The Price of Being Near-Sighted

Fabian Kuhn, Thomas Moscibroda, Roger Wattenhofer
{kuhn,moscitho,wattenhofer}@tik.ee.ethz.ch
Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland

Abstract

Achieving a global goal based on local information is challenging, especially in complex and large-scale networks such as the Internet or even the human brain. In this paper, we provide an almost tight classification of the possible trade-off between the amount of local information and the quality of the global solution for general covering and packing problems. Specifically, we give a distributed algorithm using only small messages which obtains an $(\rho\Delta)^{1/k}$-approximation for general covering and packing problems in time $O(k^2)$, where ρ depends on the LP's coefficients. If message size is unbounded, we present a second algorithm that achieves an $O(n^{1/k})$ approximation in $O(k)$ rounds. Finally, we prove that these algorithms are close to optimal by giving a lower bound on the approximability of packing problems given that each node has to base its decision on information from its k-neighborhood.

1 Introduction

Many of the most fascinating and fundamental systems in the world are large and complex networks, such as the human society, the Internet, or the human brain. Such systems have in common that their entirety is composed of a multiplicity of individual *entities*; human beings in society, hosts in the Internet, or neurons in the brain. As diverse as these systems may be, they share the key characteristic that the capability of direct communication of each individual entity is restricted to only a small subset of neighboring entities. Most human communication, for instance, is between acquaintances or within the family, and neurons are directly linked with merely a relatively small number of other neurons for neurotransmission. On the other hand, in spite of each node thus being inherently "near-sighted," i.e., restricted to *local communication*, the entirety of the system is supposed to come up with some kind of global solution, or to keep an equilibrium.

Achieving a *global goal* based on *local information* is challenging. Many of the systems which are the focus of computer science fall exactly into the above mentioned category of networks. In the Internet, large-scale peer-to-peer systems, or mobile ad hoc and sensor networks, no node in the network is capable of keeping global information on the network. Instead, these nodes have to perform their intended (global) task based on local information only. In other words, all computation in these systems is inherently local! Not surprisingly, studying the fundamental possibilities and limitations of *local computation* is therefore of interest to theoreticians in approximation theory, distributed computing, and graph theory.

The study of local computation has been initiated by the pioneering work of Linial [11], and Naor and Stockmeyer [15] more than a decade ago. Also, the work of Peleg [17] has resulted in numerous interesting and deep results. But nonetheless, there remains a great number of important open problems related to questions such as what kind of global tasks can be performed by individual entities that have to base their decisions on local information, or how much local information is required in order to come up with a globally optimal solution. For instance, the open question from [16], that is, characterizing the trade-off between communication among agents to exchange information and the global utility achieved, has been unanswered. It is the goal of this paper to make a step towards answering this open problem and, more generally, to bring forward some of the underlying principles and trade-offs governing local computation.

Not surprisingly, many global criteria such as counting the total number of nodes in the network or obtaining a minimum spanning tree cannot be met if every node's decision is based solely on local knowledge. On the other hand, many fundamental coordination tasks and applications in large-scale networks appear to be easier to handle from a "local-global" perspective. Specifically, classic graph theory problems such as *dominating set* or *matching* can be formulated as standard covering and packing problems. The nature of simple covering and packing problems like minimum vertex cover or maximum matching appears to be local and intuitively, one may think that each node's (edge's) decision is not affected by very distant nodes (edges). Interestingly, we prove in this paper that this intuition is misleading even for the most basic packing problems.

On the positive side, we show that there exist *distributed approximation algorithms* that almost achieve the optimal trade-off, even in the practically important case in which the amount of information exchanged in each message is limited. Specifically, we give the following results:

- Consider a network with n nodes and maximum degree Δ. Assume that each node in a network graph has to base its decision on its k-hop neighborhood. We present an efficient deterministic distributed algorithm that operates with small messages of size $O(\log n)$ bits. The algorithm achieves a $(\rho\Delta)^{1/k}$-approximation for general covering and packing problems in $O(k^2)$ communication rounds, where ρ depends on the coefficients of the underlying LP.
- When message size is unbounded, each network node can easily gather the entire information from its $O(k)$-neighborhood in $O(k)$ communication rounds. Hence, in this case, the achievable trade-off is a true consequence of locality restriction only. We present an algorithm producing an $O(n^{1/k})$-approximation if each node knows its $O(k)$-neighborhood.
- In combination with (distributed) randomized rounding, the above algorithms can be transformed into *constant-time distributed algorithms* having non-trivial approximation ratios for various combinatorial problems.
- Finally, we show that the trade-off achieved by our algorithms is almost tight. Specifically, we prove that even the most simple packing problem, (fractional) maximum matching, cannot be approximated within $\Omega(n^{c/k^2}/k)$ and $\Omega(\Delta^{1/k}/k)$, respectively. This implies $\Omega(\sqrt{\log n/\log\log n})$ and $\Omega(\log\Delta/\log\log\Delta)$ time lower bounds for (possibly randomized) distributed algorithms in order to obtain a constant or polylogarithmic approximation for maximum matching and packing problems, even if message size is unbounded. This lower bound extends a similar result that has recently been achieved for the minimum vertex cover problem in [9].

Note that by giving upper and lower bounds for general covering and packing LPs, we show that many different natural problems behave similarly with regard to *local approximability*. This is of great theoretical interest since such a classification of problems may provide a completely new insight into the impact of locality on algorithms.

Related work and the model of computation are described in Sections 2 and 3. In subsequent Sections 4 and 5, we give distributed algorithm for general covering and packing LPs in the bounded and unbounded message model, respectively. The packing lower bound is derived in subsequent Section 6. Section 7 concludes the paper. Due to lack of space, some proofs are omitted from this extended abstract.[1]

2 Related Work

Little is known about the fundamental limitations of locality-based approaches. Fich and Ruppert, for instance, describe a numerous lower bounds and impossibility results in distributed computing [5]. But most of them apply to other computational models where locality is no issue or there are additional, more restrictive limiting factors, such as bounded message size [4]. There have been virtually no nontrivial lower bounds for local computation, besides Linial's seminal $\Omega(\log^* n)$ time lower bound for constructing a maximal independent set on a ring [11]. In addition, we have shown that minimum vertex cover and thus covering problems cannot be approximated better than $\Omega(n^{c/k^2}/k)$ and $\Omega(\Delta^{1/k}/k)$ if each node's information is restricted to its k-neighborhood [9]. On the positive side, it was shown by Naor and Stockmeyer [15] that there exist locally checkable labelings which can be computed in distributed constant time, i.e., with completely local information only.

The focus of this paper is to understand locality in problems that can be formulated as packing and covering LPs. There are a number of (parallel) algorithms for solving such LPs which are faster than interior-point methods that can be applied to general LPs (e.g. [6, 14, 18, 22]). All these algorithms need at least some global information to work. The problem of approximating positive LPs using only local information has been introduced in [16]. The first algorithm achieving a constant approximation in polylogarithmic time is described in [2]. Distributed algorithms targeted for specific covering and packing problems include algorithms for the minimum dominating set problem [3, 8, 19] as well as algorithms for maximal matchings and maximal independent sets [1, 7, 13]. This implies a constant approximation for maximum matching.

All described distributed algorithms have a time complexity which is at least logarithmic in n. That is, each node may gather information which is as far away as $O(\log n)$ hops. Hence, while these algorithms provide solutions to particular problems, they do not fully explore the trade-off between local knowledge and solution quality. A distributed algorithm for minimum dominating set running in an arbitrary, possibly constant number of rounds is found in [10].

[1] A version containing all proofs can be found as TIK technical report 229 at ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report229.pdf

3 Model

We describe the network as an undirected graph $G = (V, E)$. The vertices $V = \{v_1, \ldots, v_n\}$ represent the network entities or *nodes* (e.g. processors) and the edges represent bidirectional communication channels. We distinguish two prototypical and classic message passing models [17], $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$, depending on how much information can be sent in each message.

In the $\mathcal{LOCAL}$ model (e.g., [11, 15, 17]), knowing your k-neighborhood and performing k communication rounds are equivalent. It is assumed that in every communication round, each node in the network can send an *arbitrarily long message* to each of its neighbors. Local computations are for free. Each node has a unique identifier and initially, nodes have no knowledge about the network graph. In k communication rounds, a node v may collect the IDs and interconnections of all nodes up to distance k from v, because messages are unbounded. Hence, each node has a partial (local) view of the graph; it knows its entire vicinity up to distance k. Let $\mathcal{T}_{v,k}$ be the topology seen by v after k rounds. $\mathcal{T}_{v,k}$ is the graph induced by the k-neighborhood of v without all edges between nodes at distance exactly k. The labeling (i.e., the assignment of IDs) of $\mathcal{T}_{v,k}$ is denoted by $\mathcal{L}(\mathcal{T}_{v,k})$. The *view* of a node v is the pair, $\mathcal{V}_{v,k} := (\mathcal{T}_{v,k}, \mathcal{L}(\mathcal{T}_{v,k}))$. The *view* of an edge $e = (u, v)$ is the union of views of its incident nodes. The best a local algorithm can do in time k, is to have every node v collect its k-neighborhood and base its decision on $\mathcal{V}_{v,k}$. Since the $\mathcal{LOCAL}$ model abstracts away other aspects arising in the design of distributed algorithms (congestion, fast local computation, ...), it is the most fundamental model when studying the phenomenon of locality; particularly for lower bounds.

In practice, the amount of information exchanged between two neighbors in one communication step is limited. The $\mathcal{CONGEST}$ model [4, 17] takes into account the *volume of communication*. This model limits the information that can be sent in one message to $O(\log n)$ bits. Given this additional restriction, even problems on the complete network graph, which could be solved optimally in a single communication round in the $\mathcal{LOCAL}$ model, become nontrivial.

A fractional covering problem (PP) and its dual fractional packing problem (DP), are linear programs of the canonical forms

$$\begin{array}{lll} \min & \underline{c}^{\mathrm{T}}\underline{x} & \\ \text{s.t.} & A \cdot \underline{x} \geq \underline{b} & \\ & \underline{x} \geq \underline{0} & \end{array} \qquad \begin{array}{lll} \max & \underline{b}^{\mathrm{T}}\underline{y} & \\ \text{s.t.} & A^{\mathrm{T}} \cdot \underline{y} \leq \underline{c} & \\ & \underline{y} \geq \underline{0} & \end{array}$$

where all a_{ij}, b_i, and c_i are non-negative. We will use the term primal LP (PP) for the minimization and dual LP (DP) 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 coefficient and $a_{\min} := \min_{i,j}\{a_{ij}, b_i, c_i\} \setminus \{0\}$ be the minimum non-zero coefficient of (PP) and (DP). $\rho := a_{\max}/a_{\min}$ is the maximum ratio between any two coefficients.

Analogously to [2, 16], 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 (PP), i.e., $v_i^{(p)}$ and $v_j^{(d)}$ are connected iff $a_{ji} > 0$[2]. The degrees of $v_i^{(p)}$ and $v_j^{(d)}$ are called $\delta_i^{(p)}$ and $\delta_j^{(d)}$, respectively. $\Delta_p := \max_i \delta_i^{(p)}$ and $\Delta_d := \max_j \delta_j^{(d)}$ are called the primal and dual degree, 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.

4 Bounded Messages

In this section, we describe an efficient distributed algorithm to approximate covering and packing linear programs in the $\mathcal{CONGEST}$ model. For our algorithm, we need the LPs (PP) and (DP) to be of the following special form:

$$\forall i, j \ : \ b_i = 1, \quad a_{ij} = 0 \text{ or } a_{ij} \geq 1. \tag{4.1}$$

The transformation to (4.1) 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 (4.1) 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. Note that the described transformation can be computed locally in a constant number of rounds. For the rest of this section,

[2] Note that in order to solve such a problem in a real network setting where only some variables correspond to nodes, the other variables may be simulated by the nodes as well. Variables associated to edges (like in vertex cover or maximum matching) can be simulated by incident nodes.

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

1: $x_i := 0$;
2: **for** $e_p := k_p - 2$ **to** $-f - 1$ **by** -1 **do**
3: **for** 1 **to** h **do**
4: $(* \ \gamma_i := \frac{c_{\max}}{c_i} \sum_j a_{ji} r_j \ *)$
5: **for** $e_d := k_d - 1$ **to** 0 **by** -1 **do**
6: $\tilde{\gamma}_i := \frac{c_{\max}}{c_i} \sum_j a_{ji} \tilde{r}_j$;
7: **if** $\tilde{\gamma}_i \geq \Gamma_p^{e_p/k_p}$ **then**
8: $x_i^+ := 1/\Gamma_d^{e_d/k_d}$; $x_i := x_i + x_i^+$;
9: **fi**;
10: **send** x_i^+, $\tilde{\gamma}_i$ to dual neighbors;
11:
12:
13:
14:
15: **receive** $\tilde{r}_j$ from dual neighbors
16: **od**;
17:
18: **receive** r_j from dual neighbors
19: **od**
20: **od**;
21: $x_i := x_i / \min_{j \in N_i^{(p)}} \sum_\ell a_{j\ell} x_\ell$

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

1: $y_i := y_i^+ := w_i := f_i := 0$; $r_i := 1$;
2: **for** $e_p := k_p - 2$ **to** $-f - 1$ **by** -1 **do**
3: **for** 1 **to** h **do**
4: $\tilde{r}_i := r_i$;
5: **for** $e_d := k_d - 1$ **to** 0 **by** -1 **do**
6:
7:
8:
9:
10: **receive** x_j^+, $\tilde{\gamma}_j$ from primal neighbors;
11: $y_i^+ := y_i^+ + \tilde{r}_i \sum_j a_{ij} x_j^+ / \tilde{\gamma}_j$;
12: $w_i^+ := \sum_j a_{ij} x_j^+$;
13: $w_i := w_i + w_i^+$; $f_i := f_i + w_i^+$;
14: **if** $w_i \geq 1$ **then** $\tilde{r}_i := 0$ **fi;**
15: **send** $\tilde{r}_i$ to primal neighbors
16: **od**;
17: **increase_duals()**;
18: **send** r_i to primal neighbors
19: **od**
20: **od**;
21: $y_i := y_i / \max_{j \in N_i^{(d)}} \frac{1}{c_j} \sum_\ell a_{\ell j} y_\ell$

Algorithm 1: Distributed LP Approximation Algorithm

we assume that the coefficients of the LP are given according to (4.1).

We start the description of the algorithm with a general outline. As our algorithm borrows from the greedy dominating set/set cover algorithm, it is useful to view the distributed LP algorithm in this context. The greedy minimum dominating set (MDS) algorithm starts with an empty set and sequentially adds the node which covers the most not yet covered nodes. The LP relaxation of MDS asks for variables x_i for the nodes v_i such that the sum of the x_i in the 1-neighborhood of every node is at least 1. Analogous to the sequential greedy approach, we also start with all x_i set to 0 and we give priority to nodes with many uncovered neighbors when increasing the x_i. In particular, we always increase the x_i of all the nodes whose number of uncovered neighbors is maximum up to a certain factor (active nodes). In order not to 'over-cover' a node with many active neighbors, we have to carefully choose the increment of the x_i at active nodes. As we proceed, we simultaneously compute a solution for the dual LP such that the objective values of the solutions stay the same. In the end, each node is covered at least f times and each dual constraint is fulfilled up to a factor αf. Hence by dividing by f and αf, we obtain feasible, α-approximate primal and dual solutions, respectively.

In order to achieve that every primal inequality is fulfilled f times, 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 (cf. [21]). The decision whether a primal node $v_i^{(p)}$ is active and can increase x_i is based on the efficiency per cost ratio γ_i which is defined 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

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

At the price of a considerably more complicated (and less readable) algorithm, it is possible to get rid of this assumption. For details, we refer to the full paper.

The detailed algorithm is given by Algorithm 1 along with the procedure **increase_duals()** which is

procedure increase_duals():

1: **if** $w_i \geq 1$ **then**
2: **if** $f_i \geq f$ **then**
3: $y_i := y_i + y_i^+$; $y_i^+ := 0$;
4: $r_i := 0$; $w_i := 0$
5: **else if** $w_i \geq 2$ **then**
6: $y_i := y_i + y_i^+$; $y_i^+ := 0$;
7: $r_i := r_i / \Gamma_p^{\lfloor w_i \rfloor / k_p}$
8: **else**
9: $\lambda := \max\{\Gamma_d^{1/k_d}, \Gamma_p^{1/k_p}\}$;
10: $y_i := y_i + \min\{y_i^+, r_i\lambda/\Gamma_p^{e_p/k_p}\}$;
11: $y_i^+ := y_i^+ - \min\{y_i^+, r_i\lambda/\Gamma_p^{e_p/k_p}\}$;
12: $r_i := r_i / \Gamma_p^{1/k_p}$
13: **fi**;
14: $w_i := w_i - \lfloor w_i \rfloor$
15: **fi**

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 lemmas which establish all the necessary details to analyze Algorithm 1.

The goal of the outer e_p-loop is to reduce the maximum "weighted primal degree" γ_i. This is reflected by the following lemma.

LEMMA 4.1. *For each primal node $v_i^{(p)}$, at all times during Algorithm 1, $\gamma_i \leq \Gamma_p^{(e_p+2)/k_p}$.*

One complete run (k_d iterations) of the innermost e_d-loop can be seen as one parallel greedy step. Primal nodes with large γ_i increase their x_i such that the corresponding increases y_i^+ of the dual variables are almost feasible.

LEMMA 4.2. *Each time a dual node enters **increase_duals()** in Algorithm 1,*

$$y_i^+ \leq r_i \cdot \frac{w_i}{\Gamma_p^{e_p/k_p}} \text{ and } y_i^+ \leq r_i \cdot \frac{\Gamma_d^{1/k_d} + 1}{\Gamma_p^{e_p/k_p}}. \tag{4.2}$$

As shown in Lemma 4.4, all the increases of the dual variables together render the dual constraints feasible up to a small factor times $(k_p + f + 1)$. We first need the following helper lemma.

LEMMA 4.3. *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

$$\beta_j \leq \frac{\Gamma_p^{1/k_p} \cdot \max\{\Gamma_p^{1/k_p}, \Gamma_d^{1/k_d}\}}{\Gamma_p^{e_p/k_p}(\Gamma_p^{1/k_p} - 1)} \cdot r_j^-. \tag{4.3}$$

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 (4.3), we again consider the cases where $w_j \geq 2$ and where $1 \leq w_j < 2$. If $w_j \geq 2$, by Lemma 4.2, $\beta_j = y_j^+ \leq r_j(1 + \Gamma_d^{1/k_d})/\Gamma_p^{e_p/k_p}$. 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 (4.3):

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

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 4.4. *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 4.3 still holds. The analysis is exactly the same as for the case $w_j \geq 2$ in Lemma 4.3. By Lemma 4.3, 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:

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

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

At the end of the algorithm, all primal constraints are satisfied at least f times. Further, the primal and dual objective functions are the same.

LEMMA 4.5. *After the loops at line 20, $\forall i$: $r_i = 0$ and $f_i \geq f$ and $\sum_{i=1}^m c_i x_i = c_{\max} \sum_{j=1}^n y_j$.*

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 (PP) 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 $f_i \geq f$.

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

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

Because $f_j \geq f$, all y_j^+ are 0 in the end and thus y_j is equal to the sum of all increases of y_j^+. □

Combining the above lemmas, we get the following theorem.

THEOREM 4.1. *For arbitrary $k_p, k_d \geq 1$, Algorithm 1 approximates* (PP) *and* (DP) *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 4.5, each primal constraint is satisfied at least f times. Therefore, all primal variables are divided by at least f. Due to Lemma 4.4, 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 4.4. Dividing all dual variables by $(k_p + f + 1)Q$ therefore renders the dual solution feasible. By Lemma 4.5, the ratio between the objective functions of the primal and the dual solutions is

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

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

As for the time complexity, note that each iteration of the inner-most loop (e_d-loop) requires two rounds. Hence, the algorithm has time complexity $\mathrm{O}(k_d(k_p + f)h)$. The claim follows from substituting the actual values for f and h. For $k_p \in \mathrm{O}(\log \Gamma_p)$, $\Gamma_p^{1/k_p} - 1$ is a constant and therefore the time complexity simplifies to $\mathrm{O}(k_d k_p)$. □

COROLLARY 4.1. *For sufficiently small ε, Algorithm 1 computes a $(1 + \varepsilon)$-approximation for* (PP) *and* (DP) *in $\mathrm{O}\left(\log \Gamma_p \log \Gamma_d / \varepsilon^4\right)$ rounds. In particular, a constant factor approximation can be achieved in time $\mathrm{O}(\log \Gamma_p \log \Gamma_d)$.*

Remark: Using methods similar to the ones described in [2, 14], it is possible to get rid of the dependency on the coefficients $\rho := a_{\max}/a_{\min}$. As a result, the running time and approximation ratio would depend on the number of nodes m and n instead of the degrees Δ_p and Δ_d.

Distributed Randomized Rounding We can apply our distributed LP approximation algorithms together with standard distributed randomized rounding techniques to obtain distributed approximation algorithms for a number of combinatorial problems. We can prove that given an α-approximate solution for the LP relaxation of problems for which the matrix elements $a_{ij} \in \{0, 1\}$, we can compute in a constant number of rounds a $O(\alpha \log \Delta_p)$-approximation for the corresponding covering IP and a $O(\alpha\Delta_d)$-approximation for the packing IP.

5 Unbounded Messages

In [12], Linial and Saks presented a randomized distributed algorithm to decompose a graph into subgraphs of limited diameter. We use their algorithm to decompose the linear program into sub-programs which can be solved locally in the $\mathcal{LOCAL}$ model. For a general graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n 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.[3]

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

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

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

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

$d(u, v)$ denotes the distance between two nodes u and v on $\mathcal{G}$. We apply the algorithm of [12] to obtain connected components of G with the following properties.

(I) The components have small diameter.

(II) 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.

(III) 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. We apply the decomposition process in parallel often enough such that w.h.p. each node has been selected a logarithmic number of times.

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

For the decomposition of (PP) and (DP), we need the following lemma.

LEMMA 5.1. *Let $\{y'_1, \ldots, y'_{m'}\}$ be a subset of the dual variables of DP and let $x'_1, \ldots, x'_{n'}$ be the primal variables which are adjacent to the given subset of the dual variables. Further let PP' and DP' 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 PP' makes the corresponding primal inequalities in PP feasible and every feasible solution for DP' is feasible for DP (variables not occurring in PP' and DP' are set to 0). Further, the values of the objective functions for the optimal solutions of PP' and DP' are upper bounded by the optimal values for PP and DP.*

We call PP' and DP' the sub-LPs induced by the subset $\{y'_1, \ldots, y'_{m'}\}$ of dual variables. We apply the graph decomposition algorithm of [12] to obtain PP' and DP' (as in Lemma 5.1) which can be solved locally.

For the decomposition of the linear program, we define $\mathcal{G}$ such that the node set $\mathcal{V}$ is the set of dual nodes of the graph G and 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 primal nodes in G whose variables occur in the same constraint of (PP). 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 dual node v to approximate PP and DP then works as follows:

1: Run graph decomposition of [12] on $\mathcal{G}$;
2: **if** $v \in \mathcal{S}$ **then**
3: **send** IDs of primal neighbors to $\ell(v)$.
4: **fi**;
5: **if** $v = \ell(u)$ for some $u \in \mathcal{S}$ **then**
6: compute local PLP/DLP (cf. Lemma 5.1) of variables of $u \in \mathcal{S}$ for which $v = \ell(u)$.
7: **send** resulting values to nodes holding the respective variables.
8: **fi**

The primal 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, dual variables belonging to different local LPs cannot occur in the same dual constraint (otherwise, the according dual nodes had to be neighbors in $\mathcal{G}$). The analogous fact holds for primal variables since dual nodes belonging to different local LPs have distance at least 6 on G and thus primal 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 5.1).

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.[4] Finally, N can be chosen to optimize the approximation ratio.

THEOREM 5.1. *Let $N = \alpha e n^{1/k} \ln n$ 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, yields an $\alpha e n^{1/k}$ approximation of (*PP*)/(*DP*) w.h.p. The algorithm requires $O(k)$ rounds.*

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

6 Lower Bound

We derive time lower bounds for distributed approximability of packing problems, even in the $\mathcal{LOCAL}$ model. More precisely, we prove lower bounds for the most basic packing problems, the fractional maximum matching problem (FMM). Our general approach follows [9] in which similar results are obtained for minimum vertex cover which is a covering problem. Specifically, our packing lower bound graph is structurally similar (although with subtle differences) to the one used in [9].

Let E_i denote the set of edges incident to node v_i. FMM is the natural LP relaxation of MM and defined as $\max \sum_{e_j \in E} y_j$, subject to $\sum_{v_j \in E_i} y_j \leq 1, \forall v_i \in V$ and $y_j \geq 0, \forall e_j \in E$. The outcome of an edge's decision (y_j) in a k-local computation is entirely based on the information gathered within its k-neighborhood. The idea for the lower bound is to construct a graph family $G_k = (V, E)$ in which, after k rounds of communication, two adjacent edges see exactly the same graph topology. Informally speaking, both of them are equally qualified to join the matching. However, in G_k, taking the wrong decision will be ruinous and yields a suboptimal global approximation. The construction of G_k is a two step process. First, the general structure of G_k is defined using the concept of a *cluster-graph* CG_k. Secondly, we construct an instance of G_k obeying the properties imposed by CG_k.

[4]The primal and dual variables x_i and y_j are divided by $\min_{j \in N_i} \frac{1}{b_j} \sum_\ell a_{j\ell} x_\ell$ and $\max_{i \in N_j} \frac{1}{c_i} \sum_\ell a_{\ell i} y_\ell$, respectively.

6.1 The Cluster Graph

The nodes $v \in V$ in G_k are grouped into disjoint sets which are linked to each other as bipartite graphs. The structural properties of G_k are described using a directed *cluster graph* $CG_k = (\mathcal{C}, \mathcal{A})$ with doubly labeled arcs $\ell : \mathcal{A} \to \mathbb{N} \times \mathbb{N}$. A node $C \in \mathcal{C}$ represents a *cluster*, i.e., one of the disjoint sets of nodes in G_k. An arc $a = (C, D) \in \mathcal{A}$ with $\ell(a) = (\delta^c, \delta^d)$ denotes that the clusters C and D are linked as a bipartite graph in which each node $u \in C$ has degree δ^c and each node $v \in D$ has degree δ^d. It follows that $|C| \cdot \delta^c = |D| \cdot \delta^d$.

The cluster graph consists of two equal subgraphs, so-called *cluster-trees* CT_k as defined in [9]. In CG_k, we additionally add an arc $\ell(C_i, C_i') := (1, 1)$ between two corresponding nodes of the two cluster trees. Formally, CT_k and CG_k are defined as follows. We call clusters adjacent to exactly one other cluster *leaf-clusters*, and all other clusters *inner-clusters*.

DEFINITION 6.1. *[9] For a given δ and a positive integer k, the **cluster tree** CT_k is recursively defined as follows:*

$$\begin{aligned} CT_1 &:= (\mathcal{C}_1, \mathcal{A}_1), \quad \mathcal{C}_1 := \{C_0, C_1, C_2, C_3\} \\ \mathcal{A}_1 &:= \{(C_0, C_1), (C_0, C_2), (C_1, C_3)\} \\ \ell(C_0, C_1) &:= (\delta, \delta^2), \quad \ell(C_0, C_2) := (\delta^2, \delta^3), \\ \ell(C_1, C_3) &:= (\delta, \delta^2) \end{aligned}$$

Given CT_{k-1}, CT_k is obtained in two steps: For each inner-cluster C_i, add a new leaf-cluster C_i' with $\ell(C_i, C_i') := (\delta^{k+1}, \delta^{k+2})$. For each leaf-cluster C_i with $(C_p, C_i) \in \mathcal{A}$ and $\ell(C_p, C_i) = (\delta^p, \delta^{p+1})$, add new leaf-clusters C_j' with $\ell(C_i, C_j') := (\delta^j, \delta^{j+1})$ for $j = 1 \ldots k+1, j \neq p+1$.

DEFINITION 6.2. *Let T_k and T_k' be two instances of CT_k. Further, let C_i and C_i' be corresponding clusters in T_k and T_k', respectively. We obtain the **cluster graph** CG_k by adding an arc $\ell(C_i, C_i') := (1, 1)$ for all clusters $C_i \in CT_k$. Further, we define $n_0 := |C_0 \cup C_0'|$. This uniquely defines the size of all clusters.*

Figure 1 shows CT_2 and CG_2. The shaded subgraphs correspond to CT_1 and CG_1, respectively, the dashed lines represent the links $\ell(C_i, C_i') := (1, 1)$. Note that neither CT_k nor CG_k define the adjacency on the level of nodes. They merely prescribe for each node the number of neighbors in each cluster. We define $S_0 := C_0 \cup C_0'$ and $S_1 := C_1 \cup C_1'$. The *layer* of a cluster is the distance to C_0 in the cluster tree. T_k and T_k' denote the two cluster trees constituting CG_k.

6.2 The Lower Bound Graph G_k

Having defined the cluster graph CG_k, it is now our goal to obtain

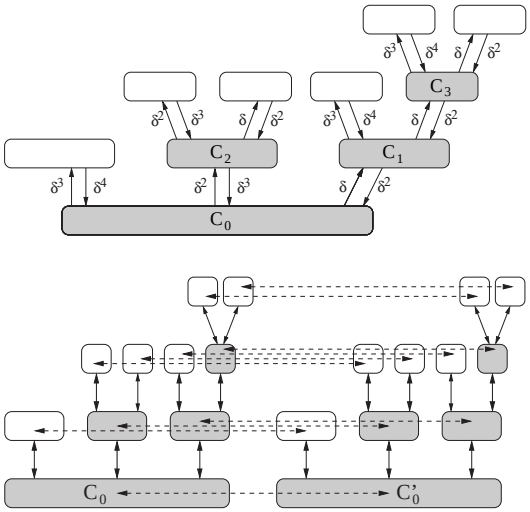

Figure 1: Cluster-Tree CT_2 and Cluster-Graph CG_2.

a realization of G_k which has the structure imposed by CG_k and features the additional property that there are no short cycles. As we must prove that the topologies seen by nodes in S_0 and S_1 are identical, the absence of short cycles is of great help. Particularly, if there are no cycles of length $2k+1$ and less, all nodes see a tree locally. The *girth* of a graph G, denoted by $g(G)$, is the length of the shortest cycle in G. Lemma 6.1 states that it is indeed possible to construct G_k as described above.

LEMMA 6.1. *If $k+1 \leq \delta/2$, G_k can be constructed such that the following conditions hold:*

(I) *G_k follows the structure of CG_k.*

(II) *The girth of G_k is at least $g(G_k) \geq 2k+1$.*

(III) *G_k has $n \leq 4^{2k}\delta^{4k^2}$ nodes.*

Next we show that all nodes in S_0 and S_1 have the same view and consequently, all edges in E' see the same topology. Using the following result from [9] facilitates this task.

LEMMA 6.2. *[9] Let G_k be an instance of a cluster tree CT_k with girth $g(G_k) \geq 2k+1$. The views of all nodes in clusters C_0 and C_1 are identical up to distance k.*

Because G_k has girth at least $2k+1$ by Lemma 6.1, the two cluster-trees T_k and T'_k constituting G_k must have girth $2k+1$ as well. It follows from Lemma 6.2 that the desired *equality of views* holds for both T_k and T'_k. Based on this fact, it is now easy to show that equality of views holds in G_k, too.

LEMMA 6.3. *Let G_k be an instance of a cluster graph CG_k with girth $g(G_k) \geq 2k+1$. The views of all nodes in clusters S_0 and S_1 are identical up to distance k.*

6.3 Analysis We now derive the lower bounds on the approximation ratio for k-local FMM algorithms. Let OPT be the optimal solution for FMM and let ALG be the solution computed by any algorithm. All nodes in S_0 and S_1 have the same view and therefore, every edge in E' sees the same topology $\mathcal{V}_{e,k}$.

LEMMA 6.4. *When applied to $G_k = (V, E)$ as constructed in Subsection 6.2, any distributed, possibly randomized algorithm which runs for at most k rounds computes, in expectation, a solution of at most $ALG \leq |S_0|/(2\delta^2) + (|V| - |S_0|)$.*

Proof. The fractional value assigned to $e_i = (u, v)$ by an algorithm is denoted by y_i. The decision of which value y_i is assigned to edge e_i depends only on the view the topologies $\mathcal{T}_{u,k}$ and $\mathcal{T}_{v,k}$ and the labelings $\mathcal{L}(\mathcal{T}_{u,k})$ and $\mathcal{L}(\mathcal{T}_{v,k})$, which e_i can collect during the k communication rounds. Assuming that the labeling of G_k is chosen uniformly at random, the labeling $\mathcal{L}(\mathcal{T}_{u,k})$ for any node u is also chosen uniformly at random.

All edges connecting nodes in S_0 and S_1 see the same topology. If the labels are chosen uniformly at random, it follows that the distribution of the views and therefore the distribution of the y_i is the same for all those edges. We call the random variables describing the distribution of the y_i, Y_i. Let $u \in S_1$ be a node of S_1. The node u has δ^2 neighbors in S_0. Therefore, for edges e_i between nodes in S_0 and S_1, by linearity of expectation, $\mathrm{E}[Y_i] \leq 1/\delta^2$ because otherwise there exist labelings for which the calculated solution is not feasible. By Lemma 6.3, edges e_j with both end-points in S_0 have the same view as edges between S_0 and S_1. Hence, also for the value y_j of e_j, $\mathrm{E}[Y_j] \leq 1/\delta^2$ must hold. There are $|S_0|/2$ such edges and therefore the expected total value contributed by edges between two nodes in S_0 is at most $|S_0|/(2\delta^2)$.

All edges which do not connect two nodes in S_0, have one end-point in $V \setminus S_0$. In order to get a feasible solution, the total value of all edges adjacent to a set of nodes V', can be at most $|V'|$. This can for example be seen by looking at the dual problem, a kind of minimum vertex cover where some edges only have one end node. Taking all nodes of V' (assigning 1 to the respective variables) yields a feasible solution for this vertex cover problem. The claim now follows by applying Yao's minimax principle.

We now derive the lower bound. Lemma 6.4 gives an upper bound on the number of nodes chosen by any k-local FMM algorithm. Choosing all edges within S_0 is feasible, hence, $|OPT| \geq |S_0|/2$. In order to establish a relationship between n, $|S_0|$, δ, and k, we bound n as $n \leq |S_0|(1 + \frac{k+1}{\delta-(k+1)})$ using a geometric series. The second lower bound then follows easily from $\Delta = \delta^{k+2}$.

THEOREM 6.1. *For all pairs (n,k) and (Δ,k), there are graphs G and a constant $c \geq 1/4$, such that in k communication rounds, every distributed algorithm for FMM on G has approximation ratios at least $\Omega(n^{c/k^2}/k)$ and $\Omega\big(\Delta^{1/k}/k\big)$, respectively.*

By setting $k = \beta\sqrt{\log n/\log\log n}$ and $k = \beta\log\Delta/\log\log\Delta$, respectively, for a constant $\beta > 0$, we obtain the following corollary.

COROLLARY 6.1. *In order to obtain a polylogarithmic or constant approximation ratio, every distributed algorithm for FMM requires at least $\Omega(\sqrt{\log n/\log\log n})$ and $\Omega(\log\Delta/\log\log\Delta)$ communication rounds. The same lower bounds hold for the construction of maximal matchings and maximal independent sets.*

Remark: The algorithm in Section 5 achieves a polylogarithmic approximation in $O(\log\Delta/\log\log\Delta)$ communication rounds. Therefore, for polylogarithmic approximations, our lower bound for FMM is *tight*.

7 Conclusions

It is interesting to view *local computation* in a wider context of computational models. *Approximation algorithms* and *online algorithms* try to bound the degradation of a globally optimal solution caused by limited computational resources and knowledge about the future, respectively. More recently, the *"price of anarchy,"* has been proposed to measure the suboptimality resulting from selfish individuals [20]. In a similar spirit, our paper sheds light on the *price of locality*, i.e., the degradation of a globally optimal solution if each individual's knowledge is restricted to its neighborhood or local environment. Specifically, the upper and lower bounds presented in this paper characterize the achievable trade-off between local information and the quality of a global solution of covering and packing problems.

References

[1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986.

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

[3] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–724, 2003.

[4] M. Elkin. Unconditional Lower Bounds on the Time-Approximation Tradeoffs for the Distributed Minimum Spanning Tree Problem. In *Proc. of the 36th ACM Symposium on Theory of Computing (STOC)*, 2004.

[5] F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Computing*, 16(2-3):121–163, 2003.

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

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

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

[9] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! In *Proc. of the 23^{rd} ACM Symp. on Principles of Distributed Computing (PODC)*, pages 300–309, 2004.

[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, 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] M. Naor and L. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

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

[17] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

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

[19] S. Rajagopalan and V. Vazirani. Primal-Dual RNC Approximation Algorithms for Set Cover and Covering Integer Programs. *SIAM Journal on Computing*, 28:525–540, 1998.

[20] T. Roughgarden and E. Tardos. How Bad is Selfish Routing? In *Proc. of the 41^{th} Symp. on Foundations of Computer Science (FOCS)*, pages 93–102, 2000.

[21] N. Young. Randomized Rounding without Solving the Linear Program. In *Proc. of the 6th Symposium on Discrete Algorithms (SODA)*, pages 170–178, 1995.

[22] N. Young. Sequential and Parallel Algorithms for Mixed Packing and Covering. In *Proc. of the 42^{nd} Symposium on Foundations of Computer Science (FOCS)*, pages 538–546, 2001.