Local approximability of max-min and min-max linear programs

Floréen, Patrik

2011

Floréen , P , Hassinen , M , Kaasinen , J , Kaski , P , Musto , T & Suomela , J 2011 , ' Local approximability of max-min and min-max linear programs ' , Theory of Computing Systems , þÿ v o l . 4 9 , n o . 4 , p p . 6 7 2 6 9 7 . h t t p s : / / d o i . o r g / 1 0 . 1 0 0 7 / s 0 0 2 2 4 - 0 1 0 - 9 3 0 3

http://hdl.handle.net/10138/28034
https://doi.org/10.1007/s00224-010-9303-6

acceptedVersion

Local approximability of max-min and min-max linear programs

Patrik Floréen, Marja Hassinen, Joel Kaasinen,
Petteri Kaski, Topi Musto, and Jukka Suomela

Helsinki Institute for Information Technology HIIT
University of Helsinki

Abstract. In a *max-min LP*, the objective is to maximise ω subject to $A\mathbf{x} \leq \mathbf{1}$, $C\mathbf{x} \geq \omega\mathbf{1}$, and $\mathbf{x} \geq \mathbf{0}$. In a *min-max LP*, the objective is to minimise ρ subject to $A\mathbf{x} \leq \rho\mathbf{1}$, $C\mathbf{x} \geq \mathbf{1}$, and $\mathbf{x} \geq \mathbf{0}$. The matrices A and C are nonnegative and sparse: each row $\mathbf{a}_i$ of A has at most Δ_I positive elements, and each row $\mathbf{c}_k$ of C has at most Δ_K positive elements.

We study the approximability of max-min LPs and min-max LPs in a distributed setting; in particular, we focus on *local algorithms* (constant-time distributed algorithms). We show that for any $\Delta_I \geq 2$, $\Delta_K \geq 2$, and $\varepsilon > 0$ there exists a local algorithm that achieves the approximation ratio $\Delta_I(1 - 1/\Delta_K) + \varepsilon$. We also show that this result is the best possible: no local algorithm can achieve the approximation ratio $\Delta_I(1 - 1/\Delta_K)$ for any $\Delta_I \geq 2$ and $\Delta_K \geq 2$.

Keywords: approximation algorithms, distributed algorithms, linear programs, local algorithms.

1 Introduction

In a *max-min linear program* (max-min LP), the objective is to

$$\begin{aligned} \text{maximise} \quad & \omega \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{1}, \\ & C\mathbf{x} \geq \omega\mathbf{1}, \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{1}$$

A *min-max linear program* (min-max LP) is analogous: the objective is to

$$\begin{aligned} \text{minimise} \quad & \rho \\ \text{subject to} \quad & A\mathbf{x} \leq \rho\mathbf{1}, \\ & C\mathbf{x} \geq \mathbf{1}, \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2}$$

In both cases, A and C are nonnegative matrices.

In this work, we study max-min LPs and min-max LPs in a distributed setting. We present a local algorithm (constant-time distributed algorithm) for approximating these LPs, and we show that the approximation factor of our algorithm is the best possible among all local algorithms.

1.1 Distributed setting

Let $\mathcal{G} = (V \cup I \cup K, E)$ be a bipartite, undirected communication graph. The nodes $v \in V$ are called *agents*, the nodes $i \in I$ are called *constraints*, and the nodes $k \in K$ are called *objectives*; the sets V, I, and K are pairwise disjoint. Each edge $e \in E$ is of the form $e = \{v, i\}$ or $e = \{v, k\}$ where $v \in V$, $i \in I$, and $k \in K$. The edges of the graph $\mathcal{G}$ have positive weights: the weight of an edge $\{i, v\} \in E$, $i \in I$, $v \in V$ is denoted by $a_{i,v}$ and the weight of an edge $\{k, v\} \in E$, $k \in K$, $v \in V$ is denoted by $c_{k,v}$. Each agent $v \in V$ is associated with a variable x_v. See Figure 1 for an illustration.

We define the shorthand notation

$$\begin{aligned} V_i &= \{v \in V : \{v, i\} \in E\}, & I_v &= \{i \in I : \{v, i\} \in E\}, \\ V_k &= \{v \in V : \{v, k\} \in E\}, & K_v &= \{k \in K : \{v, k\} \in E\} \end{aligned}$$

for all $i \in I$, $k \in K$, and $v \in V$. We assume that $|V_i| \leq \Delta_I$ and $|V_k| \leq \Delta_K$ for all $i \in I$ and $k \in K$ for some constants Δ_I and Δ_K.

Let

$$f_i(\mathbf{x}) = \sum_{v \in V_i} a_{i,v} x_v, \quad i \in I, \tag{3}$$

$$g_k(\mathbf{x}) = \sum_{v \in V_k} c_{k,v} x_v, \quad k \in K \tag{4}$$

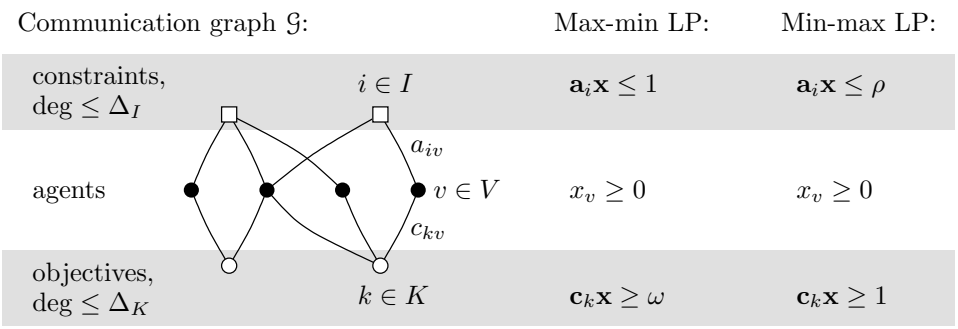

Figure 1: Communication graph $\mathcal{G}$ and the LPs associated with it. In the max-min LP, the task is to maximise ω, while in the min-max LP, the task is to minimise ρ.

and

$$\rho(\mathbf{x}) = \max_{i \in I} f_i(\mathbf{x}), \tag{5}$$

$$\omega(\mathbf{x}) = \min_{k \in K} g_k(\mathbf{x}). \tag{6}$$

In the *max-min linear program* associated with $\mathcal{G}$, the task is to

$$\begin{aligned} &\text{maximise} && \omega(\mathbf{x}) \\ &\text{subject to} && \rho(\mathbf{x}) \leq 1, \\ &&& \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{7}$$

Analogously, in the *min-max linear program* associated with $\mathcal{G}$, the task is to

$$\begin{aligned} &\text{minimise} && \rho(\mathbf{x}) \\ &\text{subject to} && \omega(\mathbf{x}) \geq 1, \\ &&& \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{8}$$

In a max-min LP, the value of the objective function $\omega(\mathbf{x})$ is called the *utility* of the solution $\mathbf{x}$, and in a min-max LP, the value of the objective function $\rho(\mathbf{x})$ is called the *cost* of the solution $\mathbf{x}$.

The optimisation problems (7) and (8) are equivalent to the LPs (1) and (2), respectively. In the distributed setting, there is a node $i \in I$ for each row $\mathbf{a}_i$ of A, and a node $k \in K$ for each row $\mathbf{c}_k$ of C. Each row of A has at most Δ_I positive elements, and each row of C has at most Δ_K positive elements.

To avoid degenerate cases, we assume that there are no isolated nodes in $\mathcal{G}$. Indeed, isolated agents and constraints can be deleted w.l.o.g. If there was an isolated objective, the optimum of (7) would be zero and (8) would be infeasible.

Remark 1. The terms "constraint" and "objective" are chosen so that they are natural from the perspective of max-min LPs. Most of our discussion focuses on max-min LPs; analogous results for min-max LPs are then derived by using reductions.

1.2 Local approximation

Each node in the communication graph $\mathcal{G}$ is a computational entity. All nodes in the network run the same distributed algorithm $\mathcal{A}$. Initially, each node knows only its *local input*, which consists of the incident edges and their weights.

During each *synchronous communication round*, each node in parallel (i) performs local computation, then (ii) sends a message to each neighbour, and finally (iii) receives a message from each neighbour. Eventually, after D communication rounds, each agent $v \in V$ in parallel produces its *local output* x_v, and the algorithm stops.

We say that $\mathcal{A}$ is a *local algorithm* if D is a constant [24]; D may depend on the parameters Δ_I and Δ_K, but it must be independent of the number of nodes in $\mathcal{G}$. The constant D is called the *local horizon* of the algorithm. For each agent $v \in V$, the output x_v is a function of the local inputs of the nodes within distance D (in number of edges) from v in the communication graph $\mathcal{G}$.

We use the convention that an approximation factor is at least 1, both in maximisation and minimisation problems. We say that $\mathcal{A}$ is an α-approximation algorithm for max-min LPs if in any graph $\mathcal{G}$, the output $\mathbf{x}$ is a feasible solution to the max-min LP associated with $\mathcal{G}$, and the value $\omega(\mathbf{x})$ is at least $1/\alpha$ times the global optimum of (7). Similarly, $\mathcal{A}$ is an α-approximation algorithm for min-max LPs if the output is a feasible solution and $\rho(\mathbf{x})$ is at most α times the global optimum of (8). We emphasise that a local approximation algorithm need not – and cannot – know the value of $\omega(\mathbf{x})$ or $\rho(\mathbf{x})$. However, it must produce a globally consistent, feasible, and near-optimal solution $\mathbf{x}$.

1.3 Applications

Immediate applications of distributed max-min LPs and min-max LPs include various tasks of fair resource allocation in contemporary networking, such as fair bandwidth allocation in a communication network and data gathering in a wireless sensor network.

Example 1 (Fair bandwidth allocation)**.** Assume that each $k \in K$ is a customer of an Internet service provider, and each $i \in I$ is an Internet access point. Construct the communication graph $\mathcal{G} = (V \cup I \cup K, E)$ as follows: for each network link between a customer $k \in K$ and an access

point $i \in I$, add a new agent v to V and the weight-1 edges $\{i, v\}$ and $\{k, v\}$ to E.

A vector $\mathbf{x}$ can be interpreted as a bandwidth allocation: if an agent $v \in V$ is adjacent to a customer $k \in K$ and an access point $i \in I$, then the customer k can transmit data at the rate x_v through the access point i. In total, we allocate $g_k(\mathbf{x})$ units of bandwidth to the customer $k \in K$, and the total load of the access point $i \in I$ is $f_i(\mathbf{x})$ units.

A solution of the max-min LP associated with the graph $\mathcal{G}$ hence gives a bandwidth allocation that maximises the minimum amount of service that we provide to each customer, subject to the constraint that each access point can handle at most 1 unit of bandwidth.

Observe that the structure of the communication graph $\mathcal{G}$ is closely related to the structure of the physical network which consists of customers, access points, and communication links between them. Hence the execution of any distributed algorithm in the graph $\mathcal{G}$ can be efficiently simulated in the physical network as well.

Example 2 (Lifetime maximisation in sensor networks)**.** Consider a two-tier wireless sensor network: wireless *sensor nodes* produce measurements of the environment, and the data is forwarded from each sensor node, through *relay nodes*, to a sink node.

Assume that each $k \in K$ is a wireless sensor node in a sensor network, and each $i \in I$ is a relay node. Construct the communication graph $\mathcal{G} = (V \cup I \cup K, E)$ like we did in the previous example: whenever a relay node $i \in I$ is within the range of the radio of a sensor node $k \in K$, add a new agent v to V and the weight-1 edges $\{i, v\}$ and $\{k, v\}$ to E. The variable x_v associated with the agent v represents the rate at which the sensor k sends data through the relay i to the sink node.

Now $g_k(\mathbf{x})$ is the total rate at which the sensor $k \in K$ produces data, and $f_i(\mathbf{x})$ is the total rate at which data is forwarded through the relay $i \in I$. If each sensor produces data at the constant rate 1, then a solution of the min-max LP associated with $\mathcal{G}$ provides data flows that minimise the maximum load of each relay. If each relay is a battery-powered device, then this data flow maximises the lifetime of the system before the first relay runs out the battery.

An algorithm for approximating max-min LPs or min-max LPs also enables one to solve approximate mixed packing and covering LPs [29]; a particular special case is finding an (approximate) solution to a nonnegative system of linear equations.

1.4 Contributions

Our work provides a complete characterisation of the local approximability of max-min LPs and min-max LPs by local algorithms. We begin with an

observation that covers the simple special cases where $\Delta_I = 1$ or $\Delta_K = 1$.

Theorem 1. *If $\Delta_I = 1$ or $\Delta_K = 1$, there are local algorithms for finding optimal solutions of max-min LPs and min-max LPs.*

Our main contribution is a matching pair of upper and lower bounds for all nontrivial values of Δ_I and Δ_K.

Theorem 2. *For any $\Delta_I \geq 2$, $\Delta_K \geq 2$, and $\varepsilon > 0$, there are local approximation algorithms for max-min LPs and min-max LPs with the approximation ratio $\Delta_I(1 - 1/\Delta_K) + \varepsilon$.*

Theorem 3. *For any $\Delta_I \geq 2$ and $\Delta_K \geq 2$, there is no local approximation algorithm for max-min LPs or min-max LPs with the approximation ratio equal to $\Delta_I(1 - 1/\Delta_K)$.*

Our results are not sensitive to the amount of auxiliary information that we have in the communication network. On the one hand, the negative result of Theorem 3 holds even if each node is assigned a unique identifier. On the other hand, the positive results in Theorems 1 and 2 hold even in the case of *anonymous networks*. Our algorithms do not need unique identifiers; we merely assume that there is a *port numbering* [1] in the graph $\mathcal{G}$, i.e., each node has chosen an ordering on its incident edges.

Our results are not sensitive to the details of the problem formulation, either. In particular, the negative result holds even if we require that A and C are 0/1 matrices, while the matching positive result holds for arbitrary nonnegative matrices. Moreover, the negative results hold even if we require that $|I_v| = 1$ and $|K_v| = 1$ for all $v \in V$, while the positive results do not depend on the size of I_v or K_v at all.

As our algorithms are *deterministic*, standard techniques [4, 5] can be used to convert our algorithms into efficient *self-stabilising* algorithms; such algorithms provide a very high degree of fault-tolerance, as they recover from an arbitrary initial configuration. We refer to Lenzen et al. [19] for more details on the connection between local and self-stabilising algorithms.

1.5 Related work

Few deterministic local algorithms are known for combinatorial problems. Most of the positive results are confined to special cases, and typically auxiliary information such as unique node identifiers are needed. Examples of the positive results include local algorithms for weak colourings in graphs where each node has an odd degree [22, 24], matchings in 2-coloured graphs [11, 12], and dominating sets in planar graphs [7, 18]. There are strong negative results that rule out the existence of local algorithms for many classical graph problems, such as finding a maximal independent set [21] or any constant-factor approximation of a maximum independent set [7, 20] in a cycle.

There have been more positive results related to local algorithms for linear programs. Prior work has primarily focused on *packing LPs*, which are nonnegative linear programs of the form

$$\begin{aligned} \text{maximise} \quad & \sum_v x_v \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{1}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

and on their duals, *covering LPs*. Among the pioneers were Papadimitriou and Yannakakis [25] who studied local algorithms for packing LPs; we will use their algorithm in Section 2.3. Kuhn et al. [15, 17] present a local approximation scheme for packing and covering LPs; for example, a $(1+\varepsilon)$-approximation can be found in $O(\varepsilon^{-4} \log^2 \Delta)$ communication rounds, assuming that A is a 0/1 matrix with at most Δ nonzero elements in any row or column.

While the distributed approximability of packing and covering LPs is nowadays fairly well understood [15, 17, 23], much less is known about more general linear programs. One of the few positive results is Kuhn et al. [16], which studies the LP relaxation of k-fold dominating sets. Problems closely related to max-min and min-max LPs have been studied from the perspective of parallel algorithms – see, for example, Young [29] – but these algorithms cannot be applied in a local, distributed setting.

To our knowledge, max-min LPs provide the first example of a natural problem where there are matching, nontrivial lower and upper bounds for the approximation factor of a deterministic local algorithm. Recently, another pair of matching lower and upper bounds for local algorithms has been discovered: in bounded-degree graphs, there is a local 2-approximation algorithm for the vertex cover problem [2, 3], and there is no local algorithm with the approximation ratio $2-\varepsilon$ for any $\varepsilon > 0$ [7, 20]. Incidentally, this is another example of a problem such that the lower bound holds even if there are unique identifiers, while the matching upper bound only needs a port numbering.

We refer to the survey [28] for more information on local algorithms.

1.6 Structure of this work

In Section 2 we show that a local approximation algorithm for max-min LPs implies a local approximation algorithm for min-max LPs and vice versa. We also prove Theorem 1.

In Section 3 we lay the groundwork for proving the main positive result, Theorem 2. We show that max-min LPs can be solved near-optimally in some special cases, provided that we have auxiliary information in the graph $\mathcal{G}$.

In Section 4 we show how to use the results of Section 3 to construct a local approximation algorithm for general max-min LPs. The reductions

from Section 2 can be used to extend the result to general min-max LPs, and Theorem 2 follows.

In Section 5 we prove the negative result of Theorem 3.

1.7 Previous versions

The present work is based on three preliminary conference and workshop reports [8, 9, 10], and it also contains material from a PhD thesis [27].

The algorithm in Sections 3 and 4 is a thoroughly revised version of the algorithm presented in our conference report [9]. The negative result in Section 5 is a special case of the construction in our workshop report [8]. All material related to min-max LPs is new.

2 Preliminaries

We begin by showing that solutions of max-min LPs and min-max LPs are related to each other via local reductions, and then show how to use this connection to prove Theorem 1.

2.1 Normalisation

To simplify the discussion we will first show that we can focus on *normalised* graphs $\mathcal{G}$ that satisfy $|I_v| \geq 1$, $|K_v| \geq 1$, $|V_i| \geq 1$, and $|V_k| \geq 1$ for all $v \in V$, $i \in I$, and $k \in K$.

If we have $I_v = \emptyset$ for an agent $v \in V$, then we can set $x_v = +\infty$ w.l.o.g., both in max-min LPs and min-max LPs. Then $g_k(\mathbf{x}) = +\infty$ for each adjacent $k \in K_v$. In effect, we can remove v and every $k \in K_v$. Similarly, if $K_v = \emptyset$ for an agent $v \in V$, we can set $x_v = 0$ w.l.o.g., both in max-min LPs and min-max LPs. In effect, we can remove v. If these two modifications create isolated nodes, we will remove them as well. We are left with a normalised graph.

A normalised graph $\mathcal{G}$ satisfies the following properties:

(i) The max-min LP associated with $\mathcal{G}$ is feasible and bounded.

(ii) The min-max LP associated with $\mathcal{G}$ is feasible and bounded.

(iii) The optimum of the max-min LP is positive. Observe that $\mathbf{x} = \varepsilon\mathbf{1}$ is a feasible solution for a sufficiently small $\varepsilon > 0$.

(iv) The optimum of the min-max LP is positive. Observe that $\rho(\mathbf{x}) = 0$ implies $\mathbf{x} = 0$ and $\omega(\mathbf{x}) = 0$.

The normalisation is a local operation; it can be implemented in a constant number of communication rounds. Hence it is sufficient to design a local algorithm for max-min LPs and min-max LPs in normalised graphs;

we can combine it with the normalisation step to obtain a local algorithm form max-min LPs and min-max LPs in general graphs.

2.2 Reductions between max-min LPs and min-max LPs

Let us now relate the optimum values of max-min LPs and min-max LPs.

Lemma 2.1. *For any normalised graph $\mathcal{G}$, the optimum of the max-min LP associated with $\mathcal{G}$ is s if and only if the optimum of the min-max LP associated with $\mathcal{G}$ is $1/s$.*

Proof. Let $\mathbf{x}$ be a feasible solution of the max-min LP with utility at least $s > 0$, that is, $\omega(\mathbf{x}) \geq s$ and $\rho(\mathbf{x}) \leq 1$. Then $\omega(\mathbf{x}/s) \geq 1$ and $\rho(\mathbf{x}/s) \leq 1/s$, i.e., $\mathbf{x}/s$ is a feasible solution of the min-max LP with cost at most $1/s$. Conversely, a feasible solution of the min-max LP with cost at most $s > 0$ provides a feasible solution of the max-min LP with utility at least $1/s$. □

Now assume that we are given a feasible solution $\mathbf{x}$ of the max-min LP associated with a normalised graph $\mathcal{G}$. We will construct a feasible solution $\mathbf{y}$ of the min-max LP associated with $\mathcal{G}$ as follows: each agent $v \in V$ sets

$$q_v \;=\; \min_{k \in K_v} g_k(\mathbf{x}), \qquad y_v \;=\; \begin{cases} 0 & \text{if } q_v = 0, \\ x_v / q_v & \text{if } q_v > 0. \end{cases} \tag{9}$$

We can compute q_v and y_v in two communication rounds: first each agent $u \in V$ sends x_u to all $k \in K_u$; then each objective $k \in K$ computes $g_k(\mathbf{x})$ and sends it to all $v \in V_k$.

Lemma 2.2. *If $\omega(\mathbf{x}) > 0$, then the vector $\mathbf{y}$ in (9) is a feasible solution of the min-max LP associated with $\mathcal{G}$, and the cost $\rho(\mathbf{y})$ is at most $1/\omega(\mathbf{x})$.*

Proof. Let us first show that $\mathbf{y}$ is a feasible solution. Consider an objective $k \in K$. Then for all $v \in V_k$ we have $0 < \omega(\mathbf{x}) \leq q_v \leq g_k(\mathbf{x})$. Therefore

$$g_k(\mathbf{y}) \;\geq\; g_k\big(\mathbf{x}/g_k(\mathbf{x})\big) \;=\; 1.$$

Let us then establish the cost of the solution. Consider a constraint $i \in I$. We have $q_v \geq \omega(\mathbf{x}) > 0$ for all $v \in V_i$. Therefore

$$f_i(\mathbf{y}) \;\leq\; f_i\big(\mathbf{x}/\omega(\mathbf{x})\big) \;\leq\; \frac{1}{\omega(\mathbf{x})}. \qquad \square$$

Then assume that we are given a feasible solution $\mathbf{x}$ of the min-max LP associated with a normalised graph $\mathcal{G}$. We will construct a feasible solution $\mathbf{y}$ of the max-min LP associated with $\mathcal{G}$ as follows: each agent $v \in V$ sets

$$p_v \;=\; \max_{i \in I_v} f_i(\mathbf{x}), \qquad y_v \;=\; \begin{cases} 0 & \text{if } p_v = 0, \\ x_v / p_v & \text{if } p_v > 0. \end{cases} \tag{10}$$

Lemma 2.3. *If $\rho(\mathbf{x}) > 0$, then the vector $\mathbf{y}$ in (10) is a feasible solution of the max-min LP associated with $\mathcal{G}$, and the utility $\omega(\mathbf{y})$ is at least $1/\rho(\mathbf{x})$.*

Proof. Let us first show that $\mathbf{y}$ is a feasible solution. Consider a constraint $i \in I$. If $f_i(\mathbf{x}) = 0$ then we have $x_v = 0$ and $y_v = 0$ for all $v \in V_i$; it follows that $f_i(\mathbf{y}) = 0$. Otherwise we have $0 < f_i(\mathbf{x}) \leq p_v$ for all $v \in V_i$. Therefore

$$f_i(\mathbf{y}) \;\leq\; f_i\big(\mathbf{x}/f_i(\mathbf{x})\big) \;=\; 1.$$

Let us then establish the utility of the solution. Consider an objective $k \in K$. We have $p_v \leq \rho(\mathbf{x})$ for all $v \in V_k$. Therefore

$$g_k(\mathbf{y}) \;\geq\; g_k\big(\mathbf{x}/\rho(\mathbf{x})\big) \;\geq\; \frac{1}{\rho(\mathbf{x})}. \qquad \square$$

Lemmas 2.1, 2.2, and 2.3 have the following corollary.

Corollary 2.4. *Given a local α-approximation algorithm for max-min LPs we can construct a local α-approximation algorithm for min-max LPs and vice versa.*

In this reduction, the running time of the algorithm increases only by 2 communication rounds. Moreover, the reduction preserves the values of the parameters Δ_I and Δ_K. For example, a local α-approximation for max-min LPs in the case $\Delta_I = 2$ and $\Delta_K = 3$ implies a local α-approximation for min-max LPs in the case $\Delta_I = 2$ and $\Delta_K = 3$.

Thanks to this reduction, we can focus on max-min LPs; we can apply Corollary 2.4 to both positive and negative results to get analogous results for min-max LPs.

2.3 The safe algorithm

Papadimitriou and Yannakakis [25] present a simple local approximation algorithm for packing LPs, the so-called *safe algorithm*; it turns out that this is a local approximation algorithm for max-min LPs as well.

In the safe algorithm, the agent v chooses

$$x_v \;=\; \min_{i \in I_v} \frac{1}{a_{i,v}|V_i|}. \tag{11}$$

Intuitively, each constraint $i \in I$ divides its "capacity" evenly among its neighbours: if i has $|V_i|$ neighbours, each neighbour $v \in V$ of i can use at most $1/|V_i|$ of the capacity – that is, $a_{i,v}x_v$ is at most $1/|V_i|$. In particular, the node v chooses the largest possible value x_v that does not violate these allotments for any of its adjacent constraints.

Lemma 2.5. *The vector $\mathbf{x}$ in (11) is a feasible, Δ_I-approximate solution of the max-min LP associated with $\mathcal{G}$.*

Proof. Feasibility is satisfied by construction. To show the approximation factor, let $\mathbf{x}^*$ be an optimal solution. As $\mathbf{x}^*$ is a feasible solution, we must have $a_{i,v}x^*_v \leq 1$ for all $i \in I$ and $v \in V_i$, that is,

$$x^*_v \leq \min_{i \in I_v} \frac{1}{a_{i,v}}.$$

Hence we have $x_v \geq x^*_v/\Delta_I$ for each agent $v \in V$. We conclude that $\omega(\mathbf{x}) \geq \omega(\mathbf{x}^*)/\Delta_I$. □

2.4 Simple special cases

The safe algorithm is clearly a local algorithm, with the running time of one communication round. In particular, Lemma 2.5 shows that a max-min LP can be solved optimally if $\Delta_I = 1$, regardless of the value of Δ_K. With Corollary 2.4, we conclude that also min-max LPs can be solved optimally if $\Delta_I = 1$.

Let us now consider another simple special case, namely $\Delta_K = 1$. In this case min-max LPs are particularly easy to solve. We can simply set

$$x_v = \max_{k \in K_v} \frac{1}{c_{k,v}} \tag{12}$$

for each $v \in V$.

Lemma 2.6. *If $\Delta_K = 1$, then the vector $\mathbf{x}$ in (12) is an optimal solution of the min-max LP associated with $\mathcal{G}$.*

Proof. Clearly $\mathbf{x}$ is a feasible solution of the min-max LP associated with $\mathcal{G}$: since $c_{k,v}x_v \geq 1$ for all $k \in K$ and $v \in V_k$, we have $\omega(\mathbf{x}) \geq 1$. On the other hand, an optimal solution $\mathbf{x}^*$ must also satisfy $c_{k,v}x^*_v \geq 1$ for all $k \in K$ and for the unique $v \in V_k$; therefore $x_v \leq x^*_v$ and $\rho(\mathbf{x}) \leq \rho(\mathbf{x}^*)$. □

Invoking Corollary 2.4 we can also construct a local algorithm that solves max-min LPs optimally in the case $\Delta_K = 1$.

In summary, both max-min LPs and min-max LPs can be solved optimally if we have either $\Delta_I = 1$ or $\Delta_K = 1$. Theorem 1 follows.

3 Layered max-min LPs

Throughout this section we focus on max-min LPs of a very specific form, which we call *layered max-min LPs*. First, each node of $\mathcal{G}$ is assigned a *layer* and each agent is also assigned one of two colours. Second, there are several structural assumptions on $\mathcal{G}$, best described by using the layers and colours.

Throughout this section, h is a positive integer constant. We use the notation $H = \{1, 2, \ldots, h\}$ and $H_0 = \{0, 1, \ldots, h\}$.

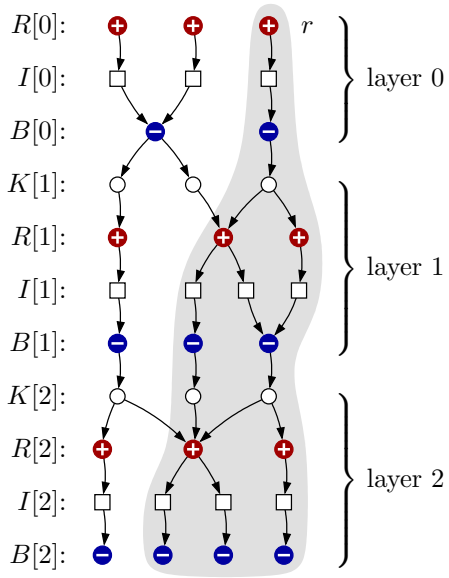

Figure 2: An example of a layered graph $\mathcal{G}$, in the case $h = 2$. The set $S^*(r)$ is highlighted.

3.1 Colours and layers

We assume that each agent $v \in V$ is assigned a *colour* red or blue. Let R be the set of red agents and let B be the set of blue agents.

We also assume that there is a *layer* $\ell(u)$ associated with each node u: for an agent $v \in V$ we have $\ell(v) \in H_0$, for a constraint $i \in I$ we have $\ell(i) \in H_0$, and for an objective $k \in K$ we have $\ell(k) \in H$. For a set of nodes U, we use the shorthand notation $U[j] = \{u \in U : \ell(u) = j\}$; for example, $R[0]$ consists of all red agents on layer 0. We refer to Figure 2 for an example.

3.2 Orientation

We assume that all edges in $\mathcal{G}$ are oriented. We write $S(u)$ for the set of immediate successors of the node u and $P(u)$ for the set of immediate predecessors of u. In those cases where the successor or predecessor is unique, we use the notation $s(u)$ and $p(u)$, respectively. Let

$$\begin{aligned} S(U) &= \bigcup_{u \in U} S(u), \\ S^0(U) &= U, \\ S^{j+1}(U) &= S(S^j(U)), \\ S^*(U) &= S^0(U) \cup S^1(U) \cup \cdots . \end{aligned}$$

We define P^*, etc., in an analogous manner. We say that a set of nodes U is *downwards closed* if $U = S^*(U)$.

3.3 Structural assumptions

The structural assumptions on $\mathcal{G}$ are as follows:

(i) Each constraint $i \in I$ has exactly one predecessor, which is a red agent on the same layer $\ell(i)$, and exactly one successor, which is a blue agent on the same layer $\ell(i)$.

(ii) Each objective $k \in K$ has exactly one predecessor, which is a blue agent on the previous layer $\ell(k) - 1$, and at least one successor, all of which are red agents on the same layer $\ell(k)$.

(iii) All agents in $V \setminus R[0]$ have at least one predecessor, and all agents in $V \setminus B[h]$ have at least one successor.

In particular, it follows that $\mathcal{G}$ is a directed acyclic graph where each directed path has length at most $4h + 2$. The nodes along any directed path are alternatingly from the sets R, I, B, and K, in this order, and the layers are non-decreasing along the path; see Figure 2.

3.4 Recursive solution

Let $\mathbf{q}$ be a vector indexed by the blue agents $b \in B$. We can define the vector $\mathbf{z}(\mathbf{q})$ indexed by the agents $v \in V$ recursively as follows:

$$z_b(\mathbf{q}) = 0, \qquad b \in B[h], \tag{13}$$

$$z_r(\mathbf{q}) = \min_{i \in S(r)} \frac{1 - a_{i,s(i)} z_{s(i)}(\mathbf{q})}{a_{i,r}}, \qquad r \in R, \tag{14}$$

$$z_b(\mathbf{q}) = \max\left\{0, \max_{k \in S(b)} \frac{q_b - \sum_{r \in S(k)} c_{k,r} z_r(\mathbf{q})}{c_{k,b}}\right\}, \quad b \in B \setminus B[h]. \tag{15}$$

We exploit this recursion in our local algorithm twice. The following observation is essential from the perspective of local computability.

Lemma 3.1. *The value $z_v(\mathbf{q})$ depends only on the local inputs of the nodes $S^*(v)$, and the values q_b for $b \in B \cap S^*(v)$.*

Proof. By the structure of (13)–(15). □

The intuition behind the recursion (13)–(15) is that each red node $r \in R$ chooses the largest possible $z_r(\mathbf{q})$ such that $f_i(\mathbf{z}(\mathbf{q})) \leq 1$ for adjacent constraints $i \in S(r)$, and each blue node $b \in B$ chooses the smallest possible $z_b(\mathbf{q}) \geq 0$ such that $g_k(\mathbf{z}(\mathbf{q})) \geq q_b$ for adjacent objectives $k \in S(b)$. We will analyse the properties of this recursion in more detail in Section 3.6.

3.5 Local algorithm

We present a local algorithm that finds a $(1+\varepsilon)$-approximation of a layered max-min LP, for any constant $\varepsilon > 0$. We begin with the description of the algorithm; we prove the correctness of the algorithm in Sections 3.6–3.7.

Phase I. Each agent $v \in V$ sets

$$\hat{x}_v = \min_{i \in I_v} \frac{1}{a_{i,v}},$$

and each objective $k \in K$ sets $\hat{g}_k = g_k(\hat{\mathbf{x}})$. Each agent $r \in R[0]$ finds the minimum of the values $\hat{g}_k$ in $S^*(r) \cap K$; let $\hat{t}_r$ be this value.

Phase II. In this phase, each node $r \in R[0]$ initiates the computation of $\mathbf{z}(\mathbf{q})$ within its local neighbourhood $S^*(r)$, using the vector $\mathbf{q} = t\mathbf{1}$ for certain values of $t > 0$. The relevant values of t are obtained using the binary search as described below.

We say that t is a *valid local estimate* for the node $r \in R[0]$ if the solution of (13)–(15) satisfies $z_v(t\mathbf{1}) \geq 0$ for all $v \in R \cap S^*(r)$. By Lemma 3.1, the node r can test whether t is a valid local estimate in $\Theta(h)$ communication rounds: First the red agent r broadcasts the query t to the subgraph $S^*(r)$. Then a solution of (13)–(15) is computed layer by layer, starting from $B[h] \cap S^*(r)$ and propagating towards r. The entire solution does not need to be transmitted to r: a flag indicating the nonnegativity of the solution suffices in order to tell whether t is a valid local estimate.

Each agent $r \in R[0]$ uses binary search to find a value t_r in the range

$$\frac{1}{2}\hat{t}_r \leq t_r \leq \hat{t}_r$$

such that (i) t_r is a valid local estimate and (ii) either $(1+\varepsilon)t_r$ is not a valid local estimate or $(1+\varepsilon)t_r > \hat{t}_r$. Note that the number of iterations only depends on the constant ε, and we can perform this procedure concurrently in parallel for all $r \in R[0]$.

Phase III. Each blue agent $b \in B$ finds the minimum of the values t_r in $P^*(b) \cap R[0]$; let s_b be this value. All agents compute recursively $\mathbf{z}(\mathbf{s})$ using (13)–(15). Finally, each agent v outputs the value $z_v(\mathbf{s})$.

3.6 Properties of the recursive solution

Before proving the correctness of the algorithm, we analyse the properties of the vectors $\mathbf{z}(\mathbf{q})$ from (13)–(15). To this end, it is good to note that (3)

and (4) can be rewritten as follows in the case of layered max-min LPs:

$$f_i(\mathbf{x}) \;=\; a_{i,p(i)} x_{p(i)} + a_{i,s(i)} x_{s(i)}, \qquad i \in I, \tag{16}$$

$$g_k(\mathbf{x}) \;=\; c_{k,p(k)} x_{p(k)} + \sum_{r \in S(k)} c_{k,r} x_r, \quad k \in K. \tag{17}$$

We begin with a technical lemma.

Lemma 3.2. *Let U be a downwards closed set of nodes in $\mathcal{G}$. Assume that $\mathbf{x}$ satisfies $x_v \geq 0$ for all $v \in V \cap U$ and $f_i(\mathbf{x}) \leq 1$ for all $i \in I \cap U$. Assume that $\mathbf{q}$ satisfies $0 \leq q_{p(k)} \leq g_k(\mathbf{x})$ for all $k \in K \cap U$. Then*

$$0 \leq z_b(\mathbf{q}) \leq x_b \quad \text{for each } b \in B \cap U, \tag{18}$$

$$x_r \leq z_r(\mathbf{q}) \quad \text{for each } r \in R \cap U. \tag{19}$$

In particular, $z_v(\mathbf{q})$ is nonnegative for all $v \in V \cap U$.

Proof. The base case, (18) for all $b \in B[h] \cap U$, is immediate from (13).

Now assume that (18) holds for all $b \in B[j] \cap U$. Let $r \in R[j] \cap U$. Then for all $i \in S(r)$ we have

$$a_{i,r} x_r + a_{i,s(i)} x_{s(i)} \;=\; a_{i,p(i)} x_{p(i)} + a_{i,s(i)} x_{s(i)} \;=\; f_i(\mathbf{x}) \;\leq\; 1.$$

Using the assumption (18) it follows that

$$x_r \;\leq\; \frac{1 - a_{i,s(i)} x_{s(i)}}{a_{i,r}} \;\leq\; \frac{1 - a_{i,s(i)} z_{s(i)}(\mathbf{q})}{a_{i,r}}$$

for all $i \in S(r)$. Hence (14) implies $x_r \leq z_r(\mathbf{q})$. We have shown that (19) holds for all $r \in R[j] \cap U$.

To complete the induction, assume that (19) holds for all $r \in R[j] \cap U$. Let $b \in B[j-1] \cap U$. For all $k \in S(b)$ we have

$$c_{k,b} x_b + \sum_{r \in S(k)} c_{k,r} x_r \;=\; c_{k,p(k)} x_{p(k)} + \sum_{r \in S(k)} c_{k,r} x_r \;=\; g_k(\mathbf{x}) \;\geq\; q_{p(k)} \;=\; q_b.$$

Using the assumption (19) it follows that

$$x_b \;\geq\; \frac{q_b - \sum_{r \in S(k)} c_{k,r} x_r}{c_{k,b}} \;\geq\; \frac{q_b - \sum_{r \in S(k)} c_{k,r} z_r(\mathbf{q})}{c_{k,b}}$$

for all $k \in S(b)$; moreover, we assumed that $x_b \geq 0$. Hence (15) implies $0 \leq z_b(\mathbf{q}) \leq x_b$. We have shown that (18) holds for all $b \in B[j-1] \cap U$. □

The following two lemmas show, among others, that *if* we knew the utility of a feasible solution $\mathbf{x}$, then we could choose an appropriate $\mathbf{q}$ such that $\mathbf{z}(\mathbf{q})$ is a feasible solution and at least as good as $\mathbf{x}$.

Lemma 3.3. *Let* $\mathbf{x} = \mathbf{z}(\mathbf{q})$*. If* $\mathbf{x}$ *is nonnegative, then it is a feasible solution of the layered max-min LP associated with* $\mathcal{G}$*. Furthermore, the utility* $\omega(\mathbf{x})$ *is at least* $\min_b q_b$*.*

Proof. To verify feasibility, we first observe that $x_v \geq 0$ for all $v \in V$ by assumption. Next, consider a constraint $i \in I$. Let $r = p(i)$ be its unique predecessor, and let $b = s(i)$ be its unique successor. Since $r \in R$ and $i \in S(r)$, in (14) we have chosen an x_r such that $a_{i,r} x_r \leq 1 - a_{i,b} x_b$, that is, $f_i(\mathbf{x}) \leq 1$.

To analyse the utility $\omega(\mathbf{x})$, consider an objective $k \in K$. Let $b = p(k)$ be its unique predecessor. Since $b \in B \setminus B[h]$ and $k \in S(b)$, in (15) we have chosen an x_b such that $g_k(\mathbf{x}) \geq q_b$. □

Lemma 3.4. *Let* $\mathbf{x}$ *be a feasible solution of the layered max-min LP associated with* $\mathcal{G}$*, let* U *be a downwards closed set of nodes in* $\mathcal{G}$*, and assume that* $0 \leq q \leq g_k(\mathbf{x})$ *for all* $k \in K \cap U$*. Then* $z_v(q\mathbf{1})$ *is nonnegative for all* $v \in V \cap U$*.*

Proof. A feasible solution $\mathbf{x}$ satisfies $x_v \geq 0$ for all $v \in V \cap U$ and $f_i(\mathbf{x}) \leq 1$ for all $i \in I \cap U$; hence we can apply Lemma 3.2 with $\mathbf{q} = q\mathbf{1}$. □

The following lemma provides us with flexibility in the choice of the vector $\mathbf{q}$: instead of choosing $\mathbf{q} = q\mathbf{1}$ using a *global* estimate q, like we did in Lemma 3.4, we can choose each q_v using a *local* estimate.

Lemma 3.5. *Let* U *be a downwards closed set of nodes in* $\mathcal{G}$*. If* $z_v(\mathbf{q}')$ *is nonnegative for all* $v \in V \cap U$*, and* $0 \leq q_b \leq q'_b$ *for all* $b \in B \cap U$*, then* $z_v(\mathbf{q})$ *is nonnegative for all* $v \in V \cap U$*.*

Proof. Set $\mathbf{x} = \mathbf{z}(\mathbf{q}')$. By assumption, $x_v \geq 0$ for all $v \in V \cap U$, and from (14) we have $f_i(\mathbf{x}) \leq 1$ for all $i \in I \cap U$; cf. the proof of Lemma 3.3. Moreover, we have $0 \leq q_{p(k)} \leq q'_{p(k)} \leq g_k(\mathbf{x})$ for all $k \in K \cap U$. Hence we can apply Lemma 3.2 to show that $z_v(\mathbf{q})$ is nonnegative for all $v \in V \cap U$. □

3.7 Proof of correctness

Let $\mathbf{x}^*$ be an optimal solution of the layered max-min LP. Let us first analyse Phase I. The values $\hat{x}_v$ are upper bounds for the variables x^*_v, as shown in the following lemma.

Lemma 3.6. *For each* $v \in V$ *it holds that* $\hat{x}_v \geq x^*_v$*.*

Proof. To reach a contradiction, assume that $x^*_v > \hat{x}_v$. Then there is a constraint $i \in I_v$ such that $x^*_v > 1/a_{i,v}$; hence $f_i(\mathbf{x}^*) > 1$. □

The following corollary is immediate.

Corollary 3.7. *For each* $k \in K$ *it holds that* $\hat{g}_k \geq g_k(\mathbf{x}^*) \geq \omega(\mathbf{x}^*)$*.* □

We can scale down the values $\hat{x}_v$ by factor $\Delta_I = 2$ to obtain a feasible solution. This is the essence of the safe algorithm (recall Section 2.3).

Lemma 3.8. *The vector $\hat{\mathbf{x}}/2$ is a feasible solution of the layered max-min LP.*

Proof. Consider a constraint $i \in I$. By the choice of $\hat{\mathbf{x}}$, we have

$$\hat{x}_{p(i)} \le \frac{1}{a_{i,p(i)}}, \quad \hat{x}_{s(i)} \le \frac{1}{a_{i,s(i)}}.$$

Therefore $f_i(\hat{\mathbf{x}}/2) \le 1$. □

Now we are ready to analyse Phase II.

Lemma 3.9. *For each $r \in R[0]$, any $t \le \omega(\mathbf{x}^*)$ is a valid local estimate.*

Proof. Set $\mathbf{x} = \mathbf{x}^*$, $U = S^*(r)$, and $q = t$ in Lemma 3.4. □

Lemma 3.10. *For each $r \in R[0]$, any $t \le \hat{t}_r/2$ is a valid local estimate.*

Proof. Lemma 3.8 shows that $\hat{\mathbf{x}}/2$ is a feasible solution, and we have $\hat{t}_r/2 \le g_k(\hat{\mathbf{x}}/2)$ for all $k \in K \cap S^*(r)$. The claim follows from Lemma 3.4, by setting $\mathbf{x} = \hat{\mathbf{x}}/2$, $U = S^*(r)$, and $q = t$. □

Lemma 3.11. *Each $r \in R[0]$ finds a valid local estimate t_r such that $(1+\varepsilon)t_r > \omega(\mathbf{x}^*)$.*

Proof. Lemma 3.5 shows that binary search can be applied: if t is a valid local estimate, then all values below t are valid as well. Moreover, Lemma 3.10 shows that the first point of the range is a valid local estimate.

If $\omega(\mathbf{x}^*) \le \hat{t}_r/2$, binary search will return a value $t_r \ge \hat{t}_r/2 \ge \omega(\mathbf{x}^*)$, and the claim follows.

Otherwise $\hat{t}_r/2 < \omega(\mathbf{x}^*) \le \hat{t}_r$ by Corollary 3.7. Furthermore, $\omega(\mathbf{x}^*)$ is a valid local estimate by Lemma 3.9; hence the binary search will return a point such that $(1+\varepsilon)t_r$ is not valid or $(1+\varepsilon)t_r > \hat{t}_r$. Both possibilities imply $(1+\varepsilon)t_r > \omega(\mathbf{x}^*)$. □

Finally, we analyse Phase III.

Lemma 3.12. *The output $\mathbf{z}(\mathbf{s})$ is a feasible, $(1+\varepsilon)$-approximate solution of the layered max-min LP associated with $\mathcal{G}$.*

Proof. Consider an $r \in R[0]$. By Lemma 3.11, t_r is a valid local estimate and hence $z_v(t_r\mathbf{1})$ is nonnegative for all $v \in V \cap S^*(r)$. Since $s_b \le t_r$ for all $b \in B \cap S^*(r)$, Lemma 3.5 implies that $z_v(\mathbf{s})$ is nonnegative for all $v \in V \cap S^*(r)$ as well.

For every $v \in V$ there is an $r \in R[0]$ such that $v \in S^*(r)$. Therefore the above reasoning shows that $z_v(\mathbf{s})$ is nonnegative for all $v \in V$.

As $\mathbf{z}(\mathbf{s})$ is nonnegative, Lemma 3.3 shows that $\mathbf{z}(\mathbf{s})$ is a feasible solution of the layered max-min LP. Moreover, the same lemma shows that

$$\omega(\mathbf{z}(\mathbf{s})) \geq \min_{b \in B} s_b.$$

For each $b \in B$ there is an $r \in R[0]$ such that $s_b = t_r$, and Lemma 3.11 shows that $(1+\varepsilon)t_r > \omega(\mathbf{x}^*)$. We conclude that $(1+\varepsilon)\omega(\mathbf{z}(\mathbf{s})) > \omega(\mathbf{x}^*)$. □

The main result of this section is summarised in the following corollary of Lemma 3.12.

Corollary 3.13. *There is a local $(1+\varepsilon)$-approximation algorithm for layered max-min LPs for any $\varepsilon > 0$ and $h \in \{1, 2, \dots\}$. The running time of the algorithm is $O(h \log 1/\varepsilon)$ synchronous communication rounds.*

In the following section we show how to apply this algorithm to solve more general max-min LPs.

Remark 2. If we used an exact LP solver instead of the simple binary search in Phase II, we could also find an exact solution of the layered max-min LP. However, that would not improve the main positive result, Theorem 2.

4 Local algorithms for max-min and min-max LPs

In this section, we prove Theorem 2. We first show how to solve max-min LPs with $\Delta_I = 2$. Then we show how the general result follows by local reductions.

4.1 Max-min LPs with $\Delta_I = 2$

Given a max-min LP with $\Delta_I = 2$, we construct a layered max-min LP, and show how to use a solution of the layered max-min LP to approximate the original max-min LP.

First, we normalise the graph as described in Section 2.1. Hence we can focus on the case $|I_v| \geq 1$, $|K_v| \geq 1$, $1 \leq |V_i| \leq \Delta_I = 2$, and $1 \leq |V_k| \leq \Delta_K$ for all $v \in V$, $i \in I$, and $k \in K$.

Now consider a constraint $i \in I$. If $V_i = \{v\}$ for some $v \in V$, we define $n(i,v) = v$ and $\bar{a}_{i,v} = a_{i,v}/2$. Otherwise $V_i = \{u, v\}$, and we define $n(i,u) = v$, $n(i,v) = u$, $\bar{a}_{i,u} = a_{i,u}$, and $\bar{a}_{i,v} = a_{i,v}$. With this notation, we have

$$f_i(\mathbf{x}) = \bar{a}_{i,v} x_v + \bar{a}_{i,n(i,v)} x_{n(i,v)}$$

for all $i \in I$ and $v \in V_i$.

Next consider an objective $k \in K$. If $V_k = \{v\}$ for some $v \in V$, we define $N(k,v) = \{v\}$ and $\bar{c}_{k,v} = c_{k,v}/2$. Otherwise $|V_k| \geq 2$; then we define

$N(k,v) = V_k \setminus \{v\}$ and $\bar{c}_{k,v} = c_{k,v}$ for every $v \in V_k$. With this notation, we have

$$g_k(\mathbf{x}) \;=\; \bar{c}_{k,v} x_v + \sum_{u \in N(k,v)} \bar{c}_{k,u} x_u$$

for all $k \in K$ and $v \in V_k$.

Fix a positive integer h. As in Section 3, we set $H = \{1, 2, \ldots, h\}$ and $H_0 = \{0, 1, \ldots, h\}$. We construct a graph $\mathcal{G}_h$ such that the max-min LP associated with $\mathcal{G}_h$ is a layered max-min LP; see Figure 3 for an illustration. The nodes of the layered graph $\mathcal{G}_h$ are as follows:

(i) A red layer-ℓ agent (v, ℓ, red) for each $v \in V$ and $\ell \in H_0$.

(ii) A blue layer-ℓ agent (v, ℓ, blue) for each $v \in V$ and $\ell \in H_0$.

(iii) A layer-ℓ objective (k, ℓ, v) for each $k \in K$, $v \in V_k$, and $\ell \in H$.

(iv) A layer-ℓ constraint (i, ℓ, v) for each $i \in I$, $v \in V_i$, and $\ell \in H_0$.

The edges of $\mathcal{G}_h$ are as follows:

(i) An edge of weight $\bar{c}_{k,v}$ from $(v, \ell - 1, \text{blue})$ to (k, ℓ, v) for each $k \in K$, $v \in V_k$, and $\ell \in H$.

(ii) An edge of weight $\bar{c}_{k,u}$ from (k, ℓ, v) to (u, ℓ, red) for each $k \in K$, $v \in V_k$, $u \in N(k,v)$, and $\ell \in H$.

(iii) An edge of weight $\bar{a}_{i,v}$ from (v, ℓ, red) to (i, ℓ, v) for each $i \in I$, $v \in V_i$, and $\ell \in H_0$.

(iv) An edge of weight $\bar{a}_{i,u}$ from (i, ℓ, v) to $(n(i,v), \ell, \text{blue})$ for each $i \in I$, $v \in V_i$, and $\ell \in H_0$.

The max-min LP associated with $\mathcal{G}_h$ is a layered max-min LP. In the following, we use notation such as $\mathbf{x}^h$ to refer to a solution of the layered max-min LP associated with $\mathcal{G}_h$. We can adapt (3)–(6) as follows:

$$f^h_{(i,\ell,v)}(\mathbf{x}^h) \;=\; \bar{a}_{i,v} x^h_{(v,\ell,\text{red})} + \bar{a}_{i,n(i,v)} x^h_{(n(i,v),\ell,\text{blue})}, \quad i \in I,\ \ell \in H_0,\ v \in V_i,$$

$$g^h_{(k,\ell,v)}(\mathbf{x}^h) \;=\; \bar{c}_{k,v} x^h_{(v,\ell-1,\text{blue})} + \sum_{u \in N(k,v)} \bar{c}_{k,u} x^h_{(u,\ell,\text{red})}, \quad k \in K,\ \ell \in H,\ v \in V_k,$$

$$\rho^h(\mathbf{x}^h) \;=\; \max_{\substack{i \in I \\ v \in V_i \\ \ell \in H_0}} f^h_{(i,\ell,v)}(\mathbf{x}^h), \quad \omega^h(\mathbf{x}^h) \;=\; \min_{\substack{k \in K \\ v \in V_k \\ \ell \in H}} g^h_{(k,\ell,v)}(\mathbf{x}^h).$$

With this notation, the objective in the layered max-min LP is to maximise $\omega^h(\mathbf{x}^h)$ subject to $\rho^h(\mathbf{x}^h) \le 1$ and $\mathbf{x}^h \ge \mathbf{0}$. The following lemma shows that the optimum of the layered max-min LP is at least as high as the optimum of the original max-min LP.

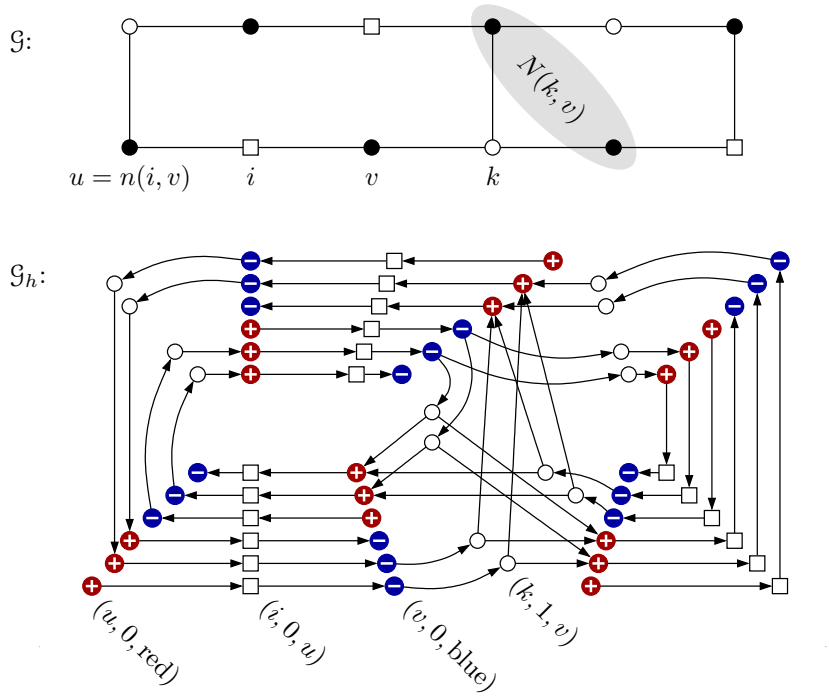

Figure 3: Constructing the layered graph $\mathcal{G}_h$; the case $h = 2$. The source nodes (red agents with no predecessor) are on layer 0, the sink nodes (blue agents with no successor) are on layer $h = 2$, and the layer increases when we traverse a curved arrow.

Lemma 4.1. *Let* $\mathbf{x}^*$ *be an optimal solution of the max-min LP associated with* $\mathcal{G}$*. Then there is a feasible solution* $\mathbf{x}^h$ *of the layered max-min LP associated with* $\mathcal{G}_h$ *such that* $\omega^h(\mathbf{x}^h) \geq \omega(\mathbf{x}^*)$*.*

Proof. Set

$$x^h_{(v,\ell,\mathrm{red})} \;=\; x^h_{(v,\ell,\mathrm{blue})} \;=\; x^*_v$$

for each agent $v \in V$ and layer $\ell \in H_0$. □

A local algorithm running in the graph $\mathcal{G}$ can simulate any local algorithm in the graph $\mathcal{G}_h$ efficiently. In particular, we can use the algorithm from Section 3.5 to find a $(1+\varepsilon)$-approximate solution of the max-min LP associated with $\mathcal{G}_h$; let $\mathbf{z}^h$ be such a solution. Each agent $v \in V$ in the original graph $\mathcal{G}$ then outputs the value

$$z_v \;=\; \frac{1}{2|H_0|} \sum_{\ell \in H_0} \Big(z^h_{(v,\ell,\mathrm{blue})} + z^h_{(v,\ell,\mathrm{red})} \Big). \tag{20}$$

We proceed to show that the solution $\mathbf{z}$ is feasible, and that the approximation factor matches the lower bound of Theorem 3, in the special case $\Delta_I = 2$.

Lemma 4.2. *The vector* $\mathbf{z}$ *is a feasible solution of the max-min LP associated with* $\mathcal{G}$*.*

Proof. First we note that $\mathbf{z}$ is nonnegative, as $\mathbf{z}^h$ is nonnegative. Then consider a constraint $i \in I$. Choose an arbitrary $v \in V_i$ and let $u = n(i,v)$; observe that $v = n(i,u)$. We have

$$\begin{aligned}
2|H_0| f_i(\mathbf{z}) \;&=\; 2|H_0| \big(\bar{a}_{i,u} z_u + \bar{a}_{i,v} z_v \big) \\
&=\; \sum_{\ell \in H_0} \bar{a}_{i,u} z^h_{(u,\ell,\mathrm{red})} \;+\; \sum_{\ell \in H_0} \bar{a}_{i,u} z^h_{(u,\ell,\mathrm{blue})} \\
&\quad\; + \sum_{\ell \in H_0} \bar{a}_{i,v} z^h_{(v,\ell,\mathrm{red})} \;+\; \sum_{\ell \in H_0} \bar{a}_{i,v} z^h_{(v,\ell,\mathrm{blue})} \\
&=\; \sum_{\ell \in H_0} f^h_{(i,u,\ell)}(\mathbf{z}^h) \;+\; \sum_{\ell \in H_0} f^h_{(i,v,\ell)}(\mathbf{z}^h) \\
&\leq\; \sum_{\ell \in H_0} 1 \;+\; \sum_{\ell \in H_0} 1 \;=\; 2|H_0|.
\end{aligned}$$

□

Lemma 4.3. *The vector* $\mathbf{z}$ *satisfies*

$$\frac{\omega^h(\mathbf{z}^h)}{\omega(\mathbf{z})} \leq 2 \left(1 + \frac{1}{h}\right) \left(1 - \frac{1}{\Delta_K}\right).$$

Proof. Consider an objective $k \in K$. Let $v \in V_k$ and $\delta = |N(k,v)|$; observe that δ does not depend on the choice of v. For any function $\phi(\cdot)$ it holds that

$$\sum_{v \in V_k} \sum_{u \in N(k,v)} \phi(u) = \delta \sum_{v \in V_k} \phi(v).$$

Hence we have

$$\begin{aligned}
\frac{2\delta |H_0||V_k|}{\delta + 1} g_k(\mathbf{z}) &= \frac{2\delta |H_0|}{\delta + 1} \sum_{v \in V_k} \left(\bar{c}_{k,v} z_v + \sum_{u \in N(k,v)} \bar{c}_{k,u} z_u \right) \\
&= 2|H_0| \sum_{v \in V_k} \sum_{u \in N(k,v)} \bar{c}_{k,u} z_u \\
&= \sum_{v \in V_k} \sum_{u \in N(k,v)} \sum_{\ell \in H_0} \left(\bar{c}_{k,u} z^h_{(u,\ell,\mathrm{blue})} + \bar{c}_{k,u} z^h_{(u,\ell,\mathrm{red})} \right) \\
&\geq \sum_{v \in V_k} \sum_{u \in N(k,v)} \sum_{\ell \in H} \left(\bar{c}_{k,u} z^h_{(u,\ell-1,\mathrm{blue})} + \bar{c}_{k,u} z^h_{(u,\ell,\mathrm{red})} \right) \\
&= \sum_{\ell \in H} \sum_{v \in V_k} \left(\delta \bar{c}_{k,v} z^h_{(v,\ell-1,\mathrm{blue})} + \sum_{u \in N(k,v)} \bar{c}_{k,v} z^h_{(v,\ell,\mathrm{red})} \right) \\
&\geq \sum_{\ell \in H} \sum_{v \in V_k} \left(\bar{c}_{k,v} z^h_{(v,\ell-1,\mathrm{blue})} + \sum_{u \in N(k,v)} \bar{c}_{k,v} z^h_{(v,\ell,\mathrm{red})} \right) \\
&= \sum_{\ell \in H} \sum_{v \in V_k} g^h_{(k,\ell,v)}(\mathbf{z}^h) \\
&\geq |H||V_k| \omega^h(\mathbf{z}^h).
\end{aligned}$$

The claim follows since $\delta \leq \Delta_K - 1$, $|H| = h$, and $|H_0| = h + 1$. □

Putting together Corollary 3.13 and Lemmas 4.1, 4.2, and 4.3, we obtain the following result.

Corollary 4.4. *There is a local $2(1+\varepsilon)(1+1/h)(1-1/\Delta_K)$-approximation algorithm for max-min LPs with $\Delta_I = 2$ and $\Delta_K \geq 2$ for any $\varepsilon > 0$ and $h \in \{1, 2, \dots\}$. The running time of the algorithm is $O(h \log 1/\varepsilon)$ synchronous communication rounds.*

To prove Theorem 2, we need to extend the result of Corollary 4.4 to the case $\Delta_I > 2$.

4.2 General max-min LPs

Given a general max-min LP, we replace each constraint $i \in I$ with $|V_i| > 2$ by the $\binom{|V_i|}{2}$ constraints

$$a_{i,u} x_u + a_{i,v} x_v \leq \frac{2}{|V_i|}, \quad \forall\, u, v \in V_i,\ u \neq v. \tag{21}$$

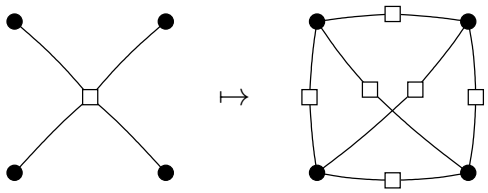

Figure 4: Replacing a constraint of degree 4 by $\binom{4}{2}$ constraints of degree 2.

See Figure 4 for an illustration of how the underlying communication graph $\mathcal{G}$ changes in the case $|V_i| = 4$.

Let us first show that a feasible solution of the modified instance is a feasible solution to the original problem.

Lemma 4.5. *If a solution* $\mathbf{x}$ *satisfies* (21)*, then it also satisfies the original constraint* $f_i(\mathbf{x}) \leq 1$.

Proof. The claim follows from the observation that

$$\begin{aligned} f_i(\mathbf{x}) &= \sum_{v \in V_i} a_{i,v} x_v \\ &= \frac{1}{2(|V_i|-1)} \sum_{v \in V_i} \sum_{u \in V_i \setminus \{v\}} \left(a_{i,u} x_u + a_{i,v} x_v \right) \\ &\leq \frac{1}{2(|V_i|-1)} \sum_{v \in V_i} \sum_{u \in V_i \setminus \{v\}} \frac{2}{|V_i|} = 1. \end{aligned}$$

□

Let us then observe that the optimum of the modified instance is within factor $\Delta_I/2$ of the optimum of the original instance.

Lemma 4.6. *Consider a max-min LP with* $\Delta_I > 2$*. Let* $\mathbf{x}^*$ *be an optimal solution of the max-min LP. Then* $\mathbf{x} = 2\mathbf{x}^*/\Delta_I$ *satisfies* (21) *for each constraint* $i \in I$ *with* $|V_i| > 2$.

Proof. Immediate from construction. □

Hence we can solve an arbitrary max-min LP as follows:

1. Use the reduction in this section to construct an instance with $\Delta_I = 2$.
2. Use the reduction in Section 4.1 to construct a layered max-min LP.
3. Solve the layered max-min LP by using the algorithm of Section 3.
4. Apply (20) to map the solution of the layered max-min LP to a solution of the original max-min LP.

Putting together Corollary 4.4, Lemma 4.5, and Lemma 4.6, we obtain the following approximation guarantee and running time.

Corollary 4.7. *There is a local* $\Delta_I(1+\varepsilon)(1+1/h)(1-1/\Delta_K)$*-approximation algorithm for max-min LPs with* $\Delta_I \geq 2$ *and* $\Delta_K \geq 2$ *for any* $\varepsilon > 0$ *and* $h \in \{1, 2, \dots\}$*. The running time of the algorithm is* $O(h \log 1/\varepsilon)$ *synchronous communication rounds.*

Theorem 2 now follows from Corollaries 2.4 and 4.7.

Remark 3. The local algorithm does not need to know the constants Δ_I and Δ_K. These constants are only used in the derivation of the approximation guarantee. Therefore an implementation of the algorithm can fix some values of ε and h and we can run the same algorithm in any graph. For example, if we simply choose realistic values such as $\varepsilon = 1/30$ and $h = 16$ and run the algorithm, Corollary 4.7 indicates that the worst-case approximation ratio is always within factor 1.1 of the lower bound of Theorem 3, regardless of the values of Δ_I and Δ_K. Note that the lower bound holds even if we had designed a local algorithm for particular values of the constants Δ_I and Δ_K.

4.3 Discussion

In Section 4.1, we created $2|H_0|$ copies of each agent: a red agent and a blue agent on each layer $\ell \in H_0$.

By using the two colours, red and blue, we can break the symmetry in the layered graph $\mathcal{G}_h$. The approach is somewhat similar to the use of bipartite double covers in local vertex cover algorithms [2, 26], in which a 2-coloured graph is constructed, and the colours are exploited in the design of a local algorithm.

The layers play a different role. They can be be seen as an application of the shifting strategy [6, 13], which is a classical technique for designing polynomial-time approximation algorithms.

5 Inapproximability

We proceed to prove Theorem 3. Fix any constants $\Delta_I \geq 2$ and $\Delta_K \geq 2$. Let $\mathcal{A}$ be a local approximation algorithm for max-min LPs; $\mathcal{A}$ may depend on the choice of Δ_I and Δ_K. We will show that the approximation guarantee of $\mathcal{A}$ is strictly larger than $\Delta_I(1 - 1/\Delta_K)$.

5.1 A high-girth graph $\mathcal{G}$

Let r be the local horizon of the algorithm $\mathcal{A}$; again, r may depend on Δ_I and Δ_K. W.l.o.g. we can assume that r is a multiple of 4.

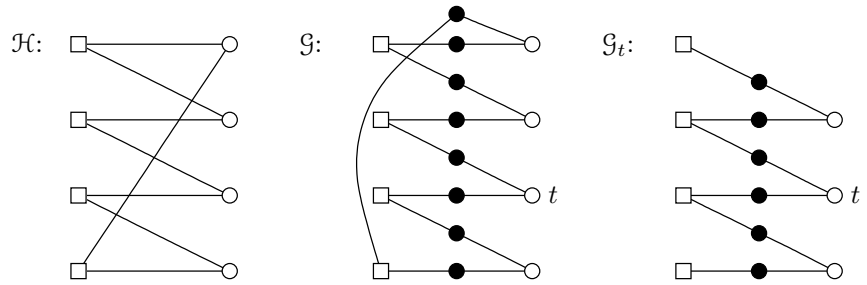

Figure 5: The lower-bound construction for $\Delta_I = 2$, $\Delta_K = 2$ and $r = 4$. The shortest cycle of $\mathcal{G}$ has length larger than $2r + 4$; hence $\mathcal{G}_t$ is a tree regardless of the choice of $t \in K$.

Our lower-bound construction is based on a high-girth graph. Let $\mathcal{G} = (V \cup I \cup K, E)$ be a communication graph with the following properties; see Figure 5 for an illustration:

(i) the degree of each constraint $i \in I$ is Δ_I,

(ii) the degree of each objective $k \in K$ is Δ_K,

(iii) each agent $v \in V$ is adjacent to 1 constraint and 1 objective,

(iv) the weight of each edge is 1,

(v) there is no cycle of length less than $2r + 5$.

Let us first show that such a graph indeed exists, for any choice of Δ_I, Δ_K, and r. We say that a bipartite graph $\mathcal{H} = (I \cup K, E)$ is (a, b)-*regular* if the degree of each node in I is a and the degree of each node in K is b.

Lemma 5.1. *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. We adapt a proof of a similar result [14, Theorem A.2] for d-regular graphs to our needs; for the sake of completeness, we repeat the proof here. 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 $\mathcal{H} = (I \cup K, E)$ 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 $\mathcal{H}_S = (I_S \cup K_S, E_S)$ as follows:

$$\begin{aligned} I_S &= \{0,1\} \times I, \\ K_S &= \{0,1\} \times K, \\ E_S &= \big\{\{(0,i),(0,k)\},\{(1,i),(1,k)\} : \{i,k\} \in S\big\} \\ &\quad \cup \big\{\{(0,i),(1,k)\},\{(1,i),(0,k)\} : \{i,k\} \in E \setminus S\big\}. \end{aligned}$$

The graph $\mathcal{H}_S$ is an (a, b)-regular bipartite graph. Furthermore, $\mathcal{H}_S$ 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 $\mathcal{H}_S$ is strictly smaller than the number of cycles of length c in $\mathcal{H}$. 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 $\mathcal{H}$, we have in $\mathcal{H}_S$ 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 $\mathcal{H}_S$ is therefore equal to the number of cycles of length c in $\mathcal{H}$. The choice $S = E$ doubles the number of such cycles; therefore some other choice necessarily decreases the number of such cycles. □

To construct a communication graph $\mathcal{G} = (V \cup I \cup K, E)$ that meets our requirements, we first apply Lemma 5.1 to find a (Δ_I, Δ_K)-regular bipartite graph $\mathcal{H} = (I \cup K, E')$, and then subdivide each edge of $\mathcal{H}$. More precisely, we begin with $V \leftarrow \emptyset$ and $E \leftarrow \emptyset$. Then for each edge $\{i, k\} \in E'$ we add a new agent v to V and the weight-1 edges $\{i, v\}$ and $\{k, v\}$ to E.

Choose the port numbering and/or node identifiers in $\mathcal{G}$ in an arbitrary manner. Then apply the local algorithm $\mathcal{A}$ to solve the max-min LP associated with $\mathcal{G}$.

Lemma 5.2. *Let* $\mathbf{y}$ *be the output of* $\mathcal{A}$ *on* $\mathcal{G}$*. Then there exists an objective* $t \in K$ *such that* $g_t(\mathbf{y}) \leq \Delta_K / \Delta_I$*.*

Proof. Vector $\mathbf{y}$ must be a feasible solution of the max-min LP. Therefore

$$\sum_{k \in K} g_k(\mathbf{y}) = \sum_{v \in V} y_v = \sum_{i \in I} f_i(\mathbf{y}) \leq |I| = \frac{\Delta_K}{\Delta_I} |K|. \qquad \square$$

5.2 A tree $\mathcal{G}_t$

Now choose a $t \in K$ as in Lemma 5.2. Let $\mathcal{G}_t$ be the subgraph of $\mathcal{G}$ induced by all nodes within distance $r + 2$ from t. Port numbers and node identifiers in $\mathcal{G}_t$ are inherited from $\mathcal{G}$.

By the construction of $\mathcal{G}$, there is no cycle in $\mathcal{G}_t$. As r is a multiple of 4, the leaves of the tree $\mathcal{G}_t$ are constraints.

Lemma 5.3. *The optimum of the max-min LP associated with* $\mathcal{G}_t$ *is greater than* $\Delta_K - 1$*.*

Proof. Construct a feasible solution $\mathbf{x}$ as follows. Let $D = \max\{\Delta_I, \Delta_K + 1\}$. If the distance between an agent $v \in V$ and the objective t in $\mathcal{G}_t$ 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 exactly Δ_I neighbours, and the distance between t and i is $4j + 2$ for some j. One of the agents adjacent to i is at distance $4j + 1$ from t while $\Delta_I - 1$ agents are at distance $4j + 3$. Thus

$$f_i(\mathbf{x}) = 1 - \frac{1}{D^{2j+1}} + \frac{\Delta_I - 1}{D^{2j+2}} < 1.$$

We proceed to show that $g_k(\mathbf{x}) > \Delta_K - 1$ for all $k \in K$. First, consider the objective t. We have

$$g_t(\mathbf{x}) = \Delta_K(1 - 1/D) > \Delta_K - 1.$$

Second, consider an objective $k \in K \setminus \{t\}$. The objective k has Δ_K neighbours and the distance between t and k is $4j$ for some j. Thus

$$g_k(\mathbf{x}) = \frac{1}{D^{2j}} + \frac{\Delta_K - 1}{1 - 1/D^{2j+1}} > \Delta_K - 1. \qquad \square$$

Now apply the algorithm $\mathcal{A}$ to the max-min LP associated with $\mathcal{G}_t$. The algorithm produces a feasible solution $\mathbf{y}'$. The radius-r neighbourhoods of the agents $v \in V_t$ are identical in $\mathcal{G}$ and $\mathcal{G}_t$; therefore the algorithm $\mathcal{A}$ must produce the same output. In particular, $g_t(\mathbf{y}') = g_t(\mathbf{y}) \leq \Delta_K/\Delta_I$.

However, Lemma 5.3 shows that the optimum of $\mathcal{G}_t$ is strictly larger than $\Delta_K - 1$; hence the approximation factor of $\mathcal{A}$ must be strictly larger than $\Delta_I(1 - 1/\Delta_K)$. Theorem 3 follows.

Acknowledgements

We thank Boaz Patt-Shamir for discussions, and anonymous reviewers for their helpful feedback.

This research was supported in part by the Academy of Finland, Grants 116547 and 117499, by Helsinki Graduate School in Computer Science and Engineering (Hecse), and by the Foundation of Nokia Corporation.

References

[1] Dana Angluin. 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)*, pages 82–93. ACM Press, New York, NY, USA, 1980.

[2] Matti Åstrand, Patrik Floréen, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. A local 2-approximation algorithm for the vertex cover problem. In *Proc. 23rd International Symposium on Distributed Computing (DISC, Elche, Spain, September 2009)*, volume

5805 of *Lecture Notes in Computer Science*, pages 191–205. Springer, Berlin, Germany, 2009.

[3] Matti Åstrand and Jukka Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proc. 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA, Santorini, Greece, June 2010)*, pages 294–302. ACM Press, New York, NY, USA, 2010.

[4] Baruch Awerbuch and Michael Sipser. Dynamic networks are as fast as static networks. In *Proc. 29th Annual Symposium on Foundations of Computer Science (FOCS, White Plains, NY, USA, October 1988)*, pages 206–219. IEEE, Piscataway, NJ, USA, 1988.

[5] Baruch Awerbuch and George Varghese. Distributed program checking: a paradigm for building self-stabilizing distributed protocols. In *Proc. 32nd Annual Symposium on Foundations of Computer Science (FOCS, San Juan, Puerto Rico, October 1991)*, pages 258–267. IEEE, Piscataway, NJ, USA, 1991.

[6] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[7] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Proc. 22nd International Symposium on Distributed Computing (DISC, Arcachon, France, September 2008)*, volume 5218 of *Lecture Notes in Computer Science*, pages 78–92. Springer, Berlin, Germany, 2008.

[8] Patrik Floréen, Marja Hassinen, Petteri Kaski, and Jukka Suomela. Tight local approximation results for max-min linear programs. In *Proc. 4th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors, Reykjavík, Iceland, July 2008)*, volume 5389 of *Lecture Notes in Computer Science*, pages 2–17. Springer, Berlin, Germany, 2008.

[9] Patrik Floréen, Joel Kaasinen, Petteri Kaski, and Jukka Suomela. An optimal local approximation algorithm for max-min linear programs. In *Proc. 21st Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA, Calgary, Canada, August 2009)*, pages 260–269. ACM Press, New York, NY, USA, 2009.

[10] Patrik Floréen, Petteri Kaski, Topi Musto, and Jukka Suomela. Approximating max-min linear programs with local algorithms. In *Proc. 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS, Miami, FL, USA, April 2008)*. IEEE, Piscataway, NJ, USA, 2008.

[11] Patrik Floréen, Petteri Kaski, Valentin Polishchuk, and Jukka Suomela. Almost stable matchings by truncating the Gale–Shapley algorithm. *Algorithmica*, 58(1):102–118, 2010.

[12] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, San Francisco, CA, USA, January 1998)*, pages 219–225. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[13] Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.

[14] Shlomo Hoory. *On Graphs of High Girth*. PhD thesis, Hebrew University, Jerusalem, March 2002.

[15] Fabian Kuhn. *The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives*. PhD thesis, ETH Zürich, 2005.

[16] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Fault-tolerant clustering in ad hoc and sensor networks. In *Proc. 26th IEEE International Conference on Distributed Computing Systems (ICDCS, Lisboa, Portugal, July 2006)*. IEEE Computer Society Press, Los Alamitos, CA, USA, 2006.

[17] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, Miami, FL, USA, January 2006)*, pages 980–989. ACM Press, New York, NY, USA, 2006.

[18] Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. What can be approximated locally? In *Proc. 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA, Munich, Germany, June 2008)*, pages 46–54. ACM Press, New York, NY, USA, 2008.

[19] Christoph Lenzen, Jukka Suomela, and Roger Wattenhofer. Local algorithms: Self-stabilization on speed. In *Proc. 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS, Lyon, France, November 2009)*, volume 5873 of *Lecture Notes in Computer Science*, pages 17–34. Springer, Berlin, Germany, 2009.

[20] Christoph Lenzen and Roger Wattenhofer. Leveraging Linial's locality limit. In *Proc. 22nd International Symposium on Distributed Computing (DISC, Arcachon, France, September 2008)*, volume 5218 of *Lecture Notes in Computer Science*, pages 394–407. Springer, Berlin, Germany, 2008.

[21] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[22] Alain Mayer, Moni Naor, and Larry Stockmeyer. Local computations on static and dynamic graphs. In *Proc. 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS, Tel Aviv, Israel, January 1995)*, pages 268–278. IEEE, Piscataway, NJ, USA, 1995.

[23] Thomas Moscibroda. *Locality, Scheduling, and Selfishness: Algorithmic Foundations of Highly Decentralized Networks*. PhD thesis, ETH Zürich, 2006.

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

[25] Christos H. Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix. In *Proc. 25th Annual ACM Symposium on Theory of Computing (STOC, San Diego, CA, USA, May 1993)*, pages 121–129. ACM Press, New York, NY, USA, 1993.

[26] Valentin Polishchuk and Jukka Suomela. A simple local 3-approximation algorithm for vertex cover. *Information Processing Letters*, 109(12):642–645, 2009.

[27] Jukka Suomela. *Optimisation Problems in Wireless Sensor Networks: Local Algorithms and Local Graphs*. PhD thesis, University of Helsinki, Department of Computer Science, Helsinki, Finland, May 2009.

[28] Jukka Suomela. Survey of local algorithms. `http://www.iki.fi/jukka.suomela/local-survey`, 2009. Manuscript submitted for publication.

[29] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proc. 42nd Annual Symposium on Foundations of Computer Science (FOCS, Las Vegas, NV, USA, October 2001)*, pages 538–546. IEEE Computer Society Press, Los Alamitos, CA, USA, 2001.