Distributed Computing
https://doi.org/10.1007/s00446-021-00391-w

Optimal distributed covering algorithms

Ran Ben-Basat[1] · Guy Even[2] · Ken-ichi Kawarabayashi[3] · Gregory Schwartzman[4]

Received: 11 November 2019 / Accepted: 7 March 2021

Abstract

We present a time-optimal deterministic distributed algorithm for approximating a minimum weight vertex cover in hypergraphs of rank f. This problem is equivalent to the Minimum Weight Set Cover problem in which the frequency of every element is bounded by f. The approximation factor of our algorithm is $(f+\varepsilon)$. Let Δ denote the maximum degree in the hypergraph. Our algorithm runs in the CONGEST model and requires $O(\log \Delta / \log\log \Delta)$ rounds, for constants $\varepsilon \in (0, 1]$ and $f \in \mathbb{N}^+$. This is the first distributed algorithm for this problem whose running time does not depend on the vertex weights nor the number of vertices. Thus adding another member to the exclusive family of *provably optimal* distributed algorithms. For constant values of f and ε, our algorithm improves over the $(f+\varepsilon)$-approximation algorithm of Kuhn et al. (SODA, 2006)whose running time is $O(\log \Delta + \log W)$, where W is the ratio between the largest and smallest vertex weights in the graph. Our algorithm also achieves an f-approximation for the problem in $O(f \log n)$ rounds, improving over the classical result of Khuller et al. (J Algorithms, 1994) that achieves a running time of $O(f \log^2 n)$. Finally, for weighted vertex cover ($f = 2$) our algorithm achieves a *deterministic* running time of $O(\log n)$, matching the *randomized* previously best result of Koufogiannakis and Young (Distrib Comput, 2011). We also show that integer covering-programs can be reduced to the Minimum Weight Set Cover problem in the distributed setting. This allows us to achieve an $(f\lceil \log_2(M)+1\rceil+\varepsilon)$-approximate integral solution in

$$O\left((1+f/\log n)\cdot\left(\frac{\log\Delta}{\log\log\Delta}+(f\cdot\log M)^{1.01}\cdot\log\varepsilon^{-1}\cdot(\log\Delta)^{0.01}\right)\right)$$

rounds, where f bounds the number of variables in a constraint, Δ bounds the number of constraints a variable appears in, and $M = \max\{1, \lceil 1/a_{\min}\rceil\}$, where $a_{\min}$ is the smallest normalized constraint coefficient.

Keywords Distributed algorithms · Approximation algorithms · Vertex cover · Set cover

1 Introduction

In the Minimum Weight Hypergraph Vertex Cover (MWHVC) problem, we are given a hypergraph $G = (V, E)$ with vertex weights $w : V \to \{1, \ldots, W\}$.[1] The goal is to find a minimum weight *cover* $U \subseteq V$ such that $\forall e \in E : e \cap U \neq \emptyset$. In this paper we develop a distributed approximation algorithm for MWHVC in the CONGEST model. The approximation ratio is $f+\varepsilon$, where f denotes the rank of the hypergraph (i.e., f is an upper bound on the size of every hyperedge). The MWHVC problem is a generalization of the Minimum Weight Vertex Cover (MWVC) problem (in which $f = 2$). The MWHVC problem is also equivalent to the Minimum Weight Set Cover Problem (the rank f of the hypergraph corresponds to the

Supported by the Zuckerman Institute, the Technion Hiroshi Fujiwara Cyber Security Research center, the Israel Cyber Directorate, and by JSPS Kakenhi Grant Number JP19K20216 and JP18H05291.

✉ Ran Ben-Basat
r.benbasat@cs.ucl.ac.uk

Guy Even
guy@eng.tau.ac.il

Ken-ichi Kawarabayashi
k_keniti@nii.ac.jp

Gregory Schwartzman
greg@jaist.ac.jp

1 University College London (UCL), London, UK

2 Tel Aviv University, Tel Aviv-Yafo, Israel

3 National Institute of Informatics, Chiyoda City, Japan

4 JAIST, Ishikawa, Japan

[1] Let $n \triangleq |V|$. We assume that $|E| = n^{O(1)}$ and $W = n^{O(1)}$.

Published online: 11 April 2021

maximum frequency of an element). Both of these problems are among the classical NP-hard problems presented in [14].

We consider the following distributed setting for the MWHVC problem. The communication network is a bipartite graph $H(E \cup V, \{\{e, v\} \mid v \in e\})$. We refer to the network vertices as *nodes* and network edges as *links*. The nodes of the network are the hypergraph vertices on one side and hyperedges on the other side. There is a network link between vertex $v \in V$ and hyperedge $e \in E$ iff $v \in e$. The computation is performed in synchronous rounds, where messages are sent between neighbors in the communication network. As for message size, we consider the CONGEST model where message sizes are bounded to $O(\log |V|)$. This is more restrictive than the LOCAL model where message sizes are unbounded.

1.1 Related work

We survey previous results for MWHVC and MWVC. A comprehensive list of previous results appears in Tables 1 and 2.

Vertex Cover. The understanding of the round complexity for distributed MWVC has been established in two papers: a lower bound in [19] and a matching upper bound in [4]. Let Δ denote the maximum vertex degree in the graph G. The lower bound states that any distributed constant-factor approximation algorithm requires $\Omega(\log \Delta / \log\log \Delta)$ rounds to terminate. This lower bound holds for every constant approximation ratio, for unweighted graphs and even if the message lengths are not bounded (i.e., LOCAL model) [19]. The matching upper bound is a $(2 + \varepsilon)$-approximation distributed algorithm in the CONGEST model, for every $\varepsilon = \Omega(\log\log \Delta / \log \Delta)$.[2] In [16] an $O(\log n)$-round 2-approximation randomized algorithm for weighted graphs in the CONGEST model is given. We note that [16] was the first to achieve this running time with no dependence on W, the maximum weight of the nodes. Recently, Ben-Basat et al. [6] showed an $O(OPT^2 \log OPT)$ rounds algorithm for computing the minimal (unweighted) vertex cover and a $O(OPT)$ rounds for a $(2 + \varepsilon)$-approximation. Here, OPT is the size of the smallest cover and thus these algorithms are adequate when a small solution exists.

Hypergraph Weighted Vertex Cover. For constant values of f, Astrand et al. [2] present an f-approximation algorithm for anonymous networks whose running time is $O(\Delta^2 + \Delta \cdot \log^* W)$. Khuller et al. [15] provide a solution that runs in $O(f \cdot \log \varepsilon^{-1} \cdot \log n)$ rounds in the CONGEST model for any $\varepsilon > 0$ and achieves an $(f + \varepsilon)$-approximation. Setting $\varepsilon = 1/W$ (recall that $W = \text{poly}(n)$) results in a f-approximation in $O(f \log^2 n)$-rounds. For constant ε and f values, Kuhn et al. [17,18] present an $(f + \varepsilon)$-approximation algorithm that terminates in $O(\log \Delta + \log W)$ rounds.

For the Minimum Cardinality (i.e., unweighted) Vertex Cover in Hypergraphs Problem, the lower bound was recently matched by [9] with an $(f + \varepsilon)$-approximation algorithm in the CONGEST model. The round complexity in [9] is $O\left(f/\varepsilon \cdot \frac{\log(f \cdot \Delta)}{\log\log(f \cdot \Delta)}\right)$, which is optimal for constant f and ε. The algorithm in [9] and its analysis is a deterministic version of the randomized maximal independent set algorithm of [10].

1.2 Our contributions

In this paper, we present a deterministic distributed $(f + \varepsilon)$-approximation algorithm for minimum weight vertex cover in f-rank hypergraphs, which completes in

$$O\left((1 + f/\log n) \cdot \left(\frac{\log \Delta}{\log\log \Delta} + (f \cdot \log M)^{1.01} \cdot \log \varepsilon^{-1} \cdot (\log \Delta)^{0.01}\right)\right)$$

rounds in the CONGEST model. For any constants $\varepsilon \in (0, 1)$ and $f \in \mathbb{N}^+$ this implies a running time of $O(\log \Delta / \log\log \Delta)$, which is optimal according to [19]. This is the first distributed algorithm for this problem whose round complexity does not depend on the node weights nor the number of vertices.

Our algorithm is one of a handful of distributed approximation algorithms for *local* problems which are *provably optimal* [3,4,7–9,11]. Among these are the classic Cole-Vishkin algorithm [8] for 3-coloring a ring, the more recent results of [3] and [4] for MWVC and Maximum Matching, and the result of [9] for Minimum Cardinality Hypergraph Vertex Cover.

Our algorithm also achieves a *deterministic* f-approximation for the problem in $O(f \log n)$ rounds. This improves over the best known deterministic result for hypergraphs $O(f \log^2 n)$ [15] and matches the best known *randomized* results for weighted vertex cover ($f = 2$) of $O(\log n)$-rounds [16].

We also show that general covering Integer Linear Programs (ILPs) can be reduced to MWHVC in the distributed setting. That is, LP constraints can be translated into hyperedges such that a cover for the hyperedges satisfies all covering constraints. This allows us to achieve an

$$O\left((1 + f/\log n) \cdot \left(\frac{\log \Delta}{\log\log \Delta} + (f \cdot \log M)^{1.01} \cdot \log \varepsilon^{-1} \cdot (\log \Delta)^{0.01}\right)\right)$$

[2] Recently, the range of ε for which the runtime is optimal was improved to $\Omega(\log^{-c} \Delta)$ for any $c = O(1)$ [5].

Table 1 Previous distributed algorithms for MWVC

Det.	Weighted	Approximation	Time	Algorithm
Yes	No	3	$O(\Delta)$	[21]
Yes	No	2	$O(\Delta^2)$	[1]
Yes	Yes	2	$O(1)$ for $\Delta \leq 3$	[1]
Yes	Yes	2	$O(\Delta + \log^* n)$	[20]
Yes	Yes	2	$O(\Delta + \log^* W)$	[2]
Yes	Yes	2	$O(\log^2 n)$	[15]
Yes	Yes	2	$O(\log n \log \Delta / \log^2 \log \Delta)$	[5]
No	Yes	2	$O(\log n)$	[12,16]
Yes	Yes	2	$O(\log n)$	This work
Yes	Yes	$2+\varepsilon$	$O(\varepsilon^{-4} \log(W \cdot \Delta))$	[13,18]
Yes	Yes	$2+\varepsilon$	$O(\log \varepsilon^{-1} \log n)$	[15]
Yes	Yes	$2+\varepsilon$	$O(\varepsilon^{-1} \log \Delta / \log \log \Delta)$	[4]
Yes	Yes	$2+\varepsilon$	$O\left(\frac{\log \Delta}{\log \log \Delta} + \frac{\log \varepsilon^{-1} \log \Delta}{\log^2 \log \Delta}\right)$	[5]
Yes	Yes	$2+\varepsilon$	$O\left(\frac{\log \Delta}{\log \log \Delta} + \log \varepsilon^{-1} \cdot (\log \Delta)^{0.001}\right)$	This work
Yes	Yes	$2+\frac{\log \log \Delta}{c \cdot \log \Delta}$	$O(\log \Delta / \log \log \Delta)$	[4], $\forall c = O(1)$
Yes	Yes	$2+(\log \Delta)^{-c}$	$O(\log \Delta / \log \log \Delta)$	[5], $\forall c = O(1)$
Yes	Yes	$2+2^{-c \cdot (\log \Delta)^{0.99}}$	$O(\log \Delta / \log \log \Delta)$	This work, $\forall c = O(1)$

In the table, $n = |V|$ and $\varepsilon \in (0, 1)$. Some of the algorithms hold only for the unweighted case and some are randomized. For randomized algorithms the running times hold in expectation or with high probability

Table 2 Previous distributed algorithms for MWHVC

Weighted	Approximation	Time	Algorithm
Yes	f	$O\left(f^2 \Delta^2 + f \Delta \log^* W\right)$	[2]
Yes	f	$O\left(f \log^2 n\right)$	[15]
Yes	f	$O\left(f \log n\right)$	This work
No	$f+\varepsilon$	$O\left(\varepsilon^{-1} \cdot f \cdot \frac{\log(f\Delta)}{\log \log(f\Delta)}\right)$	[9][3]
Yes	$f+\varepsilon$	$O\left(f \cdot \log(f/\varepsilon) \cdot \log n\right)$	[15]
Yes	$f+\varepsilon$	$O\left(\varepsilon^{-4} \cdot f^4 \cdot \log f \cdot \log(W \cdot \Delta)\right)$	[18]
Yes	$f+\varepsilon$	$O\left(f \cdot \log(f/\varepsilon) \cdot (\log \Delta)^{0.001} + \frac{\log \Delta}{\log \log \Delta}\right)$	This work
No	$f+1/c$	$O\left(\log \Delta / \log \log \Delta\right)$	[9], $\forall f, c = O(1)$
Yes	$f+2^{-c \cdot (\log \Delta)^{0.99}}$	$O\left(\log \Delta / \log \log \Delta\right)$	This work, $\forall f, c = O(1)$

In the table, $n = |V|$ and $\varepsilon \in (0, 1)$. All algorithms are deterministic. Note that [9] holds only for unweighted hypergraphs. The authors state their result for an $f(1+\varepsilon)$-approximation which removes the f factor from the runtime

rounds $(f\lceil \log_2(M)+1 \rceil + \varepsilon)$-approximate *integral* solution, where f bounds the number of variables in a constraint, Δ bounds the number of constraints a variable appears in, and $M = \max\{1, \lceil 1/a_{\min} \rceil\}$, where $a_{\min}$ is the smallest normalized constraint coefficient.

1.3 Tools and techniques

The Primal-Dual schema. The Primal-Dual approach introduces, for every hyperedge $e \in E$, a dual variable denoted by $\delta(e)$. The dual edge packing constraints are $\forall v \in V, \sum_{v \in e} \delta(e) \leq w(v)$. If for some $\beta \in [0, 1)$ it holds that $\sum_{v \in e} \delta(e) \geq (1-\beta) \cdot w(v)$, we say the node v is β-tight. Let $\beta = \varepsilon/(f+\varepsilon)$. For every feasible dual solution, the weight of the set of β-tight vertices is at most $(f+\varepsilon)$ times the weight of an optimal (fractional) solution. The algorithm terminates when the set of β-tight edges constitutes a vertex cover.

The fractional LP relaxation of MWHVC is defined as follows.

$$\text{minimize: } \sum_{v \in V} w(v) \cdot x(v) \qquad (\mathcal{P})$$

subject to:

$$\sum_{v \in e} x(v) \geq 1, \quad \forall e \in E; \qquad x(v) \geq 0, \quad \forall v \in V$$

The dual LP is an *Edge Packing* problem defined as follows:

$$\text{maximize: } \sum_{e \in E} \delta(e) \tag{$\mathcal{D}$}$$

subject to:

$$\sum_{e \ni v} \delta(e) \leq w(v), \quad \forall v \in V; \qquad \delta(e) \geq 0, \quad \forall e \in E$$

The following claim is used for proving the approximation ratio of the MWHVC algorithm.

Claim 1 *Let* opt *denote the value of an optimal fractional solution of the primal LP ($\mathcal{P}$). Let $\{\delta(e)\}_{e \in E}$ denote a feasible solution of the dual LP ($\mathcal{D}$). Let $\varepsilon \in (0, 1)$ and $\beta \triangleq \varepsilon/(f+\varepsilon)$. Define the β-tight vertices by $T_\varepsilon \triangleq \{v \in V \mid \sum_{e \ni v} \delta(e) \geq (1-\beta) \cdot w(v)\}$.*

Then $w(T_\varepsilon) \leq (f+\varepsilon) \cdot$ opt.

Proof

$$w(T_\varepsilon) = \sum_{v \in T_\varepsilon} w(v) \leq \frac{1}{1-\beta} \cdot \left(\sum_{v \in T_\varepsilon} \sum_{e \ni v} \delta(e) \right)$$
$$\leq \frac{f}{1-\beta} \sum_{e \in E} \delta(e) \leq (f+\varepsilon) \cdot \text{opt}.$$

The last transition follows from $f/(1-\beta) = f + \varepsilon$ and by weak duality. The claim follows. □

The challenge. When designing a Primal-Dual distributed algorithm, the main challenge is in controlling the rate at which we increase the dual variables. On the one hand, we must grow them rapidly to reduce the number of communication rounds. On the other hand, we may not violate the edge packing constraints. This is tricky in the distributed environments as we have to coordinate between nodes. For example, the result of [4] does not generalize to hypergraphs, as hyperedges require the coordination of more than two nodes in order to increment edge variables.

Our algorithm. The algorithm proceeds in iterations, each of which requires a constant number of communication rounds. We initialize the dual variables in a "safe" way so that feasibility is guaranteed. We refer to the additive increase of the dual variable $\delta(e)$ as bid(e). Similarly to [5] (where bids are called deals), we use *levels* to measure the progress made by a vertex. Whenever the level of a vertex increases, it sends a message about it to all incident edges, which multiply (decrease) their bids by 0.5. Intuitively, the level of a vertex equals the logarithm of its uncovered portion. Formally, we define the level of a vertex v as $\ell(v) \triangleq \left\lfloor \log \frac{w(v)}{w(v) - \sum_{e \ni v} \delta(e)} \right\rfloor$. That is, the initial level of v is 0 and it is increased as the dual variables of the edges incident to v grow. The level of a vertex never reaches $z \triangleq \lceil \log \beta^{-1} \rceil$ as this implies that it is β-tight and entered the cover. Loosely speaking, the algorithm increases the increments bid(e) exponentially (multiplication by α) provided that no vertex $v \in e$ is $(0.5^{\ell(v)}/\alpha)$-tight with respect to the bids of the previous iteration. Here, $\alpha \geq 2$ is a positive parameter that we determine later. This is achieved by checking if all the vertices in e can increase the dual variables by an α factor *while staying at the current level.* Namely, each vertex either sends a "raise" message, that signals that an α-factor increase in the dual variables of all its adjacent edges would keep it in its current level, or a "stuck" message otherwise. bid(e) is then increased only if all the edge's vertices sent a "raise" message. The analysis builds on two observations: (1) The number of times that the increment bid(e) is multiplied by α is bounded by $\log_\alpha \Delta$. (2) The number of iterations in which bid(e) is not multiplied by α is bounded by $O(f \cdot z \cdot \alpha)$. Loosely speaking, each such iteration means that for at least one vertex $v \in e$ the sum of bids is at least an $1/(2\alpha)$-fraction of its slack. Therefore, after at most $O(\alpha)$ such iterations that vertex will level up. Since there are z levels per vertex and f vertices in e, we have that the number of such iterations is at most $O(f \cdot z \cdot \alpha)$. Hence the total number of iterations is bounded by $O(\log_\alpha \Delta + f \cdot z \cdot \alpha)$.

Integer linear programs (ILPs). We show distributed reductions that allow us to compute an $(f+\varepsilon)$-approximation for general covering integer linear programs (ILPs). To that end, we first show that any Zero-One covering program (where all variables are binary) can be translated into a set cover instance in which the vertex degree is bounded by 2^f times the bound on the number of constraints each ILP variable appears in. We then generalize to arbitrary covering ILPs by replacing each variable with multiple vertices in the hypergraph, such that the value associated with the ILP variable will be the weighted sum of the vertices in the cover.

2 Problem formulation

Let $G = (V, E)$ denote a hypergraph. Vertices in V are equipped with positive integer weights $w(v)$. For a subset $U \subseteq V$, let $w(U) \triangleq \sum_{v \in U} w(v)$. Let $E(U)$ denote the set of hyperedges that are incident to some vertex in U (i.e., $E(U) \triangleq \{e \in E \mid e \cap U \neq \emptyset\}$).

The *Minimum Weight Hypergraph Vertex Cover* Problem (MWHVC) is equivalent to the Weighted Set Cover Problem and is defined as follows.

Consider a set system $(X, \mathcal{U})$, where X denotes a set of elements and $\mathcal{U} = \{U_1, \ldots, U_m\}$ denotes a collection of subsets of X. The reduction from the set system $(X, \mathcal{U})$ to a hypergraph $G = (V, E)$ proceeds as follows. The set of vertices is $V \triangleq \{u_1, \ldots, u_m\}$ (one vertex u_i per subset U_i).

The set of edges is $E \triangleq \{e_x\}_{x \in X}$ (one hyperedge e_x per element x), where $e_x \triangleq \{u_i : x \in U_i\}$. The weight of vertex u_i equals the weight of the subset U_i.

We now detail the setting for computing a distributed $(f+\varepsilon)$-approximation of the problem.

Input. The input is a hypergraph $G = (V, E)$ with nonnegative vertex weights $w : V \to \mathbb{N}^+$ and an approximation ratio parameter $\varepsilon \in (0, 1]$. We denote the number of vertices by n, the rank of G by f (i.e., each hyperedge contains at most f vertices), and the maximum degree of G by Δ (i.e., each vertex belongs to at most Δ edges).

Assumption 1 We assume that (i)Vertex weights are polynomial in $n \triangleq |V|$ so that sending a vertex weight requires $O(\log n)$ bits. (ii)Vertex degrees are polynomial in n (i.e., $|E(v)| = n^{O(1)}$) so that sending a vertex degree requires $O(\log n)$ bits. Since $|E(v)| \leq n^f$, this assumption trivially holds for constant f. (iii)The maximum degree is at least 3 so that $\log \log \Delta > 0$. (iv)All vertices know f (or an estimate $\widehat{f} = \Theta(f)$). For getting an f- (but not for an $f+\varepsilon$)−approximation, the vertices also need to know n and W. For simplicity, we assume that Δ is known, but later discuss how to remove this assumption.

Output. The nodes compute a vertex cover $C \subseteq V$. Namely, for every hyperedge $e \in E$, the intersection $e \cap C$ is not empty. The set C is maintained locally in the sense that every vertex v knows whether it belongs to C or not.

Communication Network. The communication network $N(E \cup V, \{\{e, v\} \mid v \in e\})$ is a bipartite graph. There are two types of nodes in the network: *servers* and *clients*. The set of servers is V (the vertex set of G) and the set of clients is E (the hyperedges in G). There is a link (v, e) from server $v \in V$ to a client $e \in E$ if $v \in e$. We note that the degree of the clients is bounded by f and the degree of the servers is bounded by Δ.

Notation.

- We say that an edge e is *covered* by C if $e \cap C \neq \emptyset$.
- Let $E(v) \triangleq \{e \in E \mid v \in e\}$ denote the set of hyperedges that contain v.
- For every vertex v, the algorithm maintains a subset $E'(v) \subseteq E(v)$ that consists of the uncovered hyperedges in $E(v)$ (i.e., $E'(v) = \{e \in E(v) \mid e \cap C = \emptyset\}$).

Invariants. Throughout its execution, as we prove in the analysis in Sect. 4.1, the algorithm maintains the following invariants *at the end* of each iteration:

- The level of each vertex which did not terminate correctly measures its tightness, i.e., $\ell(v) = \left\lfloor \log \frac{w(v)}{w(v) - \sum_{e \ni v} \delta(e)} \right\rfloor$.
- The dual variables constitute a feasible edge packing. Namely,

$$\sum_{e \in E(v)} \delta_i(e) \leq w(v) \qquad \text{for every vertex } v \in V,$$

$$\delta_i(e) \geq 0 \qquad \text{for every edge } e \in E.$$

3 Distributed approximation algorithm for MWHVC

3.1 Parameters and variables

- The algorithm computes an $(f+\varepsilon)$-approximation where $\varepsilon \in (0, 1]$. The parameter β is defined by $\beta \triangleq \varepsilon/(f+\varepsilon)$, where f is the rank of the hypergraph.
- Each vertex v is assigned a level $\ell(v)$ which is a nonnegative integer.
- We denote the dual variables at the end of iteration i by $\delta_i(e)$ (see Sect. 1.3 for a description of the dual edge packing linear program). The amount by which $\delta_i(e)$ is increased in iteration i is denoted by $\text{bid}_i(e)$. Namely, $\delta_i(e) = \sum_{j \leq i} \text{bid}_j(e)$.
- The parameter $\alpha \geq 2$ determines the factor by which bids are multiplied. We determine its value in the analysis in the following section.

3.2 Algorithm MWHVC

1. Initialization. Set $C \leftarrow \emptyset$. For every vertex v, set level $\ell(v) \leftarrow 0$ and uncovered edges $E'(v) \leftarrow E(v)$.
2. Iteration $i = 0$. Every edge e collects the weight $w(v)$ and degree $|E(v)|$ from every vertex $v \in e$, and sets: $\text{bid}(e) = 0.5 \cdot \min_{v \in e}\{w(v)/|E(v)|\}$. The value $\text{bid}(e)$ is sent to every $v \in e$. The dual variable is set to $\delta(e) \leftarrow \text{bid}(e)$.
3. For $i = 1$ to ∞ do:

(a) Check β-tightness. For every $v \notin C$, if $\sum_{e \in E(v)} \delta(e) \geq (1-\beta)w(v)$, then v joins the cover C, sends a message to every $e \in E'(v)$ that e is covered, and vertex v terminates.

(b) For every uncovered edge e, if e receives a message that it is covered, then it tells all its vertices that e is covered and terminates.

(c) For every vertex $v \notin C$, if it receives a message from e that e is covered, then $E'(v) \leftarrow E'(v) \setminus \{e\}$. If $E'(v) = \emptyset$, then v terminates (without joining the cover).

(d) Increment levels and scale bids. For every active (that has not terminated) vertex[3]:

While $\sum_{e \in E(v)} \delta(e) > w(v)(1 - 0.5^{\ell(v)+1})$ do

(i) $\ell(v) \leftarrow \ell(v) + 1$

[3] For simplicity of the description, we assume in step 3(d)ii that every $v \in e$ decides whether $\text{bid}(e)$ is halved. In a distributed setting, $\text{bid}(e)$ is halved if there exists a vertex $v \in e$ that requests such a halving. The distributed implementation of this step is further discussed in Appendix A.

(ii) For every $e \in E'(v)$: $\text{bid}(e) \leftarrow 0.5 \cdot \text{bid}(e)$
(e) For every active vertex, if $\sum_{e \in E'(v)} \text{bid}(e) \leq \frac{1}{\alpha} \cdot 0.5^{\ell(v)+1} \cdot w(v)$, then send the message "raise" to every $e \in E'(v)$; otherwise, send the message "stuck" to every $e \in E'(v)$.
(f) For every uncovered edge e, if *all* incoming messages are "raise", set $\text{bid}(e) \leftarrow \alpha \cdot \text{bid}(e)$.
(g) Send $\text{bid}(e)$ to every $v \in e$ (so it can track $\delta(e)$).

*Termination. Every vertex v terminates when either $v \in C$ or every edge $e \in E(v)$ is covered (i.e., $E'(v) = \emptyset$). Every edge e terminates when it is covered (i.e., $e \cap C \neq \emptyset$). We say that the algorithm has terminated if all the vertices and edges have terminated.
*Execution in CONGEST. See Section A in the Appendix for a discussion of how Algorithm MWHVC is executed in the CONGEST model.

4 Algorithm analysis

In this section, we analyze the approximation ratio and the running time of the algorithm. Throughout the analysis, we attach an index i to the variables $\text{bid}_i(e)$, $\delta_i(e)$ and $\ell_i(v)$. The indexed variable refers to its value at the end of the i'th iteration.

4.1 Feasibility and approximation ratio

The following invariants are satisfied throughout the execution of the algorithm. In the following claim we bound the sum of the bids of edges incident to a vertex.

Claim 2 *If $v \notin C$ at the end of iteration i, then*

$$\sum_{e \in E'(v)} \text{bid}_i(e) \leq 0.5^{\ell_i(v)+1} \cdot w(v).$$

Proof The proof is by induction on i. For $i = 0$, the claim holds because $\ell_0(v) = 0$ and $\text{bid}_0(e) \leq 0.5 \cdot w(v)/|E(v)|$. The induction step, for $i \geq 1$, considers two cases. (A) If v sends a "raise" message in iteration i, then Step 3e implies that $\sum_{e \in E'(v)} \text{bid}_i(e) \leq 0.5^{\ell(v)+1} \cdot w(v)$, as required. (B) Suppose v sends a "stuck" message in iteration i. By Step 3d, $\text{bid}_i(e) \leq 0.5^{\ell_i(v)-\ell_{i-1}(v)} \cdot \text{bid}_{i-1}(e)$ for every $e \in E'(v)$. The induction hypothesis states that $\sum_{e \in E'(v)} \text{bid}_{i-1}(e) \leq 0.5^{\ell_{i-1}(v)+1} \cdot w(v)$. The claim follows by combining these inequalities. □

If an edge e is covered in iteration j, then e terminates and $\delta_i(e)$ is not set for $i \geq j$. In this case, we define $\delta_i(e) = \delta_{j-1}(e)$, namely, the last value assigned to a dual variable.

Claim 3 *For every $i \geq 1$ and every a vertex v that has not terminated the following inequality holds:*

$$w(v)(1 - 0.5^{\ell_i(v)}) \leq \sum_{e \in E(v)} \delta_{i-1}(e) \leq (1 - 0.5^{\ell_i(v)+1}) \cdot w(v) \,. \tag{1}$$

In addition
the dual variables $\delta_i(e)$ constitute a feasible edge packing. Namely,

$$\sum_{e \in E(v)} \delta_i(e) \leq w(v) \qquad \text{for every vertex } v \in V, \tag{2}$$

$$\delta_i(e) \geq 0 \qquad \text{for every edge } e \in E. \tag{3}$$

Proof We prove the claim by induction on the iteration number i. To simplify the proof, we reformulate the statement of the feasibility of the dual variables to $i - 1$, i.e., $\sum_{e \in E(v)} \delta_{i-1}(e) \leq w(v)$ and $\delta_{i-1}(e) \geq 0$. We first prove the induction basis for $i = 1$.

Proof of Eq. 1 for $i = 1$. Fix a vertex v. At the end of iteration 0, $\ell_0(v) = 0$ and $0 < \text{bid}_0(e) \leq w(v)/(2|E(v)|)$, for every $e \in E(v)$. Hence $0 < \sum_{e \in E(v)} \text{bid}_0(e) \leq w(v)/2$. Because $\delta_0(e) = \text{bid}_0(e)$, the condition in Step 3d does not hold, and $\ell_1(v) = \ell_0(v) = 0$. We conclude that Eq. 1 holds for $i = 1$.

Proof of feasibility of $\delta_0(e)$ (Eq. (2) and (3)) for $i = 1$. Non-negativity follows from the fact that $\delta_0(e) = \text{bid}_0(e) > 0$. The packing constraint for vertex v is satisfied because $\sum_{e \in E(v)} \text{bid}_0(e) \leq w(v)/2$. This completes the proof of the induction basis.
We now prove the induction step assuming that Eq. 1 holds for δ_{i-1}.

Proof of Eq. 1 for $i > 1$. Since v is not in the cover it is also not β-tight. Step 3d in iteration i increases $\ell(v)$ until Eq. 1 holds for i.

Proof of feasibility of $\delta_{i-1}(e)$ (Eq. (2) and (3)) for $i > 1$. Consider a vertex v. If v joins C in iteration $i - 1$, then $\delta_{i-1} = \delta_{i-2}$, and the packing constraint of v holds by the induction hypothesis. Otherwise, then by $\delta_{i-1}(e) = \delta_{i-2}(e) + \text{bid}_{i-1}(e)$, Claim 2, and the induction hypothesis for Eq. 1, we have

$$\begin{aligned}\sum_{e \in E(v)} \delta_{i-1}(e) &= \sum_{e \in E(v)} (\delta_{i-2}(e) + \text{bid}_{i-1}(e)) \\ &\leq \left(1 - 0.5^{\ell_{i-1}(v)+1} + 0.5^{\ell_{i-1}(v)+1}\right) \cdot w(v) = w(v) \,.\end{aligned}$$

□

Let opt denote the cost of an optimal (fractional) weighted vertex cover of G.

Corollary 1 *Upon termination, the approximation ratio of Algorithm* MWHVC *is $f + \varepsilon$.*

Proof Throughout the algorithm, the set C consists of β-tight vertices. By Claim 1, $w(C) \leq (f+\varepsilon)\cdot \mathsf{opt}$. Upon termination, C constitutes a vertex cover (as an edge only terminate once covered), and the corollary follows. □

4.2 Communication rounds analysis

In this section, we prove that the number of communication rounds of Algorithm MWHVC is bounded by (where $\gamma > 0$ is a constant, e.g., $\gamma = 0.001$)

$$O\left(f\cdot\log(f/\varepsilon) + \frac{\log\Delta}{\gamma\cdot\log\log\Delta} + \min\left\{\log\Delta,\, f\cdot\log(f/\varepsilon)\cdot(\log\Delta)^{\gamma}\right\}\right).$$

It suffices to bound the number of iterations because each iteration consists of a constant number of communication rounds.
Let $z \triangleq \lceil \log_2 \frac{1}{\beta}\rceil$. Note that $z = O(\log(f/\varepsilon))$.

Claim 4 *The level of every vertex is always less than z.*

Proof Assume that $\ell_i(v) \geq z$. By Eq. 1, $\sum_{e\in E(v)}\delta_{i-1}(e) \geq w(v)\cdot(1-2^{-z}) \geq (1-\beta)\cdot w(v)$. This implies that v is β-tight and joins the cover in Line 3a before $\ell_i(v)$ reaches z. □

4.2.1 Raise or stuck iterations

Definition 1 (e-raise and v-stuck iterations) An iteration $i \geq 1$ is an *e-raise* iteration if in Line 3f we multiplied $\mathrm{bid}(e)$ by α. An iteration $i \geq 1$ is a *v-stuck* iteration if v sent the message "stuck" in iteration i.

Note that if iteration i is a v-stuck iteration and $v \in e$, then $\mathrm{bid}_i(e) \leq \mathrm{bid}_{i-1}(e)$ and i is not an e-raise iteration.
We bound the number of e-raise iterations as follows.

Lemma 1 *The number of e-raise iterations is bounded by $\log_\alpha(\Delta\cdot 2^{f\cdot z})$.*

Proof Let v^* denote a vertex with minimum normalized weight in e (that is, $v^* \in \mathrm{argmin}_{v\in e}\{w(v)/|E(v)|\}$). The first bid satisfies $\mathrm{bid}_0(e) = 0.5\cdot w(v^*)/|E(v^*)| \geq 0.5\cdot w(v^*)/\Delta$. By Claim 2, $\mathrm{bid}_i(e) \leq 0.5\cdot w(v^*)$. The bid is multiplied by α in each e-raise iteration and is halved at most $f\cdot z$ times. The bound on the number of halvings holds because the number of vertices in the edge is bounded by f, and each vertex halves the bid each time its level is incremented. The lemma follows. □

We bound the number of v-stuck iterations as follows.

Lemma 2 *For every vertex v and level $\ell(v)$, the number of v-stuck iterations is bounded by α.*

Proof Notice that when v reached the level $\ell(v)$, we had $\sum_{e\in E(v)}\delta(e) \geq w(v)(1-0.5^{\ell(v)})$. The number of v-stuck iterations is then bounded by the number of times it can send a "stuck" message without reaching $\sum_{e\in E(v)}\delta(e) > w(v)(1-0.5^{\ell(v)+1})$. Indeed, once this inequality holds, the level of v is incremented. Every stuck iteration implies, by Line 3e, that $\sum_{e\in E'(v)}\mathrm{bid}(e) > \frac{1}{\alpha}\cdot 0.5^{\ell(v)+1}\cdot w(v)$. Therefore, we can bound the number of iteration by $\frac{w(v)(1-0.5^{\ell(v)+1})-w(v)(1-0.5^{\ell(v)})}{\frac{1}{\alpha}\cdot 0.5^{\ell(v)+1}\cdot w(v)} = \alpha$. □

4.2.2 Bound on the number of rounds

Theorem 1 *For every $\alpha \geq 2$, the number of iterations of Algorithm* MWHVC *is*

$$O\left(\log_\alpha(\Delta\cdot 2^{f\cdot z}) + f\cdot z\cdot\alpha\right) = O\left(\log_\alpha\Delta + f\cdot\log(f/\varepsilon)\cdot\alpha\right).$$

Proof Fix an edge e. We bound the number of iterations until e is covered. Every iteration is either an e-raise iteration or a v-stuck iteration for some $v \in e$. Since e contains at most f vertices, we conclude that the number of iterations is bounded by the number of e-raise iterations plus the sum over $v \in e$ of the number of v-stuck iterations. The theorem follows from Lemmas 1 and 2 . □

In Theorem 2 we assume that all the vertices know the maximum degree Δ and that $\Delta \geq 3$. The assumption that the maximal degree Δ is known to all vertices is not required. Instead, each hyperedge e can compute a local maximum degree $\Delta(e)$, where $\Delta(e) \triangleq \max_{u\in e}|E(u)|$. The local maximum degree $\Delta(e)$ can be used instead of Δ to define local value of the multiplier $\alpha = \alpha(e)$. Let $T(f,\Delta,\varepsilon)$ denote the round complexity of Algorithm MWHVC. By setting α appropriately, we bound the running time as follows.

Theorem 2 [4] *Let $\gamma > 0$ denote a constant and set*

$$\alpha = \begin{cases} \max\left(2, \frac{\log\Delta}{f\cdot\log(f/\varepsilon)\cdot\log\log\Delta}\right) & \text{if } \frac{\log\Delta}{f\cdot\log(f/\varepsilon)\cdot\log\log\Delta} \geq (\log\Delta)^{\gamma/2} \\ 2 & \text{otherwise.} \end{cases}$$

Then, the round complexity of Algorithm MWHVC *satisfies:*

$$T(f,\Delta,\varepsilon) = O\left(f\cdot\log(f/\varepsilon) + \frac{\log\Delta}{\log\log\Delta} + \min\left\{\log\Delta,\, f\cdot\log(f/\varepsilon)\cdot(\log\Delta)^{\gamma}\right\}\right).$$

Proof We prove the theorem using a case analysis. **Case** $\alpha = \frac{\log\Delta}{f\cdot\log(f/\varepsilon)\cdot\log\log\Delta} \geq 2$ and $\alpha \geq (\log\Delta)^{\gamma/2}$.

[4] The statement of the theorem is asymptotic (in Δ). This means that for every constant γ, it holds that either $\log^{\gamma/2}\Delta \geq \log\log\Delta$ or Δ is bounded by a constant (determined by γ) in which case expressions involving Δ can be omitted from the asymptotic expression.

This means that the runtime is bounded by $O\left(\log_\alpha \Delta + f \cdot \log(f/\varepsilon) \cdot \alpha\right) = O\left(\frac{\log \Delta}{\log\log \Delta}\right)$.

Case $\alpha = 2$ and $2 > \frac{\log \Delta}{f \cdot \log(f/\varepsilon) \cdot \log\log \Delta} \geq (\log \Delta)^{\gamma/2}$.

Notice that in this case $2 > (\log \Delta)^{\gamma/2}$ which implies that Δ is constant. Therefore, the round complexity of our algorithm is bounded by

$$O\left(\log \Delta + f \cdot \log(f/\varepsilon)\right) = O\left(f \cdot \log(f/\varepsilon)\right).$$

Case $\frac{\log \Delta}{f \cdot \log(f/\varepsilon) \cdot \log\log \Delta} < (\log \Delta)^{\gamma/2}$.
This implies that

$$\begin{aligned}\log \Delta &\leq \min\left\{\log \Delta,\ f \cdot \log(f/\varepsilon) \cdot (\log \Delta)^{\gamma/2} \cdot \log\log \Delta\right\} \\ &= O\left(\min\left\{\log \Delta,\ f \cdot \log(f/\varepsilon) \cdot (\log \Delta)^{\gamma}\right\}\right).\end{aligned}$$

Therefore, since $\alpha = 2$ in this case, the runtime is bounded by

$$\begin{aligned}&O\left(\log \Delta + f \cdot \log(f/\varepsilon)\right) \\ &= O\left(f \cdot \log(f/\varepsilon) + \min\left\{\log \Delta,\ f \cdot \log(f/\varepsilon) \cdot (\log \Delta)^{\gamma}\right\}\right).\end{aligned}$$

□

Let $W \triangleq \max_v w(v)$. By setting $\varepsilon = 1/(nW)$, we conclude the following result for an f-approximation (recall that the vertex degrees and weights are polynomial in n):

Corollary 2 *Algorithm* MWHVC *computes an f-approximation in $O(f \log n)$ rounds.*

Additionally, we get the following range of parameters for which the round complexity is optimal:

Corollary 3 *Let $f = O\left((\log \Delta)^{0.99}\right)$ and $\varepsilon = (\log \Delta)^{-O(1)}$. Then, Algorithm* MWHVC *computes an $(f+\varepsilon)$-approximation in $O\left(\frac{\log \Delta}{\log\log \Delta}\right)$ rounds.*

For $f = O(1)$ we also get an extension of range of parameters for which the round complexity is optimal. This extension is almost exponential compared to the $\varepsilon = (\log \Delta)^{-O(1)}$ of [5].

Corollary 4 *Let $f = O(1)$ and $\varepsilon = 2^{-O\left((\log \Delta)^{0.99}\right)}$. Then our algorithm computes an $(f+\varepsilon)$-approximation and terminates in $O\left(\frac{\log \Delta}{\log\log \Delta}\right)$ rounds.*

5 Approximation of covering ILPs

In this section, we present a reduction from solving covering integer linear programs (ILPs) to MWHVC. This reduction implies that one can distributively compute approximate solutions to covering ILPs using a distributed algorithm for MWHVC.

Notation. Let $\mathbb{N}$ denote the set of natural numbers, including 0. Let A denote a real $m \times n$ matrix, $\mathbf{b} \in \mathbb{R}^n$, and $\mathbf{w} \in \mathbb{R}^n$. Let $LP(A, \mathbf{b}, \mathbf{w})$ denote the linear program $\min\ \mathbf{w}^T \cdot \mathbf{x}$ subject to $A \cdot \mathbf{x} \geq \mathbf{b}$ and $\mathbf{x} \geq 0$. Let $ILP(A, \mathbf{b}, \mathbf{w})$ denote the integer linear program $\min\ \mathbf{w}^T \cdot \mathbf{x}$ subject to $A \cdot \mathbf{x} \geq \mathbf{b}$ and $\mathbf{x} \in \mathbb{N}^n$.

Definition 2 The linear program $LP(A, \mathbf{b}, \mathbf{w})$ and integer linear program $ILP(A, \mathbf{b}, \mathbf{w})$ are *covering programs* if all the components in $A, \mathbf{b}, \mathbf{w}$ are non-negative.

5.1 Distributed setting

We denote the number of rows of the matrix A by m and the number of columns by n. Let $f(A)$ (resp., $\Delta(A)$) denote the maximum number of nonzero entries in a row (resp., column) of A.

Given a covering ILP, $ILP(A, \mathbf{b}, \mathbf{w})$, the communication network $N(ILP)$ over which the ILP is solved is the bipartite graph $N = (X \times C, E)$, where: $X = \{x_j\}_{j \in [n]}$, $C = \{c_i\}_{i \in [m]}$, and $E = \{(x_j, c_i) \mid A_{i,j} \neq 0\}$. We refer to the nodes in X as variable nodes, and to those in C as constraint nodes. Note that the maximum degree of a constraint node is $f(A)$ and the maximum degree of a variable node is $\Delta(A)$.

We assume that the local input of every variable node x_j consists of w_j and the j'th column of A (i.e., $A_{i,j}$, for $i \in [m]$). Every constraint vertex c_i is given the value of b_i as its local input. We assume that these inputs can be represented by $O(\log(nm))$ bits. In $(f+1)$ rounds, every variable node v_j can learn all the components in the rows i of A such that $A_{i,j} \neq 0$ as well as the component b_i.

5.2 Zero-one covering programs

The special case in which a variable may be assigned only the values 0 or 1 is called a *zero-one program*. We denote the zero-one covering ILP induced by a matrix A and vectors $\mathbf{b}$ and $\mathbf{w}$ by $ZO(A, \mathbf{b}, \mathbf{z})$. Every instance of the MWHVC problem is a zero-one program in which the matrix A is the incidence matrix of the hypergraph. The following lemma deals with the converse reduction.

Lemma 3 *Every feasible zero-one covering program $ZO(A, \mathbf{b}, \mathbf{w})$ can be reduced to an* MWHVC *instance with rank $f' < f(A)$ and degree $\Delta' < 2^{f(A)} \cdot \Delta(A)$.*

Proof Let A_i denote the i'th row of the matrix A. For a subset $S \subseteq [n]$, let I_S denote the indicator vector of S. Let $\mathbf{x} \in \{0,1\}^n$ and let $\sigma_i \triangleq \{j \in [n] \mid A_{i,j} \neq 0\}$. Feasibility implies that $A_i \cdot I_{\sigma_i} \geq b_i$, for every row i. Let $\mathcal{S}_i$ denote the set of all subsets $S \subset [n]$ such that the indicator vector I_S does not

satisfy the i'th constraint, i.e., $A_i \cdot I_S < b_i$ The i'th constraint is not satisfied, i.e., $A_i \cdot \mathbf{x} < b_i$, if and if only there exists a set $S \in \mathcal{S}_i$ such that $\mathbf{x} = I_S$. Hence, $A_i \cdot \mathbf{x} < b_i$ if and only if the truth value of the following DNF formula is true: $\varphi_i(x) \triangleq \bigvee_{S \in \mathcal{S}_i} \bigwedge_{j \in \sigma_i \setminus S} \text{not}(x_j)$. By De Morgan's law, we obtain that $\text{not}(\varphi_i(x))$ is equivalent to a monotone CNF formula $\psi_i(x)$ such that $\psi_i(x)$ has less than $2^{f(A)}$ clauses, each of which has length less than $f(A)$. We now construct the hypergraph H for the MWHVC instance as follows. For every row i and every $S \in \mathcal{S}_i$, add the hyperedge $e_{i,S} = \sigma_i \setminus S_i$. (Feasibility implies that $e_{i,S}$ is not empty.) Given a vertex cover C of the hypergraph, every hyperedge is stabbed by C, and hence I_C satisfies all the formulae $\psi_i(x)$, where $i \in [m]$. Hence, $A_i \cdot I_C \geq b_i$, for every i. The converse direction holds as well, and the lemma follows. □

How does $N(ILP)$ (a bipartite graph with $m+n$ vertices) simulate the execution of MWHVC over the hypergraph H? Each variable node x_j simulates all hyperedges $e_{i,S}$, where $j \in \sigma_i$ and $S \in \mathcal{S}_i$. First, the variable nodes exchange their weights with the variables nodes they share a constraint with in $O(f(A))$ rounds – first every x_j broadcasts its own weight and then each c_i sends all neighbors the weights it received. At each iteration, the variable node sends a raise/stuck message and whether its level was incremented. [5] Notice that the number of rounds required for each such iteration is $O(1 + f(A)/\log n)$ (i.e., constant for $f(A) = O(\log n)$). Each edge node c_i then broadcasts to all vertices two $f(A)$-bit messages that indicate two subsets of vertices of the edge: those that sent a raise message (the complement sent a stuck message) and those that incremented their level. Each variable node x_j knows how to update its bid with every $e_{i,S}$ for which $j \in \sigma_i$ and $S \in \mathcal{S}_i$.

We summarize the complexity of the distributed algorithm for solving a zero-one covering program.

Claim 5 *Denoting by $T(f, \Delta, \varepsilon)$ is the running time of Algorithm* MWHVC, *There exists a distributed* CONGEST *algorithm for computing an $(f + \varepsilon)$-approximate solution for zero-one covering programs with running time of $O((1 + f(A)/\log n) \cdot T(f(A), 2^{f(A)} \cdot \Delta(A), \varepsilon))$.*

5.3 Reduction of covering ILPs to zero-one covering

Consider the covering ILP $ILP(A, \mathbf{b}, \mathbf{w})$. We present a reduction of the ILP to a zero-one covering program.

Definition 3 Define $M(A, \mathbf{b}) \triangleq \max_j \max_i \left\{ \frac{b_i}{A_{i,j}} \mid A_{i,j} \neq 0 \right\}$.

We abbreviate and write M for $M(A, \mathbf{b})$ when the context is clear.

[5] One needs to modify the MWHVC algorithm slightly so that, in each iteration, the level of every vertex is increased by at most 1. We defer the details to the full version.

Proposition 1 *Limiting $\mathbf{x}$ to the box $[0, M]^n$ does not increase the optimal value of the ILP.*

Claim 6 *Every covering ILP, $ILP(A, \mathbf{b}, \mathbf{w})$, can be approximated by a zero-one covering program $ZO(A', \mathbf{b}, \mathbf{w'})$, where $f(A') \leq f(A) \cdot \lceil \log_2(M) + 1 \rceil$ and $\Delta(A') = \Delta(A)$.*

Proof Let $B = \lceil \log_2 M \rceil$. Limiting each variable x_j by M means that we can replace x_j by B zero-one variables $\{x_{j,\ell}\}_{\ell=0}^{B-1}$ that correspond to the binary representation of x_j, i.e., $x_j = \sum_{\ell=0}^{B-1} 2^\ell \cdot x_{j,\ell}$. This replacement means that the dimensions of the matrix A' are $m \times n'$, where $n' = n \cdot B$. The j'th column $A^{(j)}$ of A is replaced by B columns, indexed 0 to $B-1$, where the ℓ'th column equals $2^\ell \cdot A^{(j)}$. The vector $\mathbf{w'}$ is obtained by duplicating and scaling the entries of $\mathbf{w}$ in the same fashion. □

Combining Claims 5 and 6 , we obtain the following result.

Theorem 3 *There exists a distributed* CONGEST *algorithm for computing an $(f(A') + \varepsilon) = (f(A) \cdot \lceil \log_2(M) + 1 \rceil + \varepsilon)$-approximate solution for covering integer linear programs $ILP(A, \mathbf{b}, \mathbf{w})$ with running time of $O((1 + f(A)/\log n) \cdot T(f(A) \cdot \log M, 2^{f(A)} \cdot M \cdot \Delta(A), \varepsilon))$, where $T(f, \Delta, \varepsilon)$ is the running time of Algorithm* MWHVC.[6]

Proof Let f_{HVC}, f_{ZO}, f_{ILP} denote the ranks of the hypergraph vertex cover instance, zero-one covering program, and ILP, respectively. We use the same notation for maximum degrees. The reduction of zero-one programs to MWHVC in Lemma 3 implies that $f_{HVC} \leq f_{ZO}$ and $\Delta_{HVC} \leq 2^{f_{ZO}} \cdot \Delta_{ZO}$. The reduction of covering ILPs to zero-one programs in Claim 6 implies that $f_{ZO} \leq f_{ILP} \cdot (1 + \log M)$ and $\Delta_{ZO} \leq \Delta_{ILP}$. The composition of the reductions gives $f_{HVC} \leq f_{ILP} \cdot (1 + \log M)$ and $\Delta_{HVC} \leq 2^{f_{ILP} \cdot (1 + \log M)} \cdot \Delta_{ILP} = 2^{f_{ILP}} \cdot 2M \cdot \Delta_{ILP}$. □

After some simplifications, the running time of the resulting algorithm for $(f \cdot \lceil \log_2(M) + 1 \rceil + \varepsilon)$-approximate integer covering linear programs is

$$O\left((1 + f/\log n) \cdot \left(\frac{\log \Delta}{\log \log \Delta} + (f \cdot \log M)^{1.01} \cdot \log \varepsilon^{-1} \cdot (\log \Delta)^{0.01} \right) \right)$$

[6] In the conference version of this paper, the theorem mistakenly claim that an $(f + \varepsilon)$ approximation is achieved.

A adaptation to the CONGEST model

We need to show that the message lengths in Algorithm MWHVC are $O(\log n)$.

1. In round 0, every vertex v sends its weight $w(v)$ and degree $|E(v)|$ to every hyperedge in $e \in E(v)$. We assume that the weights and degrees are polynomial in n, hence the length of the binary representations of $w(v)$ and $|E(v)|$ is $O(\log n)$. Every hyperedge e sends back to every $v \in e$ the pair $(w(v_e), |E(v_e)|)$, where v_e has the smallest normalized weight, i.e., $v_e = \text{argmin}_{v \in e}\{w(v)/|E(v)|\}$. Every vertex $v \in e$ locally computes $\text{bid}_0(e) = w(v_e)/(2 \cdot |E(v_e)|)$ and $\delta_0(e) = \text{bid}_0(e)$.
2. In round $i \geq 1$, every vertex sends messages. Messages of the sort: "e is covered", "raise", or "stuck" require only a constant number of bits. the increment of v's level needs to be sent to the edges in $e(v)$. these increments require $o(\log z) = o(\log n)$ bits.
3. Every edge e sends to every $v \in e$ the number of times that $\text{bid}(e)$ is halved in this iteration. This message is $O(\log z)$ bits long.
4. Every edge sends the final bid to the vertices. Instead of sending the value of the bid, the edge can send a single bit indicating whether the bid was multiplied by α.
5. Finally, if $\alpha = \alpha(e)$ is set locally based on the local maximum degree $\max_{v \in e} |E(v)|$, then every vertex v sends its degree to all the edges $e \in E(v)$. The local maximum degree for e is sent to every vertex $v \in V$, and this parameter is used to compute $\alpha(e)$ locally.

B algorithm with at most one level increment per iteration

We propose to do a single change that will ensure that no vertex levels up more than once per iteration. To that end, we modify Line 3f of our MWHVC algorithm to

– For every uncovered edge e, if *all* incoming messages are "raise" $\text{bid}(e) \leftarrow \alpha \cdot \text{bid}(e)$. Send $\text{bid}(e)$ to every $v \in e$, who updates $\delta(e) \leftarrow \delta(e) + \text{bid}(e)/2$.

That is, the algorithm remains intact except that the dual variables $\delta(e)$ are raised by $\text{bid}(e)/2$ rather than by $\text{bid}(e)$. Intuitively, this guarantees that a vertex's slack does not reduce by more than 50% in each iteration and therefore its level may increase by at most one. We revisit the proof of 3, and, specifically, the *Proof of feasibility of* $\delta_{i-1}(e)$ for $i > 1$.

Proof of feasibility of $\delta_{i-1}(e)$ for $i > 1$. Consider a vertex v. If v joins C in iteration $i-1$, then $\delta_{i-1} = \delta_{i-2}$, and the packing constraint of v holds by the induction hypothesis. If $v \notin C$, then by $\delta_{i-1}(e) = \delta_{i-2}(e) + \text{bid}_{i-1}(e)/2$, Claim 2, and the induction hypothesis for Eq. 1, we have

$$\begin{aligned}\sum_{e \in E(v)} \delta_{i-1}(e) &= \sum_{e \in E(v)} (\delta_{i-2}(e) + \text{bid}_{i-1}(v)/2) \\ &\leq \left(1 - 0.5^{\ell_{i-1}(v)+1} + 0.5^{\ell_{i-1}(v)+2}\right) \\ &\quad \cdot w(v) = \left(1 - 0.5^{(\ell_{i-1}(v)+1)+1}\right) \cdot w(v) \,. \end{aligned} \tag{4}$$

□

As evident by Eq. 4, the vertex v's level may increase by at most once in each iteration.

Corollary 5 *For any iteration $i \geq 1$ and vertex v: $\ell_i(v) \leq \ell_{i-1}(v) + 1$.*

We note that the change to the algorithm does not affect the correctness of claims 2, 3, 4 and Lemma 1. Lemma 2 changes slightly, as there can now be twice as many v-stuck iterations:

Lemma 4 *For every vertex v and level $\ell(v)$, the number of v-stuck iterations is bounded by 2α.*

Proof Notice that when v reached the level $\ell(v)$, we had $\sum_{e \in E(v)} \delta(e) \geq w(v)(1 - 0.5^{\ell(v)})$. The number of v-stuck iterations is then bounded by the number of times it can send a "stuck" message without reaching $\sum_{e \in E(v)} \delta(e) > w(v)(1 - 0.5^{\ell(v)+1})$. Indeed, once this inequality holds, the level of v is incremented. Every stuck iteration implies, by Line 3e, that $\sum_{e \in E'(v)} \text{bid}(e) > \frac{1}{\alpha} \cdot 0.5^{\ell(v)+1} \cdot w(v)$. Therefore, we can bound the number of iteration by $\frac{w(v)(1-0.5^{\ell(v)+1}) - w(v)(1-0.5^{\ell(v)})}{\frac{1}{\alpha} \cdot 0.5^{\ell(v)+1} \cdot w(v)/2} = 2\alpha$. □

We conclude that the algorithm remains correct and its asymptotic complexity does not change.

References

1. Astrand, M., Floréen, P., Polishchuk, V., Rybicki, J., Suomela, J., Uitto, J.: A local 2-approximation algorithm for the vertex cover problem. In: DISC (2009)
2. Astrand, M., Suomela, J.: Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In: SPAA (2010)

3. Bar-Yehuda, R., Censor-Hillel, K., Ghaffari, M., Schwartzman, G.: Distributed approximation of maximum independent set and maximum matching. In: ACM, PODC (2017)
4. Bar-Yehuda, R., Censor-Hillel, K., Schwartzman, G.: A distributed $(2+\varepsilon)$- approximation for vertex cover in $o(log\Delta/\varepsilon loglog\Delta)$ rounds. J. ACM (2017)
5. Bar-Yehuda, R., Censor-Hillel, K., Schwartzman, G.: A distributed $(2+\varepsilon)$- approximation for vertex cover in $o(log\Delta/\varepsilon loglog\Delta)$ rounds. J. ACM (2017)
6. Ben-Basat, R., Even, G., Kawarabayashi, K.-I., Schwartzman, G.: A Deterministic distributed 2-approximation for weighted vertex cover in $O(\log n \log \Delta / \log^2 \log \Delta)$ rounds. In: SIROCCO (2018)
7. Chang, Y., Kopelowitz, T., Pettie, S.: An exponential separation between randomized and deterministic complexity in the LOCAL model. In: FOCS (2016)
8. Cole, R., Vishkin, U.: Deterministic coin tossing with applications to optimal parallel list ranking. Inf Control (1986)
9. Even, G., Ghaffari, M., Medina, M.: Distributed set cover approximation: primal-dual with optimal locality. In: DISC (2018)
10. Ghaffari, M.: An improved distributed algorithm for maximal independent set. Soc. Ind. Appl. Math. SODA (2016)
11. Ghaffari, M., Su, H.: Distributed degree splitting, edge coloring, and orientations. In: SODA (2017)
12. Grandoni, F., Könemann, J., Panconesi, A.: Distributed weighted vertex cover via maximal matchings. ACM Trans. Algorithms (2008)
13. Hochbaum, D.S.: Approximation algorithms for the set covering and vertex cover problems. SIAM J. Comput. (1982)
14. Karp, R.M.: Reducibility among combinatorial problems. In: Proceedings of a symposium on the Complexity of Computer Computations (1972)
15. Khuller, S., Vishkin, U., Young, N.E.: A primal-dual parallel approximation technique applied to weighted set and vertex covers. J. Algorithms (1994)
16. Koufogiannakis, C., Young, N.E.: Distributed algorithms for covering, packing and maximum weighted matching. Distrib. Comput. (2011)
17. Kuhn, F.: The price of locality: exploring the complexity of distributed coordination primitives. PhD thesis, ETH Zurich (2005)
18. Kuhn, F., Moscibroda, T., Wattenhofer, R.: The price of being near-sighted. In: SODA (2006)
19. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local computation: Lower and upper bounds. J. ACM (2016)
20. Panconesi, A., Rizzi, R.: Some simple distributed algorithms for sparse networks. Distrib. Comput. (2001)
21. Polishchuk, V., Suomela, J.: A simple local 3-approximation algorithm for vertex cover. Inf. Process. Lett. (2009)