u_i is chosen uniformly at random from B.
The instance we created is feasible, as we can build a solution by taking all the singletons with color 0 in the first layer, and the set X with color 1 in the second layer.

Claim 25. *If G is a YES instance w.h.p $(X, \mathcal{S}, c)$ has a solution of size* $1 + 2q$.

Proof. Since G is a YES instance, there exists a set of k vertices $V' \subset V$ touching at least q edges $E' \subseteq E$. Build a solution to $(X, \mathcal{S}, c)$ as follows:

- The first layer consists of all sets $\{v_i\}$, s.t $v \in V'$, $i = 1...t$ with color $g(\{v_i\}) = 0$.
- The second layer is all sets $S_{(v,w)}$ s.t $(v, w) \in E'$, with color $g(S_{(v,w)}) = 1$.
- In the third layer, take the set X with color $g(X) = 0$.

All the elements of color 0 are covered and with the right color. In addition all the elements we covered from B, are covered correctly. We are left to check what is the probability that all elements

in B are covered. Using union bound, for a constant c the probability that all elements in B are covered is:

$$Pr[B \subseteq \bigcup_{(u,v)\in E'} S_{(u,v)}] \geq 1 - |B| \cdot (1 - \frac{2\ln n}{q})^q \to 1 - \frac{q}{c \cdot n^2 \cdot \ln(n)} \to 1$$

Thus, w.h.p we get a solution of size $1 + q + t \cdot k = 1 + q + \frac{q}{k} \cdot k = 1 + 2q$ □

Claim 26. *If G is a NO instance any solution to $(X, \mathcal{S}, c)$ is of size at least $\frac{q}{2\ln n} + \frac{q}{k} \cdot k' = q(\frac{1}{2\ln(n)} + \sqrt{\frac{\rho}{\ln n}})$.*

Proof. Consider two possible types of solutions.

- Solutions which include the set X with color 1. Since we use at most 3 layers, the only option to cover all elements of color 0 is using all singletons with color 0 in the first layer. This solution is of size $1 + tn = 1 + \frac{q}{k} \cdot n$.
- Solutions which don't include the set X with color 1. In order to color the elements of B without using the set X, we need at least $|B| = \frac{q}{2\ln(n)} = \frac{q}{\rho'}$ sets of the form S_e. Since G is a NO instance, every $|B|$ edges touch at least k' vertices. All the elements associated with these vertices are need to be covered in the first layer, and therefore we need to include at least $t \cdot k'$ singletons. In total, we get that each solution of this form contains at least $\frac{q}{2\ln n} + t \cdot k' = \frac{q}{2\ln n} + \frac{q}{k} \cdot k'$ sets.

We can assume $k \leq n/\rho$, otherwise by a greedy algorithm which removes a vertex of smallest degree in each step we can solve the *ρ–DkS–gap problem* in polynomial time. Thus, $\frac{q}{2\ln(n)} + \frac{q}{k} \cdot k' = \frac{q}{2\ln n} + q \cdot \sqrt{\frac{\rho}{\ln(n)}} \leq q \cdot \rho < q \cdot \frac{n}{k} < 1 + \frac{q}{k} \cdot n$. □

2.6 The Min-3-OC(α_0, α_1) problem

Theorem 10. *For any $\alpha_0, \alpha_1 \geq 2$, there exists a polynomial time approximation algorithm for Min–3–OC(α_0, α_1) achieving an approximation ratio of $\alpha_1(\alpha_0 - 1)$.*

Moreover, under the Unique Games Conjecture this approximation ratio is best possible (up to low order terms) for every constant α_1, α_0.

First we show an $\alpha_1(\alpha_0 - 1)$–approximation algorithm for the Min–3–OC(α_0, α_1) problem for any $\alpha_0, \alpha_1 > 1$. Then we show a hardness of approximation result achieving the same ratio up to low order terms for constant α_0, α_1.

2.6.1 $\alpha_1(\alpha_0 - 1)$–approximation algorithm for Min–3–OC(α_0, α_1)

We assume $\alpha_0, \alpha_1 > 1$. Note that if $\alpha_1 = 1$, or $\alpha_0 = 1$ and the input is feasible, there is only one way to cover the elements of color 0 (or 1). Thus, the problem is equivalent to using one color and therefore to *Set Cover*.

We begin by formulating 3–OC(α_0, α_1) as an Integer Programming problem. Given an input $(X, \mathcal{S}, c)$, with parameters $|X| = n, |S| = m$ we divide S to 3 distinct sets. Monochromatic sets of color 0 denoted by $W = \{S_j : S_j \subseteq c^{-1}(0)\}$, monochromatic sets of color 1 denoted by $B = \{S_j : S_j \subseteq c^{-1}(1)\}$, and "mixed sets" $M = \mathcal{S} \setminus (W \cup B)$. Note that in every solution, the first layer contains only sets from W and the second layer is contained in $B \cup M$. Since every set from

W included in the third layer can also be included in the first layer, we can assume w.lo.g the third layer contains only sets from M.

The variables: For each set $S_j \in W$, we introduce a variable y_j^1 with the intended meaning that $y_j^1 = 1$ if the set S_j is selected in the first layer and 0 otherwise. Similarly, for each set $S_j \in B \cup M$ introduce a variable y_j^2 to indicate whether it is chosen in the second layer, and for each $S_j \in M$, y_j^3 holds for the third layer. In addition, for each x with color $c(x) = 0$ introduce a variable z_x with the intended meaning that $z_x = 1$ if x is covered in the first layer and 0 if x is covered only in the third layer. If x is contained only in sets from W, fix $z_x = 1$ and if x is contained only in sets from M fix $z_x = 0$. Thus, Min–3–OC can be formulated as the following Integer Program.

$$\text{minimize } \sum y_i^k$$

subject to

$$\forall x \in X : c(x) = 1 :$$

$$\sum_{S_j \in B \cup M : x \in S_j} y_j^2 \geq 1 \tag{4}$$

$$\forall x \in X : c(x) = 0 :$$

$$\sum_{S_j \in W : x \in S_j} y_j^1 \geq z_x \tag{5}$$

$$\sum_{S_j \in M : x \in S_j} y_j^3 \geq 1 - z_x \tag{6}$$

$$\forall S_j \in M : x \in S_j \quad y_j^2 \leq z_x \tag{7}$$

$$\text{All variables have value } \{0, 1\} \tag{8}$$

For the linear program relaxation change constraints (8) to non-negativity constraints.

Rounding Given a fractional solution to the LP y^*, we construct a solution using three layers

$$L_1 = \{S_j : y_j^{1*} \geq \frac{1}{\alpha_1(\alpha_0 - 1)}\} \ , \ L_2 = \{S_j : y_j^{2*} \geq \frac{1}{\alpha_1}\} \ , \ L_3 = \{S_j : y_j^{3*} \geq \frac{1}{\alpha_1(\alpha_0 - 1)}\}$$

If a set is contained in more than one layer, include it only in the first layer it appears in. We can construct a solution (T, g) by simply concatenating all sets in L_1, L_2 and L_3 one after the other to a tuple T, and defining the color function to be $g(S_j) = 1$ for $S_j \in L_2$ and $g(S_j) = 0$ for $S_j \in L_1 \cup L_3$

Claim 27. *T with coloring g is a legal solution.*

Proof. Let $x \in X$ be an element of color $c(x) = 1$. Since y^* is a feasible solution to the LP, it satisfies constraint (4). x is contained in at most α_1 sets, therefore for some $S_j \in B \cup M$ it holds that $y_j^{2*} \geq \frac{1}{\alpha_1}$ and x is covered in the second layer with color 1. Recall that we introduce variables of the form y_j^1 only to sets in W thus, x is not covered in the first layer.

For $x \in X$ s.t $c(x) = 0$, we consider the following cases:

- If there exists a set $S_j \in L_1$ containing x we are done.

- If x is contained only in sets from M then $z_x = 0$. Due to constraint (7) x is not covered in the second layer. By constraint (6) and the definition of α_0, there exists a set S_j containing x such that $y_j^{3*} \geq \frac{1}{\alpha_0} \geq \frac{1}{\alpha_1(\alpha_0-1)}$ (we assumed $\alpha_0, \alpha_1 > 1$). Therefore, x is covered in L_3 for the first time with the right color.

- Otherwise, there are sets both in W and in M containing x. Since x is contained in at most α_0 sets and $W \cap M = \emptyset$, each of the sums in constraints (5) and (6) contain at most $\alpha_0 - 1$ summands. x is not covered in the first layer, therefore $z_x \leq \sum_{S_j \in W: x \in S_j} y_j^{1*} < \frac{(\alpha_0-1)}{\alpha_1(\alpha_0-1)} = \frac{1}{\alpha_1}$, meaning that x is not covered in the second layer. From constraint (6) we can conclude there exist $S_j \in M$, such that $x \in S_j$, and $y_j^{3*} \geq \frac{1}{\alpha_0-1} \cdot (1 - \frac{1}{\alpha_1}) \geq \frac{1}{\alpha_1(\alpha_0-1)}$. Hence x is covered in the third layer for the first time, with the right color.

□

Note that $\alpha_1(\alpha_0 - 1) \geq \alpha_1$. Thus, the size of the solution obtained by the rounding procedure is at most $\alpha_1(\alpha_0 - 1) \cdot val(LP) \leq \alpha_1(\alpha_0 - 1) \cdot OPT$ where OPT is the size of a minimal solution.

2.6.2 Hardness of approximation

Under the Unique Games Conjecture we show that for any constant α_0, α_1 and for any $\epsilon > 0$, Min–3–OC(α_0, α_1) is hard to approximate better than $\alpha_1(\alpha_0 - 1) - \epsilon$. Recall that the same ratio up to lower order terms is achieved by the algorithm described in 2.1.

For every $\epsilon' > 0$ and constant k, Khot and Regev [19] show a reduction starting from an instance of the Unique Games I, to a k-uniform hypergraph $G(V, E)$ and value q s.t:

1. If I is a YES instance, G has a vertex cover of size at most q.

2. If I is a NO instance, every vertex cover of G contain at least $(k - \epsilon')q$ vertices.

Given ϵ, α_0, α_1 starting from an instance I of the unique games, via the reduction stated in [19], for a small enough ϵ' s.t $(\alpha_1 - \epsilon')(\alpha_0 - 1 - \epsilon') \geq \alpha_1(\alpha_0 - 1) - \epsilon/2$ construct two hypergraphs, an α_1 uniform hypergraph $G(V_1, E_1)$ with parameters $|V_1| = n_1, |E_1| = m_1$ and a value q, and an $(\alpha_0 - 1)$ uniform hypergraph $H(V_2, E_2)$ with parameters $|V_2| = n_2, |E_2| = m_2$ and a value h. Next we construct an instance of the 3–OC(α_0, α_1) problem $(X, \mathcal{S}, c)$ using G and H. We duplicate H into n_1^2 copies denoted by

$$H_1^1, H_2^1, ..., H_{n_1}^1 \ , \ H_1^2, H_2^2, ..., H_{n_1}^2 \ \ ... \ \ H_1^{n_1}, H_2^{n_1}, ..., H_{n_1}^{n_1}$$

The elements: The edges of G are elements of color 1 and the edges of the n_1^2 graphs H_i^j are elements of color 0.
The sets: For each vertex v in each H_i^j, we introduce a set S_v containing all edges touching v in H_i^j. For each vertex u_t in G, introduce a set S_{u_t} containing all edges touching u_t in G and in addition all edges of the n_1 graphs $H_1^t ... H_{n_1}^t$.

See Figure 2 for an illustration of the construction.

Claim 28. *If I is a YES instance, $(X, \mathcal{S}, c)$ has a solution containing at most $n_1 + n_1 \cdot q \cdot h$ sets.*

Proof. If I is a YES instance, G has a vertex cover of size at most q and H has a vertex cover of size at most h. Thus, in order to cover all the elements of color 1 i.e. the edges of G in the second layer, we can use at most q sets of the form S_{u_t}. For each S_{u_t} used in the second layer, all edges of the n_1 graphs of the form H_i^t are covered in the first layer ($q \cdot n_1$ graphs). Since H has a vertex

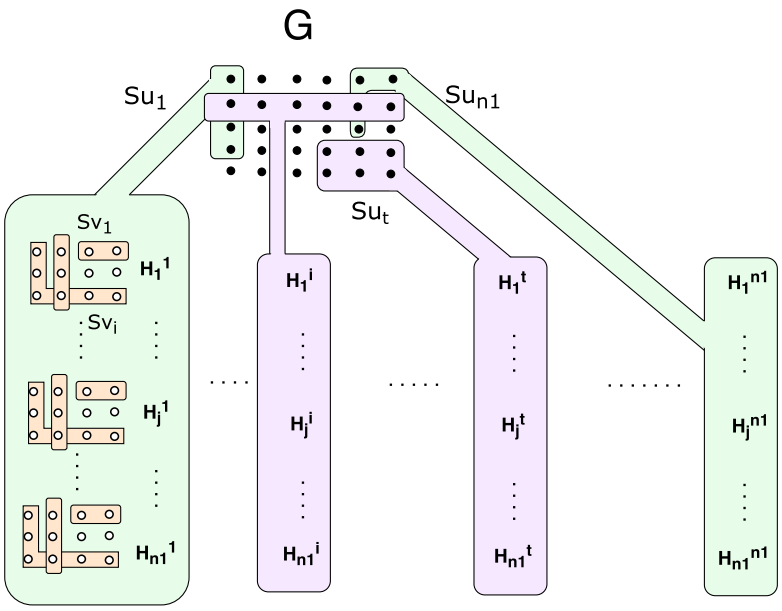

Figure 2: Each edge in G is an element of color 1 (black in the figure) and the edges of the n_1^2 copies of H are element of color 0 (white). For each vertex in each hypergraph H_j^i add a set containing all edges touching it, for example S_{v_1} in H_1^1. For each vertex u_t in G add a set containing all edges in G touching it and all edges of the n_1 graphs H_i^t, for example S_{u_1}

.

cover of size at most h, we can cover the edges of these graphs using at most $n_1 \cdot q \cdot h$ sets. In order to cover the rest of the graphs H_i^j in the third layer, use the sets S_{u_t} corresponding to the vertices of G that are not used in the second layer. The solution consists of $n_1 \cdot q \cdot h$ sets in the first layer, q sets in the second layer and $n_1 - q$ sets in the third layer, all together $n_1 + n_1 \cdot q \cdot h$ sets. □

Claim 29. *If I is a NO instance every solution to $(X, \mathcal{S}, c)$ uses at least $n_1 + n_1 \cdot q \cdot h \cdot (\alpha_1 - \epsilon')(\alpha_0 - 1 - \epsilon')$ sets.*

Proof. Since I is a NO instance, covering the edges of G in the second layer requires at least $(\alpha_1 - \epsilon') \cdot q$ sets of the form S_{u_t}. For each of these sets, we need to cover the edges of n_1 graphs of the form H_i^t in the first layer, each requires at least $h(\alpha_0 - 1 - \epsilon')$ sets. The most efficient way to cover the rest of the graphs H_i^j edges is in the third layer using the sets corresponding to the vertices of G which are not used in the second layer. Hence, every solution contains at least $n_1 + n_1 \cdot q \cdot h \cdot (\alpha_1 - \epsilon')(\alpha_0 - 1 - \epsilon')$ sets. □

The approximation ratio achieved is

$$\frac{1 + q \cdot h(\alpha_1 - \epsilon')(\alpha_0 - 1 - \epsilon')}{1 + q \cdot h} \geq \frac{1 + q \cdot h \cdot (\alpha_1(\alpha_0 - 1) - \epsilon/2)}{1 + q \cdot h} = \alpha_1(\alpha_0 - 1) - \epsilon/2 - \frac{\alpha_1(\alpha_0 - 1) - \epsilon/2 - 1}{1 + q \cdot h}$$

For a big enough q, h s.t $\frac{2\alpha_1(\alpha_0 - 1)}{\epsilon} \leq q \cdot h$ (q, h are polynomial in n_1, n_2 hence this inequality holds), we get the required $\alpha_1(\alpha_0 - 1) - \epsilon$ ratio.

Acknowledgments

We wish thank Paz Carmi and Yael Stein for introducing to us problems which inspired this work. This work was supported in part by the Israel Science Foundation (grant No. 621/12) and by the

I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation (grant No. 4/11).

References

[1] Sanjeev Arora, László Babai, Jacques Stern, Elizabeth Sweedyk: The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. J. Comput. Syst. Sci. 54(2): pp. 317-331, 1997

[2] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. STOC 2010 : pp. 201–210.

[3] Vasek Chvátal, Perfectly orderable graphs, in Berge, Claude; Topics in Perfect Graphs, Annals of Discrete Mathematics 21, Amsterdam: North-Holland, pp. 6-68 1984.

[4] Stephen A. Cook and Ravi Sethi, Storage requirements for deterministic polynomial time recognizable languages, J. Comput. System Sci., 13, pp. 25-37, 1976.

[5] Irit Dinur, Venkatesan Guruswami, Subhash Khot, Oded Regev: A New Multilayered PCP and the Hardness of Hypergraph Vertex Cover. SIAM J. Comput. 34(5): pp. 1129-1146 2005.

[6] Irit Dinur and David Steurer. Analytical approach to parallel repetition. STOC 2014: pp. 624-633

[7] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. Annals of Mathematics 162 (1): pp. 439–485, 2005.

[8] Uriel Feige, A Threhold of ln n for approximating set cover, Journal of the ACM, 45(4): pp. 634–652 , 1998.

[9] Uriel Feige. Relations between average case complexity and approximation complexity. IEEE Conference on Computational Complexity 2002: 5

[10] Uriel Feige, László Lovász, Prasad Tetali: Approximating Min Sum Set Cover. Algorithmica 40(4): pp. 219-234, 2004

[11] Rudolf Fleischer, Jiajin Yu: A Survey of the Game "Lights Out!". Space-Efficient Data Structures, Streams, and Algorithms 2013: pp. 176-198

[12] Michael R. Garey, David S. Johnson:Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman 1979, ISBN 0-7167-1044-7

[13] Eran Halperin: Improved Approximation Algorithms for the Vertex Cover Problem in Graphs and Hypergraphs. SIAM J. Comput. 31(5): pp. 1608–1623, 2002

[14] Sanjeev Arora, Carsten Lund: Hardness of Approximation. In Dorit S. Hochbaum: Approximation algorithms for NP-hard problems, PWS Publishing Company, Boston, MA, USA, 1997.

[15] Russell Impagliazzo, Ramamohan Paturi, Francis Zane: Which Problems Have Strongly Exponential Complexity? J. Comput. Syst. Sci. 63(4): pp. 512–530, 2001

[16] David S. Johnson, Mario Szegedy: What are the Least Tractable Instances of max Tndependent Set? SODA 1999: pp. 927–928

[17] George Karakostas: A better approximation ratio for the vertex cover problem. ACM Trans. Algorithms 5(4), 2009

[18] Subhash Khot, Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique, SIAM Journal on Computing 36: pp. 1025–1071, 2006.

[19] Subhash Khot, Oded Regev: Vertex cover might be hard to approximate to within 2-epsilon. J. Comput. Syst. Sci. 74(3): pp. 335–349, 2008

[20] Subhash Khot, On the power of unique 2-Prover 1-Round games, in: Proc. 34th ACM Symp. on Theory of Computing, pp. 767–775, 2002.

[21] Guy Kortsarz: On the Hardness of Approximating Spanners. Algorithmica 30(3): pp. 432–450, 2001

[22] László Lovász: On the ratio of optimal integral and fractional covers. Discrete Mathematics, 13: pp. 383–390, 1975.

[23] László Lovász: Coverings and colorings of hypergraphs. Proc. 4th Southeastern Conf. on Comb., Utilitas Math. pp 3-12, 1973

[24] David S. Johnson: Approximation Algorithms for Combinatorial Problems. Journal of Computer and System Sciences, 9: pp. 256–278, 1974.

[25] Matthias Middendorf, Frank Pfeiffer: On the complexity of recognizing perfectly orderable graphs. Discrete Mathematics 80(3): pp. 327-333, 1990

[26] Dana Moshkovitz, Ran Raz: Two Query PCP with Sub-Constant Error. FOCS 2008: pp. 314–323

Appendix A Exact algorithms

The *exponential time hypothesis (ETH)* introduced in [15] states that 3SAT has no $2^{o(n)}$ time algorithm. *Vertex Cover* has no algorithm in time $O^*(2^{o(n+m)})$, where n is the number of vertices and m is the number of edges [15, 16]. As shown in Observation 1.2.1, *Vertex Cover* is a special case of all variations of OC and therefore this bounds holds for these problems as well. Regarding exact algorithms for Min–OC the following holds.

Theorem 30. *For an instance with n elements and m sets, the Min-OC problem can be solved in $O^*(2^{\min(n,m)})$ time, where O^* notation hides a polynomial factor in n and m. Assuming the* Exponential Time Hypothesis*, Min–OC cannot be solved in time $O^*(2^{o(\min(n,m))})$.*

Moreover, for all k the same result also applies for Min–k–OC.

In order to prove Theorem 30, we give two algorithms for Min–OC and Min–k–OC, one with time complexity of $O^*(2^m)$ and the other with time complexity $O^*(2^n)$. The O^* notation hides a polynomial factor in n and m.

A.1 $O^*(2^m)$ algorithm

As stated in Section 1.2.2, given an instance of OC, constructing a solution if one exists can be done in polynomial time. Denote the polynomial time procedure computing this task on elements Y, sets $\mathcal{T}$ and cost function c by $P(X, \mathcal{T}, c)$. Given an input $(X, \mathcal{S}, c)$, with parameters $|\mathcal{S}| = m$, and $|X| = n$, consider the following algorithm for Min-OC.

- Starting from $i = 1$ to m, for each subset $\mathcal{T} \subseteq \mathcal{S}$ of size $|\mathcal{T}| = i$, invoke $P(X, \mathcal{T}, c)$.
- If P returns a solution $\rightarrow$ return it and halt.
- If P returns false $\rightarrow$ continue to the next subset.
- If P failed on all subsets of $\mathcal{S}$, the input is not feasible $\rightarrow$ return false.

For every $\mathcal{T} \subseteq \mathcal{S}$ which contains the sets of a solution tuple, the procedure $P(X, \mathcal{T}, c)$ return a solution. Hence if the input is feasible, the algorithm constructs a solution of minimal size.

Regarding the running time, denote the running time of the procedure P by $T(n, m)$. The running time of the algorithm is bounded by:

$$\sum_{i=1}^{m} \binom{m}{i} T(n, i) \leq T(n, m) \cdot 2^m = poly(n, m) \cdot 2^m = O^*(2^m)$$

Extension for Min–k–OC For any k, by replacing P with the polynomial time procedure which decides feasibility of k–OC introduced in Section 1.2.2, we obtain an algorithm for Min-k–OC with the same time complexity.

A.2 $O^*(2^n)$ algorithm

We introduce an algorithm based on dynamic programming, first for Min–OC, and then for Min–k–OC.

Let $(X, \mathcal{S}, c)$ be an input to OC with parameters $|X| = n$ and $|\mathcal{S}| = m$. For each subset $Y \subseteq X$, define $f(Y)$ to be the size of a minimal solution for $(Y, \mathcal{S}|_Y, c)$, where $\mathcal{S}|_Y = \{S \cap Y : S \in \mathcal{S}\}$. We build a table T computing f inductively in the following way:

- $T(\emptyset) = 0$
- $\forall Y \subseteq X$, $T(Y) = \min\{T(Y \setminus S_j) + 1 : S_j \in \mathcal{S} \text{ and } S_j \cap Y \text{ is monochromatic}\}$.

Claim 31. *$T(Y) = f(Y)$ for all $Y \subseteq X$*

Proof. First note that if $(X, \mathcal{S}, c)$ is feasible, so is $(Y, \mathcal{S}|_Y, c)$. We now prove the claim by induction on $|Y|$. For $|Y| = 0, Y = \emptyset$ and therefore $f(Y) = T(Y) = 0$. For $Y \neq \emptyset$, let $(S_1..S_l)$ be a solution of minimum size for $(Y, \mathcal{S}|_Y, c)$. Since it is a legal solution, $S_1 \cap Y$ is monochromatic and non–empty. By the minimality of l, the tuple $(S_2..S_l)$ is a minimal size solution to $Y \setminus S_1$ with the same coloring. Thus, $f(Y) = l = f(Y \setminus S_1) + 1 = T(Y \setminus S_1) + 1 = T(Y)$, where the last equalities derives from the induction step and the minimality of l. □

In order to construct a solution, in each cell $T(Y)$ save which set achieved the minimum value when constructing T. Starting from $T(X)$ by adding at each step the corresponding set to the end of the tuple and updating the color function, we construct a legal solution of minimum size.

Complexity The algorithm uses a table of size 2^n, where each cell contains a set index and a value of a solution. Thus, the space complexity is $O^*(2^n)$.

Regarding the time complexity, building each cell takes $O(m)$ time. Given the table T, constructing the solution tuple takes $O(n)$ time. If we assume the computational model allows random access to the table, the running time is $O^*(2^n)$.

Extension for Min–k–OC We can think of a solution to k–OC as k monochromatic layers of sets. The algorithm for Min–k–OC is similar to the algorithm for Min–OC, with the following modulation. Given an input (X, S, c), for every $Y \subseteq X, a \in \{0,1\}, j \leq k$, we define the function $f(Y, j, a)$ to be the size of a minimal solution for $(Y, \mathcal{S}|_Y, c)$ using j layers, where the first layer is of color a, or ∞ if it is not feasible. Building the table T:

- $T(\emptyset, j, a) = 0$ for all j, a
- $T(Y, 0, a) = \infty$ for every $Y \neq \emptyset$
- Else,

$$T(Y,k,a) = \min \begin{array}{l} \{T(Y \setminus S_j, k, a) + 1 : S_j \cap Y \text{ is monochromatic with color } a)\}, \\ \{T(Y \setminus S_j, k-1, \overline{a}) + 1 : S_j \cap Y \text{ is monochromatic with color } \overline{a})\}, \\ \infty \end{array}$$

The equivalence of T and f can be shown by a simple induction similarly to Claim 31. In order to build a solution, we save at each step which set achieved the minimum. Starting from $T(X, k, 0)$, by adding at each step the corresponding set to the end of the tuple and updating the color function, we obtain a solution of minimum size.

Since we can assume $k \leq n$, the time and space complexity remains $O^*(2^n)$.

Appendix B Multiple colors

For simplicity in the main body of the paper we only consider instances where the coloring function uses at most two colors. However, all results can be extended to multiple colors.

The *OC-ℓ-colors* problem is a generalization of OC in which the input color function c uses ℓ colors instead of two ($c : X \to [\ell]$). As discussed in Section 1.2.1, *Set Cover* and *Vertex Cover* are special cases of *Min-OC-ℓ-colors* and *Min-k-OC-ℓ-colors* using only one color. Regarding feasibility, Proposition 1 stated in 1.2.2 holds for the OC-ℓ-colors problem as well, and therefore checking if an instance is feasible, i.e. has a solution can be done in polynomial time.

B.1 Hardness of approximation Results.

Since OC is a special case of OC–ℓ–colors for any $\ell \geq 2$, all hardness results shown for Min–OC, and Min–3–OC in Theorems 8, 10 and 9 also hold for Min-OC–ℓ–colors and Min-3–OC–ℓ–colors.

B.2 3–OC(α_0, α_1)-ℓ-colors

Instances of 3–OC(α_0, α_1)-ℓ-colors which use more than three colors are not feasible since there is no solution using only two alternations between colors. For $\ell \leq 2$, the problem is the same as

3–OC(α_0, α_1), and the algorithmic result stated in Theorem 10 holds . Regarding $\ell = 3$, using the same arguments used in Section 1.2.4 for $k = 2$, the 3–OC(α_0, α_1)-3-colors problem can be translated into three independent instances of *Set Cover*.

B.3 Exact algorithms

The algorithms described in Appendix A for Min–OC and Min–k–OC also solve the Min–OC-ℓ-colors and Min–k–OC-ℓ-colors problems, and therefore Theorem 30 holds as well.

B.4 *Colored Cover-ℓ-colors* – omitting the order constraints

Similarly to the CC problem described in Section 1.2.3, denote by *CC-ℓ-colors* the variation of OC–ℓ-colors where we omit the constrains regarding the order of the sets in a solution. Since CC is a sub-problem of CC-ℓ-colors, Proposition 3 holds and it is NP-hard to check if a solution exists.

The CC(2,2)-ℓ-colors problem is CC-ℓ-colors restricted to instances where each element is contained in at most two sets. Similarly to CC(2,2), checking feasibility for OC(2,2)-ℓ-colors can be done in polynomial time.

Proposition 32. *Deciding whether an instance to CC(2,2)-ℓ-colors is feasible, i.e. has a solution, can be done in polynomial time.*

Proof. We reduce the problem of deciding whether an input to $CC(2,2)$-ℓ-colors is feasible to a *2–SAT* problem. For each set $S \in \mathcal{S}$, we introduce ℓ variables $S^1, ..., S^\ell$ with the intended meaning that $S_i = 1$ if S is included in the solution with color i. To ensure that each set is included in the solution with at most one color, for each $i, j \in [\ell]$, add a constraint $\overline{S^i} \vee \overline{S^j}$. For each element x with color i contained in sets $S, T \in \mathcal{S}$, add a constraint $S^i \vee T^i$. Since ℓ is bounded by the number of elements, the size of the *2–SAT* instance is polynomial in the size of the input to $CC(2,2)$-ℓ-color. The *2–SAT* problem is in P, and consequently deciding feasibility of CC(2,2)-ℓ-colors can be done in polynomial time.

□

B.4.1 Min–OC(2,2)-ℓ-colors

The same result presented in Theorem 5 holds for Min–OC(2,2)-ℓ-colors. The 2–approximation algorithm for Min–OC(2,2)-ℓ-colors is similar to the algorithm for Min–OC(2,2) presented in Section 2.1, with the following modifications to the LP.

For every $v \in V$, introduce ℓ variables $v^1, ...v^\ell$ with the intended meaning that $v^i = 1$ if v is included in the solution with color i. The Linear Program for CC(2,2)-ℓ-colors is as follows.

$$\begin{aligned}
\text{minimize } & \sum_{v \in V} v^0 + v^1 + .. + v^\ell \\
\text{subject to} & \\
& \sum_{i=1}^{\ell} v^i \leq 1 && , v \in V \\
& u^i + v^i \geq 1 && , \{u, v\} \in E,\ c(\{u, v\}) = i \\
& v^i \geq 0 && , v \in V,\ i \in [\ell]
\end{aligned}$$

B.4.2 Min–OC(2,2)-ℓ-colors on bipartite graphs

Using similar arguments to Claim 14, the constraint matrix of the LP relaxation for CC(2,2)-ℓ-colors presented above is totally unimodular. Hence, as Lemma 13 states the solution to the LP on bipartite graphs is integral and therefore the problem can be solved in polynomial time.

Appendix C Integrality gap for Min–3–OC(α_0, α_1)

First we give an example showing that the analysis of the rounding procedure is tight.

C.1 Tightness of rounding

Consider a complete α_1–partite hypergraph H on n vertices divided into α_1 sets $U_1 ... U_{\alpha_1}$ of size $|U_i| = \frac{n}{\alpha_1}$, with edges $U_1 \times U_2 ... \times U_{\alpha_1}$. In addition, we introduce n complete $(\alpha_0 - 1)$–partite hypergraphs $G_1 ... G_n$. Each G_i has n vertices divided into $\alpha_0 - 1$ sets $V_1^i ... V_{\alpha_0-1}^i$ of size $|V_1^i| = \frac{n}{\alpha_0-1}$ and edges $V_1^i \times V_2^i ... V_{\alpha_0-1}^i$. Using these graphs we define an instance of 3–OC(α_0, α_1).

The elements: The edges of H are the elements of color 1. The edges of $G_1 ... G_n$ are the elements of color 0.

The sets: For each vertex v in G_i we introduce a set S_v containing all edges touching v in G_i. For each vertex u_i in H, we add a set S_{u_i} containing all edges touching u_i in H and all edges of the graph G_i.

Note that each element of color 0 is included in α_0 sets, and each element of color 1 is included in α_1 sets.

Optimal value: In order to cover all elements of color 1 i.e the edges of H in the second layer, we need at least $\frac{n}{\alpha_1}$ sets of the form S_{u_i}. Thus, we need to cover at least $\frac{n}{\alpha_1}$ graphs G_i edges in the first layer, each requires $\frac{n}{\alpha_0-1}$ sets of the form S_v. The rest of the graphs G_i can be covered in the third layer with the sets S_{u_i} not used in the second layer. All together, the optimal value is

$$n + \frac{n}{\alpha_1} \cdot \frac{n}{\alpha_0 - 1} = \Theta(\frac{n^2}{\alpha_1(\alpha_0 - 1)})$$

The value achieved by the algorithm: In order to satisfy the constraints regarding the elements of color 1 in a minimal way, for each set associated with a vertex in H, give value $y^2 = \frac{1}{\alpha_1}$. In order to cover the elements of color 0, for each set associated with a vertex of G_i, give value $y^1 = \frac{1}{\alpha_1(\alpha_0-1)}$, and for all sets S_{u_i} for $u_i \in V(H)$, give value $y^3 = 1 - \frac{1}{\alpha_1}$. In total the LP value is $\frac{n^2}{\alpha_1(\alpha_0-1)} + n$, which is equal to the optimal value.

Using the rounding procedure, a solution of size $n^2 + n$ is obtained and therefore a ratio of approximately $\alpha_1(\alpha_0 - 1)$.

C.2 Integrality Gap

We give two bounds for the integrality gap, one for Min–3–OC(α_1, α_0) with constant α_1, α_0, and one for Min–3–OC without limiting the number of sets an element is contained in.

C.2.1 Integrality Gap for constant α_0, α_1

We show an example where the integrality gap is approximately $\alpha_1(\alpha_0 - 1)$.
For parameters n, α_1 and α_0 consider the following instance.

- Elements of color 1: The edges of an α_1-uniform complete hypergraph on $b = \sqrt{n}$ vertices denoted by G. ($\binom{b}{\alpha_1}$ elements.)
- Elements of color 0: The edges of (α_0-1)-uniform complete hypergraphs on n vertices denoted by $H_1...H_b$. ($b \cdot \binom{n}{\alpha_0-1}$ elements.)
- The sets: For each vertex v in each H_j, introduce a set S_v containing all edges touching v in H_j. For each vertex u_i in G, introduce a set T_{u_i} containing all edges touching u_i and in addition all edges of the graph H_i.

LP value: In order to satisfy the constraints regarding the elements of color 1 i.e the edges of G, for all sets $S_{u_1}...S_{u_b}$ set value $y^2_j = \frac{1}{\alpha_1}$. Regarding the constraints corresponding the elements of color 0 i.e the edges of $H_1,..H_b$, define $z_e = \frac{1}{\alpha_1}$, $y^1_j = \frac{1}{\alpha_1(\alpha_0-1)}$, $y^3_j = 1 - \frac{1}{\alpha_1}$ for all variables. This defines a feasible solution to the LP with value $b + \frac{bn}{\alpha_1(\alpha_0-1)} \approx \frac{n^{1\frac{1}{2}}}{\alpha_1(\alpha_0-1)}$.

Optimal value: In order to cover the elements of color 1 in the second layer, we need at least $b - \alpha_1 - 1$ sets of the form S_{u_i}. Hence, at least $b - \alpha_1 - 1$ Graphs H_i edges are covered in the first layer, each requires at least $n - \alpha_0 - 2$ monochromatic sets of the form S_v. The rest of the elements with color 0 can be covered in the third layer using the sets S_{u_i} that were not used in the second layer. Hence, any solution consists of at least $b + (b - 1 - \alpha_1) \cdot (n - \alpha_0 - 2) = \Theta(n^{1\frac{1}{2}})$ sets (we assume α_1, α_0 are constants).

Thus, we get a ratio of approximately $\alpha_1(\alpha_0 - 1)$.

C.2.2 Integrality gap for Min–3–OC

Distinguishing between a random graph $G(n, n^{-1/2})$ and a random graph $G(n, n^{-1/2})$ with an induced subgraph on $\sqrt{n}$ vertices replaced with $G(\sqrt{n}, n^{-(\frac{1}{4}+\epsilon)})$ is an open problem, and seems to be a barrier for obtaining a better approximation algorithm for *DkS* [2]. We will use the reduction from *DkS* to Min–3–OC described in Section 2.5 in order to build an example for which w.h.p. the integrality gap is approximately $N^{\frac{1}{10}}$, where N is the number of elements.

The parameters we use in the notation of the reduction are

$$q = \frac{n^{\frac{3}{4}}}{2} \ , \ k = \sqrt{n} \ , \ \rho = \frac{n^{\frac{1}{4}}}{\ln n}$$

Let $G(V, E)$ be a random graph picked from $G(n, n^{-1/2})$. We build an instance of 3–OC denoted as $(X, \mathcal{S}, c)$ in the following way.

The elements: For $t = q/k = \frac{n^{1/4}}{2}$, for each $v \in V$ introduce t elements $v_1...v_t$ of color 0. In addition, introduce $\frac{q}{2\ln n}$ elements $B = \{u_1...u_{\frac{q}{2\ln(n)}}\}$ of color 1 ($N = \Theta(n^{1\frac{1}{4}})$).

The sets: Include X as a set. For every $v \in V$, $i \in [t]$ add a singleton $\{v_i\}$. For each $(v, w) \in E$ introduce a random set $S_{(v,w)} = \{v_1...v_t, w_1...w_t, u_i\}$, where u_i is chosen uniformly at random from B.

Using union bound, with high probability G is a NO instance of ρ–*DkS*. Thus, every set of k vertices touches at most q/ρ edges. If G is a NO instance as stated in Claim 26 Section 2.5, the optimal value for $(X, \mathcal{S}, c)$ is at least

$$q \cdot (\frac{1}{2\ln n} + \sqrt{\frac{\rho}{\ln n}}) = \Theta(\frac{n^{\frac{7}{8}}}{\ln n})$$

LP value

With high probability the number of edges in G is at least $\frac{n^{1\frac{1}{2}}}{2}$. Using Chernoff and union bound, w.h.p. every element $u \in B$ is part of at least $\beta \cdot n^{\frac{3}{4}} \cdot \ln n$ sets of the form S_e for some constant β.

In order to satisfy the constrains regarding the elements of color 1 i.e the elements in B, for each set S_e give value $y_e^2 = \frac{1}{\beta \cdot n^{\frac{3}{4}} \cdot \ln n}$. Regarding the elements of color 0 (the t copies of V) define $z_{v_i} = y^1_{\{v_i\}} = \frac{1}{\beta \cdot n^{\frac{3}{4}} \cdot \ln n}$ for each v_i, and for the variable associated with the set X give value $y_X^3 = 1$. By Chernoff w.h.p the number of edges is at most $2n^{1\frac{1}{2}}$. All together w.h.p we get a solution for the LP of size

$$\frac{1}{\beta \cdot n^{\frac{3}{4}} \cdot \ln n} \cdot (t \cdot n + 2n^{1\frac{1}{2}}) + 1 = \Theta(\frac{n^{3/4}}{\ln n})$$

Thus, the ratio between the optimal value and the value of the LP is at least $\Theta(n^{1/8}) = \Theta(N^{1/10})$

Appendix D Covering one designated element

We show that the OC problem is equivalent to the problem of covering only one designated element correctly without "miscoloring" the other elements.

The x_0-OC problem In the x_0–OC problem, the input is a finite set of n elements X, a color function $c : X \to \{0, 1\}$, a collection $\mathcal{S}$ of m subsets of X, and a designated element $x_0 \in X$. The goal is to output a tuple of sets $T = (S_1 ... S_k)$ and coloring $g : \{S_i\}_{i=1}^k \to \{0, 1\}$ such that:

- $x_0 \in \bigcup_{j=1}^k S_j$
- T, g is a solution for $(\bigcup_{j=1}^k S_j, \mathcal{S}, c)$ as an instance of OC.

We show gap preserving reductions between x_0–OC and OC.

D.0.1 From the OC problem to x_0-OC.

Given input $(X, \mathcal{S}, c)$ to OC, construct an input to x_0–OC denoted by $(X', \mathcal{S}', c', x_0)$ as follows. We introduce two new elements x_0, x_1, and two new sets $T_0 = \{x_0, x_1\} \cup c^{-1}(1)$, $T_1 = \{x_1\} \cup c^{-1}(0)$. Define the elements to be $X' = X \cup \{x_0, x_1\}$, the sets $\mathcal{S}' = \mathcal{S} \cup \{T_0, T_1\}$, and the color function $c'(x_0) = 0$, $c'(x_1) = 1$, $c'(x) = c(x)$ $\forall x \in X$.

The only set containing x_0 is T_0, and the only sets containing x_1 are T_0 and T_1. We can assume that in every solution x_0 is contained in the last set of the tuple, and therefore that every solution to $(X', \mathcal{S}', c', x_0)$ contains T_0 with color 0 as the last set, and T_1 with color 1. Given a solution $(S_1, ..S_i, T_1, S_{i+1}, ..S_k, T_0)$ with coloring g for $(X', \mathcal{S}', c', x_0)$, by the definition of T_1, the sets $S_1, ..S_i$ cover all the elements of color 0 in X correctly. Similarly, all elements of color 1 in X are covered correctly by the sets $S_1, ...S_k$. Hence, the tuple $(S_1, , ..S_k)$ with the same coloring g is a solution for $(X, \mathcal{S}, c)$ as an instance of OC.

On the other hand, if $(S_1 ... S_k)$ and g is a solution for $(X, \mathcal{S}, c)$, the tuple $(S_1 ..S_k, T_1, T_0)$ with coloring $g'(T_0) = 0, g'(T_1) = 1, g'(S_j) = g(S_j)$ is a solution for $(X', \mathcal{S}', c')$.

We can conclude that $(X, \mathcal{S}, c)$ has a solution of size k if and only if $(X', \mathcal{S}', c', x_0)$ has a solution of size $k + 2$.

D.0.2 From the x_0-OC problem to OC.

Starting from an input $(X, \mathcal{S}, c, x_0)$ to the x_0–OC problem, construct an instance of the OC problem denoted by $(X', \mathcal{S}', c')$ as follows. Without loss of generality we can assume that $c(x_0) = 0$. We introduce two new elements y_1, y_0 and two new sets $T_0 = \{y_1, y_0\} \cup X \setminus \{x_0\}$ and $T_1 = \{y_1, x_0\} \cup c^{-1}(1)$. Define the elements to be $X' = X \cup \{y_1, y_0\}$, the sets $\mathcal{S}' = \mathcal{S} \cup \{T_0, T_1\}$ and the color function $c'(y_1) = 1$, $c'(y_0) = 0$, $c'(x) = c(x)$ $\forall x \in X$.

Observe that any solution to $(X, \mathcal{S}, c, x_0)$ can be extended to a solution for $(X', \mathcal{S}', c')$ by concatenating T_1 and then T_0 at the end of the tuple with colors $g(T_0) = 0$ and $g(T_1) = 1$.

On the other hand, since the only set containing y_0 is T_0 and the only sets containing y_1 are T_0 and T_1, every solution to $(X', \mathcal{S}', c)$ contains the set T_0 with color 0 and later in the tuple T_1 with color 1. Given a solution $(S_1, ..S_i, T_1, S_{i+1}, ..S_k, T_0, S_{k+1}, ..S_\ell)$ to $(X', \mathcal{S}', c')$, the set T_1 colors the element x_0 with color 1. Thus, $x_0 \in \bigcup_{j=1}^{i} S_j$ and the tuple $(S_1, ..., S_i)$ is a solution to $(X, \mathcal{S}, c, x_0)$. We can conclude that $(X, \mathcal{S}, c, x_0)$ has a solution of size at most k if and only if $(X', \mathcal{S}', c')$ has a solution of size at most $k + 2$.