Distributed Combinatorial Optimization – Extended Abstract

Fabian Kuhn, Roger Wattenhofer
{kuhn,wattenhofer}@inf.ethz.ch
Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland

Abstract

Approximating integer linear programs by solving a relaxation to a linear program (LP) and afterwards reconstructing an integer solution from the fractional one is a standard technique in a non-distributed scenario. Surprisingly, the method has not often been applied for distributed algorithms. In this paper, we show that LP relaxation is a powerful technique also to obtain fast distributed approximation algorithms. We present a novel deterministic distributed algorithm which computes a constant factor approximates for fractional covering and packing problems in only $\log^2$ rounds, using messages of logarithmic size. If messages are allowed to be larger, we show that a constant approximation can be achieved in a logarithmic number of rounds only. Finally, we show that by combining our LP approximation algorithms with randomized rounding techniques, we obtain efficient distributed approximation algorithms for a number of combinatorial problems.

1 Introduction

Achieving a global goal based on local information only is one of the key challenges when developing fast distributed algorithms. In k rounds of communication, a network node can only gather information about nodes which are at most k hops away. Not surprisingly, many global criteria such as obtaining a spanning tree cannot be met by a local algorithm, i.e. by an algorithm whose time complexity is much smaller than the diameter of the network graph. However, for many interesting problems, there are local algorithms. Probably the most prominent example is an algorithm for 3-coloring a rooted tree in $\mathrm{O}(\log^* n)$ rounds only [3, 11].

In the present paper, we look at distributed approximation algorithms for optimization problems with a global objective function. We consider trade-offs between time complexity and approximation ratio. In particular, we are interested in the approximation ratios we can achieve using algorithms which are *extremely* fast. What can we achieve in a polylogarithmic or even in a constant number of rounds? In view of recent trends in networking, focusing on locality rather than on the quality of the solution has become increasingly important. Peer-to-peer networks and mobile ad-hoc networks have become popular research areas. Both share the property that their structures are changing rapidly, calling for algorithms which are fast enough to keep up with this dynamic behavior. In peer-to-peer networks, changes of the topology occur when nodes join and leave whereas in ad-hoc networks, the mobility of nodes is the main reason for the dynamic behavior. In ad-hoc networks, the scarceness of the resources energy and bandwidth adds yet another need for low message and time complexities.

For standard, non-distributed scenarios, LP relaxations are a widely used method to obtain approximation algorithms for combinatorial optimization problems. Surprisingly, in a distributed setting, LP relaxation has not found many applications. The only example we are aware of is a distributed dominating set algorithm which approximates the LP (fractional dominating set) and finds an integer solution using randomized rounding [10]. In this paper, we generalize and improve the results of [10], showing that LP relaxation combined with randomized rounding can be an efficient means to break symmetries. The main contribution of the paper is a novel distributed algorithm to approximate the LPs corresponding to fractional covering and packing problems, a.k.a. positive LPs. The algorithm is deterministic and achieves an $(1+\varepsilon)$-approximation

in $\mathrm{O}\big(\log(\rho\Delta_d)\log(\rho\Delta_p)/\varepsilon^4\big)$ rounds where Δ_p and Δ_d are the maximum number of times a variable occurs in the inequalities of the primal and the dual LP, respectively, and where ρ denotes the ratio between the largest and the smallest non-zero coefficient of the LP. For large (i.e. constant) ε this is at least by a factor $\log(\rho m)$ faster than a previously developed algorithm which approximates the same class of LPs in time $\mathrm{O}\big(\log^2(\rho m)\log(\rho mn/\varepsilon)/\varepsilon^3\big)$ [1]. Here, m and n denote the number of primal and dual variables, respectively. Additionally, our algorithm already achieves a non-trivial approximation ratio if the number of rounds is chosen to be constant.

In a second part, we show that if message size is no issue, approximations for positive LPs can be obtained even faster. Based on a randomized graph decomposition algorithm developed by Linial and Saks [12], we achieve a constant factor approximation in $\mathrm{O}(\log m)$ communication rounds. If the number of rounds is fixed to $\mathrm{O}(k)$, the algorithm's approximation ratio is still $\mathrm{O}\big(m^{1/k}\big)$.

Finally, in the third part of the paper, we show that, combined with randomized rounding techniques, the devised LP approximation algorithms can be used to efficiently approximate a number of combinatorial problems. We show that for a number of problems our approach matches or outperforms the best known problem-specific algorithms in terms of running time and approximation quality.

The remainder of the paper is organized as follows. In Section 2, we give a summary on related work. The technical part starts with a description of the model and some notations in Section 3. Sections 4 and 5 discuss the two distributed LP approximations. They are combined with randomized rounding in Section 6 to obtain distributed approximation algorithms of integer linear programs. Finally, Section 7 concludes the paper.

2 Related Work

In the standard non-distributed setting, LP relaxation is a widely used method to approximate combinatorial problems. For an introduction, we refer to [7] or [20]. Integer covering and packing problems in particular have been studied extensively since many important optimization problems fall into this category. In this context, the technique of randomized rounding has been introduced [17, 18, 19].

As mentioned above, covering and packing problems occur in a variety of applications. It is therefore no surprise that there has been a considerable effort to find algorithms for the corresponding LPs which are faster than inner-point methods which can be applied to general LPs (e.g. [4, 5, 6, 14, 16]). All these algorithms need global information to work. The distributed approximation of positive LPs has been studied in [1], [10], and [15]. Our algorithm builds on techniques developed in [4] and [10]. The algorithms of [4], [10], and of this paper are all related to the classical greedy set cover algorithm [2]. [4] describes a (non-distributed) algorithm for the fractional set cover problem. In [10], for the fractional dominating set problem, a distributed implementation of greedy set cover is presented. In time $\mathrm{O}\big(\log^2\Delta\big)$, the algorithm achieves an approximation ratio of $\mathrm{O}(\log\Delta)$ for this special linear program.

There have been distributed approximation algorithms for a number of integer covering and packing problems such as (weighted) minimum dominating set [9, 10] or (weighted) maximum matching and vertex cover [8, 13, 21].

3 Model and Notation

In this section, we give a formal description of the model, as well as some definitions which will be used throughout the paper. A fractional covering problem is a linear program of the canonical form

$$\begin{aligned} \min \quad & \underline{c}^{\mathrm{T}}\underline{x} \\ \text{subject to} \quad & A\cdot\underline{x} \geq \underline{b} \\ & \underline{x} \geq \underline{0}. \end{aligned} \tag{LP}$$

where all a_{ij}, b_i and c_i are non-negative. The dual LP for (LP) has the form

$$\begin{aligned} \max \quad & \underline{b}^{\mathrm{T}}\underline{y} \\ \text{subject to} \quad & A^{\mathrm{T}}\cdot\underline{y} \leq \underline{c} \\ & \underline{y} \geq \underline{0}. \end{aligned} \tag{DLP}$$

and is called a fractional packing problem. Throughout the paper, we will use the term primal

LP for the minimization and dual LP for the maximization problem. The number of primal and dual variables are denoted by m and n, respectively. Let $a_{\max} := \max_{i,j}\{a_{ij}, b_i, c_i\}$ be the maximum and $a_{\min} := \min_{i,j}\{a_{ij}, b_i, c_i\} \setminus \{0\}$ be the minimum coefficient of (LP) and (DLP). $\rho := a_{\max}/a_{\min}$ is defined to be the maximum ratio between any two coefficients.

Analogously to [1, 15], we consider the following distributed setting. The linear program is bound to a network graph $G = (V, E)$. Each primal variable x_i and each dual variable y_j is associated with a node $v_i^{(p)} \in V$ and $v_j^{(d)} \in V$, respectively. There are communication links between primal and dual nodes wherever the respective variables occur in the corresponding inequality. Thus, $(v_i^{(p)}, v_j^{(d)}) \in E$ if and only if x_i occurs in the j^{th} inequality of (LP). Formally, this means that $v_i^{(p)}$ and $v_j^{(d)}$ are connected if and only if $a_{ji} > 0$. The degrees of $v_i^{(p)}$ and $v_j^{(d)}$ are called $\delta_i^{(p)}$ and $\delta_j^{(d)}$, respectively. $\Delta_p := \max_i \delta_i^{(p)}$ is called the primal degree and $\Delta_d := \max_j \delta_j^{(d)}$ is called the dual degree of the problem, i.e. Δ_p and Δ_d are the maximum number of times a primal or dual variable occurs in an inequality of (LP) or (DLP), respectively. The set of dual neighbors of $v_i^{(p)}$ is denoted by $N_i^{(p)}$, the set of primal neighbors of $v_i^{(d)}$ by $N_i^{(d)}$. Where convenient, $N_i^{(p)}$ and $N_j^{(d)}$ also denote the sets of the indices of the respective nodes.

For simplicity, we assume a purely synchronous communication model. In an asynchronous environment, the same approximation ratios and time complexities can be achieved at the cost of a higher message complexity.

4 LP Algorithm

We will now present the main result of this paper: An efficient distributed algorithm to approximate positive linear programs. For our algorithm, we need the LPs (LP) and (DLP) to be of the following special form:

$$\forall i, j \; : \; b_i = 1, \quad a_{ij} = 0 \text{ or } a_{ij} \geq 1. \qquad (1)$$

The transformation to (1) is done in two steps. First, every a_{ij} is replaced by $\hat{a}_{ij} := a_{ij}/b_i$ and b_i is replaced by 1. In the second step, the c_i and $\hat{a}_{ij}$ are divided by $\lambda_i := \min_j\{\hat{a}_{ji}\} \setminus \{0\}$. The optimal objective values of the transformed LPs are the same. A feasible solution for the transformed LP (1) can be converted to a feasible solution of the original LP by dividing all x-values by the corresponding λ_i and by dividing the y-values by the corresponding b_i. This conserves the values of the objective functions. For the rest of this section, we assume that the coefficients of the LP given are according to (1).

procedure increase_duals():

if $w_i \geq 1$ **then**
  if $f_i \geq f$ **then**
    $y_i := y_i + y_i^+$; $y_i^+ := 0$;
    $r_i := 0$; $w_i := 0$
  else if $w_i \geq 2$ **then**
    $y_i := y_i + y_i^+$; $y_i^+ := 0$;
    $r_i := r_i / \Gamma_p^{\lfloor w_i \rfloor / k_p}$
  else
    $\lambda := \max\{\Gamma_d^{1/k_d}, \Gamma_p^{1/k_p}\}$;
    $y_i := y_i + \min\{y_i^+, r_i\lambda/\Gamma_p^{e_p/k_p}\}$;
    $y_i^+ := y_i^+ - \min\{y_i^+, r_i\lambda/\Gamma_p^{e_p/k_p}\}$;
    $r_i := r_i/\Gamma_p^{1/k_p}$
  fi;
  $w_i := w_i - \lfloor w_i \rfloor$
fi

We start the description of the algorithm with a general outline. At the beginning, all primal and dual variables are set to 0. During the algorithm, primal and dual variables are gradually increased by the same total amount. Primal nodes which have a high degree and low cost c_i increase their variables first. At the end, each primal inequality is fulfilled f times and the sums of the dual constraints are only f times a small factor too high (f is to be determined later). The algorithm builds on techniques developed in [4] and [10].

Apart from the variable y_i, each dual node $v_i^{(d)}$ needs a requirement $r_i \leq 1$ which is decreased every time the corresponding primal constraint is achieved and a variable f_i which counts how many times the primal constraint has been fulfilled. To measure the efficiency per cost ratio of a primal node $v_i^{(p)}$, we

LP Approximation Algorithm for Primal Node $v_i^{(p)}$:

$x_i := 0$;
for $e_p := k_p - 2$ **to** $-f - 1$ **by** -1 **do**
  for 1 **to** h **do**
    $(* \gamma_i := \frac{c_{\max}}{c_i} \sum_j a_{ji} r_j *)$
    for $e_d := k_d - 1$ **to** 0 **by** -1 **do**
      $\tilde{\gamma}_i := \frac{c_{\max}}{c_i} \sum_j a_{ji} \tilde{r}_j$;
      if $\tilde{\gamma}_i \geq 1/\Gamma_p^{e_p/k_p}$ **then**
        $x_i^+ := 1/\Gamma_d^{e_d/k_d}$; $x_i := x_i + x_i^+$;
      fi;
      send x_i^+, $\tilde{\gamma}_i$ to dual neighbors;
      receive $\tilde{r}_j$ from dual neighbors
    od;
    receive r_j from dual neighbors
  od
od;
$x_i := x_i / \min_{j \in N_i^{(p)}} \sum_\ell a_{j\ell} x_\ell$

LP Approximation Algorithm for Dual Node $v_i^{(d)}$:

$y_i := y_i^+ := w_i := f_i := 0$; $r_i := 1$;
for $e_p := k_p - 2$ **to** $-f - 1$ **by** -1 **do**
  for 1 **to** h **do**
    $\tilde{r}_i := r_i$;
    for $e_d := k_d - 1$ **to** 0 **by** -1 **do**
      receive x_j^+, $\tilde{\gamma}_j$ from primal neighbors;
      $y_i^+ := y_i^+ + \tilde{r}_i \sum_j a_{ij} x_j^+ / \tilde{\gamma}_j$;
      $w_i^+ := \sum_j a_{ij} x_j^+$;
      $w_i := w_i + w_i^+$; $f_i := f_i + w_i^+$;
      if $w_i \geq 1$ **then** $\tilde{r}_i := 0$ **fi;**
      send $\tilde{r}_i$ to primal neighbors
    od;
    increase_duals();
    send r_i to primal neighbors
  od
od;
$y_i := y_i / \max_{j \in N_i^{(d)}} \frac{1}{c_j} \sum_\ell a_{\ell j} y_\ell$

Algorithm 1: Distributed LP Approximation Algorithm

define γ_i as follows:

$$\gamma_i := \frac{c_{\max}}{c_i} \sum_j a_{ji} r_j.$$

For simplicity, we assume that all nodes know $c_{\max} := \max\{c_i\}$ as well as two other global quantities Γ_p and Γ_d which are defined as

$$\Gamma_p := \max_i \frac{c_{\max}}{c_i} \cdot \sum_{j=1}^{n} a_{ji} \text{ and } \Gamma_d := \max_i \sum_{j=1}^{m} a_{ij}.$$

We can get rid of this assumption by techniques similar to the ones used in [10]. For details, we refer to the full version of this paper. We can also assume that instead of knowing the real values the nodes have upper bounds on the values $c_{\max}$, Γ_p, and Γ_d. Time complexity and approximation ratio are then also dependent on the upper bounds instead of the real values.

The detailed algorithm is given by Algorithm 1 along with the procedure **increase_duals()** which is used by the dual nodes. The algorithm has two parameters $k_p \geq 1$ and $k_d \geq 1$ which determine the trade-off between time complexity and approximation quality. The bigger k_p and k_d, the better the approximation ratio of the algorithm. On the other hand, smaller k_p and k_d lead to a faster algorithm. Algorithm 1 also makes use of two values f and h which are defined as follows:

$$f := \left\lceil \frac{k_p + 1}{\Gamma_p^{1/k_p} - 1} \right\rceil \text{ and } h := \left\lceil 1 + \frac{k_p}{\Gamma_p^{1/k_p} \ln \Gamma_p} \right\rceil.$$

In the following, we present a number of lemmas which establish all the necessary details to analyze Algorithm 1.

Lemma 4.1. *For each primal node $v_i^{(p)}$, at all times*

during Algorithm 1,

$$\gamma_i \leq \Gamma_p^{(e_p+2)/k_p}.$$

Proof. We prove the lemma by induction over the iterations of the outer-most loop (e_p-loop). For $e_p = k_p - 2$, the lemma follows from the definitions of γ_i, Γ_p, and r_j.

To see how fast γ_i decreases, we have to look at the behavior of the inner-most loop (e_d-loop). The value γ_i is $c_{\max}/c_i$ times the the sum of the $a_{ji}r_j$ of all dual neighbors $v_j^{(d)}$ of $v_i^{(p)}$ whereas $\tilde{\gamma}_i$ is the same sum but only for the dual neighbors for which the corresponding primal inequality has not been fulfilled since the last time **increase_duals()** was called ($w_i < 1$ and $\tilde{r}_i > 0$). When **increase_duals()** is called (i.e. after the last iteration of the e_d-loop) it holds that $\tilde{\gamma}_i < \Gamma_p^{e_p/k_p}$. If not, x_i^+ would have been set to 1 in the last e_d-loop iteration ($e_d = 0$). Because all $a_{ij} \geq 1$ and all $b_j = 1$, all dual neighbors $v_j^{(d)}$ of $v_i^{(p)}$ would then have $w_j \geq 1$ and therefore $\tilde{r}_j = 0$ after the last e_d-loop iteration. Thus, if $\tilde{\gamma}_i \geq \Gamma_p^{e_p/k_p}$ before the last e_d-loop iteration, $\tilde{\gamma}_i = 0$ when **increase_duals()** is called.

All dual nodes $v_j^{(d)}$ which set $\tilde{r}_j := 0$ divide r_j by at least Γ_p^{1/k_p} while executing **increase_duals()**. Therefore, after the call to **increase_duals()**, we have

$$\gamma_i' \leq \tilde{\gamma}_i + \frac{\gamma_i - \tilde{\gamma}_i}{\Gamma_p^{1/k_p}} \leq \Gamma_p^{e_p/k_p} + \frac{\gamma_i - \Gamma_p^{e_p/k_p}}{\Gamma_p^{1/k_p}}$$

where γ_i and γ_i' denote the values before and after executing **increase_duals()**, respectively. Before going to the next e_p-loop iteration, the e_d-loop/**increase_duals()** part is executed h times. By the induction hypothesis, $\gamma_i \leq \Gamma_p^{(e_p+2)/k_p}$ before the h iterations of the inner part and therefore, after the h iterations, we have

$$\gamma_i \;\leq\; \Gamma_p^{e_p/k_p} + \frac{\Gamma_p^{(e_p+2)/k_p} - \Gamma_p^{e_p/k_p}}{\Gamma_p^{h/k_p}}. \qquad (2)$$

We have to show that the right-hand side of Inequality (2) is at most $\Gamma_p^{(e_p+1)/k_p}$. Dividing the right-hand side of (2) and $\Gamma_p^{(e_p+1)/k_p}$ by $\Gamma_p^{e_p/k_p}$, this results in

$$\left(\Gamma_p^{1/k_p} - 1\right) \cdot \left(\Gamma_p^{1/k_p} + 1\right) \leq \left(\Gamma_p^{1/k_p} - 1\right) \cdot \Gamma_p^{h/k_p}$$

which is true for

$$h \;\geq\; \frac{\ln\left(\Gamma_p^{1/k_p} + 1\right)}{\ln\left(\Gamma_p^{1/k_p}\right)}.$$

Because $\ln(x)$ is a concave function, $\ln(x+1) \leq \ln(x) + 1/x$ and therefore

$$\begin{aligned}\frac{\ln\left(\Gamma_p^{1/k_p} + 1\right)}{\ln\left(\Gamma_p^{1/k_p}\right)} \;&\leq\; 1 + \frac{1}{\Gamma_p^{1/k_p} \ln\left(\Gamma_p^{1/k_p}\right)} \\ &=\; 1 + \frac{k_p}{\Gamma_p^{1/k_p} \ln\left(\Gamma_p\right)} \;\leq\; h\end{aligned}$$

by the definition of h. □

Lemma 4.2. *After line 12 of Algorithm 1, for each dual node $v_i^{(d)}$, either $\tilde{r}_i = 0$ or*

$$w_i^+ := \sum_j a_{ij} x_j^+ \leq \Gamma_d^{1/k_d}.$$

Proof. For the sake of contradiction, assume that $\tilde{r}_i > 0$ and $w_i^+ > \Gamma_d^{1/k_d}$. Let $w_i'^+$ be the sum of the $a_{ij}x_j^+$ of the preceding iteration of the e_d-loop. All primal variables which are increased in line 8 have also been increased in the preceding iteration because the $\tilde{\gamma}_j$ of the primal neighbors of $v_i^{(d)}$ can only have decreased. Therefore

$$w_i'^+ \geq \frac{w_i^+}{\Gamma_d^{1/k_d}} > 1$$

which is a contradiction to $\tilde{r}_i > 0$ because $w_i'^+ > 1$ implies that $\tilde{r}_i$ has been set to 0 in the preceding iteration of the e_d-loop. For the first iteration of the e_d-loop ($e_d = k_d - 1$), we have

$$w_i^+ \leq \Gamma_d x_i^+ = \frac{\Gamma_d}{\Gamma_d^{(k_d-1)/k_d}}$$

which completes the proof. □

Lemma 4.3. *Each time a dual node enters **increase_duals()** in Algorithm 1,*

$$y_i^+ \leq r_i \cdot \frac{w_i}{\Gamma_p^{e_p/k_p}} \text{ and } y_i^+ \leq r_i \cdot \frac{\Gamma_d^{1/k_d} + 1}{\Gamma_p^{e_p/k_p}}. \qquad (3)$$

Proof. We start by showing the left inequality. First, we prove that the condition can only be violated inside **increase_duals()**. Outside, r_i is not changed and y_i^+ and w_i are always increased simultaneously in lines 11 and 13. Because $\tilde{r_i} \leq r_i$ and $\tilde{\gamma_j} \geq \Gamma_p^{e_p/k_p}$, these increases do not violate Condition (3). Decreasing e_p increases the right-hand side of the inequality and therefore Condition (3) is not invalidated by this either.

Inside **increase_duals()**, we consider the 3 cases $w_i < 1$, $1 \leq w_i < 2$, and $w_i \geq 2$. If $w_i < 1$, nothing happens and we are done. If $w_i \geq 2$ or $f_i \geq f$, y_i^+ is set to 0 and therefore the Condition (3) trivially holds. The interesting case is when $1 \leq w_i < 2$.

We assume that Condition (3) holds before entering **increase_duals()**. We define $w_i' := w_i - 1$ to be the fractional part of w_i. Inside **increases_duals()** $r_i\lambda/\Gamma_p^{e_p/k_p}$ is subtracted from y_i^+ and r_i is divided by Γ_p^{1/k_p}. If Condition (3) has to be true after **increase_duals()**, the following inequality must hold (r_i before the division):

$$r_i \frac{1 + w_i' - \lambda}{\Gamma_p^{e_p/k_p}} \;\leq\; \frac{r_i}{\Gamma_p^{1/k_p}} \frac{w_i'}{\Gamma_p^{e_p/k_p}}.$$

This gives

$$\begin{aligned} \lambda\Gamma_p^{1/k_p} - \Gamma_p^{1/k_p} - w_i'\Gamma_p^{1/k_p} + w_i' &\geq \\ \Gamma_p^{2/k_p} - \Gamma_p^{1/k_p} - w_i'\Gamma_p^{1/k_p} + w_i' &= \\ \left(\Gamma_p^{1/k_p} - 1\right) \cdot \left(\Gamma_p^{1/k_p} - w_i'\right) &\geq 0, \end{aligned}$$

which is true because $\lambda \geq \Gamma_p^{1/k_p} \geq 1$ and $w_i' < 1$.

To prove the right inequality of Condition (3), note that y_i^+ is only increased in the e_d-loop as long as $w_i < 1$ before entering the loop. Otherwise, $\tilde{r_i}$ would have been set to zero. The second inequality of the lemma now follows from the first inequality and from Lemma 4.2. □

Lemma 4.4. *Let $v_i^{(p)}$ be a primal node and let $Y_i := \sum_j a_{ji} y_j$ be the weighted sum of the y-values of its dual neighbors. Further, let Y_i^+ be the increase of Y_i and γ_i^- be the decrease of γ_i during an execution of **increase_duals()**. We have*

$$Y_i^+ \;\leq\; \frac{\Gamma_p^{3/k_p} \cdot \max\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\}}{\gamma_i(\Gamma_p^{1/k_p} - 1)} \cdot \frac{c_i}{c_{\max}} \cdot \gamma_i^-.$$

Proof. We prove the lemma by showing that the inequality holds for every dual neighbor $v_j^{(d)}$ of $v_i^{(p)}$. Let β_j be the increase of y_j and let r_j^- be the decrease of r_j. We show that

$$\beta_j \leq \frac{\Gamma_p^{1/k_p} \cdot \max\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\}}{\Gamma_p^{e_p/k_p}(\Gamma_p^{1/k_p} - 1)} \cdot r_j^-. \qquad (4)$$

The lemma then follows because $\gamma_i \leq \Gamma_p^{(e_p+2)/k_p}$ (Lemma 4.1) and because

$$Y_i^+ = \sum_j a_{ji}\beta_j \quad \text{and} \quad \gamma_i^- = \frac{c_{\max}}{c_i} \sum_j a_{ji} r_j^-.$$

To prove Inequality (4), we again consider the cases where $w_j \geq 2$ and where $1 \leq w_j < 2$. If $w_j \geq 2$, by Lemma 4.3, $\beta_j = y_j^+ \leq r_j(1 + \Gamma_d^{1/k_d})/\Gamma_p^{e_p/k_p}$. The requirement r_j is divided by at least Γ_p^{2/k_p} and therefore $r_j^- \geq r_j(\Gamma_p^{2/k_p} - 1)/\Gamma_p^{2/k_p}$. Together, we get

$$\begin{aligned} \beta_j &\leq \frac{1 + \Gamma_d^{1/k_d}}{\Gamma_p^{e_p/k_p}} \cdot \frac{\Gamma_p^{2/k_p}}{\Gamma_p^{2/k_p} - 1} \cdot r_j^- \\ &\leq \frac{\left(1 + \Gamma_p^{1/k_p}\right)\Gamma_p^{1/k_p} \max\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\}}{\Gamma_p^{e_p/k_p}\left(\Gamma_p^{1/k_p} + 1\right)\left(\Gamma_p^{1/k_p} - 1\right)} r_j^-. \end{aligned}$$

For $1 \leq w_j < 2$, the proof is along the same lines. Here, $\beta_j \leq r_j \max\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\}/\Gamma_p^{e_p/k_p}$ and $r_j^- = r_j(\Gamma_p^{1/k_p} - 1)/\Gamma_p^{1/k_p}$. Again, we obtain Inequality (4):

$$\beta_j \leq \frac{\max\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\}}{\Gamma_p^{e_p/k_p}} \cdot \frac{\Gamma_p^{1/k_p}}{\Gamma_p^{1/k_p} - 1} \cdot r_j^-.$$

We do not have to consider the case $f_j \geq f$ explicitly because the same analysis as for $w_j \geq 2$ applies in this case. □

Lemma 4.5. *Let $v_i^{(p)}$ be a primal node and $Y_i = \sum_j a_{ji} y_j$ be the weighted sum of the y-values of the dual neighbors of $v_i^{(p)}$. After the main part of the algorithm (i.e., after the loops at line 20),*

$$Y_i \leq \frac{c_i}{c_{\max}}(k_p + f + 1)\Gamma_p^{3/k_p} \max\left\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\right\}.$$

Proof. For simplicity, we define

$$Q := \frac{1}{c_{\max}} \Gamma_p^{3/k_p} \max\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\}.$$

Before γ_i is decreased for the last time, we have $\gamma_i \geq 1/\Gamma_p^{(f-1)/k_p}$ because at least one r_j in the dual neighborhood of $v_i^{(p)}$ has to be greater than 0. If we assume that the last time γ_i is decreased it is only reduced to $\gamma_i = 1/\Gamma_p^{(f+1)/k_p}$, Lemma 4.4 still holds. The analysis is exactly the same as for the case $w_j \geq 2$ in Lemma 4.4. By Lemma 4.4, Y_i is therefore bounded by the area under the curve $c_i Q/(\Gamma_p^{1/k_p} - 1) \cdot 1/x$ for x between $1/\Gamma_p^{(f+1)/k_p}$ and Γ_p:

$$\begin{aligned} Y_i &\leq \frac{c_i Q}{\Gamma_p^{1/k_p} - 1} \cdot \int_{\frac{1}{\Gamma_p^{(f+1)/k_p}}}^{\Gamma_p} \frac{1}{x} dx \\ &= \frac{c_i Q \ln\left(\Gamma_p^{(k_p+f+1)/k_p}\right)}{\Gamma_p^{1/k_p} - 1} \\ &= \frac{c_i (k_p + f + 1) Q \ln\left(\Gamma_p^{1/k_p}\right)}{\Gamma_p^{1/k_p} - 1} \\ &\leq c_i (k_p + f + 1) Q. \end{aligned}$$

The last inequality follows from $\ln(1+t) \leq t$. □

Lemma 4.6. *At the end of Algorithm 1, we have*

$$\forall i: \; r_i = 0 \text{ and } f_i \geq f.$$

Proof. When entering the e_p-loop for the last time, by Lemma 4.1,

$$\Gamma_p^{(-f+1)/k_p} \geq \gamma_j \geq \sum_i a_{ij} r_i \geq \sum_{i \in N_j^{(p)}} r_i.$$

γ_j can only be greater than 0 if there is exactly one r_i in the dual neighborhood of $v_j^{(p)}$ which is greater than zero. If r_i is still greater than 0 when $e_d = 0$, x_j will be increased by 1 which makes $w_j \geq 1$ and therefore $r_i = 0$ after the next call to **increase_duals()**.

f_i counts the number of times the i^{th} constraint of (LP) is satisfied. It is increased together with w_i in line 13 of Algorithm 1. Every time the integer part of w_i is increased, r_i is divided by $\Gamma_p^{\lfloor w_i \rfloor / k_p}$ and w_i is set to $w_i - \lfloor w_i \rfloor$. Therefore, $r_i = 0$ implies $f_i \geq f$. □

Lemma 4.7. *After the main part of Algorithm 1 (i.e., after the loops at line 20), we have*

$$\sum_{i=1}^{m} c_i x_i = c_{\max} \sum_{j=1}^{n} y_j.$$

Proof. Let $v_i^{(p)}$ be a primal node which increases x_i by x_i^+ (line 8). All dual neighbors $v_j^{(d)}$ of $v_i^{(p)}$ increase y_j^+ by $a_{ji}\tilde{r_j} x_i^+ / \tilde{\gamma_i}$. Hence, the sum of the y_j^+-increases over all dual neighbors of $v_i^{(p)}$ is

$$\frac{x_i^+}{\tilde{\gamma_i}} \sum_j a_{ji} \tilde{r_j} = x_i^+ \frac{\sum_j a_{ji} \tilde{r_j}}{\frac{c_{\max}}{c_i} \sum_j a_{ji} \tilde{r_j}} = \frac{c_i}{c_{\max}} x_i^+.$$

By Lemma 4.6, all y_j^+ are 0 in the end and thus y_j is equal to the sum of all increases of y_j^+. □

Theorem 4.8. *For arbitrary $k_p, k_d \geq 1$, Algorithm 1 approximates* (LP) *and* (DLP) *by a factor*

$$\Gamma_p^{4/k_p} \max\left\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\right\}.$$

The time complexity of Algorithm 1 is

$$\mathrm{O}\left(k_d k_p \left(1 + \frac{1}{\Gamma_p^{1/k_p} - 1}\right)\left(1 + \frac{k_p}{\Gamma_p^{1/k_p} \log \Gamma_p}\right)\right).$$

For $k_p \in \mathrm{O}(\log \Gamma_p)$, this simplifies to $\mathrm{O}(k_d k_p)$.

Proof. For the approximation ratio, we have to look at line 21 of Algorithm 1 where all x and y values are divided by the largest possible values to keep/make the primal/dual solution feasible. By Lemma 4.6, each primal constraint is satisfied at least f times. Therefore, all primal variables are divided by at least f. According to Lemma 4.5, for each primal node, the sum of the y values of its dual neighbors is at most $c_i(k_p + f + 1)Q$ for Q as defined in Lemma 4.5. Dividing all dual variables by $(k_p + f + 1)Q$ therefore makes the dual solution feasible. Thus, using Lemma 4.7, the ratio between the objective functions of the primal and the dual solutions becomes

$$\begin{aligned} \frac{\sum_{i=1}^m c_i x_i}{\sum_{j=1}^n y_i} &\leq c_{\max} \frac{k_p + f + 1}{f} Q \\ &\leq c_{\max} \frac{k_p + \frac{k_p+1}{\Gamma_p^{1/k_p}-1} + 1}{\frac{k_p+1}{\Gamma_p^{1/k_p}-1}} Q \\ &= c_{\max} \Gamma_p^{1/k_p} Q \\ &= \Gamma_p^{4/k_p} \max\left\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\right\}. \end{aligned}$$

Because of the duality theorem for linear programming, this ratio is an upper bound on the approximation ratio for (LP) and (DLP).

To obtain the time complexity of Algorithm 1, note that the number of rounds is proportional to the number of iterations of the inner-most loop (e_d-loop). Each iteration of the inner-most loop takes two rounds. Hence, the algorithm has time complexity $\mathrm{O}(k_d(k_p+f)h)$. Substituting the actual values for f and h, we obtain the desired expression. For the case $k_p \in \mathrm{O}(\log \Gamma_p)$, $\Gamma_p^{1/k_p}-1$ is a constant and therefore the time complexity simplifies to $\mathrm{O}(k_d k_p)$. □

Collorary 4.9. *For sufficiently small ε, Algorithm 1 computes a $(1+\varepsilon)$-approximation for (LP) and (DLP) in $\mathrm{O}\big(\log\Gamma_p \log\Gamma_d/\varepsilon^4\big)$ rounds. In particular, a constant factor approximation can be achieved in time $\mathrm{O}(\log\Gamma_p \log\Gamma_d)$.*

Proof. We choose k_p and k_d such that $\Gamma_p^{1/k_p} = \Gamma_d^{1/k_d} = 1+\varepsilon'$. By this, we get

$$k_p = \frac{\log \Gamma_p}{\log(1+\varepsilon')} \text{ and } k_d = \frac{\log \Gamma_d}{\log(1+\varepsilon')}.$$

For constant ε', we then have $k_p = \mathrm{O}(\log\Gamma_p)$ and $k_d = \mathrm{O}(\log\Gamma_d)$. For small enough ε', using first order Taylor approximations, we can estimate the expressions as $k_p \approx \log\Gamma_p/\varepsilon'$ and $k_d \approx \log\Gamma_d/\varepsilon'$. For $\varepsilon \in \Theta(\varepsilon')$, the Corollary then follows by plugging in the values for k_p and k_d in the expressions of Theorem 4.8. □

Remark 1: Recall that $\rho := a_{\max}/a_{\min}$ has been defined to be the ratio between the largest and the smallest coefficient of (LP) and (DLP). By the definitions of Γ_p and Γ_d, we have $\Gamma_p \le \rho\Delta_p$ and $\Gamma_d \le \rho\Delta_d$, i.e. the running time and the approximation ratio are dependent on the values of the coefficients. It is possible to get rid of this dependence by using methods which are described in [1, 14].

Remark 2: Algorithm 1 can be implemented such that all messages are of size $\mathrm{O}(k_p)$ or $\mathrm{O}(k_d)$. Thus, message sizes are $\mathrm{O}(\log(\Gamma_p+1/\varepsilon))$ or $\mathrm{O}(\log(\Gamma_d+1/\varepsilon))$ if $(1+\varepsilon)$ is the approximation ratio achieved.

5 Fast LP Algorithm

In the last section, we presented a distributed algorithm to efficiently (in terms of time and communication) approximate positive LPs. In this section, we show that if message size does not have to be bounded, there are faster algorithms to approximate (LP) and (DLP). In [12], Linial and Saks presented a randomized distributed algorithm to decompose a graph into sub-graphs of limited diameter. We use their algorithm to decompose the linear program into sub-programs which can be solved locally. We only give a broad overview here, for details we refer to Appendix B. Using the algorithm of [12], we can select connected components of G with the following properties.

(I) Different components are far enough from each other such that we can define a local linear program for each component in a way in which the LPs of any two components do not interfere.

(II) Each node belongs to one of the components with probability at least p where p depends on the diameter we allow the components to have.

Because of the limited diameter, the LPs of each component can then be computed locally. In parallel, the described decomposition is applied often enough such that w.h.p. each node has been selected a logarithmic number of times. Theorem 5.1 summarizes the results which can be achieved.

Theorem 5.1. *Using the network decomposition algorithm of [12], in only $\mathrm{O}(k)$ rounds (LP) and (DLP) can be approximated by a factor $\mathrm{O}\big(m^{1/k}\big)$ w.h.p. For $k \in \Theta(\log m)$, this gives a constant factor approximation in $\mathrm{O}(\log m)$ rounds.*

6 Randomized Rounding

In the following, we apply our distributed LP approximation algorithms together with standard randomized rounding techniques to obtain distributed approximation algorithms for a number of combinatorial problems. In particular, we give algorithms for integer covering and packing problems for which the matrix elements $a_{ij} \in \{0,1\}$ and where the components of the solution vectors are restricted to integers

(i.e., $\underline{x}' \in \mathbb{N}^m$ and $\underline{y}' \in \mathbb{N}^n$). In the sequel, the solution vectors for the integer program are denoted by $\underline{x}'$ and $\underline{y}'$ whereas the solution vectors for the corresponding LPs are called $\underline{x}$ and $\underline{y}$.

We start with the consideration of covering problems (problems of the form of (LP)). Because the a_{ij} and the x_i are restricted to integer values, w.l.o.g., we can round up all b_j to the next integer value. After solving/approximating the LP, each primal node $v_i^{(p)}$ executes the following algorithm (λ to be defined later):

1: **if** $x_i \geq 1/(\lambda \ln \Delta_p)$ **then**
2: $\quad x'_i := \lceil x_i \rceil$;
3: **else**
4: $\quad p_i := x_i \cdot \lambda \ln \Delta_p$;
5: $\quad x'_i := 1$ with prob. p_i and 0 otherwise;
6: **fi**;

The expected value of the objective function is $\mathrm{E}[\underline{c}^{\mathrm{T}}\underline{x}'] \leq \lambda \ln \Delta_p \cdot \underline{c}^{\mathrm{T}}\underline{x}$. But, no matter how we choose λ, there remains a non-zero probability that the obtained integer solution is not feasible. To overcome this, we have to increase some of the x'_i. Assume that the j^{th} constraint is not satisfied. Let $\underline{a}_j$ be the j^{th} row of the matrix A and let $b'_j := b_j - \underline{a}_i^{\mathrm{T}}\underline{x}'$ be the missing weight to make the j^{th} row feasible. Further, let $i_{j_{\min}}$ be the index of the minimum c_i for which $a_{ji} = 1$. We set $x'_{i_{j_{\min}}} := x'_{i_{j_{\min}}} + b'_j$. Applied to all non-satisfied primal constraints, this gives a feasible solution for the considered integer covering problem. Theorem 6.1 shows that the approximation ratio of the described algorithm is $\mathrm{O}(\log \Delta_p)$.

Theorem 6.1. *Let $\mathrm{IP_c}$ be an integer covering problem with $a_{ij} = \{0,1\}$, $b_j \in \mathbb{N}$, and $x'_i \in \mathbb{N}$. Furthermore, let $\underline{x}$ be an α-approximate solution for the LP relaxation of $\mathrm{IP_c}$. The described algorithm computes an $\mathrm{O}(\alpha \log \Delta_p)$-approximation $\underline{x}'$ for $\mathrm{IP_c}$ in a constant number of rounds.*

Proof. As stated above, the expected approximation ratio of the first part of the algorithm is $\lambda \ln \Delta_p$. In order to bound the additional weight of the second part, where $x'_{i_{j_{\min}}}$ is increased by b'_j, we define dual variables $\tilde{y}_j := b'_j c_{i_{j_{\min}}}/b_j$. For each unsatisfied primal constraint, the increase $c_{i_{j_{\min}}} b'_j$ of the primal objective function is equal to the increase $b_j \tilde{y}_j$ of the dual objective function. If the j^{th} constraint is not satisfied, we have $b'_j \geq 1$. Therefore, $\mathrm{E}[\tilde{y}_j] \leq q_j c_{i_{j_{\min}}}$, where q_j is the probability that the j^{th} primal inequality is not fulfilled.

In order to get an upper bound on the probability q_j, we have to look at the sum of the x'_i before the randomized rounding step in line 5 of the algorithm. Let $\beta_j := b_i - \underline{a}_i^{\mathrm{T}}\underline{x}'$ be the missing weight in row j before line 5. Because the x-values correspond to a feasible solution for the LP, the sum of the p_i involved in row j is at least $\beta_j \lambda \ln \Delta_p$. For the following analysis, we assume that $\ln \Delta_p \geq 1$.[1] Using the Chernoff bound of Appendix A (Theorem A.1), we can bound q_j:

$$\begin{aligned} q_j &< e^{-\beta_j \lambda \ln \Delta_p (1-1/(\lambda \ln \Delta_p))^2/2} \\ &\leq \left(\frac{1}{\Delta_p}\right)^{\lambda(1-1/\lambda)^2/2} \leq \frac{1}{\Delta_p}. \end{aligned}$$

In the second inequality, we use that $\beta_j \geq 1$. For the last inequality, we have to choose λ such that $\lambda(1-1/\lambda)^2/2 \geq 1$ (i.e., $\lambda \geq 2+\sqrt{3}$). Thus, the expected value of $\tilde{y}_j$ is $\mathrm{E}[\tilde{y}_j] \leq c_{i_{j_{\min}}}/\Delta_p$. Hence, by definition of $c_{i_{j_{\min}}}$, in expectation the $\tilde{y}$-values form a feasible solution for (DLP). Therefore, the expected increase of the objective function $\underline{c}^{\mathrm{T}}\underline{x}'$ in the last step after the randomized rounding is upper bounded by the objective function of an optimal solution for (LP). □

We now turn our attention to integer packing problems. We have an integer LP of the form of (DLP) where all $a_{ij} \in \{0,1\}$ and where $\underline{y}' \in \mathbb{N}^n$. We can assume that the c_j are integers as well because rounding down each c_j to the next integer has no influence on the feasible region. Each dual node $v_i^{(d)}$ applies the following algorithm.

1: **if** $y_i \geq 1$ **then**
2: $\quad y'_i := \lfloor y_i \rfloor$;
3: **else**
4: $\quad p_i := 1/(2e\Delta_d)$;
5: $\quad y'_i := 1$ with prob. p_i and 0 otherwise;
6: **fi**;
7: **if** $y'_i \in$ 'non-satisfied constraint' **then**
8: $\quad y'_i := \lfloor y_i \rfloor$;
9: **fi**

[1] If $\ln \Delta_p < 1$, applying only the last step of the described algorithm gives a simple distributed 2-approximation for the considered integer program.

Clearly, this yields a feasible solution for the problem. The approximation ratio of the algorithm is given by the next theorem.

Theorem 6.2. *Let* $\mathrm{IP_p}$ *be an integer covering problem with* $a_{ij} = \{0,1\}$, $c_j \in \mathbb{N}$, *and* $y'_i \in \mathbb{N}$. *Furthermore, let* $\underline{y}$ *be an* α*-approximate solution for the LP relaxation of* $\mathrm{IP_p}$. *The given algorithm computes an* $\mathrm{O}(\alpha\Delta_d)$*-approximation* $\underline{y}'$ *for* $\mathrm{IP_p}$ *in a constant number of rounds.*

Proof. After line 6, the expected value of the objective function is $\underline{b}^{\mathrm{T}}\underline{y}' \geq \underline{b}^{\mathrm{T}}\underline{y}/(2e\Delta_d)$. We will now show that a non-zero y'_i stays non-zero with a constant probability in line 8. Let q_j be the probability that the j^{th} constraint of the integer program is not satisfied given that y'_i has been set to 1 in line 5. For convenience, we define $Y'_j := \sum_i a_{ij}y'_i$. If $c_j \geq 2$, we apply the Chernoff bound of Appendix A (Theorem A.2):

$$\begin{aligned} q_j &= \Pr\big[Y'_j > c_j \,\big|\, y'_i = 1\big] \leq \Pr\big[Y'_j > c_j - 1\big] \\ &< \left(\frac{e^{e\Delta_d - 1}}{(e\Delta_c)^{e\Delta_d}}\right)^{c_j/(2e\Delta_d)} < \frac{1}{\Delta_d}. \end{aligned}$$

If $c_j = 1$, we get

$$\begin{aligned} q_j &\leq 1 - \Pr\big[Y'_j = 0\big] = 1 - \prod_{i \in N_j^{(p)}} (1 - p_i) \\ &\leq 1 - \left(1 - \frac{1}{2e\Delta_d}\right) = \frac{1}{2e\Delta_d}. \end{aligned}$$

The probability that all dual constraints containing y'_i are satisfied is lower bounded by the product of the probabilities for each constraint [19]. Therefore, under the natural assumption that $\Delta_d \geq 2$:

$$\Pr\big[y'_i = 1 \text{ after line 8}\big] \geq \left(1 - \frac{1}{\Delta_d}\right)^{\Delta_d} \geq \frac{1}{4}.$$

Thus the expected value of the objective function of $\mathrm{IP_p}$ is $\mathrm{E}[\underline{b}^{\mathrm{T}}\underline{y}'] \geq 8e\Delta_d \cdot \underline{b}^{\mathrm{T}}\underline{y}$. □

Remark: Using more elaborate randomized rounding techniques, it would be possible to get slightly better result (cf. [18, 17, 19]). However, the objective of this section is to show that the techniques can be applied in a distributed scenario and not to find the best possible constants. Note that the derandomization techniques introduced in the context with randomized rounding cannot be applied because it can be shown that it is not possible to approximate general integer covering and packing problems by a distributed algorithm which is deterministic and local.

7 Conclusion

We have presented two approximation algorithms for fractional covering and packing problems. The first algorithm is deterministic, uses only small messages, and achieves a constant approximation ratio in $\mathrm{O}(\log(\rho\Delta_d)\log(\rho\Delta_p))$ rounds. Furthermore at the cost of a factor $1/\varepsilon^4$, the algorithm computes a $(1+\varepsilon)$-approximation. With the second randomized algorithm, a constant-factor approximation can be achieved in $\mathrm{O}(\log m)$ rounds where m is the number of primal variables. Both algorithms show that even in a constant number of rounds, positive linear programs can be approximated by a non-trivial factor.

Combining the randomized rounding techniques of Section 6 with the presented LP approximation algorithms, we improve the results of [10] for the minimum dominating set (MDS) problem. Using small messages, the results of this paper give rise to a $\mathrm{O}(\log\Delta)$-approximation of the MDS problem with time complexity $\mathrm{O}\big(\log^2\Delta\big)$. By this, our approach matches the results of [9] for MDS and also for weighted MDS. If message size in unbounded, the time complexity goes down to $\mathrm{O}(\log\Delta)$. For the minimum vertex cover problem, Algorithm 1 alone (with $k_d = 1$) can be used to obtain a constant factor approximation in $\mathrm{O}(\log\Delta)$ rounds. Maximum matching (the dual of minimum vertex cover) can also be approximated by a constant factor in $\mathrm{O}(\log\Delta)$ rounds using our approach. For the weighted version of vertex cover and matching, the time complexity goes up to $\mathrm{O}(\log(\Delta w_{\max}))$. For matching, our results meet the results of [8, 13, 21]. Note that we also show that all the above problems have non-trivial constant-time distributed approximations.

8 Acknowledgments

We would like to thank Baruch Awerbuch for inspiring discussions and Thomas Moscibroda for checking the proofs of Algorithm 1.

References

[1] Y. Bartal, J. W. Byers, and D. Raz. Global Optimization Using Local Information with Applications to Flow Control. In *Proc. of the 38th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

[2] V. Chvátal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3), August 1979.

[3] R. Cole and U. Vishkin. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Information and Control*, 70(1):32–53, 1986.

[4] F. Eisenbrand, S. Funke, N. Garg, and J. Könemann. A Combinatorial Algorithm for Computing a Maximum Independent Set in a *t*-perfect Graph. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 517–522, 2003.

[5] L. Fleischer. Approximating Fractional Multicommodity Flow Independent of the Number of Commodities. *SIAM Journal on Discrete Math.*, 13(4):505–520, 2000.

[6] L. Fleischer. A Fast Approximation Scheme for Fractional Covering Problems with Variable Upper Bounds. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004 (to appear).

[7] D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997.

[8] A. Israeli and A. Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Information Processing Letters*, 22:77–80, 1986.

[9] L. Jia, R. Rajaraman, and R. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In *Proc. of the 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2001.

[10] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[11] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, February 1992.

[12] N. Linial and M. Saks. Low Diameter Graph Decompositions. *Combinatorica*, 13(4):441–454, 1993.

[13] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[14] M. Luby and N. Nisan. A Parallel Approximation Algorithm for Positive Linear Programming. In *Proc. of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 448–457, 1993.

[15] C. Papadimitriou and M. Yannakakis. Linear Programming without the Matrix. In *Proc. of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 121–129, 1993.

[16] S. Plotkin, D. Shmoys, and E. Tardos. Fast Approximation Algorithms for Fractional Packing and Covering Problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[17] P. Raghavan. Randomized Rounding Algorithms in Combinatorial Optimization. In *Proc. FST & TCS Conference*, pages 300–317, 1994.

[18] P. Raghavan and C. D. Thompson. Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs. *Combinatorica*, 7(4):365–374, 1987.

[19] A. Srinivasan. Improved approximations of packing and covering problems. In *Proc. of the 27th ACM Symposium on Theory of Computing*, pages 268–276, 1995.

[20] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[21] M. Wattenhofer and R. Wattenhofer. Distributed Weighted Matching. Technical Report 420, ETH Zurich, Department of Computer Science, 2003.

Appendix

A Chernoff Bounds

The Chernoff bounds describe the tail behavior of the distribution of the sum of independent Bernoulli experiments.

Theorem A.1. (Lower Tail) *Let $X_1, X_2, \ldots, X_N$ be independent Bernoulli variables with $\Pr[X_i = 1] = p_i$. Let $X := \sum_i X_i$ denote the sum of the X_i and let $\mu := \mathrm{E}[X] := \sum_i p_i$ be the expected value for X. For $\delta \in]0, 1]$,*

$$\begin{aligned}\Pr[X < (1-\delta)\mu] &< \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{\mu} \\ &< e^{-\mu\delta^2/2}.\end{aligned}$$

Theorem A.2. (Upper Tail) *Let $X_1, X_2, \ldots, X_N$ be independent Bernoulli variables with $\Pr[X_i = 1] = p_i$. Let $X := \sum_i X_i$ denote the sum of the X_i and let $\mu := \mathrm{E}[X] := \sum_i p_i$ be the expected value for X. For $\delta > 0$,*

$$\Pr[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}.$$

B Fast LP Approximation Algorithm – Details

For the decomposition of (LP) and (DLP), we need the following lemma.

Lemma B.1. *Let $\{x'_1, \ldots, x'_{m'}\}$ be a subset of the primal variables of* (LP) *and let $y'_1, \ldots, y'_{n'}$ be the dual variables which are adjacent to the given subset of the primal variables. Further let* LP′ *and* DLP′ *be LPs where the matrix A' consists only of the columns and rows corresponding to the variables in $\underline{x}'$ and $\underline{y}'$. Every feasible solution for* LP′ *makes the corresponding primal inequalities in* (LP) *feasible and every feasible solution for* DLP′ *is feasible for* (DLP) *(variables not occurring in* LP′ *and* DLP′ *are set to 0). Further, the values of the objective functions for the optimal solutions of* LP′ *and* DLP′ *are smaller than the the optimal values for* (LP) *and* (DLP).

Proof. The feasibilities directly follow from the definition of LP′ and DLP′. The optimal values for the objective functions of LP′ and DLP′ are smaller than the optimal values for (LP) and (DLP) because of the (DLP)-feasibility of a dual feasible solution for DLP′. □

We call LP′ and DLP′ the sub-LPs induced by the subset $\{x'_1, \ldots, x'_{m'}\}$ of primal variables. We apply the graph decomposition algorithm of [12] to obtain LP′ and DLP′ (as in Lemma B.1) which can be solved locally. For a general graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with m nodes, the algorithm of [12] yields a subset $\mathcal{S} \subseteq \mathcal{V}$ of $\mathcal{V}$ such that each node $u \in \mathcal{S}$ has a leader $\ell(u) \in \mathcal{V}$ and such that the following properties hold[2].

(I) $\forall u \in \mathcal{S}: \ \mathrm{d}(u, \ell(u)) < k$

(II) $\forall u, v \in \mathcal{S}: \ \ell(u) \neq \ell(v) \longrightarrow (u, v) \notin E.$

(III) $\mathcal{S}$ can be computed in k rounds.

(IV) $\forall u \in \mathcal{V}: \ \Pr[u \in \mathcal{S}] \geq \frac{1}{em^{1/k}}$.

$d(u, v)$ denotes the distance between two nodes u and v on $\mathcal{G}$. In order to apply this to decompose the linear program, we define $\mathcal{G}$ as follows. The node set $\mathcal{V}$ is the set of primal nodes of the graph G as defined in Section 3. The edge set $\mathcal{E}$ is

$$\mathcal{E} := \left\{(u, v) \,\middle|\, u, v \in \mathcal{V} \wedge \mathrm{d}_G(u, v) \leq 4\right\}.$$

By this, we can guarantee that non-adjacent nodes in $\mathcal{G}$ do not have neighboring dual nodes in G whose variables occur in the same constraint of (DLP). Further, a message over an edge of $\mathcal{G}$ can be sent in 4 rounds on the network graph G. The basic algorithm for a primal node v to approximate (LP) and (DLP) then works as follows:

1: Run graph decomposition of [12] on $\mathcal{G}$;
2: **if** $v \in \mathcal{S}$ **then**
3: **send** IDs of dual neighbors to $\ell(v)$.
4: **fi**;
5: **if** $v = \ell(u)$ for some $u \in \mathcal{S}$ **then**
6: compute local LP/DLP (cf. Lemma B.1) of variables of $u \in \mathcal{S}$ for which $v = \ell(u)$.
7: **send** resulting values to nodes holding the respective variables.

[2] We use $p = 1/m^{1/k}$ in the algorithm of Section 4 of [12], the properties then directly follow from Lemma 4.1.

8: **fi**

The dual nodes only forward messages in steps 1, 3, and 7 and receive the values for their variables in step 7. We now have a closer look at the locally computed LPs in line 6. By Property (II) of the graph decomposition algorithm, primal variables belonging to different local LPs cannot occur in the same primal constraint (otherwise, the according primal nodes had to be neighbors in $\mathcal{G}$). The analogous fact holds for dual variables since primal nodes belonging to different local LPs have distance at least 6 on G and thus dual nodes belonging to different local LPs have distance at least 4 on G. Therefore, the local LPs do not interfere and together they form the sub-LPs induced by $\mathcal{S}$ (cf. Lemma B.1).

The complete LP approximation algorithm now consists of N independent parallel executions of the described basic algorithm. The variables of the N sub-LPs are added up and in the end, primal/dual nodes divide their variables by the maximum/minimum possible value to keep/make all constraints they occur in feasible (as in line 21 of Algorithm 1). The following theorem states how N has to be chosen such that the obtained approximation ratio is best possible.

Theorem B.2. *Let $N = \alpha e m^{1/k} \ln m$ for $\alpha \approx 4.51$. Executing the basic algorithm N times, summing up the variables of the N execution and dividing these sums as described in line 21 of Algorithm 1, yields an $\alpha e m^{1/k}$ approximation of (LP)/(DLP) w.h.p. The algorithm needs $\mathrm{O}(k)$ rounds to complete.*

Proof. We begin the proof with the running time. The N executions can be done completely in parallel, we therefore only have to care about one instance of the basic algorithm. By Property (I), the topology collecting and variable distribution in lines 3 and 7 can be done in $\mathrm{O}(k)$ time. By Property (III), the graph decomposition in line 1 can be computed in the same time.

For the approximation ratio, we have to bound the ratio of the factors by which the primal and the dual variables are divided in the end. By Lemma B.1, the dual variables of each of the N sub-LPs constitute a feasible solution for (DLP). Therefore, the sums of the dual variables of the sub-LPs have to be divided by at most N to obtain a feasible solution for (DLP). For the primal variables, we have to count the number occurrences in sub-LPs for each primal constraint. This is lower-bounded by the number of times each primal node has been chosen to be in $\mathcal{S}$. By (IV), for each primal node, the probability in each of the N executions is at least $1/(em^{1/k})$. We use the Chernoff bounds to obtain an upper bound on the probability that primal node v occurs in less than $\ln m$ sub-LPs. Let X denote the number of times, v is chosen by the graph decomposition algorithm of [12]:

$$\begin{aligned}\Pr[X < \ln m] &< \left(\frac{\alpha^{1/\alpha}}{e^{1-1/\alpha}}\right)^{\alpha \ln m} \\ = \frac{e^{\ln\alpha \ln m}}{e^{(\alpha-1)\ln m}} &= \frac{1}{m^{\alpha-1-\ln\alpha}} < \frac{1}{m^2}\end{aligned}$$

for $\alpha > 4.51$. Thus, with probability at least $1-1/m$ all primal variables can be divided by $\ln m$. □

Remark 1: Choosing $k \in \mathrm{O}(\log m)$, the above algorithm approximates all fractional covering and packing problems with a constant factor in $\mathrm{O}(\log m)$ rounds.

Remark 2: If $n < m$, $\mathcal{G}$ can be defined such that the node set $\mathcal{V}$ consists of the dual nodes. We can then obtain a similar algorithm, which computes a $\alpha e n^{1/k}$-approximation in $\mathrm{O}(k)$ rounds.