

# The Online Disjoint Set Cover Problem and its Applications

Ashwin Pananjady, Vivek Kumar Bagaria  
Department of Electrical Engineering  
Indian Institute of Technology Madras  
Email: {ee10b047, ee10b025}@ee.iitm.ac.in

Rahul Vaze  
School of Technology and Computer Science  
Tata Institute of Fundamental Research  
Email: vaze@tcs.tifr.res.in

**Abstract**—Given a universe  $U$  of  $n$  elements and a collection of subsets  $\mathcal{S}$  of  $U$ , the maximum disjoint set cover problem (DSCP) is to partition  $\mathcal{S}$  into as many set covers as possible, where a set cover is defined as a collection of subsets whose union is  $U$ . We consider the online DSCP, in which the subsets arrive one by one (possibly in an order chosen by an adversary), and must be irrevocably assigned to some partition on arrival with the objective of minimizing the competitive ratio. The competitive ratio of an online DSCP algorithm  $A$  is defined as the maximum ratio of the number of disjoint set covers obtained by the optimal offline algorithm to the number of disjoint set covers obtained by  $A$  across all inputs. We propose an online algorithm for solving the DSCP with competitive ratio  $\ln n$ . We then show a lower bound of  $\Omega(\sqrt{\ln n})$  on the competitive ratio for any online DSCP algorithm. The online disjoint set cover problem has wide ranging applications in practice, including the online crowd-sourcing problem, the online coverage lifetime maximization problem in wireless sensor networks, and in online resource allocation problems.

## I. INTRODUCTION

Consider a universe  $U$  consisting of  $n$  elements, i.e.,  $U = \{1, 2, \dots, n\}$ . Let  $\mathcal{S} = \{S_1, S_2, \dots\}$  be a collection of subsets of  $U$ , where  $S_i \subseteq U \forall i$ . Then the disjoint set cover problem (DSCP) is to find as many partitions of  $\mathcal{S}$  as possible such that the union of the subsets in each partition is  $U$ . The DSCP is known to be NP-hard [1], and has an optimal approximation ratio of  $\ln n$  with any polynomial time algorithm, by [2], [3].

The DSCP is a fundamental combinatorial optimization problem that has widespread applications. Maximizing the coverage lifetime (MLCP) of a sensor network [3], [4] is one example. Here,  $U$  is the set of targets, and each sensor  $i$  can cover/track targets  $S_i \subseteq U$ . The objective is to find a sensor operation (on/off) schedule such that the total time for which all targets are covered is maximized. One common approach to solve the MLCP is to find the DSCP solution, and use each of the disjoint set covers in distinct time slots [1], [3], [5].

Another DSCP application of interest is in resource allocation and scheduling problems [6]. A canonical example of a resource allocation problem is where the universe  $U$  represents the set of files or sub-files of a movie or large file, and each server  $i$  contains a subset  $S_i$  of those files. Each server has limited capability, and can serve at most one user at any given time. Thus, maximizing the number of users that can access all files of the movie (and consequently the revenue) is equivalent to solving the DSCP.

The new paradigm of crowd-sourcing also involves solving the DSCP. In crowd-sourcing [7], [8], a platform (public utility) advertises the set of tasks (the universe  $U$ ) that it wants to be accomplished (e.g. pothole tracking, community service, etc.). Each user  $i$  submits a list of tasks  $S_i$  that it can perform, and the platform has to group/cluster subset of users in as many groups as possible such that each group ensures task coverage, i.e. all tasks should be performed by at least one of the users in the group. The crowd-sourcing problem with task coverage is equivalent to the DSCP.

The DSCP is also relevant for finding efficient supply chain management solutions [9], where  $n$  distinct raw materials/sub-tasks are required for producing a particular good by a machine, and each sub-contractor/auxiliary-machine  $i$  can supply/accomplish only a subset of raw materials/sub-tasks  $S_i$  [10]. Finding the optimal allocation/routing of supplier/sub-tasks to distinct machines in order to maximize the number of simultaneously producible goods is equivalent to the DSCP.

In this paper, we consider an online version of the DSCP, where subsets arrive sequentially in time, and each subset has to be assigned to a partition irrevocably without knowledge of future subset arrivals. The objective is similar to the earlier case: to maximize the number of partitions such that the union of subsets in each partition is equal to  $U$  at the end of all subset arrivals, but now relative to the optimal offline algorithm. The offline algorithm refers to the case when the sequence of arrival of subsets is revealed to the algorithm in advance. To study the effectiveness of online algorithms, the competitive ratio, which measures the ratio of the profit of the optimal offline algorithm and a particular online algorithm, is the metric of choice [11]. The smaller the competitive ratio of an online algorithm, the better its performance. Note that here we make no assumption on the complexity of the offline/online algorithm, the optimal offline or any online algorithm is allowed to have exponential complexity.

Studying the online version of DSCP is important, since finding an efficient online algorithm for the DSCP is equivalent to solving the online versions of the optimization problems defined above. For example, the online DSCP corresponds to the online MLCP in a natural way, where sensors arrive/wake-up sequentially in time, and each sensor's on-off schedule has to be decided in an online fashion so as to maximize the coverage lifetime. Similarly, the online DSCP corresponds

to online crowd-sourcing, where users submit their requests sequentially in time, and must be grouped with other users irrevocably without knowledge of task sets of future users.

There is also a natural correspondence between the online DSCP and the online resource allocation problem, where servers become active or acquire the subset of files at arbitrary times, and server-user association has to be done in an online fashion without knowledge of future server arrivals. The online version of supply chain management has a similar correspondence to the online DSCP.

Let  $F_{min}$  be the minimum number of times any universe element occurs across all the subsets. In this paper, we make the following contributions.

- We show that any online algorithm must be given  $F_{min}$  in advance to perform with a non-trivial competitive ratio.
- Through the online polychromatic coloring of a specially defined hypergraph, we propose an online algorithm for the online DSCP that has a competitive ratio of  $O(\ln n)$ .
- We also show a lower bound on the competitive ratio, of  $\Omega(\sqrt{\ln n})$ , i.e. no online algorithm can have a better competitive ratio than  $\Omega(\sqrt{\ln n})$ . Note that no complexity assumptions are made on the online algorithm, and the lower bound holds for online algorithms even with exponential complexity.

## II. MODEL AND PRELIMINARIES

The universe of *elements* is represented by  $U = \{1, 2, 3, \dots, n-1, n\}$ , unless stated otherwise. We denote the collection of subsets provided by  $\mathcal{S} = \{S_1, S_2, \dots\}$ , where  $S_j \subseteq U \forall j$ . The number of subsets is therefore  $|\mathcal{S}|$ . We define a set cover as a collection of subsets  $C \subseteq \mathcal{S}$  such that  $\cup_{S_i \in C} S_i = U$ . We assume that at least one set cover exists in  $\mathcal{S}$ , i.e.,  $\cup_{S_i \in \mathcal{S}} S_i = U$ .

Note that we are interested in an *online* scenario, in which subsets arrive one at a time, and so the order of the subsets becomes important. In order to take this into account, we define an ordered tuple of subsets, which we will call a subset sequence, by  $\mathbf{S} = [S_1, S_2, \dots]$ . Note that a permutation of  $\mathbf{S}$  is distinct from  $\mathbf{S}$ . When  $\mathbf{S}$  is mentioned in the context of an offline algorithm however, it is equivalent to  $\mathcal{S}$ , since all subset arrivals are known in advance to an offline algorithm. We also define the *concatenation* of two subset sequences  $\mathbf{S}_a = [S_1^a, S_2^a, \dots, S_x^a]$  and  $\mathbf{S}_b = [S_1^b, S_2^b, \dots, S_y^b]$  by  $\mathbf{S}_a \wedge \mathbf{S}_b = [S_1^a, S_2^a, \dots, S_x^a, S_1^b, S_2^b, \dots, S_y^b]$ . Similarly,  $\mathbf{S} \wedge S$  for some subset  $S$  corresponds to adding  $S$  to the end of subset sequence  $\mathbf{S}$ . For the creation of  $\mathbf{S}$  from subsets, we use the familiar ordered set notation, e.g.  $\mathbf{S} = [S_j : S_j = \{j\}, j \in U]$  is effectively  $\mathbf{S} = [\{1\}, \{2\}, \dots, \{n\}]$ .

Define an *allocation*  $\mathcal{M}$  as a partition of  $\mathcal{S}$  into  $P_1, P_2, \dots$ . An allocation made by an algorithm  $\mathcal{A}$  on a sequence of subsets  $\mathbf{S}$  is denoted by  $\mathcal{M}_{\mathcal{A}}(\mathbf{S})$ . We would like to make an allocation  $\mathcal{M}$  such that the maximum number of  $P_i$ s form set covers. This is equivalent to solving the *maximum disjoint set cover problem* (DSCP) [3], defined as follows:

**Definition 1 (DSCP [3])** *Given a universe  $U$  and a set of subsets  $\mathcal{S}$ , find as many set covers  $C$  as possible such that they are all pairwise disjoint (i.e.  $C_i \cap C_j = \emptyset \forall i \neq j$ ).*

In the *offline* case of the DSCP, an allocation is made with knowledge of  $\mathcal{S}$  in its entirety. However, in this paper, we are interested in the scenario in which the algorithm is forced to make an allocation *online*. An online algorithm must assign a subset to one of the partitions  $P_i$  as soon as it arrives, and this assignment cannot be changed subsequently. The objective is to maximize the number of  $P_i$ s that form set covers at the end of all subset arrivals, relative to the optimal offline algorithm.

As an analogy, one can picture the online version of the DSCP as in Figure 1. Subsets are represented by balls containing elements from  $U$ , and partitions by bins. The online algorithm has the objective of assigning balls to bins, or in other words, making an allocation. The balls arrive one by one, and the algorithm must drop each into a particular bin as and when it arrives. The objective is to ensure that after all arrivals, the number of bins that form set covers is maximized, relative to the optimal offline allocation.

Define  $T(\mathcal{M}, \mathbf{S})$  as the number of  $P_i$ s that form set covers after an online allocation  $\mathcal{M}$  is made on a subset sequence  $\mathbf{S}$ . We also define  $\mathcal{M}_{off}^*(\mathbf{S})$  as the offline allocation that maximizes  $T(\mathcal{M}, \mathbf{S})$ , and call this maximum  $T(\mathcal{M}_{off}^*, \mathbf{S})$ .

Since,  $\cup_{S_i \in \mathcal{S}} S_i = U$ , the following is trivially established:

$$T(\mathcal{M}_{off}^*, \mathbf{S}) \geq 1. \quad (1)$$

To analyse the performance of online algorithms for the DSCP, a figure of merit is the *competitive ratio* [11]. In this paper, we will denote the competitive ratio of an online algorithm  $\mathcal{A}$  on a sequence of subsets  $\mathbf{S}$  by  $\mu_{\mathbf{S}}(\mathcal{A})$ , defined as

$$\mu_{\mathbf{S}}(\mathcal{A}) = \frac{T(\mathcal{M}_{off}^*, \mathbf{S})}{T(\mathcal{M}_{\mathcal{A}}, \mathbf{S})}. \quad (2)$$

Note that  $\mu_{\mathbf{S}}(\mathcal{A}) \geq 1 \forall \mathbf{S}, \mathcal{A}$ . Also define the worst case competitive ratio  $\mu(\mathcal{A})$  of an algorithm  $\mathcal{A}$  as its highest competitive ratio over all subset sequences.

$$\mu(\mathcal{A}) = \max_{\mathbf{S}} \mu_{\mathbf{S}}(\mathcal{A}) = \max_{\mathbf{S}} \frac{T(\mathcal{M}_{off}^*, \mathbf{S})}{T(\mathcal{M}_{\mathcal{A}}, \mathbf{S})}. \quad (3)$$

Our objective is to design online algorithms with minimum worst-case competitive ratio. From hereon, the term competitive ratio will be used to mean worst case competitive ratio, unless mentioned otherwise. A good online algorithm  $\mathcal{A}$  will have  $\mu(\mathcal{A})$  close to unity; an online algorithm  $\mathcal{A}$  with  $\mu(\mathcal{A})$  increasing with the input parameters is not desirable.

We now introduce some notation that will be used in the rest of this paper. Given  $U$  and  $\mathcal{S}$  (or  $\mathbf{S}$ ), it is useful to define the frequency  $F_i(\mathcal{S})$  of element  $i \in U$  as the number of subsets in  $\mathcal{S}$  that it appears in, i.e.  $F_i(\mathcal{S}) = \#\{S_j \in \mathcal{S} : i \in S_j\}$ . We also define  $F_{min}(\mathcal{S}) = \min_i F_i(\mathcal{S})$ . In order to simplify notation, we will use just  $F_i$  and  $F_{min}$  when there is no ambiguity about the sequence of subsets under consideration.

It is easy to see that  $F_{min}$  is an upper bound on the optimal solution of the DSCP, since each set cover must contain the

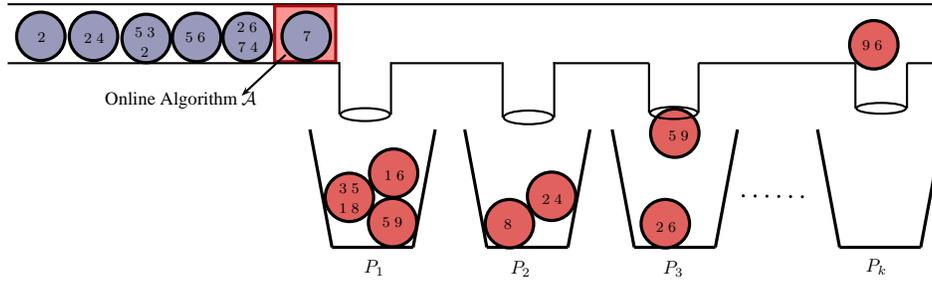


Fig. 1. An online allocation  $\mathcal{M}_A$  by an online algorithm  $\mathcal{A}$  for the universe  $U = \{1, 2, \dots, 9\}$  and a sequence of subsets  $\mathbf{S}$  depicted by balls. Notice that  $P_1$  requires the elements 2, 4 and 7 to form a set cover. Also, red balls can no longer be reallocated.  $T(\mathcal{M}_{off}^*, \mathbf{S}) = 2$  and  $T(\mathcal{M}_A, \mathbf{S}) \leq 1$ , since  $F_1 = 2$ . Note also that the online algorithm may choose to either place a ball along with others in a pre-existing bin, or create another bin.

element with frequency  $F_{min}$  and all set covers must be disjoint. Therefore, for all algorithms  $\mathcal{A}$ :

$$T(\mathcal{M}_A, \mathbf{S}) \leq T(\mathcal{M}_{off}^*, \mathbf{S}) \leq F_{min}. \quad (4)$$

Finding  $T(\mathcal{M}_{off}^*, \mathbf{S})$  optimally for arbitrary  $\mathbf{S}$  (or providing an offline algorithm to the DSCP) is NP-complete [1]. In this paper, however, we will calculate the competitive ratio of our online algorithms  $\mu$  without restricting the optimal offline algorithm to run in polynomial time. We will therefore take  $T(\mathcal{M}_{off}^*, \mathbf{S})$  to be the number of disjoint set covers returned by the optimal offline algorithm, notwithstanding its time complexity. However, finding even an expression for  $T(\mathcal{M}_{off}^*, \mathbf{S})$  combinatorially is not tractable, and so finding  $\mu$  exactly is difficult. We will therefore use (4) to bound  $\mu$ .

We will now analyse in the next section exactly how much information an online DSCP algorithm would need in advance in order for it to perform with a non-trivial competitive ratio.

### III. ESSENTIAL INFORMATION FOR AN ONLINE DSCP ALGORITHM

Before going into the crux of this section, we first introduce the trivial online algorithm GreedyCover  $\mathcal{V}$ .

**Definition 2 (GreedyCover  $\mathcal{V}$ )** *The allocation  $\mathcal{M}_{\mathcal{V}}$  is made such that it always completes a set cover before moving on to the next one. In other words, each incoming subset is placed in partition  $P_i$  until  $P_i$  becomes a set cover, after which the next subset is placed in  $P_{i+1}$ .*

**Lemma III.1** *The competitive ratio of  $\mathcal{V}$ ,  $\mu(\mathcal{V}) \leq F_{min}$ .*

*Proof:* Algorithm  $\mathcal{V}$  will always return at least one set cover, since  $\cup_{S_i \in \mathbf{S}} S_i = U \forall \mathbf{S}$ . In other words,  $T(\mathcal{M}_{\mathcal{V}}, \mathbf{S}) \geq 1 \forall \mathbf{S}$ . The proof of Lemma III.1 follows from (4), since  $F_{min}$  is an upper bound on  $T(\mathcal{M}_{off}^*, \mathbf{S})$ . ■

Similarly, it is clear that  $\mu(\mathcal{A}) \leq F_{min}$  for all algorithms  $\mathcal{A}$  that always produce at least one set cover. We will now show that no online algorithm can perform better than algorithm  $\mathcal{V}$  with prior knowledge of just  $U$  and  $|\mathcal{S}|$ .

**Theorem III.2** *The competitive ratio of any online algorithm is lower bounded by  $F_{min}$ , i.e.,  $\mu(\mathcal{A}) \geq F_{min}$  even when  $\mathcal{A}$  is given the universe  $U$  and number of subsets  $|\mathcal{S}|$  a-priori.*

*Proof:* We will present two sequences of subsets  $\mathbf{S}_1$  and  $\mathbf{S}_2$  such that  $\min_{\mathcal{A}} \max(\mu_{\mathbf{S}_1}(\mathcal{A}), \mu_{\mathbf{S}_2}(\mathcal{A})) = F_{min}$ , where the minimum is taken over all online algorithms  $\mathcal{A}$ . This would serve as a proof of Theorem III.2, by (3).

Let the universe be  $U = \{1, 2, \dots, n\}$ . This is given to the online algorithm a-priori, along with  $|\mathcal{S}|$ . Therefore,  $|\mathbf{S}_1| = |\mathbf{S}_2| = |\mathcal{S}|$  is fixed. Let  $|\mathcal{S}|$  be sufficiently large.

Define  $\mathbf{S}_{com} = [\{1, 2\}, \{1, 3\}, \{1, 4\}, \dots, \{1, n\}]$ . Note that  $|\mathbf{S}_{com}| = n - 1$ . Also define  $\mathbf{S}_{1r} = [\{1\}, \{1\}, \dots, |\mathcal{S}| - (n - 1) \text{ times}]$ . Now, let  $\mathbf{S}_1 = \mathbf{S}_{com} \wedge \mathbf{S}_{1r}$ . Note that  $T(\mathcal{M}_{off}^*, \mathbf{S}_1) = 1$ , since a set cover can be obtained by the combination of all the subsets in  $\mathbf{S}_{com}$ .

Now, let  $\mathbf{S}_{2r} = [S_j : S_j = U \setminus \{1, j + 1\}, 1 \leq j \leq n - 1]$ . Also define  $\mathbf{S}_{2e} = [\{2\}, \{2\}, \dots, |\mathcal{S}| - (2n - 2) \text{ times}]$ . Let  $\mathbf{S}_2 = \mathbf{S}_{com} \wedge \mathbf{S}_{2r} \wedge \mathbf{S}_{2e}$ . It is clear that  $T(\mathcal{M}_{off}^*, \mathbf{S}_2) = n - 1 = F_{min}(\mathbf{S}_2)$ , since the  $j$ th subset of  $\mathbf{S}_{2r}$  is the complement of the  $j$ th subset in  $\mathbf{S}_{com}$  with respect to  $U$ .

Notice that the first  $n - 1$  subsets of both  $\mathbf{S}_1$  and  $\mathbf{S}_2$  are identical, and represented by the sequence  $\mathbf{S}_{com}$ . Consider two classes of online algorithms represented by  $\mathcal{X}$  and  $\mathcal{Y}$ . Let any algorithm  $\mathcal{A} \in \mathcal{X}$  make an allocation  $\mathcal{M}_A$  such that all subsets in  $\mathbf{S}_{com}$  end up in the same partition, and let any algorithm  $\mathcal{B} \in \mathcal{Y}$  make an allocation  $\mathcal{M}_B$  such that all subsets in  $\mathbf{S}_{com}$  do not end up in the same partition. Note that the classes  $\mathcal{X}$  and  $\mathcal{Y}$  are disjoint and together span all online algorithms for the DSCP. The following relations become immediately clear:

$$\begin{aligned} \mu_{\mathbf{S}_1}(\mathcal{A}) &= 1 \quad \forall \mathcal{A} \in \mathcal{X}, \\ \mu_{\mathbf{S}_1}(\mathcal{B}) &= \infty \quad \forall \mathcal{B} \in \mathcal{Y}, \\ \mu_{\mathbf{S}_2}(\mathcal{A}) &= F_{min} \quad \forall \mathcal{A} \in \mathcal{X}, \\ \mu_{\mathbf{S}_2}(\mathcal{B}) &= 1 \quad \text{for some } \mathcal{B} \in \mathcal{Y}. \end{aligned}$$

We therefore arrive at the fact that

$$\min_{\mathcal{A} \in \mathcal{X}} \max(\mu_{\mathbf{S}_1}(\mathcal{A}), \mu_{\mathbf{S}_2}(\mathcal{A})) = F_{min}, \quad (5)$$

$$\min_{\mathcal{B} \in \mathcal{Y}} \max(\mu_{\mathbf{S}_1}(\mathcal{B}), \mu_{\mathbf{S}_2}(\mathcal{B})) = \infty. \quad (6)$$

This shows that

$$\min_{\mathcal{C}} \max(\mu_{\mathbf{S}_1}(\mathcal{C}), \mu_{\mathbf{S}_2}(\mathcal{C})) = F_{min}, \quad (7)$$

where the minimum over  $\mathcal{C} = \mathcal{X} \cup \mathcal{Y}$  is taken over all online algorithms for the DSCP. The proof is therefore complete. We also point out here that  $F_{\min}(\mathbf{S}_1) = 1$  and  $F_{\min}(\mathbf{S}_2) = n - 1$ , so if the online algorithm was provided with  $F_{\min}$  it could have chosen to use just one partition for  $\mathbf{S}_1$  and  $n - 1$  partitions for  $\mathbf{S}_2$ , thus improving its worst case competitive ratio. ■

No online algorithm can therefore outperform the trivial GreedyCover algorithm  $\mathcal{V}$  (Definition 2) with knowledge of just  $U$  and  $|\mathcal{S}|$ , from Lemmas III.1 and III.2. One can similarly show that even if  $\max_i F_i$  or  $\frac{1}{n} \sum_i F_i$  are provided along with  $U$  to an online algorithm, its competitive ratio is lower bounded by  $F_{\min}$ . We will therefore assume the following.

**Remark III.3** *From hereon, all online algorithms know  $U$  and  $F_{\min}$  a-priori.*

In the next section, we will construct an online algorithm that is provided  $F_{\min}$  in advance and performs with a non-trivial competitive ratio.

#### IV. AN ONLINE DSCP ALGORITHM

We will use the offline DSCP algorithm of [3], and make modifications such that it becomes an online algorithm. We will first present a few ideas and the algorithm from [3].

##### A. Hypergraph representation

The universe  $U$  and set of subsets  $\mathcal{S}$  can also be represented as a hypergraph, as shown in [12]. Define a hypergraph  $\mathcal{H}_{U,\mathcal{S}}(V, E)$  representing a universe  $U$  and subsets  $\mathcal{S}$ , having vertex set  $V$  and hyperedge set  $E$  as follows:

**Definition 3** ( $\mathcal{H}_{U,\mathcal{S}}(V, E)$ ) *Each subset  $S_j \in \mathcal{S}$  is represented by a vertex  $v_j \in V$ . A hyperedge  $e_i \in E$  contains vertex  $v_j$  if element  $i \in U$  is such that  $i \in S_j$ .*

While dealing with a sequence of subsets  $\mathbf{S}$ , we will denote the corresponding hypergraph representation by  $\mathcal{H}_{U,\mathbf{S}}$ .

Figure 2 illustrates an example construction of  $\mathcal{H}_{U,\mathcal{S}}(V, E)$  from  $U$  and  $\mathcal{S}$ . Note that there are  $|\mathcal{S}|$  vertices and  $n$  hyperedges in  $\mathcal{H}_{U,\mathcal{S}}(V, E)$ . The set of vertices in hyperedge  $e_i$  is denoted by  $V(e_i)$ , and so  $|V(e_i)| = F_i$ . Hence, the smallest number of vertices in any hyperedge  $\min_i |V(e_i)| = F_{\min}$ .

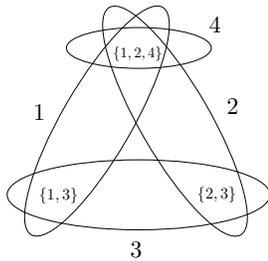


Fig. 2. Hypergraph  $\mathcal{H}_{U,\mathcal{S}}$  for  $U = \{1, 2, 3, 4\}$  and  $\mathcal{S} = \{\{1, 2, 4\}, \{2, 3\}, \{1, 3\}\}$

The DSCP on  $U$  and  $\mathcal{S}$  can be solved offline on  $\mathcal{H}_{U,\mathcal{S}}$  by an operation known as *polychromatic colouring*.

**Definition 4 (Polychromatic Colouring)** *Colour the vertices of the hypergraph with the maximum number of colours such that each hyperedge contains vertices of all colours.*

Note that each colour in a polychromatic colouring of  $\mathcal{H}_{U,\mathcal{S}}$  corresponds to a set cover of  $U$  using subsets in  $\mathcal{S}$ . The fact that vertices must have different colours forces the set covers to be disjoint, and maximizing the number of colours maximizes the number of disjoint set covers, i.e., solves the DSCP.

##### B. Deterministic Offline Algorithm [3]

The offline algorithm in [3] solves the DSCP through the polychromatic colouring of  $\mathcal{H}_{U,\mathcal{S}}$ . Let us first refresh some notation from [3], and then present a slightly different exposition of the algorithm from [3]. Let  $V(e)$  denote the set of vertices in hyperedge  $e$ . An incomplete colouring of a hypergraph is one in which there exist colours that are not present in all hyperedges. These colours are said to be *invalid*, since the subset collections that they correspond to do not form set covers. Let us colour the hypergraph using  $\ell = F_{\min} / \ln(n \ln n)$  colours, using the set  $[\ell]$ . Given an incomplete colouring of a hypergraph using colours in set  $[\ell]$ , we denote by random variable  $L$  the number of invalid colours. We also define an indicator random variable  $X_c$  for each  $c \in [\ell]$ , which is 1 if colour  $c$  is invalid and 0 otherwise. The following relation is readily established, as

$$L = \sum_{c \in [\ell]} X_c. \quad (8)$$

We also define another indicator random variable  $Y_{e,c}$  for each hyperedge-colour pair  $(e \in E, c \in [\ell])$ , which is 1 if hyperedge  $e$  does not contain any vertex coloured with colour  $c$ , and 0 otherwise. The relation between  $X_c$  and  $Y_{e,c}$  is as follows:

$$X_c \leq \sum_{e \in E} Y_{e,c}, \quad \forall c \in [\ell]. \quad (9)$$

And so, by (8) and (9),

$$L \leq \sum_{c \in [\ell]} \sum_{e \in E} Y_{e,c}. \quad (10)$$

The offline algorithm of [3] POLYOff operates in two phases. In Phase I, it colours all vertices uniformly randomly using  $\ell = F_{\min} / \ln \ln n$  colours. At this point, note from (10) that

$$\begin{aligned} \mathbf{E}[L] &\leq \sum_{c \in [\ell]} \sum_{e \in E} \mathbf{P}[Y_{e,c} = 1] \leq n \cdot \ell \cdot \left(1 - \frac{1}{\ell}\right)^{|V(e)|}, \\ &\leq n \ell e^{-|V(e)|/\ell} \leq n \ell e^{-\ln(n \ln n)}, \\ &= \ell / \ln n. \end{aligned} \quad (11)$$

This essentially achieves a randomized polychromatic colouring with at most  $\ell / \ln n$  invalid colours. After this comes Phase II of the POLYOff algorithm - the recolouring phase - which works as follows to achieve a deterministic colouring. Order the vertices arbitrarily as  $v_1, v_2, \dots$  and *recolour* them in this order. Let vertex  $v_i$  be recoloured by colour  $c_i$ . Let  $|V^u(e)|$  denote the number of vertices in hyperedge  $e$  that have not been recoloured. Now for all hyperedges  $e$ , the probability that

$e$  does not contain colour  $c$  given that the vertices  $v_1, \dots, v_i$  have been recoloured with colours  $c_1, \dots, c_i$  is 0 if there exists a vertex in  $e$  which has already been recoloured with colour  $c$ . Otherwise, the probability is given by  $(1 - 1/\ell)^{|V^u(e)|}$ , since vertices  $v_{i+1}, \dots, v_{|S|}$  were each coloured uniformly randomly with one of the  $\ell$  colours in Phase I. So,

$$\mathbf{P}[Y_{e,c} = 1 | c_1, c_2, \dots, c_i] = \begin{cases} 0, & \text{if } \exists q \leq i \text{ s.t. } v_q \in V(e) \\ & \text{and } c_q = c \\ (1 - 1/\ell)^{|V^u(e)|} & \text{otherwise.} \end{cases} \quad (12)$$

After the vertices  $v_1, v_2, \dots, v_j$  are recoloured, we denote the conditional expectation of the number of invalid colours by  $\mathbf{E}[L | c_1, c_2, \dots, c_j]$ . Note that

$$\mathbf{E}[L | c_1, c_2, \dots, c_i] \leq \sum_{e \in E} \sum_{c \in [\ell]} \mathbf{P}[Y_{e,c} = 1 | c_1, c_2, \dots, c_i].$$

Recolour  $v_1$  uniformly randomly from  $[\ell]$ . Given that  $v_1, \dots, v_{i-1}$  have been recoloured, recolour  $v_i$  such that

$$\mathbf{E}[L | c_1, c_2, \dots, c_{i-1}, c_i] \leq \mathbf{E}[L | c_1, c_2, \dots, c_{i-1}] \quad (13)$$

Note that such a recolouring exists, since due to the colouring of Phase I being uniformly random,

$$\mathbf{E}[L | c_1, c_2, \dots, c_{i-1}] = (1/\ell) \sum_{c_i \in [\ell]} \mathbf{E}[L | c_1, c_2, \dots, c_{i-1}, c_i],$$

which is a convex combination. So there exists at least one colour  $c_i$  for which (13) holds. Before recolouring (at the end of Phase I), (11) tells us that the number of invalid colours was less than  $\ell / \ln n$ , and we can ensure throughout the recolouring process that that number does not increase. Therefore, at the end of the algorithm, the number of colours that form a polychromatic colouring of  $\mathcal{H}_{U,S}$  is at least

$$\ell - \frac{\ell}{\ln n} = \frac{F_{\min}}{\ln n} \left( 1 - \frac{\ln \ln n + 1}{\ln(n \ln n)} \right). \quad (14)$$

The vertices coloured with the invalid colours do not correspond to set covers. The `POLYOFF` algorithm therefore obtains  $\frac{F_{\min}}{\ln n} (1 - o(1))$  disjoint set covers, where the  $o(1)$  term is  $< 1/2$  and goes to zero as  $n \rightarrow \infty$ .

### C. The Online Extension

Recall Remark III.3, by which we will assume that all online algorithms know  $F_{\min}$  in advance. We will now extend ideas from the `POLYOFF` algorithm presented in Section IV-B to produce the `POLYON` online algorithm.

Before we do that however, we introduce a randomized online algorithm, `RANDCOLOUR`, which colours each subset with one of  $F_{\min} / \ln n$  colours uniformly randomly on arrival. This effectively performs Phase I of the `POLYOFF` algorithm on the incoming subsets and produces, in expectation, at least  $F_{\min} / \ln n (1 - o(1))$  disjoint set covers, by (14).

We now make the following claim:

**Theorem IV.1** *There exists a deterministic online algorithm for the DSCP with competitive ratio  $\ln n$ .*

We will prove Theorem IV.1 by constructing a deterministic online algorithm called the `POLYON` algorithm. Note that the hypergraph  $\mathcal{H}_{U,S}$  is not available in advance to any online algorithm, and must be constructed as and when subsets arrive. To aid in this construction, we introduce a few concepts.

**Definition 5 (Shrinking)** *Shrinking a hyperedge  $e$  of a hypergraph  $G(V, E)$  to  $k$  vertices (or size  $k$ ) corresponds to removing elements of  $V(e)$  such that  $|V(e)|$  becomes equal to  $k$ . The removed elements of  $V(e)$  can be arbitrary. Note that the vertex set  $V$  is not disturbed; hyperedge  $e$  is simply made to connect fewer vertices.*

Let  $i(e)$  represent the element  $i \in U$  that was represented by hyperedge  $e$  in  $\mathcal{H}_{U,S}$ . Note that shrinking a hyperedge  $e$  in  $\mathcal{H}_{U,S}$  to  $k$  vertices corresponds to removing some  $F_{i(e)} - k$  occurrences of the element  $i(e)$  from the subsets that contain it. For example, in Figure 2, the hyperedge 1 can be shrunk to size 1 either by modifying the subset  $\{1, 2, 4\}$  to  $\{2, 4\}$ , or by modifying the subset  $\{1, 3\}$  to  $\{3\}$ .

We will now present the following Lemma.

**Lemma IV.2** *Let the hyperedges in hypergraph  $\mathcal{H}$  be shrunk to arbitrary sizes to produce another hypergraph  $\mathcal{H}'$ . Then any feasible polychromatic colouring of  $\mathcal{H}'$  will be a feasible polychromatic colouring of  $\mathcal{H}$ .*

*Proof:* Let hyperedge edge  $e \in \mathcal{H}$  be shrunk to  $e' \in \mathcal{H}'$ . Consider a feasible polychromatic colouring of  $\mathcal{H}'$  with some  $c$  colours. By definition, all hyperedges  $e' \in \mathcal{H}'$  contain all  $c$  colours. By Definition 5, for every  $e' \in \mathcal{H}'$ ,  $V(e') \subseteq V(e)$  where  $e$  is the corresponding hyperedge in  $\mathcal{H}$ . Therefore, each hyperedge  $e \in \mathcal{H}$  contains all  $c$  colours. This is a feasible polychromatic colouring of  $\mathcal{H}$ . ■

Let a hypergraph  $\mathcal{H}_{U,S}^{\min}$  represent the hypergraph constructed from  $\mathcal{H}_{U,S}$  by shrinking each of its hyperedges to size  $F_{\min}$ . As a corollary to Lemma IV.2, we therefore have:

**Corollary IV.3** *Any polychromatic colouring of  $\mathcal{H}_{U,S}^{\min}$  with  $k$  colours is a polychromatic colouring of  $\mathcal{H}_{U,S}$  with  $k$  colours.*

The `POLYON` algorithm will aim to construct hypergraph  $\mathcal{H}_{U,S}^{\min}$  and polychromatically colour it with  $F_{\min} / \ln n$  colours, online. Note that two things must be accomplished together: **(a)** Online shrinking of  $\mathcal{H}_{U,S}$  to produce  $\mathcal{H}_{U,S}^{\min}$  from  $\mathbf{S}$ , and **(b)** Online polychromatic colouring of  $\mathcal{H}_{U,S}^{\min}$ .

The `POLYON` algorithm will accomplish **(a)** by the following `ONLINESHRINK` method. Note that hypergraph  $\mathcal{H}_{U,S}^{\min}$  can be constructed from a sequence of subsets  $\mathbf{S}$  by constructing another sequence  $\mathbf{S}^{\min}$  as and when subsets in  $\mathbf{S}$  arrive, such that  $F_i(\mathbf{S}^{\min}) = F_{\min}(\mathbf{S}) \forall i \in U$ . One can simply do this online by ignoring all occurrences of all elements  $i$  after their  $F_{\min}(\mathbf{S})$ th occurrence. To present this formally, let us denote by  $\mathbf{S}_j$  the sub-sequence of  $\mathbf{S}$  containing its first  $j$  subsets and let  $S_j$  denote the  $j$ th subset of  $\mathbf{S}$ . Similarly define  $\mathbf{S}_j^{\min}$  and subset  $S_j^{\min}$ . For every subset arrival  $S_j$ , as long as  $F_i(\mathbf{S}_j) \leq F_{\min}(\mathbf{S}) \forall i \in U$ , we set  $S_j^{\min} = S_j$ . If after some subset  $S_j$  arrives, if  $F_i(\mathbf{S}_j) > F_{\min}(\mathbf{S})$  for some  $i \in U$ , we set  $S_j^{\min} = S_j \setminus \{i\}$ . We can thereby ensure that after all

subset arrivals,  $F_i(\mathbf{S}^{min}) = F_{min}(\mathbf{S}) \forall i \in U$ . We construct  $\mathcal{H}_{U,\mathbf{S}}^{min}$  as  $\mathcal{H}_{U,\mathbf{S}^{min}}$ , in an online fashion.

Since the shrinking of hypergraph  $\mathcal{H}_{U,\mathbf{S}}$  to produce  $\mathcal{H}_{U,\mathbf{S}}^{min}$  can be accomplished online, from hereon, we will assume that the OnlineShrink process is carried out by the PolyOn algorithm for any arrival sequence of subsets. Now, we can assume that the subset sequence  $\mathbf{S}^{min}$  arrives, and aim to achieve (b) simultaneously with (a), i.e. to polychromatically colour  $\mathcal{H}_{U,\mathbf{S}^{min}}$  (or  $\mathcal{H}_{U,\mathbf{S}}^{min}$ ) online, as follows, by the OnlineColour method:

The PolyOn algorithm will first create the set of  $F_{min}/\ln(n \ln n)$  colours  $[\ell]$ . It will assume that all subsets (vertices) in  $\mathbf{S}^{min}$  have already been coloured uniformly randomly with a colour from  $[\ell]$  prior to arrival. It will now recolour each incoming vertex with the prior knowledge that all hyperedges in  $\mathcal{H}_{U,\mathbf{S}^{min}}$  have exactly  $F_{min}$  vertices, i.e.  $|V(e)| = F_{min} \forall e$ . Recall Phase II of the PolyOff algorithm, in which we calculated the probability that hyperedge  $e$  does not contain colour  $c$  after vertices  $v_1$  through  $v_j$  had been recoloured, as  $\mathbf{P}[Y_{e,c} = 1 | c_1, c_2, \dots, c_j]$ . The expression for  $\mathbf{P}[Y_{e,c} = 1 | c_1, c_2, \dots, c_j]$  was given by (12). The PolyOn algorithm will attempt to recolour vertex  $v_j$  similarly. but it is not provided  $|V^u(e)|$  directly. It can, however, calculate  $|V^u(e)| = |V(e)| - |V^d(e)| = F_{min} - |V^d(e)|$ , where  $|V^d(e)|$  is the number of vertices in hyperedge  $e$  that have been recoloured. This is all information that is available to the online algorithm, and it can therefore calculate

$$\mathbf{P}[Y_{e,c} = 1 | c_1, c_2, \dots, c_j] = \begin{cases} 0, & \text{if } \exists q \leq j \text{ s.t. } v_q \in V(e) \\ & \text{and } c_q = c \\ (1 - 1/\ell)^{F_{min} - |V^d(e)|} & \text{else.} \end{cases}$$

for each hyperedge-colour pair  $(e, c)$  in an online fashion! The PolyOn algorithm can then use (IV-C) to calculate the conditional expectation of the number of invalid colours  $\mathbf{E}[L | c_1, c_2, \dots, c_i] \leq \sum_{e \in E} \sum_{c \in [\ell]} \mathbf{P}[Y_{e,c} = 1 | c_1, c_2, \dots, c_i]$ . It will then recolour vertex  $v_{i+1}$  such that  $\mathbf{E}[L | c_1, c_2, \dots, c_{i-1}, c_{i+1}] \leq \mathbf{E}[L | c_1, c_2, \dots, c_i], \forall i$ . Note that after all vertex arrivals, this will result in a polychromatic colouring of the hypergraph  $\mathcal{H}_{U,\mathbf{S}^{min}}$  with  $F_{min}/\ln n$  colours, by an equation similar to (14). By Corollary IV.3, this is also a polychromatic colouring of the hypergraph  $\mathcal{H}_{U,\mathbf{S}}$  with  $F_{min}/\ln n$  colours, and so the PolyOn algorithm has produced  $F_{min}/\ln n$  disjoint set covers of  $U$  from  $\mathbf{S}$  in an online manner, by executing both the OnlineShrink and OnlineColour operations together.

We know that the PolyOn algorithm (which we represent now by  $\mathcal{P}$ ) returns at least  $F_{min}/\ln n$  set covers for all input sequences  $\mathbf{S}$ . Using (4), we can see that

$$\mu(\mathcal{P}) = \max_{\mathbf{S}} \frac{T(\mathcal{M}_{off}^*, \mathbf{S})}{T(\mathcal{M}_{\mathcal{P}}, \mathbf{S})} = \frac{T(\mathcal{M}_{off}^*, \mathbf{S})}{\min_{\mathbf{S}} T(\mathcal{M}_{\mathcal{P}}, \mathbf{S})} \leq \ln n.$$

This proves Theorem IV.1.

#### D. Advantages of the PolyOn Online Algorithm

Firstly, notice that for all input sequences in which  $F_{min} \geq \ln n$ , the PolyOn algorithm performs better than the trivial

GreedyCover algorithm  $\mathcal{V}$ .

The second advantage is that the PolyOn algorithm is polynomial time optimal. It was shown in [2] that no polynomial time algorithm (offline or online) can produce more than  $T(\mathcal{M}_{off}^*, \mathbf{S})/\ln n$  disjoint set covers  $\forall \mathbf{S}$  unless  $NP \subseteq DTIME(n^{O(\log \log n)})$ . Therefore, no online algorithm that runs in polynomial time can have a better competitive ratio than the PolyOn algorithm. The PolyOn algorithm also matches the performance of the best polynomial time offline algorithm.

The following Lemma confirms the third advantage.

**Lemma IV.4** *There exists an infinite family of subset sequences  $\mathbf{S}$  for which the competitive ratio of the PolyOn algorithm is 1, i.e.  $\mu_{\mathbf{S}}(\mathcal{P}) = 1$ .*

*Proof:* We will show an infinite family of subset sequences  $\mathbf{S}$  for which  $T(\mathcal{M}_{off}^*, \mathbf{S}) \leq F_{min}/\ln n$ . Note that for these sequences,  $\mu_{\mathbf{S}}(\mathcal{P}) = 1$ . To show this, consider the *minimum set cover* problem (MinSetCover), in which given  $U$  and  $\mathcal{S}$ , we are required to find the set cover  $C \subseteq \mathcal{S}$  such that  $|C|$  is minimized. The integer programming (IP) formulation of MinSetCover is known to have an integrality gap  $\ln n$  [13], or in other words, for any problem instance  $\mathcal{I}$ , the ratio of the IP's optimal solution ( $OPT(IP_{\mathcal{I}})$ ) to the LP relaxation's optimal solution ( $OPT(LP_{\mathcal{I}})$ ), in which each subset is given a coefficient between 0 and 1, is at most  $\ln n$ . Consider one such  $\ln n$  integrality gap instance  $\mathcal{I}_0$  in which the frequency of all elements is equal to  $F_{min}$ , and  $\frac{OPT(IP_{\mathcal{I}_0})}{OPT(LP_{\mathcal{I}_0})} = \ln n$ . For an example of such an instance  $\mathcal{I}_0$ , we refer the reader to Example 13.4 of [14]. Note that  $OPT(LP_{\mathcal{I}_0}) = |\mathcal{S}|/F_{min}$ , which is accomplished by setting the coefficient of each subset in  $\mathcal{S}$  to  $1/F_{min}$ . The optimal IP solution  $OPT(IP_{\mathcal{I}_0})$  will therefore be at least  $\frac{|\mathcal{S}|}{F_{min}} \ln n$ , since this is a  $\ln n$  integrality gap instance. Each set cover in this instance therefore consists of at least  $N = \frac{|\mathcal{S}|}{F_{min}} \ln n$  subsets, and so the number of disjoint set covers can be at most  $|\mathcal{S}|/N = F_{min}/\ln n$ . We have therefore shown an infinite family (for different  $n$ ) for which  $T(\mathcal{M}_{off}^*, \mathbf{S}) \leq F_{min}/\ln n$  and  $\mu_{\mathbf{S}}(\mathcal{P}) = 1$ . ■

## V. THE LOWER BOUND

In this section, for ease of exposition, we will first show a lower bound of  $\Omega((\ln n)^{1/3})$  on the competitive ratio of any online algorithm, which we will then improve to  $\Omega((\ln n)^{1/2})$ . We first present a brief overview of our method.

**Overview 1** *We will generate a subset sequence for an online algorithm consisting of 3 sub-sequences - let us call them  $X$ ,  $Y$  and  $Z$  - in that order. We will first provide the sequence  $X$ , on which an online algorithm  $\mathcal{A}$  will make an allocation  $\mathcal{M}_{\mathcal{A}}(X)$ . Depending on  $\mathcal{M}_{\mathcal{A}}(X)$ , we will then provide an adversarial sequence  $Y(\mathcal{M}_{\mathcal{A}}(X))$  that upper bounds the number of disjoint set covers that can be formed by  $\mathcal{A}$ . Lastly, we will provide sequence  $Z(\mathcal{M}_{\mathcal{A}}(X))$  such that the optimal offline algorithm can construct a larger number of disjoint set covers by making reallocations, which is not allowed for  $\mathcal{A}$ .*

We thus obtain a lower bound on the competitive ratio of all online algorithms.

In this section, we define the universe as  $U = \{0, 1\}^q$ . There are therefore a total of  $n = 2^q$  elements, in which the  $i$ th element  $u_i$  is represented by the  $q$ -bit binary representation of  $i$ , and  $0 \leq i \leq 2^q - 1$ . We also define  $u_i^k$  as the  $k$ th bit of  $u_i$ , for  $0 \leq k \leq q - 1$ . We now form a sequence of subsets  $\mathbf{S}_{com} = [S_j : S_j = \{u_i : u_i^j = 1\}, 1 \leq j \leq q]$ , i.e. subset  $j$  of  $\mathbf{S}_{com}$  contains all elements  $u_i$  that have 1 in their  $j$ th position. Note that  $|\mathbf{S}_{com}| = q$  and that  $|S_j| = 2^{q-1} \forall S_j \in \mathbf{S}_{com}$ . We also see by definition that the frequency of element  $u_i$  in  $\mathbf{S}_{com}$ ,  $F_i(\mathbf{S}_{com}) = \#\{k : u_i^k = 1\} = q - \#\{k : u_i^k = 0\}$ . Also,  $T(\mathcal{M}_{off}^*, \mathbf{S}_{com}) = 0$ , since  $F_{min}(\mathbf{S}_{com}) = 0$ . This is because the all-zero element does not appear in any of the subsets in  $\mathbf{S}_{com}$ . Therefore,  $\mathbf{S}_{com}$  does not contain any set cover.

We will use  $\mathbf{S}_{com}$  as the start of the subset sequence, i.e. sequence  $X$  in Overview 1, in the proofs of Theorems V.1 and V.5. To that end, note the following. Let any online algorithm  $\mathcal{A}$  make an allocation  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$  on  $\mathbf{S}_{com}$ , and partition its subsets into  $P_1, P_2, \dots$ . After all subsets in  $\mathbf{S}_{com}$  have arrived and been allocated, denote the elements  $u_i \in U$  that are missing from partition  $P_j$  by the set  $E_j(\mathcal{M})$ . We can assume the following without loss of generality, for all allocations  $\mathcal{M}$

$$E_1(\mathcal{M}) = \{u_i : u_i^k = 0, k = \{0, 1, \dots, r(1)\} \text{ for some } r(1) \geq 0\}. \quad (15)$$

In words, partition  $P_1$  does not contain all those elements  $u_i$  that have zeroes in their first  $r(1)$  places, and hence contains the first  $r(1) + 1$  subsets in  $\mathbf{S}_{com}$ . This can always be accomplished by reordering the  $q$  bits. Also, by the definition of subsets in  $\mathbf{S}_{com}$ , the number of subsets in  $P_1$  is  $r(1) + 1$ . Similar to (15), we see that

$$E_2(\mathcal{M}) = \{u_i : u_i^k = 0, k = \{r(1) + 1, \dots, r(2)\} \text{ for some } r(2) > r(1)\}, \quad (16)$$

$$E_j(\mathcal{M}) = \{u_i : u_i^k = 0, k = \{r(j) + 1, \dots, r(j+1)\} \text{ for some } r(j+1) > r(j)\} \forall \mathcal{M}. \quad (17)$$

Let us denote the number of subsets in partition  $P_j$  by  $d_j = r(j) - r(j-1) \forall j \geq 1$  (we define  $r(0) = -1$  for consistency). Without loss of generality, we can assume that  $d_{j+1} \geq d_j$  for all allocations  $\mathcal{M}$ , by reordering partitions if necessary.

After an allocation  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$ , let the maximum number of subsets placed in any partition be  $L$ , where  $0 \leq L \leq q$ . Define the sets  $D_\ell(\mathcal{M}) = \{j : d_j = \ell\}$  for each  $1 \leq \ell \leq L$ . In words,  $D_\ell$  represents the indices of those partitions that contain exactly  $\ell$  subsets from  $\mathbf{S}_{com}$  after an online allocation  $\mathcal{M}$ . To simplify notation, we will use  $D_\ell$  to represent  $D_\ell(\mathcal{M})$  when there is no ambiguity about the allocation  $\mathcal{M}$ .

After the allocation  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$  is done, we also define a *bottleneck element* for each partition  $P_j$  as  $u_{b(j)}$  such that  $u_{b(j)} \in E_j(\mathcal{M})$  and  $\#\{k : u_{b(j)}^k = 0\}$  is minimum, i.e.  $u_{b(j)}$  is the element with the fewest zeroes that does not appear in partition  $j$ . For example, take an allocation in which  $d_1 = 1$  and  $d_2 = 2$ , i.e. partition  $P_1$  contains one subset  $S_1 = \{u_i : u_i^1 =$

$1\}$  and partition  $P_2$  contains 2 subsets  $S_2 = \{u_i : u_i^2 = 1\}$  and  $S_3 = \{u_i : u_i^3 = 1\}$ . In that case, the bottleneck elements for partitions  $P_1$  and  $P_2$  are  $u_{b(1)} = 01111 \dots q - 1$  times and  $u_{b(2)} = 10011 \dots q - 3$  times, respectively. All elements of the universe that are not bottleneck elements are called *non-bottleneck elements*. Note that the frequency in  $\mathbf{S}_{com}$  of a bottleneck element of partition  $j$  is  $F_{b(j)}(\mathbf{S}_{com}) = q - d_j$ . We will combine these bottleneck elements appropriately to form the next sequence (corresponding to  $Y$  in Overview 1) that upper bounds the number of disjoint set covers obtainable by any online algorithm.

Note that all the quantities have been defined with respect to allocations of  $\mathbf{S}_{com}$ . We are now ready to prove that:

**Theorem V.1** *The competitive ratio of any online DSCP algorithm is lower bounded by  $\Omega((\ln n)^{1/3})$ , i.e.  $\mu(\mathcal{A}) = \Omega((\ln n)^{1/3}) \forall \mathcal{A}$ .*

*Proof:* We will provide a subset sequence  $\mathbf{S}_1$  for which  $T(\mathcal{M}_{off}^*, \mathbf{S}_1) = \Omega(F_{min}(\mathbf{S}_1))$ , and  $T(\mathcal{M}_{\mathcal{A}}, \mathbf{S}_1) = O\left(\frac{F_{min}(\mathbf{S}_1)}{(\ln n)^{1/3}}\right) \forall$  online algorithms  $\mathcal{A}$ . The first  $q$  elements of  $\mathbf{S}_1$  will be identical to  $\mathbf{S}_{com}$ . We will then create the remaining elements of  $\mathbf{S}_1$  adversarially depending on the allocation of  $\mathbf{S}_{com}$  in order to limit the number of disjoint set covers that can be created by any online algorithm.

Consider an online algorithm  $\mathcal{A}$  that makes an allocation  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$  on the first  $q$  subsets. We will assume (15), (16) and (17), which hold without loss of generality. We now construct the next sequence of subsets  $\mathbf{S}_a$  adversarially. Create one copy of the subset  $S_1^a = \{x : x = u_{b(j)}, j \in D_1\}$ , which contains bottleneck elements of all partitions that contain exactly 1 subset from  $\mathbf{S}_{com}$ . Note that  $F_{b(j)}(\mathbf{S}_{com} \wedge S_1^a) = q \forall j \in D_1$ . We will ensure that these bottleneck elements  $u_{b(j)}$ ,  $j \in D_1$  never arrive later in sequence  $\mathbf{S}_1$ . Now look at all the partitions  $P_j$  where  $j \in D_1$ . They all require their respective bottleneck elements in order to form set covers, and yet there is only one available subset  $S_1^a$  that contains these bottleneck elements. Therefore, at most 1 partition among  $P_j$ ,  $j \in D_1$  can form a set cover. Extending this further, we now create the adversarial sequence  $\mathbf{S}_a$  containing  $\ell$  copies of the subsets  $S_\ell^a = \{x : x = u_{b_j}, j \in D_\ell\}$  for all  $1 \leq \ell \leq L$ , in some arbitrary order. Note that through this,  $F_{b(j)}(\mathbf{S}_{com} \wedge \mathbf{S}_a) = q, \forall j$ . Similar to the argument for partitions  $P_j$  where  $j \in D_1$ , we can argue that a maximum of  $\ell$  partitions among  $P_j$  for  $j \in D_\ell$  can form set covers. Also define a sequence  $\mathbf{S}_{inf}$  which contains an arbitrarily large number of singleton subsets of all non-bottleneck elements in any arbitrary order. Let  $\mathbf{S}_1 = \mathbf{S}_{com} \wedge \mathbf{S}_a \wedge \mathbf{S}_{inf}$ .

Note that  $F_{min}(\mathbf{S}_1) = F_{b(j)}(\mathbf{S}_1)$  for any  $j$ , and  $F_{b(j)}(\mathbf{S}_1) = q \forall j$ , since the non-bottleneck elements appear an infinite number of times. Now that we have constructed  $\mathbf{S}_1$  depending on  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$ , we present the following Lemma.

**Lemma V.2** *The number of disjoint set covers that can be formed by any online algorithm from  $\mathbf{S}_1$  is  $O(q^{2/3})$ , i.e.  $T(\mathcal{M}_{\mathcal{A}}, \mathbf{S}_1) = O(q^{2/3}) \forall$  online algorithms  $\mathcal{A}$ .*

*Proof:* Let  $A_\ell$  denote the number of disjoint set covers that can be formed by partitions that have  $\ell$  subsets from  $\mathbf{S}_{com}$ . Thus,  $A_\ell = \#\{P_j : j \in D_\ell, P_j \text{ forms a set cover.}\}$ . By definition,  $A_\ell \leq |D_\ell|$ . Recall that we have ensured by providing  $\mathbf{S}_a$  that a maximum of  $\ell$  partitions among  $P_j$  for  $j \in D_\ell$  can form set covers for any allocation  $\mathcal{M}_A(\mathbf{S}_{com})$ . In other words,  $A_\ell \leq \min(\ell, |D_\ell|)$  partitions can form set covers for each  $D_\ell$ . The optimization problem over all possible online allocations  $\mathcal{M}_A(\mathbf{S}_{com})$  is therefore the following:

$$\text{Maximize : } \sum_{\ell=1}^L A_\ell \quad (18)$$

$$\text{Subject to } \sum_{\ell=1}^L \ell \cdot |D_\ell| = q, \quad (19)$$

$$A_\ell \leq \min(\ell, |D_\ell|) \quad \forall 1 \leq \ell \leq L \quad (20)$$

where (18) corresponds to maximizing the number of disjoint set covers, (19) is the constraint on the number of subsets available in  $\mathbf{S}_{com}$ , and (20) is the constraint imposed by the bottleneck elements in  $\mathbf{S}_a$ . The optimization is over  $|D_\ell|$  (each  $|D_\ell| = |D_\ell(\mathcal{M})|$  corresponds to some allocation  $\mathcal{M}_A(\mathbf{S}_{com})$ ).

The optimal solution to (18) occurs when  $|D_\ell| = \ell, \forall \ell$ . That would imply that the optimal online allocation corresponds to creating  $\ell$  partitions  $P_j$  each containing  $\ell$  subsets from  $\mathbf{S}_{com}$ , i.e. 1 partition with 1 subset, 2 partitions with 2 subsets and so on. Even after the sequence  $\mathbf{S}_a$  is provided, each partition created by  $\mathcal{M}_A(\mathbf{S}_{com})$  can form a set cover. This is because  $\mathbf{S}_a$  can provide each partition with its bottleneck element, and  $\mathbf{S}_{inf}$  provides all non-bottleneck elements. Therefore, the constraint (19) evaluates to

$$\sum_{\ell=1}^L \ell \cdot |D_\ell| = \sum_{\ell=1}^L \ell^2 = q, \quad (21)$$

which implies that  $O(L^3) = q$ , so  $L = O(q^{1/3})$ . The number of disjoint set covers is therefore

$$\sum_{\ell=1}^L \min(\ell, |D_\ell|) = \sum_{\ell=1}^L \ell = O(L^2) = O(q^{2/3}). \quad (22)$$

Therefore,  $T(\mathcal{M}_A, \mathbf{S}_1) = O(q^{2/3}) \forall$  online algorithms  $\mathcal{A}$ . ■

**Lemma V.3** *The number of disjoint set covers that can be formed from  $\mathbf{S}_1$  by the optimal offline algorithm is lower bounded by  $q/2$ , i.e.  $T(\mathcal{M}_{off}^*, \mathbf{S}_1) \geq q/2$ .*

*Proof:* Since  $\mathbf{S}_1$  was created adversarially based on the allocation made by the online algorithm, we will find the optimal offline solution by reallocating subsets from the online solution. Consider an allocation  $\mathcal{M}_A(\mathbf{S}_1)$ . The following proposition will clarify how the reallocation must be done.

**Proposition V.4** *The union of any two subsets  $S_x, S_y \in \mathbf{S}_{com}$  taken such that  $S_x \in P_s$  and  $S_y \in P_t$  and  $s \neq t$ , will contain all bottleneck elements  $u_{b(j)} \forall j$ .*

*Proof:* Note that by definition,  $S_x \cup S_y$  contains all bottleneck elements  $u_{b(j)} \forall j \neq \{s, t\}$ . In order to show that it also contains  $u_{b(s)}$  and  $u_{b(t)}$ , we will show that

$u_{b(s)} \in S_y$  and  $u_{b(t)} \in S_x$ . For some  $g \neq h$ , by definition,  $S_x = \{u_i : u_i^g = 1\}$ , and  $S_y = \{u_i : u_i^h = 1\}$ . Also,  $u_{b(s)} \in E_s(\mathcal{M})$ , and therefore by (17),  $u_{b(s)}$  is such that  $u_{b(s)}^k = 0 \forall k = \{r(s)+1, \dots, r(s+1)\}$  and  $u_{b(s)}^k = 1$  otherwise. Similarly,  $u_{b(t)}$  is such that  $u_{b(t)}^k = 0 \forall k = \{r(t)+1, \dots, r(t+1)\}$  and  $u_{b(t)}^k = 1$  otherwise. Now note that  $g \notin \{r(t)+1, \dots, r(t+1)\}$  and  $h \notin \{r(s)+1, \dots, r(s+1)\}$ , since partitions  $P_s$  and  $P_t$  are distinct. Therefore,  $u_{b(s)} \in S_y$  and  $u_{b(t)} \in S_x$ . ■

From Proposition V.4, it is clear that if an allocation  $\mathcal{M}_A(\mathbf{S}_{com})$  is such that subsets can be chosen pairwise from different partitions, the offline algorithm can produce at least  $|\mathbf{S}_{com}|/2 = q/2$  disjoint set covers. This is because all bottleneck elements corresponding to the allocation  $\mathcal{M}_A(\mathbf{S}_{com})$  are covered by the union of two subsets from different partitions, and  $\mathbf{S}_{inf}$  provides an infinite supply of non-bottleneck elements. However, for allocations  $\mathcal{M}_A(\mathbf{S}_{com})$  such that  $d_j > |\mathbf{S}_{com}|/2$  for some partition  $j$ , such a pairwise choice of subsets from different partitions is impossible. A proof for that case is provided in the Appendix. ■

From Lemmas V.2 and V.3,  $\mu_{S_1}(\mathcal{A}) = \Omega(q^{1/3}) \forall$  online algorithms  $\mathcal{A}$ . Since  $q = \log_2 n$ , Theorem V.1 is proved. ■

We presented Theorem V.1 to show that the grouping of bottleneck elements to form the subset sequence  $\mathbf{S}_a$  allows us to lower bound  $T(\mathcal{M}_A, \mathbf{S}_1)$ . We will build on these ideas in the presentation of Theorem V.5.

**Theorem V.5** *The competitive ratio of any online DSCP algorithm is lower bounded by  $\Omega((\ln n)^{1/2})$ , i.e.  $\mu(\mathcal{A}) = \Omega((\ln n)^{1/2}) \forall \mathcal{A}$ .*

*Proof:* We will use a technique similar to the proof of Theorem V.1. Instead of creating the subset sequence  $\mathbf{S}_1$ , however, we will now create the subset sequence  $\mathbf{S}_2$ , using a different adversarial sequence  $\mathbf{S}_b$ . The first  $q$  subsets in  $\mathbf{S}_2$  are still the sequence  $\mathbf{S}_{com}$ . Depending on the allocation  $\mathcal{M}_A(\mathbf{S}_{com})$ , we will use the subsets in sequence  $\mathbf{S}_a$  as defined in the proof of Theorem V.1 to create the adversarial subsets in  $\mathbf{S}_b$ . Let  $S_1^b = S_1^a \cup S_2^a \cup \dots \cup S_L^a$ ,  $S_2^b = S_2^a \cup S_3^a \cup \dots \cup S_L^a$ , and so on, with  $S_\ell^b = \cup_{r=\ell}^L S_r^a$ . Let  $\mathbf{S}_b = [S_j : S_j = S_j^b, 1 \leq j \leq L]$ . Again, let  $\mathbf{S}_{inf}$  be the infinite sequence of non-bottleneck elements for the allocation  $\mathcal{M}_A(\mathbf{S}_{com})$ . Let  $\mathbf{S}_2 = \mathbf{S}_{com} \wedge \mathbf{S}_b \wedge \mathbf{S}_{inf}$ .

Note:  $F_{min}(\mathbf{S}_2) = F_{b(j)}(\mathbf{S}_2)$  for any  $j$ ,  $F_{b(j)}(\mathbf{S}_2) = q \forall j$ .

**Lemma V.6** *The number of disjoint set covers that can be formed by any online algorithm from  $\mathbf{S}_2$  is  $O(q^{1/2})$ , i.e.  $T(\mathcal{M}_A, \mathbf{S}_2) = O(q^{1/2}) \forall$  online algorithms  $\mathcal{A}$ .*

*Proof:* Let  $A_\ell = \#\{P_j : j \in D_\ell, P_j \text{ forms a set cover.}\}$ , defined as before. Note that the total number of disjoint set covers that can be formed is upper-bounded by  $L$ , since there are only  $L$  subsets in  $\mathbf{S}_b$  that contain bottleneck elements, and each partition requires its bottleneck element in order to form a set cover. Note that only 1 partition  $P_j, j \in D_1$  can be made into a set cover, i.e.  $A_1 \leq 1$ , since only one subset  $S_1^b \in \mathbf{S}_b$  contains the bottleneck elements  $u_{b(j)} \forall j \in D_1$ . After that

set cover is formed, only 1 partition  $P_j$ ,  $j \in D_2$  can be made into a set cover by using the subset  $S_2^b \in \mathbf{S}_b$ , which is the only remaining subset in  $\mathbf{S}_b$  that contains the bottleneck elements  $u_{b(j)} \forall j \in D_2$ . Alternatively, an online algorithm could have allocated  $S_1^b$  and  $S_2^b$  to 2 partitions  $P_j$ ,  $j \in D_2$  and made them set covers. Mathematically,  $A_2 \leq 2 - A_1$ . This logic can be extended for all partitions  $P_j$ ,  $j \in D_\ell \forall 3 \leq \ell \leq L$ , to give rise to the constraint  $A_\ell \leq \min(|D_\ell|, \ell - \sum_{x=1}^{\ell-1} A_x)$ . The optimization problem over all possible online allocations  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$  is therefore the following:

$$\text{Maximize : } \sum_{\ell=1}^L A_\ell \quad (23)$$

$$\text{Subject to } \sum_{\ell=1}^L \ell \cdot |D_\ell| = q, \quad (24)$$

$$A_\ell \leq \min(|D_\ell|, \ell - \sum_{x=1}^{\ell-1} A_x) \forall 1 \leq \ell \leq L. \quad (25)$$

The objective function (23) represents the maximization of the number of disjoint set covers. Constraint (24) is because of the number of subsets available in  $\mathbf{S}_{com}$ , and constraint (25) arises out of the structure of subsets in  $\mathbf{S}_b$ , in the fashion explained above. The optimal solution to this problem occurs for  $|D_\ell| = 1 \forall 1 \leq \ell \leq L$ , for which  $A_\ell = 1 \forall 1 \leq \ell \leq L$ . It is possible to intuitively see the reason for this, since an allocation with constraint (24) will try to maximize the number of partitions that contain only 1 subset from  $\mathbf{S}_{com}$ , and then two subsets from  $\mathbf{S}_{com}$ , and so on, after which constraint (25) will ensure that a maximum of one partition containing  $\ell$  subsets from  $\mathbf{S}_{com}$  can form a set cover. For this solution, it is clear from constraint (24) that  $L = O(q^{1/2})$ , and is the number of disjoint set covers. ■

Note that  $T(\mathcal{M}_{off}^*, \mathbf{S}_2) = q/2$ . The proof is identical to that of Lemma V.3, since  $\mathbf{S}_2$  also contains  $\mathbf{S}_{com}$  and  $\mathbf{S}_{inf}$ . Like with Lemma V.3, there is a slight technicality, which is dealt with in the Appendix. So from Lemma V.6,  $\mu_{\mathbf{S}_2}(\mathcal{A}) = \Omega(q^{1/2}) \forall$  online algorithms  $\mathcal{A}$ , where  $q = \log_2 n$ . ■

## VI. SIMULATIONS

For lack of space, we present only one simulation result for the  $\text{POLYON}$  algorithm  $\mathcal{P}$ . We considered the online resource allocation problem, in which each server acquires the files uniformly randomly, each with probability  $p$ , from the universe of  $n$  files to form the subset sequence  $\mathbf{S}$ . Note that  $E[F_{min}(\mathbf{S})] = |\mathbf{S}|p = k$  (say). We then appended subsets to ensure that  $F_{min} = k$ . Simulations were carried out for three different values of  $F_{min}$ . The plot of the number of set covers returned by the  $\text{POLYON}$  algorithm  $T(\mathcal{M}_{\mathcal{P}}, \mathbf{S})$  versus  $n$  is provided in Figure 3. We can see that we get approximately  $F_{min}/\ln n$  set covers for all the three scenarios, thus validating our theoretical analysis in Section IV-C.

## VII. CONCLUDING REMARKS

We presented and analysed online algorithms for the DSCP in terms of their worst case competitive ratios. We found a

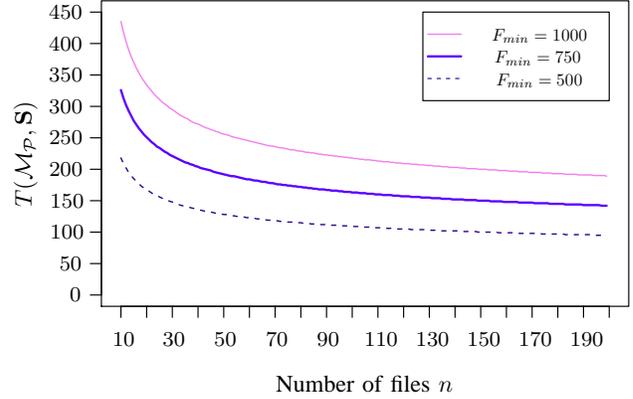


Fig. 3. Plot of the number of set covers obtained by the  $\text{POLYON}$  algorithm  $T(\mathcal{M}_{\mathcal{P}}, \mathbf{S})$  versus the number of files  $n$  for 3 values of  $F_{min}$ .

lower bound on the competitive ratio for all online algorithms, and presented an online algorithm that performed comparable to that lower bound for reasonable  $n$ . We conjecture that the tight lower bound on the competitive ratio is  $\ln n$ , but it will most likely require an entirely different approach to show. Analysis of the average case competitive ratio of online algorithms for the DSCP is still an open problem.

The results of this paper can be extended to produce online algorithms with competitive ratio  $\ln n$  for all problems that involve finding disjoint bases in a polymatroid [15]. Some examples of such problems are the domatic number problem [2], which has applications in the connectivity of WSNs [16], and in packing element-disjoint Steiner trees [17]. The lower bound on the competitive ratio also carries over to the general polymatroid problem of [15].

## REFERENCES

- [1] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005.
- [2] U. Feige, M. M. Halldórsson, G. Kortsarz, and A. Srinivasan, "Approximating the domatic number," *SIAM Journal on computing*, vol. 32, no. 1, pp. 172–195, 2002.
- [3] V. K. Bagaria, A. Pananjady, and R. Vaze, "Optimally approximating the lifetime of wireless sensor networks," *arXiv preprint arXiv:1307.5230*, 2013.
- [4] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in sensor networks," in *IEEE WCNC. 2004 Wireless Communications and Networking Conference, 2004.*, vol. 4. IEEE, 2004, pp. 2329–2334.
- [5] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *IEEE International Conference on Communications, 2001. ICC 2001.*, vol. 2. IEEE, 2001, pp. 472–476.
- [6] A. Pananjady, V. K. Bagaria, and R. Vaze, "Maximizing utility among selfish users in social groups," in *Communications (NCC), 2014 Twentieth National Conference on.* IEEE, 2014, pp. 1–6.
- [7] A. Subramanian, G. S. Kanth, and R. Vaze, "Offline and online incentive mechanism design for smart-phone crowd-sourcing," *arXiv preprint arXiv:1310.1746*, 2013.
- [8] X. Sheng, J. Tang, and W. Zhang, "Energy-efficient collaborative sensing with mobile phones," in *INFOCOM, 2012 Proceedings IEEE.* IEEE, 2012, pp. 1916–1924.
- [9] D. Lu, *Fundamentals of supply chain management.* Bookboon, 2011.

- [10] M. F. Tompkins, "Optimization techniques for task allocation and scheduling in distributed multi-agent operations," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.
- [11] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 1998.
- [12] B. Bollobás, D. Pritchard, T. Rothvoß, and A. Scott, "Cover-decomposition and polychromatic numbers," *SIAM Journal on Discrete Mathematics*, vol. 27, no. 1, pp. 240–256, 2013.
- [13] U. Feige, "A threshold of  $\ln n$  for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.
- [14] V. V. Vazirani, *Approximation algorithms*. Springer, 2001.
- [15] G. Călinescu, C. Chekuri, and J. Vondrák, "Disjoint bases in a polymatroid," *Random Structures & Algorithms*, vol. 35, no. 4, pp. 418–430, 2009.
- [16] T. Moscibroda and R. Wattenhofer, "Maximizing the lifetime of dominating sets," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 8–pp.
- [17] J. Cheriyan and M. R. Salavatipour, "Packing element-disjoint steiner trees," *ACM Transactions on Algorithms*, vol. 3, no. 4, p. 47, 2007.

#### APPENDIX

**Offline Solution for the special case:** When  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$  was such that  $d_j \geq q/2$  for some  $j = k$  (say), the offline solution could no longer be found by pairing subsets from different partitions. Note that only one such partition  $P_k$  can exist. For this case alone, after the allocation  $\mathcal{M}_{\mathcal{A}}(\mathbf{S}_{com})$ , we generate the adversarial subset sequence (either  $\mathbf{S}_a$  or  $\mathbf{S}_b$ ), differently. We consider partition  $P_k$  to consist of 2 partitions  $P_{k_1}$  and  $P_{k_2}$ , each of size less than  $q/2$ , and consider each to have its own bottleneck element. We then construct the adversarial sequence with this assumption of an additional bottleneck element. Now, all online algorithms are subject to all the constraints of (18) (and (23)), except that they can, in addition, make  $P_k$  a set cover. Therefore, the statements of Lemmas V.2 (and V.6) still hold. The offline algorithm, however, will produce  $q/2$  disjoint set covers, where 2 subsets can now be chosen pairwise from  $P_{k_1}$  and  $P_{k_2}$ . For this case too, Theorems V.1 and V.5 hold.