Tight Local Approximation Results for Max-Min Linear Programs

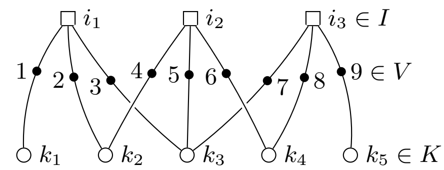
Patrik Floréen, Marja Hassinen, Petteri Kaski, and Jukka Suomela

Helsinki Institute for Information Technology HIIT
Helsinki University of Technology and University of Helsinki
P.O. Box 68, FI-00014 University of Helsinki, Finland
patrik.floreen@cs.helsinki.fi, marja.hassinen@cs.helsinki.fi,
petteri.kaski@cs.helsinki.fi, jukka.suomela@cs.helsinki.fi

Abstract. In a bipartite max-min LP, we are given a bipartite graph $\mathcal{G} = (V \cup I \cup K, E)$, where each agent $v \in V$ is adjacent to exactly one constraint $i \in I$ and exactly one objective $k \in K$. Each agent v controls a variable x_v. For each $i \in I$ we have a nonnegative linear constraint on the variables of adjacent agents. For each $k \in K$ we have a nonnegative linear objective function of the variables of adjacent agents. The task is to maximise the minimum of the objective functions. We study local algorithms where each agent v must choose x_v based on input within its constant-radius neighbourhood in $\mathcal{G}$. We show that for every $\epsilon > 0$ there exists a local algorithm achieving the approximation ratio $\Delta_I(1 - 1/\Delta_K) + \epsilon$. We also show that this result is the best possible – no local algorithm can achieve the approximation ratio $\Delta_I(1 - 1/\Delta_K)$. Here Δ_I is the maximum degree of a vertex $i \in I$, and Δ_K is the maximum degree of a vertex $k \in K$. As a methodological contribution, we introduce the technique of graph unfolding for the design of local approximation algorithms.

1 Introduction

As a motivating example, consider the task of data gathering in the following sensor network.



Each open circle is a sensor node $k \in K$, and each box is a relay node $i \in I$. The graph depicts the communication links between sensors and relays. Each sensor produces data which needs to be routed via adjacent relay nodes to a base station (not shown in the figure).

For each pair consisting of a sensor k and an adjacent relay i, we need to decide how much data is routed from k via i to the base station. For each such decision, we introduce an *agent* $v \in V$; these are shown as black dots in the figure. We arrive at a bipartite graph $\mathcal{G}$ where the set of vertices is $V \cup I \cup K$ and each edge joins an agent $v \in V$ to a node $j \in I \cup K$.

Associated with each agent $v \in V$ is a variable x_v. Each relay constitutes a bottleneck: the relay has a limited battery capacity, which sets a limit on the total amount of data that can be forwarded through it. The task is to maximise the minimum amount of data gathered from a sensor node. In our example, the variable x_2 is the amount of data routed from the sensor k_2 via the relay i_1, the battery capacity of the relay i_1 is an upper bound for $x_1 + x_2 + x_3$, and the amount of data gathered from the sensor node k_2 is $x_2 + x_4$. Assuming that the maximum capacity of a relay is 1, the optimisation problem is to

$$\begin{array}{ll} \text{maximise} & \min \{x_1,\ x_2 + x_4,\ x_3 + x_5 + x_7,\ x_6 + x_8,\ x_9\} \\ \text{subject to} & x_1 + x_2 + x_3 \le 1, \\ & x_4 + x_5 + x_6 \le 1, \\ & x_7 + x_8 + x_9 \le 1, \\ & x_1, x_2, \dots, x_9 \ge 0. \end{array} \tag{1}$$

In this work, we study *local algorithms* [1] for solving max-min linear programs (LPs) such as (1). In a local algorithm, each agent $v \in V$ must choose the value x_v solely based on its constant-radius neighbourhood in the graph $\mathcal{G}$. Such algorithms provide an extreme form of scalability in distributed systems; among others, a change in the topology of $\mathcal{G}$ affects the values x_v only in a constant-radius neighbourhood.

1.1 Max-Min Linear Programs

Let $\mathcal{G} = (V \cup I \cup K, E)$ be a bipartite, undirected communication graph where each edge $e \in E$ is of the form $\{v, j\}$ with $v \in V$ and $j \in I \cup K$. The elements $v \in V$ are called *agents*, the elements $i \in I$ are called *constraints*, and the elements $k \in K$ are called *objectives*; the sets V, I, and K are disjoint. We define $V_i = \{v \in V : \{v, i\} \in E\}$, $V_k = \{v \in V : \{v, k\} \in E\}$, $I_v = \{i \in I : \{v, i\} \in E\}$, and $K_v = \{k \in K : \{v, k\} \in E\}$ for all $i \in I$, $k \in K$, $v \in V$.

We assume that $\mathcal{G}$ is a bounded-degree graph; in particular, we assume that $|V_i| \le \Delta_I$ and $|V_k| \le \Delta_K$ for all $i \in I$ and $k \in K$ for some constants Δ_I and Δ_K.

A *max-min linear program* associated with $\mathcal{G}$ is defined as follows. Associate a variable x_v with each agent $v \in V$, associate a coefficient $a_{iv} \ge 0$ with each edge $\{i, v\} \in E$, $i \in I$, $v \in V$, and associate a coefficient $c_{kv} \ge 0$ with each edge $\{k, v\} \in E$, $k \in K$, $v \in V$. The task is to

$$\begin{array}{llll} \text{maximise} & \omega = \min_{k \in K} \sum_{v \in V_k} c_{kv} x_v & & \\ \text{subject to} & \sum_{v \in V_i} a_{iv} x_v \le 1 & \forall\, i \in I, \\ & x_v \ge 0 & \forall\, v \in V. \end{array} \tag{2}$$

We write ω^* for the optimum of (2).

1.2 Special Cases of Max-Min LPs

A max-min LP is a generalisation of a *packing LP*. Namely, in a packing LP there is only one linear nonnegative function to maximise, while in a max-min LP the goal is to maximise the minimum of multiple nonnegative linear functions.

Our main focus is on the *bipartite version* of the max-min LP problem. In the bipartite version we have $|I_v| = |K_v| = 1$ for each $v \in V$. We also define the 0/1 *version* [2]. In that case we have $a_{iv} = 1$ and $c_{kv} = 1$ for all $v \in V, i \in I_v, k \in K_v$. Our example (1) is both a bipartite max-min LP and a 0/1 max-min LP.

The *distance* between a pair of vertices $s, t \in V \cup I \cup K$ in $\mathcal{G}$ is the number of edges on a shortest path connecting s and t in $\mathcal{G}$. We write $B_{\mathcal{G}}(s, r)$ for the set of vertices within distance at most r from s. We say that $\mathcal{G}$ has *bounded relative growth* $1 + \delta$ *beyond radius* $R \in \mathbb{N}$ if

$$\frac{|V \cap B_{\mathcal{G}}(v, r+2)|}{|V \cap B_{\mathcal{G}}(v, r)|} \leq 1 + \delta \qquad \text{for all } v \in V,\ r \geq R.$$

Any bounded-degree graph $\mathcal{G}$ has a constant upper bound for δ. Regular grids are a simple example of a family of graphs where δ approaches 0 as R increases [3].

1.3 Local Algorithms and the Model of Computation

A local algorithm [1] is a distributed algorithm in which the output of a node is a function of input available within a fixed-radius neighbourhood; put otherwise, the algorithm runs in a constant number of communication rounds. In the context of distributed max-min LPs, the exact definition is as follows.

We say that the *local input* of a node $v \in V$ consists of the sets I_v and K_v and the coefficients a_{iv}, c_{kv} for all $i \in I_v, k \in K_v$. The local input of a node $i \in I$ consists of V_i and the local input of a node $k \in K$ consists of V_k. Furthermore, we assume that either (a) each node has a *unique identifier* given as part of the local input to the node [1, 4]; or, (b) each vertex independently introduces an ordering of the edges incident to it. The latter, strictly weaker, assumption is often called *port numbering* [5]; in essence, each edge $\{s, t\}$ in $\mathcal{G}$ has two natural numbers associated with it: the port number in s and the port number in t.

Let $\mathcal{A}$ be a deterministic distributed algorithm executed by each of the nodes of $\mathcal{G}$ that finds a feasible solution x to any max-min LP (2) given locally as input to the nodes. Let $r \in \mathbb{N}$ be a constant independent of the input. We say that $\mathcal{A}$ is a *local algorithm* with *local horizon* r if, for every agent $v \in V$, the output x_v is a function of the local input of the nodes in $B_{\mathcal{G}}(v, r)$. Furthermore, we say that $\mathcal{A}$ has the *approximation ratio* $\alpha \geq 1$ if $\sum_{v \in V_k} c_{kv} x_v \geq \omega^*/\alpha$ for all $k \in K$.

1.4 Contributions and Prior Work

The following local approximability result is the main contribution of this paper.

Theorem 1. *For any $\Delta_I \geq 2$, $\Delta_K \geq 2$, and $\epsilon > 0$, there exists a local approximation algorithm for the bipartite max-min LP problem with the approximation ratio $\Delta_I(1 - 1/\Delta_K) + \epsilon$. The algorithm assumes only port numbering.*

We also show that the positive result of Theorem 1 is tight. Namely, we prove a matching lower bound on local approximability, which holds even if we assume both 0/1 coefficients and unique node identifiers.

Theorem 2. *For any $\Delta_I \geq 2$ and $\Delta_K \geq 2$, there exists no local approximation algorithm for the max-min LP problem with the approximation ratio $\Delta_I(1 - 1/\Delta_K)$. This holds even in the case of a bipartite, 0/1 max-min LP and with unique node identifiers given as input.*

Considering Theorem 1 in light of Theorem 2, we find it somewhat surprising that unique node identifiers are not required to obtain the best possible local approximation algorithm for bipartite max-min LPs.

In terms of earlier work, Theorem 1 is an improvement on the *safe algorithm* [3, 6] which achieves the approximation ratio Δ_I. Theorem 2 improves upon the earlier lower bound $(\Delta_I + 1)/2 - 1/(2\Delta_K - 2)$ [3]; here it should be noted that our definition of the local horizon differs by a constant factor from earlier work [3] due to the fact that we have adopted a more convenient graph representation instead of a hypergraph representation.

In the context of packing and covering LPs, it is known [7] that any approximation ratio $\alpha > 1$ can be achieved by a local algorithm, assuming a bounded-degree graph and bounded coefficients. Compared with this, the factor $\Delta_I(1 - 1/\Delta_K)$ approximation in Theorem 1 sounds somewhat discouraging considering practical applications. However, the constructions that we use in our negative results are arguably far from the structure of, say, a typical real-world wireless network. In prior work [3] we presented a local algorithm that achieves a factor $1 + (2 + o(1))\delta$ approximation assuming that $\mathcal{G}$ has bounded relative growth $1 + \delta$ beyond some constant radius R; for a small δ, this is considerably better than $\Delta_I(1 - 1/\Delta_K)$ for general graphs. We complement this line of research on bounded relative growth graphs with a negative result that matches the prior positive result [3] up to constants.

Theorem 3. *Let $\Delta_I \geq 3$, $\Delta_K \geq 3$, and $0 < \delta < 1/10$. There exists no local approximation algorithm for the max-min LP problem with an approximation ratio less than $1 + \delta/2$. This holds even in the case of a bipartite max-min LP where the graph $\mathcal{G}$ has bounded relative growth $1 + \delta$ beyond some constant radius R.*

From a technical perspective, the proof of Theorem 1 relies on two ideas: *graph unfolding* and the idea of *averaging local solutions* of local LPs.

We introduce the unfolding technique in Sect. 2. In essence, we expand the finite input graph $\mathcal{G}$ into a possibly infinite tree $\mathcal{T}$. Technically, $\mathcal{T}$ is the *universal covering* of $\mathcal{G}$ [5]. While such unfolding arguments have been traditionally used to obtain impossibility results [8] in the context of distributed algorithms, here we use such an argument to simplify the design of local algorithms. In retrospect, our earlier approximation algorithm for 0/1 max-min LPs [2] can be interpreted as an application of the unfolding technique.

The idea of averaging local LPs has been used commonly in prior work on distributed algorithms [3, 7, 9, 10]. Our algorithm can also be interpreted as a generalisation of the safe algorithm [6] beyond local horizon $r = 1$.

To obtain our negative results – Theorems 2 and 3 – we use a construction based on regular high-girth graphs. Such graphs [11–14] have been used in prior work to obtain impossibility results related to local algorithms [4, 7, 15].

2 Graph Unfolding

Let $\mathcal{H} = (V, E)$ be a connected undirected graph and let $v \in V$. Construct a (possibly infinite) rooted tree $\mathcal{T}_v = (\bar{V}, \bar{E})$ and a labelling $f_v : \bar{V} \to V$ as follows. First, introduce a vertex $\bar{v}$ as the root of $\mathcal{T}_v$ and set $f_v(\bar{v}) = v$. Then, for each vertex u adjacent to v in $\mathcal{H}$, add a new vertex $\bar{u}$ as a child of $\bar{v}$ and set $f_v(\bar{u}) = u$. Then expand recursively as follows. For each unexpanded $\bar{t} \neq \bar{v}$ with parent $\bar{s}$, and each $u \neq f(\bar{s})$ adjacent to $f(\bar{t})$ in $\mathcal{H}$, add a new vertex $\bar{u}$ as a child of $\bar{t}$ and set $f_v(\bar{u}) = u$. Mark $\bar{t}$ as expanded.

This construction is illustrated in Fig. 1. Put simply, we traverse $\mathcal{H}$ in a breadth-first manner and treat vertices revisited due to a cycle as new vertices; in particular, the tree $\mathcal{T}_v$ is finite if and only if $\mathcal{H}$ is acyclic.

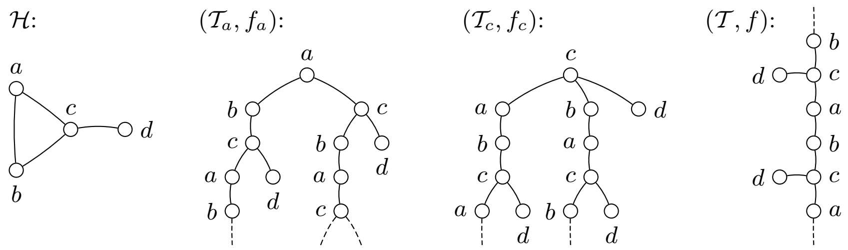


Fig. 1. An example graph $\mathcal{H}$ and its unfolding $(\mathcal{T}, f)$.

The rooted, labelled trees $(\mathcal{T}_v, f_v)$ obtained in this way for different choices of $v \in V$ are isomorphic viewed as unrooted trees [5]. For example, the infinite labelled trees $(\mathcal{T}_a, f_a)$ and $(\mathcal{T}_c, f_c)$ in Fig. 1 are isomorphic and can be transformed into each other by rotations. Thus, we can define the *unfolding* of $\mathcal{H}$ as the labelled tree $(\mathcal{T}, f)$ where $\mathcal{T}$ is the unrooted version of $\mathcal{T}_v$ and $f = f_v$; up to isomorphism, this is independent of the choice of $v \in V$.

2.1 Unfolding in Graph Theory and Topology

We briefly summarise the graph theoretic and topological background related to the unfolding $(\mathcal{T}, f)$ of $\mathcal{H}$.

From a graph theoretic perspective, using the terminology of Godsil and Royle [17, §6.8], the surjection f is a homomorphism from $\mathcal{T}$ to $\mathcal{H}$. Moreover, it is a *local isomorphism*: the neighbours of $\bar{v} \in \bar{V}$ are in one-to-one correspondence with the neighbours of $f(\bar{v}) \in V$. A surjective local isomorphism f is a *covering map* and $(\mathcal{T}, f)$ is a *covering graph* of $\mathcal{H}$.

Covering maps in graph theory can be interpreted as a special case of covering maps in topology: $\mathcal{T}$ is a *covering space* of $\mathcal{H}$ and f is, again, a covering map. See, e.g., Hocking and Young [18, §4.8] or Munkres [19, §53].

In topology, a simply connected covering space is called a *universal covering space* [18, §4.8], [19, §80]. An analogous graph-theoretic concept is a tree: unfolding $\mathcal{T}$ of $\mathcal{H}$ is equal to the *universal covering* $\mathcal{U}(\mathcal{H})$ of $\mathcal{H}$ as defined by Angluin [5].

Unfortunately, the term "covering" is likely to cause confusion in the context of graphs. The term "lift" has been used for a covering graph [13, 20]. We have borrowed the term "unfolding" from the field of model checking; see, e.g., Esparza and Heljanko [21].

2.2 Unfolding and Local Algorithms

Let us now view the graph $\mathcal{H}$ as the communication graph of a distributed system, and let $(\mathcal{T}, f)$ be the unfolding of $\mathcal{H}$. Even if $\mathcal{T}$ in general is countably infinite, a local algorithm $\mathcal{A}$ with local horizon r can be designed to operate at a node of $v \in \mathcal{H}$ exactly *as if* it was a node $\bar{v} \in f^{-1}(v)$ in the communication graph $\mathcal{T}$. Indeed, assume that the local input at $\bar{v}$ is identical to the local input at $f(\bar{v})$, and observe that the radius r neighbourhood of the node $\bar{v}$ in $\mathcal{T}$ is equal to the rooted tree $\mathcal{T}_v$ trimmed to depth r; let us denote this by $\mathcal{T}_v(r)$. To gather the information in $\mathcal{T}_v(r)$, it is sufficient to gather information on all walks of length at most r starting at v in $\mathcal{H}$; using port numbering, the agents can detect and discard walks that consecutively traverse the same edge.

Assuming that only port numbering is available, the information in $\mathcal{T}_v(r)$ is in fact *all* that the agent v can gather. Indeed, to assemble, say, the subgraph of $\mathcal{H}$ induced by $B_{\mathcal{H}}(v, r)$, the agent v in general needs to distinguish between a short cycle and a long path, and these are indistinguishable without node identifiers.

2.3 Unfolding and Max-Min LPs

Let us now consider a max-min LP associated with a graph $\mathcal{G}$. The unfolding of $\mathcal{G}$ leads in a natural way to the unfolding of the max-min LP. We show in this section that in order to prove Theorem 1, it is sufficient to design a local approximation algorithm for unfoldings of a max-min LP.

Unfolding requires us to consider max-min LPs where the underlying communication graph is countably infinite. The graph is always a bounded-degree graph, however. This allows us to circumvent essentially all of the technicalities otherwise encountered with infinite problem instances; cf. Anderson and Nash [16]. For the purposes of this work, it suffices to define that x is a *feasible solution with utility at least* ω if (x, ω) satisfies

$$\begin{aligned} \textstyle\sum_{v \in V_k} c_{kv} x_v &\ge \omega \qquad \forall\, k \in K, \\ \textstyle\sum_{v \in V_i} a_{iv} x_v &\le 1 \qquad \forall\, i \in I, \\ x_v &\ge 0 \qquad \forall\, v \in V. \end{aligned} \tag{3}$$

Observe that each of the sums in (3) is finite. Furthermore, this definition is compatible with the finite max-min LP defined in Sect. 1.1. Namely, if ω^* is the optimum of a finite max-min LP, then there exists a feasible solution x^* with utility at least ω^*.

Let $\mathcal{G} = (V \cup I \cup K, E)$ be the underlying finite communication graph. Unfold $\mathcal{G}$ to obtain a (possibly infinite) tree $\mathcal{T} = (\bar{V} \cup \bar{I} \cup \bar{K}, \bar{E})$ with a labelling f. Extend this to an unfolding of the max-min LP by associating a variable $x_{\bar{v}}$ with each agent $\bar{v} \in \bar{V}$, the coefficient $a_{\bar{\iota}\bar{v}} = a_{f(\bar{\iota}),f(\bar{v})}$ for each edge $\{\bar{\iota}, \bar{v}\} \in \bar{E}$, $\bar{\iota} \in \bar{I}$, $\bar{v} \in \bar{V}$, and the coefficient $c_{\bar{\kappa}\bar{v}} = c_{f(\bar{\kappa}),f(\bar{v})}$ for each edge $\{\bar{\kappa}, \bar{v}\} \in \bar{E}$, $\bar{\kappa} \in \bar{K}$, $\bar{v} \in \bar{V}$. Furthermore, assume an arbitrary port numbering for the edges incident to each of the nodes in $\mathcal{G}$, and extend this to a port numbering for the edges incident to each of the nodes in $\mathcal{T}$ so that the port numbers at the ends of each edge $\{\bar{u}, \bar{v}\} \in \bar{E}$ are identical to the port numbers at the ends of $\{f(\bar{u}), f(\bar{v})\}$.

Lemma 1. *Let $\bar{\mathcal{A}}$ be a local algorithm for unfoldings of a family of max-min LPs and let $\alpha \geq 1$. Assume that the output x of $\bar{\mathcal{A}}$ satisfies $\sum_{v \in V_k} c_{kv} x_v \geq \omega'/\alpha$ for all $k \in K$ if there exists a feasible solution with utility at least ω'. Furthermore, assume that $\bar{\mathcal{A}}$ uses port numbering only. Then, there exists a local approximation algorithm $\mathcal{A}$ with the approximation ratio α for this family of max-min LPs.*

Proof. Let x^* be an optimal solution of the original instance, with utility ω^*. Set $x_{\bar{v}} = x^*_{f(\bar{v})}$ to obtain a solution of the unfolding. This is a feasible solution because the variables of the agents adjacent to a constraint $\bar{\iota}$ in the unfolding have the same values as the variables of the agents adjacent to the constraint $f(\bar{\iota})$ in the original instance. By similar reasoning, we can show that this is a feasible solution with utility at least ω^*.

Construct the local algorithm $\mathcal{A}$ using the assumed algorithm $\bar{\mathcal{A}}$ as follows. Each node $v \in V$ simply behaves as if it was a node $\bar{v} \in f^{-1}(v)$ in the unfolding $\mathcal{T}$ and simulates $\bar{\mathcal{A}}$ for $\bar{v}$ in $\mathcal{T}$. By assumption, the solution x computed by $\bar{\mathcal{A}}$ in the unfolding has to satisfy

$$\begin{aligned} \textstyle\sum_{\bar{v} \in V_{\bar{\kappa}}} c_{\bar{\kappa}\bar{v}} x_{\bar{v}} &\geq \omega^*/\alpha \qquad && \forall\, \bar{\kappa} \in \bar{K}, \\ \textstyle\sum_{\bar{v} \in V_{\bar{\iota}}} a_{\bar{\iota}\bar{v}} x_{\bar{v}} &\leq 1 && \forall\, \bar{\iota} \in \bar{I}. \end{aligned}$$

Furthermore, if $f(\bar{u}) = f(\bar{v})$ for $\bar{u}, \bar{v} \in \bar{V}$, then the neighbourhoods of $\bar{u}$ and $\bar{v}$ contain precisely the same information (including the port numbering), so the deterministic $\bar{\mathcal{A}}$ must output the same value $x_{\bar{u}} = x_{\bar{v}}$. Giving the output $x_v = x_{\bar{v}}$ for any $\bar{v} \in f^{-1}(v)$ therefore yields a feasible, α-approximate solution to the original instance. □

We observe that Lemma 1 generalises beyond max-min LPs; we did not exploit the linearity of the constraints and the objectives.

3 Approximability Results

We proceed to prove Theorem 1. Let $\Delta_I \geq 2$, $\Delta_K \geq 2$, and $\epsilon > 0$ be fixed. By virtue of Lemma 1, it suffices to consider only bipartite max-min LPs where the graph $\mathcal{G}$ is a (finite or countably infinite) tree.

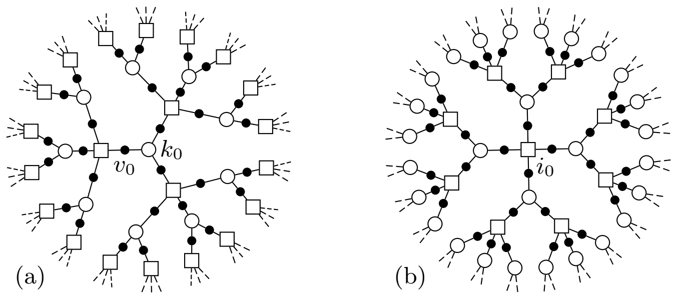


Fig. 2. Radius 6 neighbourhoods of (a) an objective $k_0 \in K$ and (b) a constraint $i_0 \in I$ in the regularised tree $\mathcal{G}$, assuming $\Delta_I = 4$ and $\Delta_K = 3$. The black dots represent agents $v \in V$, the open circles represent objectives $k \in K$, and the boxes represent constraints $i \in I$.

To ease the analysis, it will be convenient to *regularise* $\mathcal{G}$ to a countably infinite tree with $|V_i| = \Delta_I$ and $|V_k| = \Delta_K$ for all $i \in I$ and $k \in K$.

To this end, if $|V_i| < \Delta_I$ for some $i \in I$, add $\Delta_I - |V_i|$ new *virtual* agents as neighbours of i. Let v be one of these agents. Set $a_{iv} = 0$ so that no matter what value one assigns to x_v, it does not affect the feasibility of the constraint i. Then add a new virtual objective k adjacent to v and set, for example, $c_{kv} = 1$. As one can assign an arbitrarily large value to x_v, the virtual objective k will not be a bottleneck.

Similarly, if $|V_k| < \Delta_K$ for some $k \in K$, add $\Delta_K - |V_k|$ new virtual agents as neighbours of k. Let v be one of these agents. Set $c_{kv} = 0$ so that no matter what value one assigns to x_v, it does not affect the value of the objective k. Then add a new virtual constraint i adjacent to v and set, for example, $a_{iv} = 1$.

Now repeat these steps and grow virtual trees rooted at the constraints and objectives that had less than Δ_I or Δ_K neighbours. The result is a countably infinite tree where $|V_i| = \Delta_I$ and $|V_k| = \Delta_K$ for all $i \in I$ and $k \in K$. Observe also that from the perspective of a local algorithm it suffices to grow the virtual trees only up to depth r because then the radius r neighbourhood of each original node is indistinguishable from the regularised tree. The resulting topology is illustrated in Fig. 2 from the perspective of an original objective $k_0 \in K$ and an original constraint $i_0 \in I$.

3.1 Properties of Regularised Trees

For each $v \in V$ in a regularised tree $\mathcal{G}$, define $K(v, \ell) = K \cap B_{\mathcal{G}}(v, 4\ell+1)$, that is, the set of objectives k within distance $4\ell+1$ from v. For example, $K(v, 1)$ consists of 1 objective at distance 1, $\Delta_I - 1$ objectives at distance 3, and $(\Delta_K - 1)(\Delta_I - 1)$ objectives at distance 5; see Fig. 2a. In general, we have

$$|K(v, \ell)| = 1 + (\Delta_I - 1)\Delta_K n(\ell), \tag{4}$$

where

$$n(\ell) = \sum_{j=0}^{\ell-1} (\Delta_I - 1)^j (\Delta_K - 1)^j.$$

Let $k \in K$. If $u, v \in V_k$, $u \neq v$, then the objective at distance 1 from u is the same as the objective at distance 1 from v; therefore $K(u, 0) = K(v, 0)$. The objectives at distance 3 from u are at distance 5 from v, and the objectives at distance 5 from u are at distance 3 or 5 from v; therefore $K(u, 1) = K(v, 1)$. By a similar reasoning, we obtain

$$K(u, \ell) = K(v, \ell) \qquad \forall\, \ell \in \mathbb{N},\ k \in K,\ u, v \in V_k. \tag{5}$$

Let us then study a constraint $i \in I$. Define

$$K(i, \ell) = \bigcap_{v \in V_i} K(v, \ell) = K \cap B_{\mathcal{G}}(i, 4\ell) = K \cap B_{\mathcal{G}}(i, 4\ell - 2).$$

For example, $K(i, 2)$ consists of Δ_I objectives at distance 2 from the constraint i, and $\Delta_I(\Delta_K - 1)(\Delta_I - 1)$ objectives at distance 6 from the constraint i; see Fig. 2b. In general, we have

$$|K(i, \ell)| = \Delta_I n(\ell). \tag{6}$$

For adjacent $v \in V$ and $i \in I$, we also define $\partial K(v, i, \ell) = K(v, \ell) \setminus K(i, \ell)$. We have by (4) and (6)

$$|\partial K(v, i, \ell)| = 1 + (\Delta_I \Delta_K - \Delta_I - \Delta_K)\, n(\ell). \tag{7}$$

3.2 Local Approximation on Regularised Trees

It now suffices to meet Lemma 1 for bipartite max-min LPs in the case when the underlying graph $\mathcal{G}$ is a countably infinite regularised tree. To this end, let $L \in \mathbb{N}$ be a constant that we choose later; L depends only on Δ_I, Δ_K and ϵ.

Each agent $u \in V$ now executes the following algorithm. First, the agent gathers all objectives $k \in K$ within distance $4L + 1$, that is, the set $K(u, L)$. Then, for each $k \in K(u, L)$, the agent u gathers the radius $4L + 2$ neighbourhood of k; let $\mathcal{G}(k, L)$ be this subgraph. In total, the agent u accumulates information from distance $r = 8L + 3$ in the tree; this is the local horizon of the algorithm.

The structure of $\mathcal{G}(k, L)$ is a tree similar to the one shown in Fig. 2a. The leaf nodes of the tree $\mathcal{G}(k, L)$ are constraints. For each $k \in K(u, L)$, the agent u forms the constant-size *subproblem* of (2) restricted to the vertices of $\mathcal{G}(k, L)$ and solves it optimally using a deterministic algorithm; let x^{kL} be the solution. Once the agent u has solved the subproblem for every $k \in K(u, L)$, it sets

$$q = 1 / \big(\Delta_I + \Delta_I(\Delta_I - 1)(\Delta_K - 1)n(L)\big), \tag{8}$$

$$x_u = q \textstyle\sum_{k \in K(u,L)} x_u^{kL}. \tag{9}$$

This completes the description of the algorithm. In Sect. 3.3 we show that the computed solution x is feasible, and in Sect. 3.4 we establish a lower bound on the performance of the algorithm. Section 3.5 illustrates the algorithm with an example.

3.3 Feasibility

Because each x^{kL} is a feasible solution, we have

$$\textstyle\sum_{v\in V_i} a_{iv}x_v^{kL} \le 1 \qquad \forall \text{ non-leaf } i \in I \text{ in } \mathcal{G}(k,L), \tag{10}$$

$$a_{iv}x_v^{kL} \le 1 \qquad \forall \text{ leaf } i \in I,\ v \in V_i \text{ in } \mathcal{G}(k,L). \tag{11}$$

Let $i \in I$. For each subproblem $\mathcal{G}(k,L)$ with $v \in V_i$, $k \in K(i,L)$, the constraint i is a non-leaf vertex; therefore

$$\begin{aligned}\sum_{v\in V_i}\sum_{k\in K(i,L)} a_{iv}x_v^{kL} &= \sum_{k\in K(i,L)}\sum_{v\in V_i} a_{iv}x_v^{kL}\\ &\overset{(10)}{\le} \sum_{k\in K(i,L)} 1\\ &\overset{(6)}{=} \Delta_I\, n(L).\end{aligned} \tag{12}$$

For each subproblem $\mathcal{G}(k,L)$ with $v \in V_i$, $k \in \partial K(v,i,L)$, the constraint i is a leaf vertex; therefore

$$\begin{aligned}\sum_{v\in V_i}\sum_{k\in \partial K(v,i,L)} a_{iv}x_v^{kL} &\overset{(11)}{\le} \sum_{v\in V_i}\sum_{k\in \partial K(v,i,L)} 1\\ &\overset{(7)}{=} \Delta_I\,\big(1+(\Delta_I\Delta_K-\Delta_I-\Delta_K)\,n(L)\big).\end{aligned} \tag{13}$$

Combining (12) and (13), we can show that the constraint i is satisfied:

$$\begin{aligned}\sum_{v\in V_i} a_{iv}x_v &\overset{(9)}{=} q\sum_{v\in V_i} a_{iv}\sum_{k\in K(v,L)} x_v^{kL}\\ &= q\Bigg(\sum_{v\in V_i}\sum_{k\in K(i,L)} a_{iv}x_v^{kL}\Bigg) + q\Bigg(\sum_{v\in V_i}\sum_{k\in \partial K(v,i,L)} a_{iv}x_v^{kL}\Bigg)\\ &\le q\Delta_I n(L) \;+\; q\Delta_I\big(1+(\Delta_I\Delta_K-\Delta_I-\Delta_K)n(L)\big)\\ &\overset{(8)}{=} 1.\end{aligned}$$

3.4 Approximation Ratio

Consider an arbitrary feasible solution x' of the unrestricted problem (2) with utility at least ω'. This feasible solution is also a feasible solution of each finite subproblem restricted to $\mathcal{G}(k,L)$; therefore

$$\textstyle\sum_{v\in V_h} c_{hv}x_v^{kL} \ge \omega' \qquad \forall\ h \in K \text{ in } \mathcal{G}(k,L). \tag{14}$$

Define

$$\alpha \;=\; \frac{1}{q(1+(\Delta_I-1)\Delta_K n(L))}. \tag{15}$$

Consider an arbitrary $k \in K$ and $u \in V_k$. We have

$$\begin{aligned}\sum_{v \in V_k} c_{kv} x_v &= q \sum_{v \in V_k} c_{kv} \sum_{h \in K(v,L)} x_v^{hL} \\ &\overset{(5)}{=} q \sum_{h \in K(u,L)} \sum_{v \in V_k} c_{kv} x_v^{hL} \\ &\overset{(14)}{\geq} q \sum_{h \in K(u,L)} \omega' \\ &\overset{(4)}{\geq} q(1 + (\Delta_I - 1)\Delta_K n(L))\, \omega' \\ &\overset{(15)}{=} \omega'/\alpha.\end{aligned}$$

By (8) and (15), we have

$$\alpha = \Delta_I \left(1 - \frac{1}{\Delta_K + 1/((\Delta_I - 1)n(L))}\right).$$

For a sufficiently large L, we meet Lemma 1 with $\alpha < \Delta_I(1 - 1/\Delta_K) + \epsilon$. This completes the proof of Theorem 1.

3.5 An Example

Assume that $\Delta_I = 4$, $\Delta_K = 3$, and $L = 1$. For each $k \in K$, our approximation algorithm constructs and solves a subproblem; the structure of the subproblem is illustrated in Fig. 2a. Then we simply sum up the optimal solutions of each subproblem. For any $v \in V$, the variable x_v is involved in exactly $|K(v, L)| = 10$ subproblems.

First, consider an objective $k \in K$. The boundary of a subproblem always lies at a constraint, never at an objective. Therefore the objective k and all its adjacent agents $v \in V_k$ are involved in 10 subproblems. We satisfy the objective exactly 10 times, each time at least as well as in the global optimum.

Second, consider a constraint $i \in I$. The constraint may lie in the middle of a subproblem or at the boundary of a subproblem. The former happens in this case $|K(i, L)| = 4$ times; the latter happens $|V_i| \cdot |\partial K(v, i, L)| = 24$ times. In total, we use up the capacity available at the constraint i exactly 28 times. See Fig. 2b for an illustration; there are 28 objectives within distance 6 from the constraint $i_0 \in I$.

Finally, we scale down the solution by factor $q = 1/28$. This way we obtain a solution which is feasible and within factor $\alpha = 2.8$ of optimum. This is close to the lower bound $\alpha > 2.66$ from Theorem 2.

4 Inapproximability Results

We proceed to prove Theorems 2 and 3. Let $r = 4, 8, \dots$, $s \in \mathbb{N}$, $D_I \in \mathbb{Z}^+$, and $D_K \in \mathbb{Z}^+$ be constants whose values we choose later. Let $\mathcal{Q} = (I' \cup K', E')$ be a bipartite graph where the degree of each $i \in I'$ is D_I, the degree of each $k \in K'$ is D_K, and there is no cycle of length less than $g = 2(4s + 2 + r) + 1$. We first show that such graphs exist for all values of the parameters.

We say that a bipartite graph $\mathcal{G} = (V \cup U, E)$ is (a, b)-*regular* if the degree of each node in V is a and the degree of each node in U is b.

Lemma 2. *For any positive integers a, b and g, there exists an (a, b)-regular bipartite graph which has no cycle of length less than g.*

Proof (sketch). We slightly adapt a proof of a similar result for d-regular graphs [13, Theorem A.2] to our needs. We proceed by induction on g, for $g = 4, 6, 8, \ldots$.

For the base case $g = 4$, we can choose the complete bipartite graph $K_{b,a}$.

Next consider $g \geq 6$. Let $\mathcal{G} = (V \cup U, E)$ be an (a, b)-regular bipartite graph where the length of the shortest cycle is $c \geq g - 2$. Let $S \subseteq E$. Construct a graph $\mathcal{G}_S = (V_S \cup U_S, E_S)$ as follows:

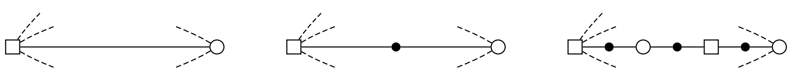
$$\begin{aligned} V_S &= \{0, 1\} \times V, \\ U_S &= \{0, 1\} \times U, \\ E_S &= \{\{(0, v), (0, u)\}, \{(1, v), (1, u)\} : \{v, u\} \in S\} \\ &\quad \cup \{\{(0, v), (1, u)\}, \{(1, v), (0, u)\} : \{v, u\} \in E \setminus S\}. \end{aligned}$$

The graph $\mathcal{G}_S$ is an (a, b)-regular bipartite graph. Furthermore, $\mathcal{G}_S$ has no cycle of length less than c. We proceed to show that there exists a subset S such that the number of cycles of length exactly c in $\mathcal{G}_S$ is strictly less than the number of cycles of length c in $\mathcal{G}$. Then by a repeated application of the same construction, we can conclude that there exists a graph which is an (a, b)-regular bipartite graph and which has no cycle of length c; that is, its girth is at least g.

We use the probabilistic method to show that the number of cycles of length c decreases for some $S \subseteq E$. For each $e \in E$, toss an independent and unbiased coin to determine whether $e \in S$. For each cycle $C \subseteq E$ of length c in $\mathcal{G}$, we have in $\mathcal{G}_S$ either two cycles of length c or one cycle of length $2c$, depending on the parity of $|C \cap S|$. The expected number of cycles of length c in $\mathcal{G}_S$ is therefore equal to the number of cycles of length c in $\mathcal{G}$. The choice $S = E$ doubles the number of such cycles; therefore some other choice necessarily decreases the number of such cycles. □

4.1 The Instance $\mathcal{S}$

Given the graph $\mathcal{Q} = (I' \cup K', E')$, we construct an instance of the max-min LP problem, $\mathcal{S}$. The underlying communication graph $\mathcal{G} = (V \cup I \cup K, E)$ is constructed as shown in the following figure.



$\mathcal{Q}$ $\qquad$ $\mathcal{G}$ for $s = 0$ $\qquad$ $\mathcal{G}$ for $s = 1$

Each edge $e = \{i, k\} \in E'$ is replaced by a path of length $4s + 2$: the path begins with the constraint $i \in I'$; then there are s segments of agent–objective–agent–constraint; and finally there is an agent and the objective $k \in K'$. There are no other edges or vertices in $\mathcal{G}$. For example, in the case of $s = 0$, $D_I = 4$, $D_K = 3$, and sufficiently large g, the graph $\mathcal{G}$ looks *locally* similar to the trees in Fig. 2, even though there may be long cycles.

The coefficients of the instance $\mathcal{S}$ are chosen as follows. For each objective $k \in K'$, we set $c_{kv} = 1$ for all $v \in V_k$. For each objective $k \in K \setminus K'$, we set $c_{kv} = D_K - 1$ for all $v \in V_k$. For each constraint $i \in I$, we set $a_{iv} = 1$. Observe that $\mathcal{S}$ is a bipartite max-min LP; furthermore, in the case $s = 0$, this is a 0/1 max-min LP. We can choose the port numbering in $\mathcal{G}$ in an arbitrary manner, and we can assign unique node identifiers to the vertices of $\mathcal{G}$ as well.

Lemma 3. *The utility of any feasible solution of $\mathcal{S}$ is at most*

$$\frac{D_K}{D_I} \cdot \frac{D_K - 1 + D_K D_I s - D_I s}{D_K - 1 + D_K s}.$$

Proof. Consider a feasible solution x of $\mathcal{S}$, with utility ω. We proceed to derive an upper bound for ω. For each $j = 0, 1, \ldots, 2s$, let $V(j)$ consist of agents $v \in V$ such that the distance to the nearest constraint $i \in I'$ is $2j + 1$. That is, $V(0)$ consists of the agents adjacent to an $i \in I'$ and $V(2s)$ consists of the agents adjacent to a $k \in K'$. Let $m = |E'|$; we observe that $|V(j)| = m$ for each j.

Let $X(j) = \sum_{v \in V(j)} x_v/m$. From the constraints $i \in I'$ we obtain

$$X(0) \;=\; \sum_{v \in V(0)} x_v/m \;=\; \sum_{i \in I'} \sum_{v \in V_i} a_{iv} x_v/m \;\leq\; \sum_{i \in I'} 1/m \;=\; |I'|/m \;=\; 1/D_I.$$

Similarly, from the objectives $k \in K'$ we obtain $X(2s) \geq \omega |K'|/m = \omega/D_K$.

From the objectives $k \in K \setminus K'$, taking into account our choice of the coefficients c_{kv}, we obtain the inequality $X(2t) + X(2t + 1) \geq \omega/(D_K - 1)$ for $t = 0, 1, \ldots, s - 1$. From the constraints $i \in I \setminus I'$, we obtain the inequality $X(2t+1) + X(2t+2) \leq 1$ for $t = 0, 1, \ldots, s-1$. Combining inequalities, we have

$$\begin{aligned} \omega/D_K - 1/D_I \;&\leq\; X(2s) - X(0) \\ &=\; \sum_{t=0}^{s-1} \Big(\big(X(2t + 1) + X(2t + 2)\big) - \big(X(2t) + X(2t + 1)\big) \Big) \\ &\leq\; s \cdot \big(1 - \omega/(D_K - 1)\big). \end{aligned}$$

The claim follows. ∎

4.2 The Instance $\mathcal{S}_k$

Let $k \in K'$. We construct another instance of the max-min LP problem, $\mathcal{S}_k$. The communication graph of $\mathcal{S}_k$ is the subgraph $\mathcal{G}_k$ of $\mathcal{G}$ induced by $B_{\mathcal{G}}(k, 4s+2+r)$. By the choice of g, there is no cycle in $\mathcal{G}_k$. As r is a multiple of 4, the leaves of the tree $\mathcal{G}_k$ are constraints. For example, in the case of $s = 0$, $D_I = 4$, $D_K = 3$, and $r = 4$, the graph $\mathcal{G}_k$ is isomorphic to the tree of Fig. 2a. The coefficients, port numbers and node identifiers are chosen in $\mathcal{G}_k$ exactly as in $\mathcal{G}$.

Lemma 4. *The optimum utility of $\mathcal{S}_k$ is greater than $D_K - 1$.*

Proof. Construct a solution x as follows. Let $D = \max\{D_I, D_K + 1\}$. If the distance between the agent v and the objective k in $\mathcal{G}_k$ is $4j + 1$ for some j, set $x_v = 1 - 1/D^{2j+1}$. If the distance is $4j + 3$, set $x_v = 1/D^{2j+2}$.

To see that x is a feasible solution, first observe that feasibility is clear for a leaf constraint. Any non-leaf constraint $i \in I$ has at most D_I neighbours, and the distance between k and i is $4j + 2$ for some j. Thus

$$\sum_{v \in V_i} a_{iv} x_v \leq 1 - 1/D^{2j+1} + (D_I - 1)/D^{2j+2} < 1.$$

Let ω_k be the utility of x. We show that $\omega_k > D_K - 1$. First, consider the objective k. We have

$$\sum_{v \in V_k} c_{kv} x_v = D_K (1 - 1/D) > D_K - 1.$$

Second, each objective $h \in K' \setminus \{k\}$ has D_K neighbours and the distance between h and k is $4j$ for some j. Thus

$$\sum_{v \in V_h} c_{hv} x_v = 1/D^{2j} + (D_K - 1)(1 - 1/D^{2j+1}) > D_K - 1.$$

Finally, each objective $h \in K \setminus K'$ has 2 neighbours and the distance between h and k is $4j$ for some j; the coefficients are $c_{hv} = D_K - 1$. Thus

$$\sum_{v \in V_h} c_{hv} x_v = (D_K - 1)(1/D^{2j} + 1 - 1/D^{2j+1}) > D_K - 1.$$

□

4.3 Proof of Theorem 2

Let $\Delta_I \geq 2$ and $\Delta_K \geq 2$. Assume that $\mathcal{A}$ is a local approximation algorithm with the approximation ratio α. Set $D_I = \Delta_I$, $D_K = \Delta_K$ and $s = 0$. Let r be the local horizon of the algorithm, rounded up to a multiple of 4. Construct the instance $\mathcal{S}$ as described in Sect. 4.1; it is a 0/1 bipartite max-min LP, and it satisfies the degree bounds Δ_I and Δ_K. Apply the algorithm $\mathcal{A}$ to $\mathcal{S}$. The algorithm produces a feasible solution x. By Lemma 3 there is a constraint k such that $\sum_{v \in V_k} x_v \leq \Delta_K / \Delta_I$.

Now construct $\mathcal{S}_k$ as described in Sect. 4.2; this is another 0/1 bipartite max-min LP. Apply $\mathcal{A}$ to $\mathcal{S}_k$. The algorithm produces a feasible solution x'. The radius r neighbourhoods of the agents $v \in V_k$ are identical in $\mathcal{S}$ and $\mathcal{S}_k$; therefore the algorithm must make the same decisions for them, and we have $\sum_{v \in V_k} x'_v \leq \Delta_K / \Delta_I$. But by Lemma 4 there is a feasible solution of $\mathcal{S}_k$ with utility greater than $\Delta_K - 1$; therefore the approximation ratio of $\mathcal{A}$ is $\alpha > (\Delta_K - 1)/(\Delta_K/\Delta_I)$. This completes the proof of Theorem 2.

4.4 Proof of Theorem 3

Let $\Delta_I \geq 3$, $\Delta_K \geq 3$, and $0 < \delta < 1/10$. Assume that $\mathcal{A}$ is a local approximation algorithm with the approximation ratio α. Set $D_I = 3$, $D_K = 3$, and $s = \lceil 4/(7\delta) - 1/2 \rceil$. Let r be the local horizon of the algorithm, rounded up to a multiple of 4.

Again, construct the instance $\mathcal{S}$. The relative growth of $\mathcal{G}$ is at most $1 + 2^j/((2^j-1)(2s+1))$ beyond radius $R = j(4s+2)$; indeed, each set of 2^j new agents can be accounted for $1+2+\cdots+2^{j-1} = 2^j - 1$ chains with $2s+1$ agents each. Choosing $j = 3$, the relative growth of $\mathcal{G}$ is at most $1+\delta$ beyond radius R.

Apply $\mathcal{A}$ to $\mathcal{S}$. By Lemma 3 we know that there exists an objective h such that $\sum_{v \in V_h} x_v \leq 2 - 2/(3s+2)$. Choose a $k \in K'$ nearest to h. Construct $\mathcal{S}_k$ and apply $\mathcal{A}$ to $\mathcal{S}_k$. The local neighbourhoods of the agents $v \in V_h$ are identical in $\mathcal{S}$ and $\mathcal{S}_k$. By Lemma 4 there is a feasible solution of $\mathcal{S}_k$ with utility greater than 2. Using the assumption $\delta < 1/10$, we obtain

$$\alpha > \frac{2}{2 - 2/(3s+2)} = 1 + \frac{1}{3s+1} \geq 1 + \frac{1}{3(4/(7\delta)+1/2)+1} > 1 + \frac{\delta}{2}.$$

This completes the proof of Theorem 3.

Acknowledgements

We thank anonymous reviewers for their helpful feedback. This research was supported in part by the Academy of Finland, Grants 116547 and 117499, and by Helsinki Graduate School in Computer Science and Engineering (Hecse).

References

1. Naor, M., Stockmeyer, L.: What can be computed locally? SIAM Journal on Computing **24**(6) (1995) 1259–1277
2. Floréen, P., Hassinen, M., Kaski, P., Suomela, J.: Local approximation algorithms for a class of 0/1 max-min linear programs (2008) Manuscript, arXiv:0806.0282 [cs.DC].
3. Floréen, P., Kaski, P., Musto, T., Suomela, J.: Approximating max-min linear programs with local algorithms. In: Proc. 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS, Miami, FL, USA, April 2008), Piscataway, NJ, USA, IEEE (2008)
4. Linial, N.: Locality in distributed graph algorithms. SIAM Journal on Computing **21**(1) (1992) 193–201
5. Angluin, D.: Local and global properties in networks of processors. In: Proc. 12th Annual ACM Symposium on Theory of Computing (STOC, Los Angeles, CA, USA, April 1980), New York, NY, USA, ACM Press (1980) 82–93
6. Papadimitriou, C.H., Yannakakis, M.: Linear programming without the matrix. In: Proc. 25th Annual ACM Symposium on Theory of Computing (STOC, San Diego, CA, USA, May 1993), New York, NY, USA, ACM Press (1993) 121–129

7. Kuhn, F., Moscibroda, T., Wattenhofer, R.: The price of being near-sighted. In: Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, Miami, FL, USA, January 2006), New York, NY, USA, ACM Press (2006) 980–989
8. Lynch, N.A.: A hundred impossibility proofs for distributed computing. In: Proc. 8th Annual ACM Symposium on Principles of Distributed Computing (PODC, Edmonton, Canada, August 1989), New York, NY, USA, ACM Press (1989) 1–28
9. Kuhn, F., Moscibroda, T.: Distributed approximation of capacitated dominating sets. In: Proc. 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA, San Diego, CA, USA, June 2007), New York, NY, USA, ACM Press (2007) 161–170
10. Kuhn, F., Moscibroda, T., Wattenhofer, R.: On the locality of bounded growth. In: Proc. 24th Annual ACM Symposium on Principles of Distributed Computing (PODC, Las Vegas, NV, USA, July 2005), New York, NY, USA, ACM Press (2005) 60–68
11. Lubotzky, A., Phillips, R., Sarnak, P.: Ramanujan graphs. Combinatorica **8**(3) (1988) 261–277
12. Lazebnik, F., Ustimenko, V.A.: Explicit construction of graphs with an arbitrary large girth and of large size. Discrete Applied Mathematics **60**(1–3) (1995) 275–284
13. Hoory, S.: On Graphs of High Girth. PhD thesis, Hebrew University, Jerusalem (March 2002)
14. McKay, B.D., Wormald, N.C., Wysocka, B.: Short cycles in random regular graphs. Electronic Journal of Combinatorics **11**(1) (2004) #R66
15. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed locally! In: Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC, St. John's, Newfoundland, Canada, July 2004), New York, NY, USA, ACM Press (2004) 300–309
16. Anderson, E.J., Nash, P.: Linear Programming in Infinite-Dimensional Spaces: Theory and Applications. John Wiley & Sons, Ltd., Chichester, UK (1987)
17. Godsil, C., Royle, G.: Algebraic Graph Theory. Volume 207 of Graduate Texts in Mathematics. Springer-Verlag, New York, NY, USA (2004)
18. Hocking, J.G., Young, G.S.: Topology. Addison-Wesley, Reading, MA, USA (1961)
19. Munkres, J.R.: Topology. 2nd edn. Prentice Hall, Upper Saddle River, NJ, USA (2000)
20. Amit, A., Linial, N., Matoušek, J., Rozenman, E.: Random lifts of graphs. In: Proc 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, Washington, DC, USA, January 2001), Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2001) 883–894
21. Esparza, J., Heljanko, K.: A new unfolding approach to LTL model checking. In: Proc. 27th International Colloquium on Automata, Languages and Programming (ICALP, Geneva, Switzerland, July 2000). Volume 1853 of Lecture Notes in Computer Science, Berlin, Germany, Springer-Verlag (2000) 475–486