


# A Simple Distributed Algorithm for Sparse Fractional Covering and Packing Problems

Qian Li  

Shenzhen International Center For Industrial And Applied Mathematics,  
Shenzhen Research Institute of Big Data, China

Minghui Ouyang  

School of Mathematical Sciences, Peking University, Beijing, China

Yuyi Wang 

Lambda Lab, China Railway Rolling Stock Corporation Zhuzhou Institute, China

---

## Abstract

This paper presents a distributed algorithm in the CONGEST model that achieves a  $(1 + \epsilon)$ -approximation for row-sparse fractional covering problems (RS-FCP) and the dual column-sparse fraction packing problems (CS-FPP). Compared with the best-known  $(1 + \epsilon)$ -approximation CONGEST algorithm for RS-FCP/CS-FPP developed by Kuhn, Moscibroda, and Wattenhofer (SODA'06), our algorithm is not only much simpler but also significantly improves the dependency on  $\epsilon$ .

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** CONGEST model, row-sparse fractional covering, column-sparse fractional packing, positive linear programming, simple algorithms

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2024.45

**Related Version** *Full Version*: <https://arxiv.org/abs/2409.16168>

**Funding** Qian Li's work is supported by the National Key R&D Program of China under Grant 2022YFA1003900, the Hetao Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone Project (No.HZQSW-S-KCCYB-2024016), and the National Natural Science Foundation of China Grants No.62002229. Minghui Ouyang's work is supported in part by the National Natural Science Foundation of China Grants No.62002229. Yuyi Wang's work is supported by the Natural Science Foundation of Hunan Province 2024JJ5128.

## 1 Introduction

A *fractional covering* problem (FCP) and its dual *fractional packing* problem (FPP) are positive linear programs (LP) of the canonical form:

$$\begin{array}{ll} \min_{\mathbf{x}} & \mathbf{c}^T \mathbf{x} \quad (\text{FCP, the primal LP}) \\ \text{s.t.} & \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array} \quad \begin{array}{ll} \max_{\mathbf{y}} & \mathbf{b}^T \mathbf{y} \quad (\text{FPP, the dual LP}) \\ \text{s.t.} & \mathbf{A}^T \mathbf{y} \leq \mathbf{c} \\ & \mathbf{y} \geq 0, \end{array}$$

where all  $A_{ij}$ ,  $b_i$ , and  $c_j$  are non-negative. This paper particularly focuses on  $k$ -row-sparse FCPs ( $k$ -RS-FCP) and  $k$ -column-sparse FPPs ( $k$ -CS-FPP). These are FCPs and FPPs in which the matrix  $\mathbf{A}$  contains at most  $k$  non-zero entries per row. They are still fairly general problems and can model a broad class of basic problems in combinatorial optimization, such as the fractional version of vertex cover, bounded-frequency weighted set cover, weighted  $k$ -uniform hypergraph matching, stochastic matching, and stochastic  $k$ -set packing.

This paper studies distributed algorithms for FCPs and FPPs in the CONGEST model. The CONGEST model features a network  $G = (V, E)$ , where each node corresponds to a processor and each edge  $(u, v)$  represents a bidirectional communication channel between



© Qian Li, Minghui Ouyang, and Yuyi Wang;

licensed under Creative Commons License CC-BY 4.0

35th International Symposium on Algorithms and Computation (ISAAC 2024).

Editors: Julián Mestre and Anthony Wirth; Article No. 45; pp. 45:1–45:8

Leibniz International Proceedings in Informatics



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

processors  $u$  and  $v$ . The computation proceeds in rounds. In one round, each processor first executes local computations and then sends messages to its neighbors. Each of the messages is restricted to  $O(\log |V|)$  bits. The algorithm complexity is measured in the number of rounds it performs.

For an FCP/FPP instance with  $m$  primal variables and  $n$  dual variables (i.e.,  $A$  has dimensions  $n \times m$ ), the network is a bipartite graph  $G = ([m], [n], E)$ . Each primal variable  $x_j$  is associated with a left node  $j \in [m]$ , and each dual variable  $y_i$  is associated with a right node  $i \in [n]$ . An edge  $(i, j)$  exists if and only if  $A_{ij} > 0$ . At the beginning of a CONGEST algorithm, each left node  $j$  only knows its corresponding cost  $c_j$  and the column vector  $(A_{ij} : i \in [n])$ , and each right node  $i$  only knows  $b_i$  and the row vector  $(A_{ij} : j \in [m])$ . At the end of the algorithm, each left node  $j$  is required to output a number  $\hat{x}_j$  and each right node  $i$  to output a  $\hat{y}_i$ , which together are supposed to form approximate solutions to the FCP and FPP respectively.

Let  $\mathbf{1}_k$  denote the  $k$ -dimensional all-ones vector. The subscript  $k$  will be dropped if it is implicit. Without loss of generality, this paper considers FCP and FPP instances of the following normal forms:

$$\min_{\mathbf{x}} \quad \mathbf{1}^T \mathbf{x} \quad \text{s.t.} \quad \mathbf{A} \mathbf{x} \geq \mathbf{1} \text{ and } \mathbf{x} \geq 0 \quad (1)$$

and

$$\max_{\mathbf{y}} \quad \mathbf{1}^T \mathbf{y} \quad \text{s.t.} \quad \mathbf{A}^T \mathbf{y} \leq \mathbf{1} \text{ and } \mathbf{y} \geq 0 \quad (2)$$

where  $A_{ij}$  is either 0 or  $\geq 1$ . The reduction to the normal form proceeds as follows:

- First, we can assume that each  $b_i > 0$  since otherwise we can set  $y_i$  to zero and delete the  $i$ -th row of  $A$ ;  
similarly, we can assume each  $c_j > 0$  since otherwise we can set  $x_j$  to  $+\infty$  and delete the  $j$ -th column of  $A$ , and then set  $y_i$  to zero and delete the  $i$ -th row for all  $i$  with  $A_{ij} > 0$ .
- Then, we can replace  $A_{ij}$  by  $\hat{A}_{ij} = \frac{A_{ij}}{b_i c_j}$ , replace  $\mathbf{b}$  and  $\mathbf{c}$  by all-ones vector, and work with variables  $\hat{x}_j = c_j x_j$  and  $\hat{y}_i = b_i y_i$ .
- Finally, we replace  $\hat{A}_{ij}$  with  $\tilde{A}_{ij} = \frac{\hat{A}_{ij}}{\min\{\hat{A}_{i'j'} \mid \hat{A}_{i'j'} > 0\}}$  and work with  $\tilde{x}_j = \hat{x}_j \cdot \min\{\hat{A}_{i'j'} \mid \hat{A}_{i'j'} > 0\}$  and  $\tilde{y}_i = \hat{y}_i \cdot \min\{\hat{A}_{i'j'} \mid \hat{A}_{i'j'} > 0\}$ .

Papadimitriou and Yannakakis [12] initiated the research on approximating FCPs/FPPs in the CONGEST model. Bartal, Byers, and Raz [3] proposed the first constant approximation ratio algorithm with  $\text{polylog}(m+n)$  rounds. After designing a distributed algorithm for a specific FCP/FPP scenario, namely the fractional dominating set problem [10], Kuhn, Moscibroda, and Wattenhofer [8] finally developed an efficient  $(1+\epsilon)$ -approximation algorithm for general FCP/FPP instances, running in  $O(\log \Gamma_p \cdot \log \Gamma_d / \epsilon^4)$  round for normalized instances where

$$\Gamma_p := \max_j \sum_{i=1}^n A_{ij}, \text{ and } \Gamma_d := \max_i \sum_{j=1}^m A_{ij}.$$

Particularly, for RS-FCPs/CS-FPPs, the round complexity becomes  $O(\log A_{\max} \cdot \log \Gamma_p / \epsilon^4)$ . Later, Awerbuch and Khandekar [2] proposed another  $(1+\epsilon)$ -approximation algorithm for general normalized FCP/FPP instances running in  $\tilde{O}(\log^2(n A_{\max}) \log^2(n m A_{\max}) / \epsilon^5)$  rounds, which has worse bound than [8] but enjoys the features of simplicity and statelessness.

Several works studied the lower bound for CONGEST algorithms to approximate linear programming. Bartal, Byers, and Raz [3] showed that  $(1+\epsilon)$ -approximation algorithms for general FCPs/FPPs require at least  $\Omega(1/\epsilon)$  rounds. Kuhn, Moscibroda, and Wattenhofer

proved that [5, 6, 7, 8, 11] no constant round, constant-factor approximation CONGEST algorithms exist for the LP relaxation of minimum vertex cover, minimum dominating set, or maximum matching in general graphs. Later, they improved this lower bound by showing that [9] no  $o(\sqrt{\log(m+n)}/\log\log(m+n))$  rounds CONGEST algorithms can constant-factor approximate the LP relaxation of minimum vertex cover, maximum matching, or by extension, the general binary RS-FCPs or CS-FPPs (i.e.,  $A_{ij} \in \{0, 1\}$ ) as well.

In this paper, we propose a CONGEST algorithm (Algorithm 1) for approximating general RS-FCP/CS-FPP instances, which is much simpler than the algorithm of [8]. Moreover, our algorithm exhibits a worse dependency on  $A_{\max}$  but improves the dependency on  $\epsilon$ . In particular, for the binary RS-FCPs or CS-FPPs, which include LP relaxations of many combinatorial problems such as minimum vertex cover, minimum dominating set, maximum matching, and maximum independent set, our algorithm runs in  $O(\log \Gamma_p/\epsilon^2)$  rounds, which is a  $1/\epsilon^2$  factor improvement over the algorithm of [8].

► **Theorem 1 (Main Theorem).** *For any  $\epsilon > 0$ , Algorithm 1 computes  $(1 + \epsilon)$ -approximate solutions to RS-FCP and CS-FPP at the same time, running in  $O(A_{\max} \cdot \log \Gamma_p/\epsilon^2)$  rounds.*

► **Remark 2.** In 2018, Ahmadi et al. [1] proposed a simple  $(1 + \epsilon)$ -approximation distributed algorithm for the LP relaxation of minimum vertex cover and maximum weighted matching problems, which are special classes of 2-RS-FCPs and 2-CS-FPPs. They claimed the algorithm runs in  $O(\log \Gamma_p/\epsilon^2)$  rounds, an  $O(A_{\max})$  factor faster than our algorithm. Unfortunately, there is a flaw in their proof<sup>1</sup>, and we do not know how to correct the proof to achieve the claimed bound.

## 2 Algorithms

In this section, we present our algorithm (Algorithm 1) and the analysis. Indeed, our algorithm applies to general FCP/FPP instances, and we will prove the following theorem:

► **Theorem 3.** *For any  $\epsilon > 0$ , Algorithm 1 computes  $(1 + \epsilon)$ -approximate solutions to (1) and (2) at the same time, running in  $O(\Gamma_d \cdot \log \Gamma_p/\epsilon^2)$  rounds.*

Our algorithm is based on the sequential fractional set cover algorithm by Eisenbrand et al. [4] and the fractional weighted bipartite matching by Ahmadi et al. [1]. It will be helpful to view (1) as a generalization of the fractional set cover problem. Specifically, there is a universe  $U = \{e_1, \dots, e_n\}$  of  $n$  elements, a collection  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  of subsets of  $U$ , and a matrix  $\{A_{eS} : e \in U, S \in \mathcal{S}\}$  indicating the covering efficiency of  $S$  on  $e$ . We say  $e \in S$  if  $A_{eS} > 0$ . Then (1) can be recast as the following generalization of the fractional set cover problem:

$$\min_{\mathbf{x} \geq 0} \sum_{S \in \mathcal{S}} x_S \quad \text{s.t.} \quad \sum_{S \ni e} A_{eS} \cdot x_S \geq 1 \text{ for any } e. \quad (3)$$

The dual (2) can be recast as:

$$\max_{\mathbf{y} \geq 0} \sum_{e \in U} y_e \quad \text{s.t.} \quad \sum_{e \in S} A_{eS} \cdot y_e \leq 1 \text{ for any } S. \quad (4)$$

<sup>1</sup> In the proof of Lemma 5.2 in the full version,  $Y_v^+$  should be defined as  $y_e^+/w_e$  rather than  $y_e^+$ ;  $\frac{\alpha^{1/w_e}}{\alpha^{1/w_e}-1} \leq \frac{\alpha}{\alpha-1}$  seems doubtful since  $\frac{1}{w_e} > 1$  in their setting.

■ **Algorithm 1** An  $(1 + \epsilon)$ -approximation algorithm for the normalized FCP/FPP.

---

```

1 Parameter:  $\alpha, f \in \mathbb{R}_{\geq 0}$  and  $L \in \mathbb{N}$  defined as in (5);
2 Initialize  $x_S := 0$  for any  $S$ , and  $y_e := 0$  and  $r_e := 1$  for any  $e$ ;
3 for  $\ell = 1$  to  $L$  do
4   for each all  $S$  in parallel do
5     if  $\rho_S \geq \frac{1}{\alpha} \cdot \max_{S' \cap S \neq \emptyset} \rho_{S'}$  then
6        $x_S := x_S + 1$ ;
7       for all  $e \in S$  do
8          $y_e := y_e + A_{eS} \cdot r_e / (\sum_{e \in S} A_{eS} \cdot r_e)$  and  $r_e := r_e / \alpha^{A_{eS}}$ ;
9         if  $r_e \leq \alpha^{-f}$  then  $r_e := 0$ ;
10 Return:  $\mathbf{x}/f$  and  $\mathbf{y}/(f \cdot (1 + \epsilon))$  as the approximate solutions to (3) and (4)
    respectively.

```

---

Our algorithm maintains a variable  $x_S$  for each subset  $S$ , and two variables  $y_e$  and  $r_e$  for each element  $e$ .  $x_S$  and  $y_e$  are initially 0 and their values can only increase throughout the algorithm; the variable  $r_e$  is initially 1 and its value can only decrease. Intuitively,  $r_e$  denotes the “requirement” of element  $e$ . Furthermore, we define the efficiency of  $S$  as  $\rho_S := \sum_{e \in S} A_{eS} \cdot r_e$ .

Our algorithm consists of  $L$  phases.  $\alpha$  and  $f$  are two other algorithmic parameters. The values of  $L, \alpha, f$  will be determined later. In the  $\ell$ -th phase, the algorithm picks all subsets  $S$  with

$$\rho_S \geq \frac{1}{\alpha} \cdot \max_{S' \cap S \neq \emptyset} \rho_{S'}.$$

and update the primal variable

$$x_S := x_S + 1,$$

as well as: for each  $e \in S$ ,

$$y_e := y_e + \frac{A_{eS} \cdot r_e}{\sum_{e \in S} A_{eS} \cdot r_e}, \text{ and } r_e := r_e / \alpha^{A_{eS}}.$$

In other words, let  $\Xi_\ell(e) := \{S \ni e \mid S \text{ is selected in the } \ell\text{-th phase}\}$ , then after the  $\ell$ -th phase, we have

$$y_e := y_e + \Delta y_e = y_e + \sum_{S \in \Xi_\ell(e)} \frac{A_{eS} \cdot r_e}{\sum_{e \in S} A_{eS} \cdot r_e}, \text{ and } r_e := r_e / \alpha^{\sum_{S \in \Xi_\ell(e)} A_{eS}}.$$

Besides, we set  $r_e = 0$  as soon as  $r_e \leq \alpha^{-f}$ . Finally, the algorithm returns  $\mathbf{x}/f$  and  $\mathbf{y}/((1 + \epsilon) \cdot f)$  as the approximation solutions. See Algorithm 1 for a formal description.

► **Remark 4.** The two algorithms in [8] and [1] both have a similar greedy fashion: it starts with all  $x_S$  set to 0, always increases the  $x_S$  whose “efficiency” is maximum up to a certain factor, and then distributes the increment of  $x_S$  among its elements and decreases the requirements  $r_e$ . Our algorithm and the two algorithms of [8] and [1] differ in specific implementations: the definition of efficiency, the distribution of increments, and the reduction of requirements. In particular, the algorithm of [8] consists of two levels of loops: the goal of the first-level loop is to reduce the maximum “weighted primal degree”, and one complete

run of the second-level loop can be seen as one parallel greedy step. This two-level structure complicates the algorithm of [8]. The algorithm of [1] only works for the LP relaxation of minimum vertex cover and maximum weighted matching problems, which are special classes of 2-RS-FCPs and 2-CS-FPPs. The distribution of increments and the reduction of requirements in our implementation is similar to [1], and the main difference is the definition of efficiency.

Before analyzing the correctness and efficiency of the algorithm, we present some helpful observations about its behavior.

► **Proposition 5.** *Throughout the algorithm, we always have*

- (a)  $\sum_S x_S = \sum_e y_e$ .
- (b) For any  $S$ , the value of  $\rho_S$  is non-increasing, and lies within  $(\alpha^{-f}, \Gamma_p] \cup \{0\}$ .
- (c) After each phase,  $\max_S \rho_S$  decreases by a factor of at least  $\alpha$ .

**Proof.**

**Part (a).** Initially,  $\sum_S x_S = \sum_e y_e = 0$ . Then, whenever we increase  $x_S$  by 1, we increase  $\sum_e y_e$  by  $\sum_{e \in S} \frac{A_{eS} \cdot r_e}{\sum_{e \in S} A_{eS} \cdot r_e} = 1$ .

**Part (b).** Since  $r_e$  is non-increasing, so is  $\rho_S := \sum_{e \in S} A_{eS} \cdot r_e$ . Besides, the initial value of  $\rho_S$  is  $\sum_{e \in S} A_{eS}$ , which is upper-bounded by  $\Gamma_p$ . Furthermore, for any non-zero  $r_e$  where  $e \in S$ , it should be strictly greater than  $\alpha^{-f}$ , since otherwise it will be set to 0. Recalling that  $A_{eS} \geq 1$ , we have  $\rho_S > \alpha^{-f}$  or  $\rho_S = 0$ .

**Part (c).** Define  $\rho_{\max} := \max_S \rho_S$ . Note that every  $S$  with  $\rho_S \geq \rho_{\max}/\alpha$  will be picked. Then for any such  $S$ ,  $r_e$  will decrease by a factor of  $\alpha^{\sum_{S' \in \Xi_\ell(e)} A_{eS}} \geq \alpha^{A_{eS}} \geq \alpha$  for any  $e \in S$ , so  $\rho_S = \sum_{e \in S} A_{eS} \cdot r_e$  decreases by a factor of at least  $\alpha$  as well. ◀

By Proposition 5 (b) and (c), it is easy to see that after  $\lceil \log_\alpha \Gamma_p + f \rceil$  phases, all  $\rho_S$  and  $r_e$  will become zero. We choose <sup>2</sup>

$$\alpha := 1 + \frac{\epsilon}{c \cdot \Gamma_d}, \text{ and } f := \frac{2}{\epsilon \cdot \ln \alpha} \cdot \ln \Gamma_p, \text{ and } L := \lceil \log_\alpha \Gamma_p + f \rceil. \quad (5)$$

where  $c$  is a sufficiently large constant. Note that each phase can be implemented in constant rounds. So the following lemma holds.

► **Lemma 6.** *Algorithm 1 runs in  $O(\Gamma_d \cdot \log \Gamma_p / \epsilon^2)$  rounds. When it terminates, all  $r_e = 0$  and all  $\rho_S = 0$ .*

What remains is to prove its correctness. Let  $\mathbf{x}^L$  and  $\mathbf{y}^L$  denote the values of  $\mathbf{x}$  and  $\mathbf{y}$  right after the  $L$ -th phase. We first prove the feasibility.

► **Theorem 7.**  *$\mathbf{x}^L/f$  and  $\mathbf{y}^L/((1 + \epsilon) \cdot f)$  are feasible solutions to (3) and (4) respectively.*

**Proof.** We first show the feasibility of  $\mathbf{x}^L/f$ . Obviously,  $\mathbf{x}^L/f$  are non-negative. Given any  $e$ , whenever we increase  $x_S$  by 1 for some  $S \ni e$ , we divide  $r_e$  by a factor  $\alpha^{A_{eS}}$ . The

<sup>2</sup> The reason behind the choices of parameters is that  $\alpha$  should be sufficiently close to 1 and  $f$  should be sufficiently large, such that  $\alpha^{\Gamma_d+1} = 1 + O(\epsilon)$  and  $\frac{\ln \Gamma_p}{\ln \alpha} \ll f$ . See the proof of Theorem 7 for details.

## 45:6 Distributed Algorithm for Sparse Fractional Covering/Packing

initial value of  $r_e$  is 1; and by Lemma 6, finally  $r_e$  becomes  $\leq \alpha^{-f}$ , and then is set to 0. We therefore have

$$\sum_{S \ni e} A_{eS} \cdot x_S^L \geq f,$$

and then conclude the feasibility of  $\mathbf{x}^L/f$ .

In the following, we prove the feasibility of  $\mathbf{y}^L/f$ . Obviously,  $\mathbf{y}^L/((1+\epsilon) \cdot f)$  are non-negative. What remains is to show that for any  $S$

$$\sum_{e \in S} A_{eS} \cdot y_e^L \leq (1+\epsilon) \cdot f.$$

For the convenience of presentation, define  $Y_S := \sum_{e \in S} A_{eS} \cdot y_e$ , which only increases during the algorithm's execution. The idea is to upper bound the increment  $\Delta Y_S$  of  $Y_S$  in terms of the decrement  $\Delta \rho_S$  of  $\rho_S$  in each phase.

On the one hand, recalling that the increment of  $y_e$  is

$$\Delta y_e = \sum_{S' \in \Xi_\ell(e)} \frac{A_{eS'} \cdot r_e}{\sum_{e \in S'} A_{eS'} \cdot r_e} = \sum_{S' \in \Xi_\ell(e)} \frac{1}{\rho_{S'}} \cdot A_{eS'} \cdot r_e,$$

we have

$$\begin{aligned} \Delta Y_S &= \sum_{e \in S} A_{eS} \cdot \Delta y_e = \sum_{e \in S} \sum_{S' \in \Xi_\ell(e)} \frac{1}{\rho_{S'}} \cdot A_{eS} \cdot A_{eS'} \cdot r_e \leq \sum_{e \in S} \sum_{S' \in \Xi_\ell(e)} \frac{\alpha}{\rho_S} \cdot A_{eS} \cdot A_{eS'} \cdot r_e \\ &= \frac{\alpha}{\rho_S} \sum_{e \in S} \sum_{S' \in \Xi_\ell(e)} A_{eS} \cdot A_{eS'} \cdot r_e. \end{aligned}$$

On the other hand,

$$\begin{aligned} \Delta \rho_S &= \sum_{e \in S} A_{eS} \cdot \Delta r_e = \sum_{e \in S} A_{eS} \cdot r_e \cdot \left( 1 - 1/\alpha^{\sum_{S' \in \Xi_\ell(e)} A_{eS'}} \right) \\ &= \sum_{e \in S} A_{eS} \cdot r_e \cdot \frac{1}{\alpha^{\sum_{S' \in \Xi_\ell(e)} A_{eS'}}} \left( \alpha^{\sum_{S' \in \Xi_\ell(e)} A_{eS'}} - 1 \right) \\ &\geq \sum_{e \in S} A_{eS} \cdot r_e \cdot \frac{1}{\alpha^{\sum_{S' \in \Xi_\ell(e)} A_{eS'}}} \left( \sum_{S' \in \Xi_\ell(e)} \ln \alpha \cdot A_{eS'} \right) \\ &\geq \frac{\ln \alpha}{\alpha^{\Gamma_d}} \sum_{e \in S} \sum_{S' \in \Xi_\ell(e)} A_{eS} \cdot A_{eS'} \cdot r_e \end{aligned}$$

Combining the two inequalities above, we get

$$\Delta Y_S \leq \frac{\alpha^{\Gamma_d+1}}{\ln \alpha} \cdot \frac{\Delta \rho_S}{\rho_S}.$$

Then, summing this inequality up over all phases, we have

$$\begin{aligned} Y_S^{\text{end}} &= \sum_{\text{each phase}} \Delta Y_S \leq \frac{\alpha^{\Gamma_d+1}}{\ln \alpha} \sum_{\text{each phase}} \frac{\Delta \rho_S}{\rho_S} \leq \frac{\alpha^{\Gamma_d+1}}{\ln \alpha} \int_{\rho_S^{\text{end}}}^{\rho_S^{\text{initial}}} \frac{1}{\rho} d\rho \\ &= \frac{\alpha^{\Gamma_d+1}}{\ln \alpha} \cdot (\ln \rho_S^{\text{initial}} - \ln \rho_S^{\text{end}}) \leq \frac{\alpha^{\Gamma_d+1}}{\ln \alpha} \cdot (\ln \alpha \cdot f + \ln \Gamma_p) \\ &= \alpha^{\Gamma_d+1} f + \frac{\alpha^{\Gamma_d+1} \ln \Gamma_p}{\ln \alpha} \leq (1+\epsilon) f. \end{aligned}$$

◀

By Proposition 5 (a), we have  $\sum_S x_S^L = \sum_e y_e^L$ , which means

$$\sum_e y_e^L / ((1 + \epsilon) \cdot f) \leq (1 + \epsilon) \sum_S x_S^L / f.$$

We therefore conclude that  $\mathbf{x}^L/f$  and  $\mathbf{y}^L/((1 + \epsilon) \cdot f)$  are  $(1 + \epsilon)$ -approximate solutions to (3) and (4) respectively. By putting it and Lemma 6 together, we finish the proof of Theorem 3.

### 3 Conclusion

This paper proposes a simple  $(1 + \epsilon)$ -approximation CONGEST algorithm for row-sparse fractional covering problems and column-sparse fractional packing problems. It runs in  $O(A_{\max} \cdot \log \Gamma_p / \epsilon^2)$  rounds, where  $\Gamma_p = \max_j \sum_i A_{ij}$  and  $\Gamma_d = \max_i \sum_j A_{ij}$ . Our algorithm is simpler than the algorithm of [8], worsens the  $A_{\max}$ -dependency, but improves the  $\epsilon$ -dependency. For future work, it is an intriguing open problem, proposed by Suomela [13], whether constant round, constant-factor approximation CONGEST algorithms exist for row-sparse, column-sparse FCP/FPP instances – a special kind of RS-FCP/CS-FPP where the number of nonzero entries in each column of  $A$  is also bounded. Our algorithm and the algorithm of [8] are both such algorithms for instances where  $A_{\max}$  is bounded.

---

#### References

- 1 Mohamad Ahmadi, Fabian Kuhn, and Rotem Oshman. Distributed approximate maximum matching in the CONGEST model. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPICs*, pages 6:1–6:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.DISC.2018.6.
- 2 Baruch Awerbuch and Rohit Khandekar. Stateless distributed gradient descent for positive linear programs. *SIAM J. Comput.*, 38(6):2468–2486, 2009. doi:10.1137/080717651.
- 3 Yair Bartal, John W. Byers, and Danny Raz. Global optimization using local information with applications to flow control. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 303–312. IEEE Computer Society, 1997. doi:10.1109/SFCS.1997.646119.
- 4 Friedrich Eisenbrand, Stefan Funke, Naveen Garg, and Jochen Könemann. A combinatorial algorithm for computing a maximum independent set in a  $t$ -perfect graph. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, pages 517–522. ACM/SIAM, 2003. URL: <http://dl.acm.org/citation.cfm?id=644108.644194>.
- 5 Fabian Kuhn. *The price of locality: exploring the complexity of distributed coordination primitives*. PhD thesis, ETH Zurich, 2005. URL: <https://d-nb.info/977273725>.
- 6 Fabian Kuhn. Local approximation of covering and packing problems. In *Encyclopedia of Algorithms*, pages 1129–1132. Springer New York, 2016. doi:10.1007/978-1-4939-2864-4\_209.
- 7 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In Soma Chaudhuri and Shay Kutten, editors, *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 300–309. ACM, 2004. doi:10.1145/1011767.1011811.
- 8 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 980–989. ACM Press, 2006. URL: <http://dl.acm.org/citation.cfm?id=1109557.1109666>.

- 9 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.
- 10 Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. *Distributed Comput.*, 17(4):303–310, 2005. doi:10.1007/S00446-004-0112-5.
- 11 Christoph Lenzen and Roger Wattenhofer. Minimum dominating set approximation in graphs of bounded arboricity. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 510–524. Springer, 2010. doi:10.1007/978-3-642-15763-9\_48.
- 12 Christos H. Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 121–129. ACM, 1993. doi:10.1145/167088.167127.
- 13 Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, 2013. doi:10.1145/2431211.2431223.