

# Distributed Set Cover Approximation: Primal-Dual with Optimal Locality

**Guy Even**


Tel-Aviv University, Israel  
guy@eng.tau.ac.il

**Mohsen Ghaffari**

ETH Zurich, Switzerland  
ghaffari@inf.ethz.ch

**Moti Medina**

Ben-Gurion University, Israel  
medinamo@bgu.ac.il

 <https://orcid.org/0000-0002-5572-3754>

---

## Abstract

This paper presents a deterministic distributed algorithm for computing an  $f(1+\varepsilon)$  approximation of the well-studied minimum *set cover* problem, for any constant  $\varepsilon > 0$ , in  $O(\log(f\Delta)/\log\log(f\Delta))$  rounds. Here,  $f$  denotes the maximum element frequency and  $\Delta$  denotes the cardinality of the largest set. This  $f(1+\varepsilon)$  approximation almost matches the  $f$ -approximation guarantee of standard centralized primal-dual algorithms, which is known to be essentially the best possible approximation for polynomial-time computations. The round complexity almost matches the  $\Omega(\log(\Delta)/\log\log(\Delta))$  lower bound of Kuhn, Moscibroda, Wattenhofer [JACM'16], which holds for even  $f = 2$  and for any  $\text{poly}(\log\Delta)$  approximation. Our algorithm also gives an alternative way to reproduce the time-optimal  $2(1+\varepsilon)$ -approximation of vertex cover, with round complexity  $O(\log\Delta/\log\log\Delta)$ , as presented by Bar-Yehuda, Censor-Hillel, and Schwartzman [PODC'17] for weighted vertex cover. Our method is quite different and it can be viewed as a locality-optimal way of performing primal-dual for the more general case of set cover. We note that the vertex cover algorithm of Bar-Yehuda et al. does not extend to set cover (when  $f \geq 3$ ).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Graph algorithms analysis, Mathematics of computing  $\rightarrow$  Graph algorithms, Theory of computation  $\rightarrow$  Distributed algorithms

**Keywords and phrases** Distributed Algorithms, Approximation Algorithms, Set Cover, Vertex Cover

**Digital Object Identifier** 10.4230/LIPIcs.DISC.2018.22

**Acknowledgements** Part of this work was done while the authors were visiting the Max Planck Institute for Informatics.

## 1 Introduction and Related Work

The *set cover* problem is one of the central problems in the study of approximation algorithms. For instance, the first chapter of the textbook of Williamson and Shmoys [27] is dedicated to illustrating “several of the central techniques of the book applied to a single problem, the set cover problem.” In this paper, we present the first time-optimal distributed approximation algorithm for the set cover problem, with an approximation guarantee that essentially matches the best known centralized approximation. Let us elaborate on this by first recalling the problem statement and centralized approximation bounds, as well as the distributed model of computation in the study of this problem.



© Guy Even, Mohsen Ghaffari, and Moti Medina;  
licensed under Creative Commons License CC-BY

32nd International Symposium on Distributed Computing (DISC 2018).

Editors: Ulrich Schmid and Josef Widder; Article No. 22; pp. 22:1–22:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1.1 Background

**Set Cover.** We are given a ground set of elements  $U$  and some sets  $S_1, S_2, \dots, S_k \subseteq U$ . The objective is to find a minimum-cardinality collection of the sets that covers all the elements, i.e., a collection  $I \subseteq \{1, \dots, k\}$  that minimizes  $|I|$  subject to  $\cup_{i \in I} S_i = U$ .

**Known Centralized Approximations and Inapproximability Bounds.** For each element  $u \in U$ , we use  $f_u = |\{i \mid u \in S_i\}|$  to denote the frequency of this element, i.e., the number of sets that contain  $u$ . We also use  $f$  to denote the maximum frequency among all elements, i.e.  $f = \max_{u \in U} f_u$ . A standard approximation guarantee for the set cover problem is an  $f$ -approximation, see e.g. [4, Theorem 2] or [27, Theorem 1.6] or [26, Theorem 15.2]. Moreover, this approximation is known to be nearly the best possible for polynomial-time central algorithms: For the special case  $f = 2$  (when the problem is better known as the *vertex cover* problem), Dinur and Safra [9] proved NP-hardness of 1.36 approximation, improving on a  $7/6 - \epsilon$  hardness by Hastad [14]. For general  $f$ , the inapproximability has been improved in a sequence of papers: Trevisan gave an  $\Omega(f^{1/19})$  bound [25]; Holmerin gave an  $\Omega(f^{1-\epsilon})$  bound [15]; Dinur, Guruswami, and Khot improved that to  $f - 3 - \epsilon$ ; which was then improved by Dinur, Guruswami, Khot, and Regev to  $f - 1 - \epsilon$  [8]. Furthermore, assuming the Unique Games Conjecture, Khot and Regev proved an inapproximability of  $f - \epsilon$  [16]. We remark that another approximation bound for the set cover is  $\ln |U|$  – see [27, Theorem 1.11]. This bound is of interest when the frequency of appearances of the elements in different sets is large. Moreover, this bound is also known to be the nearly the best possible in the worst-case: A series of works by Lund and Yannakakis [21], Feige [10], and Moshkovitz [22] showed that it is NP-hard to always approximate set cover to within  $(1 - \epsilon) \ln |U|$ , for any constant  $\epsilon > 0$ . We note that although the standard way of formulating the upper bound is  $\ln |U|$ , the actual bound can be written more precisely as  $\ln \Delta$  where  $\Delta$  denotes the cardinality of the largest set.

**Distributed Computation Model.** We consider the CONGEST [23] model, which is the standard synchronous message passing model in distributed computing. In this model, the network is abstracted as a simple graph  $G = (V, E)$  where  $n = |V|$ . There is one processor on each node of the network, which initially does not know the topology of the network. These processors can communicate in synchronous rounds where per round each processor/node can send one  $O(\log n)$  bit message to each of its neighbors.

**Distributed Formulation of Set Cover.** The standard distributed formulation of the set cover problem (see, e.g., [19]) is that we have one processor for each element in the ground set  $U$ , and also one processor for each of the sets  $S_1, S_2, \dots, S_k \subseteq U$ . The network is the natural corresponding (bipartite) graph where each element-processor is connected to the set-processors whose set contains this element. Communications on this network follow the CONGEST model of synchronous message passing, as explained above.

The above is a natural formulation. As prototypical examples, it captures the following settings: cases where we want to select as few as possible of the servers so that they can serve all of the clients, when each element can be served only by certain servers; and cases where we want to select as few as possible of monitoring agents who can control all workers, where each worker can be controlled only by certain monitoring agents<sup>1</sup>.

---

<sup>1</sup> Of course, in the practical version of each of these problems, there might be many more constraints or optimization objectives. However, that goes beyond the objective of our paper, which is to characterize the complexity of a basic and fundamental problem in distributed approximation algorithms.

## 1.2 Our Result

We present a deterministic distributed algorithm that almost matches the  $f$ -approximation mentioned above, up to a  $(1 + \varepsilon)$  factor for any arbitrary small constant  $\varepsilon > 0$ , in a time-complexity that is provably optimal:

► **Theorem 1.** *There is a deterministic distributed algorithm in the CONGEST model that computes an  $f(1 + \varepsilon)$  approximation of minimum set-cover, in  $O\left(\frac{\log(f\Delta)}{\varepsilon \log \log(f\Delta)}\right)$  rounds, in any set-system of frequency  $f$  and maximum set size  $\Delta$ , and for any  $0 < \varepsilon < 1$ . Moreover, the algorithm operates on an anonymous network and uses messages of length  $O(\varepsilon^{-1} \cdot \log(f \cdot \Delta))$ .*

The matching lower bound is due to a celebrated work of Kuhn, Moscibroda, and Wattenhofer [19]: they show that even the simple case of  $f = 2$ , where the set cover problem boils down to vertex cover, has a lower bound of  $\Omega\left(\frac{\log \Delta}{\log \log \Delta}\right)$  rounds, for any approximation up to  $\text{poly}(\log \Delta)$ . Moreover, for all cases of interest for  $f$ -approximation – i.e., when  $f$  is smaller than the other known approximation bound  $\ln \Delta$  –, the  $O\left(\frac{\log(f\Delta)}{\log \log(f\Delta)}\right)$  round complexity of the above algorithm asymptotically matches the  $\Omega\left(\frac{\log \Delta}{\log \log \Delta}\right)$  lower bound.

We note that coming up with a deterministic distributed algorithm that achieves  $\text{poly} \log \Delta$  (or even  $\text{poly} \log n$ ) approximation for set cover, even with unbounded size messages, with  $\text{poly} \log n$  number of rounds, where  $n$  is the number of processors in the network, would be a major breakthrough: as shown recently in [12, Theorem 7.5], it would imply that any randomized distributed algorithm with  $\text{poly} \log n$  number of rounds for any locally checkable problem can be derandomized and solved in  $\text{poly} \log n$  number of rounds deterministically. This includes computing a Maximal Independent Set in  $\text{poly} \log n$  number of rounds, which is an open question by Linial since the 80's [20].

## 1.3 The Main Related Work and Comparison of Techniques

**Sequential Primal-Dual.** A standard centralized approximation algorithm that gives an  $f$ -approximation for set cover is the one based on the primal-dual schema. See, e.g., Bar-Yehuda and Even [4] or Vazirani's textbook [26, Section 15.2] for a comprehensive description. Summarized, this schema works roughly as follows: there is a variable  $y_u \in [0, 1]$  for each element  $u \in U$ ; these are known as dual variables. Until all elements are covered, we iteratively pick an uncovered element, say  $u$ , and we raise its variable  $y_u$  until for one of the sets containing  $u$ , say  $S_i$ , we have  $\sum_{u' \in S_i} y_{u'} = 1$ . We call such a set tight (because its constraint in the primal linear program is tight). Then we add this tight set to the set cover to be outputted at the end, and we consider all of its elements covered. As shown in [4, Theorem 2], and [26, Theorem 15.3], this method gives an  $f$ -approximation.

**Standard Distributed Primal-Dual.** The above method is clearly sequential. However, one can easily adapt it to the distributed setting<sup>2</sup>, when we relax the approximation factor to  $f(1 + \varepsilon)$  for any arbitrarily small constant  $\varepsilon > 0$ . Initially, set  $y_u = 1/\Delta$  for each element  $u \in U$ . Then, in each iteration, we do as follows: (1) for each set  $S_i$  that has  $\sum_{u' \in S_i} y_{u'} \geq 1 - \varepsilon/2$ , add this set  $S_i$  to the output set cover (all at the same time) and consider all of its elements covered. Then, for each uncovered element  $u$ , set  $y_u \leftarrow y_u \cdot \frac{1}{1 - \varepsilon/2}$ . The method terminates in  $O(\log \Delta / \varepsilon)$  rounds and outputs an  $f(1 + \varepsilon)$  approximation.

<sup>2</sup> In fact, this adaptation is so simple and well-known that we are not sure what is the reference for it (or its first appearance). The analysis follows directly from [26, Proposition 15.1].

As a side comment, we add that Kuhn et al. [18,19] give a general algorithm for obtaining a  $(1 + \varepsilon)$  approximation of fractional packing linear programs, which can then be turned into an integral solution for  $f(1 + \varepsilon)$  approximation of set cover via a simple deterministic rounding. However, the resulting algorithm would be slower than the above.

**Sped-up Distributed Solution, via the Local-Ratio Method.** In an elegant recent work, Bar-Yehuda, Censor-Hillel, and Schwartzman [3] presented an improved algorithm for the special case of  $f = 2$  (i.e., vertex cover), based on the local-ratio method [5] which itself is closely related to the primal-dual scheme [6]. Their algorithm improves the round complexity for  $f(1 + \varepsilon) = 2(1 + \varepsilon)$  approximation of vertex cover to the optimal bound of  $O(\log \Delta / (\varepsilon \log \log \Delta))$ . Their algorithm also works for the weighted variant of the vertex cover problem. This round complexity matches the lower bound of Kuhn, Moscibroda, and Wattenhofer [19]. However, the algorithm of [3] seems especially crafted for the case of  $f = 2$  and it does not generalize<sup>3</sup> to even  $f = 3$ . In a very rough sense, the limitation is as follows: the method works by dividing the leftover space in dual constraints (i.e.,  $1 - \sum_{u' \in S_i} y_{u'}$ ) into two parts, a vault and a bank. The vault is used to initiate requests for increases in the dual variables (i.e.,  $y_{u'}$ ) and the bank is used to securely accept these dual variable increases, while making sure that  $\sum_{u' \in S_i} y_{u'} \leq 1$ . When trying to extend this to  $f = 3$  or higher, it is not clear how to make all the sets containing one element agree consistently with the amount of the raise in the dual variable, while respecting their own individual  $\sum_{u' \in S_i} y_{u'} \leq 1$  constraints, and without slowing down the process too much.

**Our Method, in a Nutshell.** We also follow the primal-dual schema. But our method can be viewed as an improved and more general way of performing primal-dual distributedly, with optimal locality (i.e., round complexity) for set cover. In a very rough sense, it is based on a natural dynamic process that, over time, flexibly adjusts the amount of increase per each dual variable, while (1) not violating any of the constraints, (2) maintaining a large step of increase for most variables at most times. We are hopeful that dynamics of the same style may lead to improvements for many other optimization problems.

**A conceptual contribution, in the context of randomized Maximal Independent Set Algorithms [2, 11].** Besides the improvement in the round complexity of the set-cover problem, we think of our solution as shedding some light on some other known prior work [2,11]. The dynamic process that we use for adjusting the increase steps in dual variables is closely related to the randomized maximal independent set algorithm of Ghaffari [11]. We note that a parameter-optimized version of the latter was used by Bar-Yehuda, Censor-Hillel, Ghaffari, and Schwartzman [2] to obtain a  $2(1 + \varepsilon)$  approximation of maximum matching in  $O(\log \Delta / \log \log \Delta)$  rounds. However, the place where we use the general dynamic process appears quite different than those of [2,11]. While in those previous papers the dynamic process was set up to adjust the probability of trying to join the MIS (or the nearly maximal matching), in our current paper, the dynamic process is used in a fully deterministic way and it governs the adjustments in the increase step of dual variables. In hindsight, this suggests (in an informal way) that one can view the probabilities in Ghaffari’s MIS algorithm [11] as fractional solutions to some linear program. The dynamic process tries to adjust these probabilities towards the “sweet spot” where per round many nodes get hit (by either joining

---

<sup>3</sup> We have also double checked this with Gregory Schwartzman, through personal communication.

MIS or having a neighbor join MIS). This is reminiscent of the standard randomized rounding method in design of approximation algorithms, where one first finds a good fractional solution to a suitable linear program formulation, and then performs a randomized rounding to turn these fractional solutions to integral; see e.g., [27, Section 1.7]. The difference is that the algorithm of [11] does not wait for these fractional variables to reach the sweet spot and only then do the rounding (i.e., deciding probabilistically for various elements). It instead performs a certain “iterative rounding” where even the interim fractional values are used for attempts of forming a good integral solution (an independent set that is adjacent to a large set of vertices).

## 1.4 Other related work

In this section we survey other related work. We start with results where  $f = 2$ , i.e., vertex cover. Recently, Ben-Basat, Even, Kawarabayashi, and Schwartzman [7] presented a 2-approximation algorithm for minimum weighted vertex cover in CONGEST with round complexity of  $O\left(\frac{\log n \log \Delta}{(\log \log \Delta)^2}\right)$ . Their approach generalizes the  $(2+\varepsilon)$ -approximation algorithm of [3] and improves the dependency on  $\varepsilon^{-1}$  to logarithmic. For a detailed overview of work on vertex cover we refer the reader to [1, 3].

We now turn to results for general  $f$ : Koufogiannakis and Young [17] presented a distributed algorithm for weighted set cover in the LOCAL model. Their algorithm achieves an approximation ratio of  $f$  in  $O(\log^2 m)$  rounds w.h.p, where  $m$  is the number of elements. Kuhn et al. [18, 19] studied covering and packing linear programs in the LOCAL model and obtained a  $(1 + \varepsilon)$ -approximation algorithm in  $O(\varepsilon^{-1} \log n)$  rounds w.h.p., where  $n$  is the number of primal and dual variables. Ghaffari, Kuhn, and Maus [13] presented a randomized distributed approximation scheme (i.e.,  $(1 + \varepsilon)$ -approximation) for arbitrary distributed covering and packing integer linear programs in the LOCAL model with round complexity  $O(\text{poly log}(n/\varepsilon))$  w.h.p., where  $n$  is the number of primal and dual variables. For more results in the LOCAL model we refer the reader to the survey by [24].

## 2 Problem Definition and Model of Computation

In this section we introduce the problem of *vertex cover in hypergraphs* (VCH). Designing a distributed CONGEST algorithm for VCH directly translates to an algorithm for set cover.

### 2.1 Preliminaries

A hypergraph  $H$  is a pair  $(V, E)$  where  $V$  denotes the set of vertices and  $E \subseteq 2^V$ . Every hyperedge  $e \in E$  is a nonempty subset of vertices. The *maximum degree* of the graph  $G$  is denoted by  $\Delta$ , and defined by  $\Delta \triangleq \max_v |\{e \in E \mid v \in e\}|$ . The *rank* of  $H$  is denoted by  $f$ , and defined by  $f \triangleq \max_{e \in E} |e|$ .

### 2.2 Vertex Cover in Hypergraphs (VCH)

A subset  $C \subset V$  is a *vertex cover* in  $H = (V, E)$  if  $C \cap e \neq \emptyset$ , for every hyperedge  $e \in E$ . The minimum cardinality vertex cover problem in hypergraphs is defined as follows.

- Problem:** Minimum Cardinality Vertex Cover in Hypergraphs (VCH)  
**Instance:** A hypergraph  $H = (V, E)$ .  
**Solution:** A vertex cover  $C$ .  
**Objective:** Minimize the cardinality of the cover  $C$ .

We denote the cardinality of an optimal vertex cover by  $\text{opt}$ .

Note that the VCH problem translates to set cover as follows:

- (i) Each element in the ground set  $U$  is an hyperedge in the VCH formulation, and every set in the set cover problem is a vertex in VCH.
- (ii) Indeed, the maximum rank of a hyperedge in VCH translates to the maximum frequency of an element in set cover and that the maximum degree in VCH translates to the maximum cardinality of a set in the set cover problem.

Also note that VCH is identical to the *hitting set* problem in set systems.

### 2.3 The Network

The network that corresponds to a hypergraph  $H = (V, E)$  is a bipartite graph  $N = (V \cup E, L)$ , where there is a processor for every vertex  $v$  and a processor for every hyperedge  $e$ . The set of links  $L$  consists of all the pairs  $(v, e) \in V \times E$  such that  $v \in e$ . Our algorithm does not require distinct IDs, namely, the network is anonymous. Moreover, the algorithm does not even rely on numbering of ports.

## 3 Algorithm Description

The algorithm is a primal-dual algorithm that updates the primal and dual variables in iterations. Each hyperedge has two variables: an auxiliary variable  $x(e)$  and an edge packing variable  $y(e)$ . We denote the value of the variables in iteration  $t$  by  $x_t(e)$  and  $y_t(e)$ .

- The dual variable  $y(e)$  is a nonnegative edge packing variable. By an edge packing variable we mean that, for every vertex  $v$  and in every iteration  $t$ ,  $\sum_{e \ni v} y_t(e) \leq 1$ . The variable  $y(e)$  is monotone non-decreasing over time.
- The auxiliary variable  $x(e)$  is initialized to  $x_0(t) = 1/K$ . The dynamics of  $x(e)$  allow to either divide or multiply  $x(e)$  by  $K$  in each iteration as long as it is bounded by  $1/K$ . Here,  $K \geq 2$  is a free parameter that is to be fixed later. The role of the auxiliary variables  $x(e)$  is to control the increase of the dual edge packing variables  $y(e)$ .

► **Definition 2.** A vertex  $v$  is  $\varepsilon$ -tight if  $\sum_{e \ni v} y_t(e) \geq 1 - \varepsilon$ .

Following the primal-dual approximation scheme, a vertex  $v$  joins the vertex cover as soon as it becomes  $\varepsilon$ -tight. The algorithm terminates when the set of  $\varepsilon$ -tight vertices covers all the hyperedges.

The following terminology is used in the algorithm and its analysis.

1. The set of edges that contain a vertex  $v$  is denoted by  $E(v)$ .
2. For a subset of edges  $A \subseteq E$ , let  $x_t[A] \triangleq \sum_{e \in A} x_t(e)$ .
3. For a vertex  $v$  let  $y_t[v] \triangleq \sum_{e \ni v} y_t(e)$ .

► **Definition 3.** The *effective degree* of an edge  $e$  is defined by

$$d_t(e) = \sum_{v \in e} x_t[E(v)].$$

Note that  $d_t(e) = \sum_{e': e' \cap e \neq \emptyset} |e \cap e'| \cdot x_t(e')$ . The “natural” definition of effective degree  $d_t(e) = \sum_{e': e' \cap e \neq \emptyset} x_t(e')$  works as well. However, it is not clear how to implement the natural definition in CONGEST.

► **Definition 4.** An edge  $e$  is *light* (in iteration  $t$ ) if  $d_t(e) < K$ . If  $d_t(e) \geq K$ , we say that the edge is *heavy*.

### 3.1 The Algorithm (ALG)

---

**Input:** Hypergraph  $H = (V, E)$  and  $0 < \varepsilon < 1$ .

**Output:** A vertex cover  $C \subseteq V$ .

**Initialization:** For every  $e \in E$ ,  $x_0(e) \leftarrow 1/K$ ,  $y_0(e) \leftarrow 0$ ,  $C \leftarrow \emptyset$ ,  $E' \leftarrow E$ .

**Invariants:** (1) The variables  $y(e)$  constitute a feasible edge packing. (2)  $C$  equals the set of  $\varepsilon$ -tight vertices.

**ALG:** The algorithm works by iterations until  $E' = \emptyset$ . Iteration  $t$  works as follows:

1. For each light edge  $e \in E'$ , set  $y_{t+1}(e) \leftarrow y_t(e) + x_t(e) \cdot \varepsilon/K$ .
2. Add all the new  $\varepsilon$ -tight vertices to  $C$ .
3. Remove covered edges:  $E' \leftarrow E' \setminus \{e \in E : e \cap C \neq \emptyset\}$ .
4. Update the auxiliary variables of edge  $e \in E'$ , as follows:

$$x_{t+1}(e) = \begin{cases} x_t(e)/K, & \text{if } d_t(e) \geq K \quad //\text{heavy edge rule} \\ \min\{K \cdot x_t(e), 1/K\}, & \text{if } d_t(e) < K \quad //\text{light edge rule.} \end{cases}$$


---

The following simple observation bounds  $d_t(e)$  for every edge  $e$  and iteration  $t$ .

► **Observation 5.** For all  $e \in E$ , and for all iterations  $t$  it holds that

$$d_t(e) \leq \frac{f\Delta}{K}, \text{ and} \tag{1}$$

$$\frac{d_t(e)}{K} \leq d_{t+1}(e) \leq K \cdot d_t(e). \tag{2}$$

## 4 Analysis

The analysis consists of two parts. In the first part, we prove that if the algorithm terminates, then it finds a vertex cover that is a  $(1 + O(\varepsilon)) \cdot f$ -approximation of a minimum cardinality vertex cover. In the second part, we prove an upper bound on the number iterations of the algorithm. Every iteration requires a constant number of communication rounds, and hence the bound on the number of communication rounds follows.

### 4.1 Approximation Ratio

► **Claim 6.** Throughout the algorithm, the variables  $y_t(e)$  constitute a feasible edge packing.

**Proof.** Fix a vertex  $v$ . The proof is by induction on  $t$ . Initially,  $y_0(e) = 0$ , hence,  $y_0$  is clearly a feasible edge packing. Assume that  $\{y_t(e)\}_e$  is an edge packing (i.e.,  $y_t[v] \leq 1$ , for every  $v$ ), we now prove that  $\{y_{t+1}(e)\}_e$  is an edge packing. If  $y_{t+1}[v] > y_t[v]$ , then  $v$  is not  $\varepsilon$ -tight in the end of iteration  $t$ , and thus  $y_t[v] < 1 - \varepsilon$ .

Let  $e^*$  denote an arbitrary edge such that  $v \in e^*$  and  $y_{t+1}(e^*) > y_t(e^*)$ . In particular, this implies that  $e^*$  is light (see Step 1 of the algorithm), i.e.,  $d_t(e^*) < K$ .

We conclude that

$$\begin{aligned} y_{t+1}[v] - y_t[v] &= \frac{\varepsilon}{K} \cdot \sum_{e \ni v, e \text{ light}} x_t(e) \\ &\leq \frac{\varepsilon}{K} \cdot d_t(e^*) < \varepsilon, \end{aligned}$$

where the second inequality holds because every light edge  $e$  that contains  $v$  contributes at least  $x_t(e)$  to  $d_t(e^*)$ . Since  $y_t[v] < 1 - \varepsilon$ , the claim follows. ◀

► **Claim 7.** *At the end of every iteration  $t$  of the algorithm, the cardinality of the set of  $\varepsilon$ -tight vertices is at most  $\frac{f}{1-\varepsilon} \cdot \text{opt}$ .*

**Proof.**

$$\begin{aligned} |\{v \mid y_t[v] \geq 1 - \varepsilon\}| &\leq \sum_{v \mid y_t[v] \geq 1 - \varepsilon} \frac{1}{1 - \varepsilon} \sum_{e \ni v} y_t(e) \\ &\leq \frac{1}{1 - \varepsilon} \sum_{e \in E} \sum_{v \in e} y_t(e) \\ &\leq \frac{f}{1 - \varepsilon} \sum_{e \in E} y_t(e) \leq \frac{f}{1 - \varepsilon} \cdot \text{opt}, \end{aligned}$$

where the first inequality follows from the definition of  $\varepsilon$ -tight vertices, the third inequality follows from the fact that  $|e| \leq f$ , and the fourth inequality follows from weak. ◀

Note that throughout the algorithm,  $C$  is the set of  $\varepsilon$ -tight vertices. Upon termination,  $E' = \emptyset$ , and thus  $C$  is a vertex cover. Hence, by Claim 7, it follows that when the algorithm terminates, the set  $C$  is vertex cover and its cardinality is  $(1 + O(\varepsilon)) \cdot f \cdot \text{opt}$ .

## 4.2 Bounding the Number of Rounds

In this section we prove the following theorem. Recall that the algorithm terminates when  $E' = \emptyset$ .

► **Theorem 8.** *Let  $K \geq 2$ , the algorithm terminates after  $O\left(\frac{\log(f\Delta)}{\log K} + \frac{K^3}{\varepsilon}\right)$  iterations.*

### 4.2.1 Golden Iterations

Let  $Light_t \triangleq \{e \in E \mid d_t(e) < K\}$ , and  $Heavy_t \triangleq \{e \in E \mid d_t(e) \geq K\}$ .

► **Definition 9.** An iteration  $t$  is a Type-1 iteration with respect to hyperedge  $e$  if it satisfies:

$$d_t(e) < K \quad \text{and} \quad x_t(e) = 1/K.$$

► **Definition 10.** An iteration  $t$  is a Type-2 iteration with respect to hyperedge  $e$  if it satisfies:

$$d_t(e) \geq 1 \quad \text{and} \quad \sum_{v \in e} x_t[E(v) \cap Light_t] \geq \frac{1}{2K^2} \cdot d_t(e).$$

An iteration  $t$  is a *golden* iteration with respect to  $e$  if it is a Type-1 or Type-2 iteration with respect to  $e$ .

Our goal is to bound the number of iterations until termination. Throughout the analysis, fix a hyperedge  $e$ , and assume that it is not covered after  $T$  iterations (i.e.,  $e \cap C = \emptyset$ ).

► **Definition 11.** For a fixed hyperedge  $e$  not covered after  $T$  iterations, define the following



subsets of iterations.

$$\begin{aligned}
G_1 &\triangleq \left\{ t \in [T] \mid d_t(e) < K \text{ and } x_t(e) = \frac{1}{K} \right\} && \text{Type-1} \\
G_2 &\triangleq \left\{ t \in [T] \mid d_t(e) \geq 1 \text{ and } \sum_{v \in e} x_t[E(v) \cap \text{Light}_t] \geq \frac{1}{2K^2} \cdot d_t(e) \right\} && \text{Type-2} \\
H &\triangleq \{ t \in [T] \mid d_t(e) \geq K \} && \text{Heavy} \\
L &\triangleq \{ t \in [T] \mid d_t(e) < K \} && \text{Light} \\
U &\triangleq \{ t \in [T] \mid x_{t+1}(e) = K \cdot x_t(e) \} && \text{Up} \\
S &\triangleq \left\{ t \in [T] \mid x_t(e) = \frac{1}{K} \right\} && \text{Saturated}
\end{aligned}$$

### 4.2.2 Useful Claims

We denote the cardinalities of these subsets using lower case letters, e.g.,  $g_1 = |G_1|$ ,  $h = |H|$ , etc.

► **Claim 12.**  $H = \{ t \in [T] \mid x_{t+1}(e) = x_t(e)/K \}$  and  $u \leq h$ .

**Proof.** The first part follows from Line 4 of the algorithm. The variable  $x_0(e)$  is initialized to  $1/K$ , never exceeds  $1/K$ , is divided by  $K$  in iterations in  $H$ , and multiplied by  $K$  in iterations in  $U$ . Hence,  $1/K \geq x_T(e) = x_0(e) \cdot K^{u-h}$ , and  $u \leq h$ , as required. ◀

► **Claim 13.**  $T \leq 3h + g_1$ .

**Proof.** Note that  $\ell \leq u + s$ . Indeed, If  $t \in L$ , then either  $t \in U$  or  $x_t(e)$  could not be multiplied by  $K$ , hence  $t \in S$ . Since  $T = h + \ell$ , by Claim 12 we conclude that  $T \leq h + u + s \leq 2h + s$ .

To conclude the proof, we show that  $s \leq g_1 + h$ . This holds simply because,  $S \setminus G_1 \subseteq H$ . ◀

► **Claim 14.**  $\max\{g_1, g_2\} \leq \frac{2K^3}{\varepsilon}$ .

**Proof.** For each Type-1 iteration  $t \in [T]$ , the update of  $y_t(e)$  due to Steps 1 and 4 is

$$y_{t+1}(e) = y_t(e) + x_t(e) \cdot \frac{\varepsilon}{K} = y_t(e) + \frac{\varepsilon}{K} \cdot \frac{1}{K}.$$

Hence  $y_{T+1}(e) \geq g_1 \cdot \frac{\varepsilon}{K^2}$ . Claim 6 implies that the  $y_{T+1}(e')$  variables constitute a feasible edge packing, i.e.,  $y_{T+1}[v] \leq 1$  for every  $v$ , then  $y_{T+1}(e) \leq 1$ , and hence  $g_1 \leq K^2/\varepsilon$ , as required.

We bound  $g_2$  as follows. Consider a Type-2 iteration  $t \in [T]$ . Then,

$$\begin{aligned}
\sum_{v \in e} y_{t+1}[v] - y_t[v] &= \sum_{v \in e} \left( \sum_{e' \ni v, e' \in \text{Light}_t} \frac{\varepsilon}{K} \cdot x_t(e') \right) \\
&= \frac{\varepsilon}{K} \cdot \sum_{v \in e} x_t[E(v) \cap \text{Light}_t] \\
&\geq \frac{\varepsilon}{K} \cdot \frac{1}{2K^2} \cdot d_t(e) \geq \frac{\varepsilon}{2K^3},
\end{aligned}$$

where the last two inequalities follow from the definition of a Type-2 golden round (see Definition 10). This implies that  $g_2(e) \leq 2K^3/\varepsilon$ , as required. ◀

## 22:10 Distributed Set Cover Approximation: Primal-Dual with Optimal Locality

► **Claim 15.** *If  $d_t(e) \geq 1$ , and  $t \notin G_2$ , then*

$$d_{t+1}(e) < \frac{3}{2K} \cdot d_t(e). \quad (3)$$

**Proof.** If  $d_t(e) \geq 1$  and  $t$  is a not Type-2 iteration with respect to  $e$ , then  $\sum_{v \in e} x_t[E(v) \cap \text{Light}_t] < \frac{1}{2K^2} \cdot d_t(e)$ . Since  $x_{t+1}(e) \leq K \cdot x_t(e)$  if  $e \in \text{Light}_t$ , and  $x_{t+1}(e) = x_t(e)/K$  if  $e \in \text{Heavy}_t$ , we conclude that

$$\begin{aligned} d_{t+1}(e) &\leq \frac{1}{K} \cdot \sum_{v \in e} x_t[E(v) \cap \text{Heavy}_t] + K \cdot \sum_{v \in e} x_t[E(v) \cap \text{Light}_t] \\ &\leq \frac{1}{K} \cdot d_t(e) + K \cdot \frac{1}{2K^2} \cdot d_t(e). \end{aligned}$$

The claim follows. ◀

► **Claim 16.**  $h \leq \frac{\log(f\Delta/k^2)}{\log(\frac{2K}{3})} + 4g_2$ .

**Proof.** Partition  $H$  into maximally contiguous (disjoint) intervals  $H = H_1 \cup \dots \cup H_z$ . Denote the endpoints of  $H_i$  by  $[t_i, b_i]$ . Define

$$a_i \triangleq \begin{cases} t_1 & \text{if } i = 1 \\ \min\{t < t_i \mid \forall r \in [t, t_i - 1] : 1 \leq d_r(e) < K\} & \text{if } z \geq i > 1. \end{cases}$$

Note that, if  $i > 1$ , then the set  $\{t < t_i \mid \forall r \in [t, t_i - 1] : 1 \leq d_r(e) < K\}$  is not empty. Indeed,  $t_i - 1$  belongs to this set as  $d_{t_i}(e) \geq K$  and  $1 \leq d_{t_i-1}(e) < K$ .

Let  $I_i \triangleq [a_i, b_i]$ . Note that the intervals  $\{I_i\}_{i=1}^z$  are pairwise disjoint.

Since  $x_0(e) = \frac{1}{K}$  for every  $e \in E$  and since  $\sum_{v \in e} |E(v)| \leq f \cdot \Delta$  we get that  $d_{a_i}(e) \leq f\Delta/K$ . Hence, by the definition of  $a_i$ , we have

$$d_{a_i}(e) \leq \begin{cases} f\Delta/K & \text{if } i = 1 \\ K & \text{if } i > 1 \end{cases}$$

By the definition of  $b_i$  we have

$$d_{b_i}(e) \geq K.$$

Now,

$$\begin{aligned} d_{b_i}(e) &\leq d_{a_i}(e) \cdot \left(\frac{3}{2K}\right)^{|I_i \cap \bar{G}_2|} \cdot K^{|I_i \cap G_2|} \\ &\leq d_{a_i}(e) \cdot \left(\frac{2K}{3}\right)^{3 \cdot |I_i \cap G_2| - |I_i \cap \bar{G}_2|}. \end{aligned}$$

The first inequality follows from Claims 5 and 15. The second inequality follows from  $K < (2K/3)^3$ , as  $K \geq 2$ . Hence,

$$|I_i \cap \bar{G}_2| \leq 3 \cdot |I_i \cap G_2| + \frac{\log\left(\frac{d_{a_i}(e)}{d_{b_i}(e)}\right)}{\log(2K/3)}$$

Since

$$\frac{d_{a_i}(e)}{d_{b_i}(e)} \leq \begin{cases} f\Delta/K^2 & \text{if } i = 1 \\ 1 & \text{if } i > 1 \end{cases},$$

by summing up over all the disjoint intervals we obtain

$$\sum_{i=1}^z |I_i \cap \overline{G}_2| \leq 3g_2 + \frac{\log(f\Delta/K^2)}{\log(2K/3)}.$$

Since  $h \leq g_2 + \sum_{i=1}^z |I_i \cap \overline{G}_2|$ , the claim follows.  $\blacktriangleleft$

### 4.2.3 Proof of Theorem 8

**Proof of Theorem 8.** Suppose that the algorithm does not terminate after  $T$  rounds because the edge  $e$  remains uncovered. Claims 13, 16, and 14 and the fact that  $K \geq 2$  and  $0 < \varepsilon < 1$  imply that

$$\begin{aligned} T &\leq 3h + g_1 \\ &\leq 3 \left( \frac{\log(f\Delta/K^2)}{\log(\frac{2K}{3})} + 4g_2 \right) + g_1 \\ &= 3 \cdot \frac{\log(f\Delta/K^2)}{\log(\frac{2K}{3})} + 12g_2 + g_1 \\ &\leq 3 \cdot \frac{\log(f\Delta/K^2)}{\log(\frac{2K}{3})} + \frac{26K^3}{\varepsilon}. \end{aligned}$$

Since  $\log(2K/3) = \Omega(\log K)$ , the theorem follows.  $\blacktriangleleft$

## 5 Distributed Implementation

In this section we present a distributed implementation of the algorithm. To simplify the presentation, we present the sequence of computations and messages performed by the vertices and the edges in a combined fashion.

**States.** Every vertex  $v$  has three states: “active” - means that  $v$  did not decide yet if it is in the cover or not, “in cover” - means that  $v$  decided to join the cover, “not in cover” - means that  $v$  decided that it will not join the cover. Every edge  $e$  has two states: “uncovered” and “covered”.

### Distributed Implementation.

1. Every edge processor  $e$  maintains the variables  $x(e)$  and  $y(e)$ . These variables are initialized as follows:  $x(e) \leftarrow 1/K$  and  $y(e) \leftarrow 0$ . The initial state of  $e$  is “uncovered”.
2. Every vertex processor  $v$  maintains a variable  $E'(v) \subseteq E(v)$ , where  $E'(v)$  denotes the subset of edges that are not covered yet. Initialize  $E'(v) \leftarrow E(v)$ . The initial state of a vertex is “active”.
3. Each iteration consists of the following steps:
  - a. For every uncovered  $e$ , send  $x(e)$  and  $y(e)$  to every  $v \in e$ .
  - b. For every active  $v$ , if  $\sum_{e \ni v} y(e) \geq 1 - \varepsilon$ , then  $v$  changes its state of  $v$  to “in cover” and sends every edge  $e \in E(v)$  a message “in cover”.<sup>4</sup>
  - c. For every active  $v$ , send  $x[E(v)] = \sum_{e \in v} x(e)$  to every edge  $e' \in E(v)$ .

<sup>4</sup> The value used for  $y(e)$  is the last value received from  $e$ . If  $e$  is uncovered, then it sends  $y(e)$  in the previous round. If  $e$  is covered, then  $v$  remembers the last received value.

- d. For every edge  $e$ , if  $e$  received an “in cover” message, then  $e$  changes its state to “covered”, and sends a “covered” message to every  $v \in e$ .
- e. For every active vertex  $v$ , if  $v$  received a “covered” message from  $e$ , then deletes  $e$  from  $E'(v)$ . If  $E'(v) = \emptyset$ , then  $v$  changes its state to “not in cover”.<sup>5</sup>
- f. For every uncovered edge  $e$ , let  $d(e) = \sum_{v \in e} x[E(v)]$ . Update  $x(e)$  as follows:

$$x(e) \leftarrow \begin{cases} x(e)/K, & \text{if } d(e) \geq K \\ \min\{Kx(e), 1/K\}, & \text{if } d(e) < K \end{cases}$$

The algorithm terminates when all the edges are covered and all the vertices are not active.

**Bound on Message Length.** The messages in the algorithm are  $x(e), y(e), x[E(v)]$  and information about the state. Our goal is to bound the length of these messages.

- **Observation 17.** For every edge  $e$  and iteration  $t$ ,  $\frac{1}{K^t} \leq x_t(e) \leq \frac{1}{K}$ .
- **Observation 18.** For every vertex  $v$  and iteration  $t$ ,  $\frac{1}{K^t} \leq x_t[E(v)] \leq \frac{\Delta}{K}$ .
- **Observation 19.** For every edge  $e$  and iteration  $t$ , if  $y_t(e) > 0$ , then  $\frac{\varepsilon}{K} \cdot \frac{1}{K^t} \leq y_t(e) \leq \frac{\varepsilon}{K} \cdot \frac{t}{K}$ .

The following lemma is implied by the observations above and by the fact that the number of bits required for encoding the numbers in  $[a, b]$  where consecutive numbers differ by  $1/K$  is  $\log(bK/a)$ .

► **Lemma 20.** Let  $T$  denote the number of rounds of the algorithm until it terminates. Then the message length of the vertex cover algorithm is  $O(\log \Delta + T \cdot \log K)$ .

## 6 Proof of the Main Result

**Proof of Theorem 1.** Setting  $K = \sqrt[3]{\frac{\log(f\Delta)}{\log \log(f\Delta)}}$  in Theorem 8, implies that the round complexity of the algorithm is  $O\left(\frac{\log(f\Delta)}{\varepsilon \log \log(f\Delta)}\right)$ .

Claim 7 implies that the set  $C$  computed by our algorithm is indeed a vertex cover, and that this cover is an  $f(1 + O(\varepsilon))$ -approximate solution.

Lemma 20 implies that the message length of our algorithm is  $O(\varepsilon^{-1} \cdot \log(f\Delta))$ , as required. ◀

## 7 Discussion

In this paper we prove that an approximation of the minimum set cover (or the equivalent vertex cover in hypergraphs) can be computed in CONGEST in a locality-optimal way of performing the primal-dual scheme. The attained approximation ratio and number of rounds are  $f(1 + \varepsilon)$  and  $O\left(\frac{\log(f\Delta)}{\varepsilon \log \log(f\Delta)}\right)$  respectively, where  $\varepsilon$  is a constant in  $(0, 1)$ . Hence, for  $f \leq \text{poly}(\log \Delta)$  the round complexity matches the lower bound of  $\Omega\left(\frac{\log \Delta}{\log \log \Delta}\right)$  by Kuhn, Moscibroda, and Wattenhofer [19].

<sup>5</sup> In fact,  $v$  only needs to count the number of received “covered” messages. Hence, IDs and port numbers are not required.

The updates of the dual set variables are governed by the effective degrees of its elements  $e$ , the natural definition of which is (roughly) the summation over the elements which share a set with  $e$ . Unfortunately, it is not clear how to implement this natural definition in CONGEST. A nice observation is that the analysis also works with an approximated definition of the effective degree above (e.g., it allows double counting of elements) which is implementable in CONGEST. Another outcome of this relaxed definition of the effective degree is that our algorithm does not require distinct IDs, namely, the network is anonymous. Moreover, the algorithm does not even rely on numbering of ports. We are hopeful that dynamics of the same style may lead to improvements for other optimization problems.

---

## References

- 1 Matti Åstrand and Jukka Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, pages 294–302. ACM, 2010.
- 2 Reuven Bar-Yehuda, Keren Censor-Hillel, Mohsen Ghaffari, and Gregory Schwartzman. Distributed approximation of maximum independent set and maximum matching. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 165–174, 2017. doi:10.1145/3087801.3087806.
- 3 Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A Distributed  $(2+\epsilon)$ -Approximation for Vertex Cover in  $O(\log \Delta/\epsilon \log \log \Delta)$  Rounds. *J. ACM*, 64(3):23:1–23:11, 2017. doi:10.1145/3060294.
- 4 Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.
- 5 Reuven Bar-Yehuda and Shimon Even. *A local-ratio theorem for approximating the weighted vertex cover problem*. Technion-Israel Institute of Technology. Department of Computer Science, 1983.
- 6 Reuven Bar-Yehuda and Dror Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM Journal on Discrete Mathematics*, 19(3):762–797, 2005.
- 7 R. Ben-Basat, G. Even, K. Kawarabayashi, and G. Schwartzman. A Deterministic Distributed 2-Approximation for Weighted Vertex Cover in  $O(\log n \log \Delta/\log^2 \log \Delta)$  Rounds. *ArXiv e-prints (Appeared in SIROCCO 2018)*, 2018. arXiv:1804.01308.
- 8 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005.
- 9 Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005.
- 10 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
- 11 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 270–277, 2016. doi:10.1137/1.9781611974331.ch20.
- 12 Mohsen Ghaffari, David G Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. *arXiv preprint arXiv:1711.02194*, 2017.
- 13 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 784–797. ACM, 2017.

- 14 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.
- 15 Jonas Holmerin. Improved inapproximability results for vertex cover on  $k$ -uniform hypergraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 1005–1016. Springer, 2002.
- 16 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within  $2 - \epsilon$ . *Journal of Computer and System Sciences*, 74(3):335–349, 2008.
- 17 Christos Koufogiannakis and Neal E Young. Distributed algorithms for covering, packing and maximum weighted matching. *Distributed Computing*, 24(1):45–63, 2011.
- 18 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 980–989. Society for Industrial and Applied Mathematics, 2006.
- 19 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *J. ACM*, 63(2):17:1–17:44, 2016. doi:10.1145/2742012.
- 20 Nathan Linial. Distributive graph algorithms global solutions from local data. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 331–335. IEEE, 1987.
- 21 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.
- 22 Dana Moshkovitz. The projection games conjecture and the np-hardness of  $\ln n$ -approximating set-cover. *Theory of Computing*, 11:221–235, 2015. doi:10.4086/toc.2015.v011a007.
- 23 David Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, 2000.
- 24 Jukka Suomela. Survey of local algorithms. *ACM Computing Surveys (CSUR)*, 45(2):24, 2013.
- 25 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 453–461. ACM, 2001.
- 26 Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.
- 27 David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.