Approximating Max-min Linear Programs with Local Algorithms

Patrik Floréen Petteri Kaski Topi Musto Jukka Suomela

Helsinki Institute for Information Technology HIIT, Department of Computer Science,
University of Helsinki, P.O. Box 68, FI-00014 University of Helsinki, Finland
E-mail: {firstname.lastname}@cs.helsinki.fi

Abstract

A local algorithm is a distributed algorithm where each node must operate solely based on the information that was available at system startup within a constant-size neighbourhood of the node. We study the applicability of local algorithms to max-min LPs where the objective is to maximise $\min_k \sum_v c_{kv}x_v$ *subject to* $\sum_v a_{iv}x_v \leq 1$ *for each* i *and* $x_v \geq 0$ *for each* v. *Here* $c_{kv} \geq 0$, $a_{iv} \geq 0$, *and the support sets* $V_i = \{v : a_{iv} > 0\}$, $V_k = \{v : c_{kv} > 0\}$, $I_v = \{i : a_{iv} > 0\}$ *and* $K_v = \{k : c_{kv} > 0\}$ *have bounded size. In the distributed setting, each agent* v *is responsible for choosing the value of* x_v, *and the communication network is a hypergraph* $\mathcal{H}$ *where the sets* V_k *and* V_i *constitute the hyperedges. We present inapproximability results for a wide range of structural assumptions; for example, even if* $|V_i|$ *and* $|V_k|$ *are bounded by some constants larger than* 2, *there is no local approximation scheme. To contrast the negative results, we present a local approximation algorithm which achieves good approximation ratios if we can bound the relative growth of the vertex neighbourhoods in* $\mathcal{H}$.

1. Introduction

We study the limits of what can and what cannot be achieved by *local algorithms* [13]. We focus on the *approximability* of a certain class of linear optimisation problems, which generalises beyond widely studied packing LPs; the emphasis is on deterministic algorithms and worst-case analysis.

1.1. Local algorithms

A local algorithm is a distributed algorithm where each node must operate solely based on the information that was available at system startup within a constant-size neighbourhood of the node. We focus on problems where the size of the input per node is bounded by a constant; in such problems, local algorithms provide an extreme form of scalability: the communication, space and time complexity of a local algorithm is constant per node, and a local algorithm scales to an arbitrarily large or even infinite network.

The study of local algorithms has several uses beyond providing highly scalable distributed algorithms. The existence of a local algorithm shows that the function can be computed by *bounded-fan-in, constant-depth Boolean circuits*; we can say that the function is in the class NC^0. A local algorithm is also an efficient centralised algorithm: the time complexity of the centralised algorithm is *linear* in the number of nodes; furthermore, due to spatial locality in memory accesses, we may be able to achieve a low I/O complexity in the *external memory* [18] model of computation. In certain problems, a local approximation algorithm can be used to construct a *sublinear time* algorithm which approximates the size of the optimal solution, assuming that we tolerate an additive error and some probability of failure [16]. A local algorithm can be turned into an efficient *self-stabilising* algorithm [3]; the time to stabilise is constant [1]. Finally, the existence and nonexistence of local algorithms gives us insight into the *algorithmic value of information* in distributed decision-making [14].

1.2. Max-min packing problem

In this section, we define the optimisation problem that we study in this work. Let V, I and K be index sets with $I \cap K = \emptyset$; we say that each $v \in V$ is an *agent*, each $k \in K$ is a *beneficiary party*, and each $i \in I$ is a *resource* (*constraint*). We assume that one unit of activity by v benefits the party k by $c_{kv} \geq 0$ units and consumes $a_{iv} \geq 0$ units of the resource i; the objective is to set the activities to provide a fair share of benefit for each party. In notation, assuming that the activity of agent v is x_v units, the objective is to

$$\begin{aligned} \text{maximise} \quad & \omega = \min_{k \in K} \sum_{v \in V} c_{kv} x_v \\ \text{subject to} \quad & \sum_{v \in V} a_{iv} x_v \leq 1 \qquad \forall\, i \in I, \qquad (1) \\ & x_v \geq 0 \qquad \forall\, v \in V. \end{aligned}$$

Throughout this work we assume that the *support sets* defined for all $i \in I$, $k \in K$, and $v \in V$ by

$$\begin{aligned} V_i &= \{v \in V : a_{iv} > 0\}, \\ V_k &= \{v \in V : c_{kv} > 0\}, \\ I_v &= \{i \in I : a_{iv} > 0\}, \\ K_v &= \{k \in K : c_{kv} > 0\} \end{aligned}$$

have bounded size. That is, we focus on instances of (1) such that $|I_v| \le \Delta_V^I$, $|K_v| \le \Delta_V^K$, $|V_i| \le \Delta_I^V$ and $|V_k| \le \Delta_K^V$ for some constants Δ_V^I, Δ_V^K, Δ_I^V and Δ_K^V. To avoid uninteresting degenerate cases, we furthermore assume that I_v, V_i and V_k are nonempty.

1.3. LP formulation

If the sets V, I and K are finite, the problem can be represented using matrix notation. Let A be the nonnegative $|I| \times |V|$ matrix where the entry at row i, column v is a_{iv}; define C analogously. We write a_i for the row i of A and c_k for the row k of C. Let x be a column vector of length $|V|$. The goal is to

$$\begin{aligned} \text{maximise} \quad & \omega = \min_{k \in K} c_k x \\ \text{subject to} \quad & Ax \le \mathbf{1}, \\ & x \ge \mathbf{0}. \end{aligned}$$

In the special case $|K| = 1$, this is the widely studied fractional packing problem:

$$\begin{aligned} \text{maximise} \quad & cx \\ \text{subject to} \quad & Ax \le \mathbf{1}, \\ & x \ge \mathbf{0}. \end{aligned}$$

This simple linear program (LP) has nonnegative coefficients in c and A. We refer to a problem of this form as a *packing LP*; the dual is a *covering LP*. Naturally the case of any finite K can also be written as a linear program, but the constraint matrix is no longer nonnegative:

$$\begin{aligned} \text{maximise} \quad & \omega \\ \text{subject to} \quad & Ax \le \mathbf{1}, \\ & \omega \mathbf{1} - Cx \le \mathbf{0}, \\ & \omega, x \ge \mathbf{0}. \end{aligned}$$

1.4. Distributed setting

We construct the hypergraph $\mathcal{H} = (V, \mathcal{E})$ where the hyperedges are

$$\mathcal{E} = \{V_i : i \in I\} \cup \{V_k : k \in K\}.$$

This is the *communication graph* in our distributed optimisation problem. The variable x_v is controlled by the agent $v \in V$, and two agents $u, v \in V$ can communicate directly with each other if they are adjacent in $\mathcal{H}$. We write $d_{\mathcal{H}}(u, v)$ for the shortest-path distance between u and v in $\mathcal{H}$. The agents are cooperating, not selfish; the difficulty arises from the fact that the agents have to make decisions based on incomplete information.

Initially, each agent $v \in V$ knows only the following local information: the identity of its neighbours in the graph $\mathcal{H}$; the sets I_v and K_v; the values a_{iv} for each $i \in I_v$; and the values c_{kv} for each $k \in K_v$. That is, v knows with whom it is competing for which resources, and with whom it is working together to benefit which parties.

When we compare the present work with previous work, we often mention the special case $|K| = 1$, as this corresponds to the widely studied packing LP. However, in this case the size of V_k is not bounded by a constant Δ_K^V: we have $V_k = V$ for the sole $k \in K$. Therefore we introduce a restricted variant of the distributed setting, which we call *collaboration-oblivious*. In this variant, the hyperedges are $\mathcal{E} = \{V_i : i \in I\}$. Whenever we study related work on the packing LP, we focus on the collaboration-oblivious setting.

1.5. Local setting

We are interested in solving the problem (1) by using a local algorithm. Let $r = 1, 2, \ldots$ be the *local horizon* of the algorithm; this is a constant which does not depend on the particular problem instance at hand. Let

$$B_{\mathcal{H}}(v, r) = \{u \in V : d_{\mathcal{H}}(u, v) \le r\}$$

be the set of nodes which have distance at most r to the node v in $\mathcal{H}$. The agent v must choose the value x_v based on the information that is initially available in the agents $B_{\mathcal{H}}(v, r)$.

We focus on the case where the size of the input is constant per node. The elements a_{iv} and c_{kv} are represented at some finite precision. Furthermore, we assume that the nodes have constant-size locally unique identifiers; i.e., any node can be identified uniquely within the local horizon.

1.6. Approximation

A local algorithm has the *approximation ratio* α for some $\alpha > 1$ if the decisions x_v are a feasible solution and the value ω is within a factor α of the global optimum. A family of local algorithms is a *local approximation scheme* if we can achieve any $\alpha > 1$ by choosing a large enough local horizon r.

1.7. Contributions

In Section 4 we show that while a simple algorithm achieves the approximation ratio Δ_I^V for (1), no local algo-

rithm can achieve an approximation ratio less than

$$\frac{\Delta_I^V + 1}{2} - \frac{1}{2\Delta_K^V - 2}$$

in the general case. In Section 5 we present a local approximation algorithm which can achieve an improved approximation ratio if we can bound the relative growth of the vertex neighbourhoods in $\mathcal{H}$.

2. Applications

Consider a two-tier sensor network: battery-powered sensor devices generate some data; the data is transmitted to a battery-powered relay node, which forwards the data to a sink node. The sensor network is used to monitor the physical areas K. Let S be the set of sensors, and let T be the set of relays; choose $I = S \cup T$.

For each sensors device $s \in S$, there may be multiple relays $t \in T$ which are within the reach of the radio of s; we say that there is a wireless link (s,t) from s to t. The set V consists of all such wireless links, and the variable $x_{(s,t)}$ indicates how much data is transmitted from s via t to the sink. Transmitting one unit of data on the link $v = (s,t) \in V$ and forwarding it to the sink consumes the fraction a_{sv} of the energy resources of the sensor s and also the fraction a_{tv} of the energy resources of the relay t.

Let $c_{kv} = 1$ for each link $v = (s,t)$ if the sensor s is able to monitor the physical area $k \in K$. Now (1) captures the following optimisation problem: choose the data flows in the sensor network so that we maximise the minimum amount of data that is received from any physical area. Equivalently, we can interpret the objective as follows: choose data flows such that the lifetime of the network (time until the first sensor or relay runs out of the battery) is maximised, assuming that we receive data at the same average rate from each physical area.

Similar constructions have applications beyond the field of sensor networks: consider, for example, the case where each $k \in K$ is a major customer of an Internet service provider (ISP), each $s \in S$ is a bounded-capacity last-mile link between the customer and the ISP, and each $t \in T$ is a bounded-capacity access router in the ISP's network.

3. Related work

Papadimitriou and Yannakakis [15] present the *safe algorithm* for the packing LP. The agent v chooses

$$x_v = \min_{i \in I_v} \frac{1}{a_{iv}|V_i|}. \tag{2}$$

This is a local Δ_I^V-approximation algorithm with horizon $r = 1$.

Kuhn et al. [9] give a distributed approximation scheme for the packing LP and covering LP. The algorithm provides a local approximation scheme for some families of packing and covering LPs. For example, let $a_{iv} \in \{0,1\}$ for all i,v. Then for each Δ_I^V, Δ_V^I and $\alpha > 1$, there is a local algorithm with some constant horizon r which achieves an α-approximation. Our work shows that such local approximation schemes do not exist for (1).

Another distributed approximation scheme by Kuhn et al. [9] forms several decompositions of $\mathcal{H}$ into subgraphs, solves the optimisation problem optimally for each subgraph, and combines the solutions. However, the algorithm is not a local approximation algorithm in the strict sense that we use here: to obtain any constant approximation ratio, the local horizon must extend (logarithmically) as the number of variables increases. Also Bartal et al. [2] present a distributed but not local approximation scheme for the packing LP.

Kuhn and Wattenhofer [10] present a family of local, constant-factor approximation algorithms of the covering LP that is obtained as an LP relaxation of the minimum dominating set problem. Kuhn et al. [7] present a local, constant-factor approximation of the packing and covering LPs in unit-disk graphs.

There are few examples of local algorithms which approximate linear problems beyond packing and covering LPs. Kuhn et al. [8] study an LP relaxation of the k-fold dominating set problem and obtain a local constant-factor approximation for bounded-degree graphs.

For combinatorial problems, there are both negative [6, 11] and positive [4, 8, 10, 13, 17] results on the applicability of local algorithms.

4. Inapproximability

Even though the safe algorithm [15] was presented for the special case of $|K| = 1$, $c = \mathbf{1}$, and finite I and V, we note that the safe solution x defined by (2) and an optimal solution x^* also satisfy

$$\begin{aligned} \min_{k \in K} \sum_{v \in V_k} c_{kv} x_v^* &\leq \min_{k \in K} \sum_{v \in V_k} c_{kv} \Delta_I^V x_v \\ &= \Delta_I^V \min_{k \in K} \sum_{v \in V_k} c_{kv} x_v. \end{aligned}$$

Therefore we obtain a local approximation algorithm with the approximation ratio Δ_I^V for (1).

One could hope that widening the local horizon beyond $r = 1$ would significantly improve the quality of approximation. In general, this is not the case: no matter what constant local horizon r we use, we cannot improve the approximation ratio beyond $\Delta_I^V/2$. In this section, we prove the following theorem.

Theorem 1. *Let $\Delta_I^V \geq 2$ and $\Delta_K^V \geq 2$ be given. There is no local approximation algorithm for* (1) *with the approximation ratio less than $\Delta_I^V/2 + 1/2 - 1/(2\Delta_K^V - 2)$. This holds even if we make the following restrictions: $a_{iv} \in \{0,1\}$, $\Delta_V^I = 1$ and $\Delta_V^K = 1$.*

We emphasise that the local algorithm could even choose any local horizon r depending on the bounds Δ_I^V, Δ_K^V, Δ_V^I and Δ_V^K. Nevertheless, an arbitrarily low approximation ratio cannot be achieved if $\Delta_I^V \geq 3$ or $\Delta_K^V \geq 3$. In the case $\Delta_I^V = \Delta_K^V = 2$ the existence of a local approximation scheme remains an open question.

Analogous proof techniques, using constructions based on regular bipartite high-girth graphs, have been applied in previous work to prove the local inapproximability of packing and covering LPs [9] and combinatorial problems [6].

4.1. Proof outline

Choose any local approximation algorithm $\mathcal{A}$ for the problem (1). Let $r \geq 1$ be the local horizon of $\mathcal{A}$ and let α be the approximation ratio of $\mathcal{A}$. We derive a lower bound for α by constructing two instances of (1), $\mathcal{S}$ and $\mathcal{S}'$, such that certain sets of nodes in the two instances have identical radius-r neighbourhoods in both instances. Consequently, the deterministic local algorithm $\mathcal{A}$ must make the same choices for these nodes in both instances. The nodes with identical views are selected based on the solution of $\mathcal{S}$ computed by $\mathcal{A}$, which enables us to obtain a lower bound on α by showing that this solution is necessarily suboptimal as a solution of $\mathcal{S}'$.

4.2. Construction of $\mathcal{S}$

We now proceed with the detailed construction of the instance $\mathcal{S}$. The constructions used in the proof are illustrated in Figure 1.

Let $\Delta_I^V \geq 2$ and $\Delta_K^V \geq 2$; without loss of generality we can assume that at least one of the inequalities is strict because setting $\Delta_I^V = \Delta_K^V = 2$ in the theorem statement yields the trivial bound $\alpha \geq 1$. Let $d = \Delta_I^V - 1$ and $D = \Delta_K^V - 1$. Observe that $dD > 1$. Let $R > r$; the precise value of R is chosen later and will depend on d, D and α only.

Let $\mathcal{Q}$ be a $d^R D^{R-1}$-regular bipartite graph with no cycles consisting of less than $4r+2$ edges. (A random regular bipartite graph with sufficiently many nodes has this property with positive probability [12].) The graph $\mathcal{Q}$ provides the template for constructing the hypergraph underlying the instance $\mathcal{S}$.

Before describing the construction, we first introduce some terminology. A *complete (d,D)-ary hypertree* of *height h* is defined inductively as follows. For $h = 0$, the hypertree consists of exactly one node and no edges; the *level* of the node is 0. For $h > 0$, start with a complete (d,D)-ary hypertree of height $h-1$. For each node v at level $h-1$, introduce a new hyperedge and new nodes as follows. If $h-1$ is even, the new hyperedge consists of the node v and d new nodes. If $h-1$ is odd, the new hyperedge consists of the node v and D new nodes. For future reference, we call these hyperedges of types I and II, respectively. The new nodes have level h in the constructed hypertree. The constructed hypertree is a complete (d,D)-ary hypertree of height h. The *root* of the hypertree is the node at level 0, the *leaves* are the nodes at level h. Each level ℓ has either $(dD)^{\ell/2}$ or $(dD)^{(\ell-1)/2}d$ nodes depending on whether ℓ is even or odd, respectively. See Figure 1 for an illustration.

We now construct the hypergraph underlying $\mathcal{S}$. Denote by Q the vertex set of $\mathcal{Q}$. Form a hypergraph $\mathcal{H}$ by taking $|Q|$ node-disjoint copies of a complete (d,D)-ary hypertree of height $2R-1$. For $q \in Q$, denote the copy corresponding to q by $\mathcal{T}_q$. Denote the node set of $\mathcal{T}_q$ by T_q. For $\ell = 0, 1, \ldots, 2R-1$, denote the set of nodes at level ℓ in $\mathcal{T}_q$ by $T_q(\ell)$. Denote the set of leaf nodes in $\mathcal{T}_q$ by $L_q = T_q(2R-1)$.

Observe that the number of leaf nodes in each $\mathcal{T}_q$ is equal to the degree of every vertex in $\mathcal{Q}$. For each vertex $q \in Q$ and each leaf node $v \in L_q$, associate with v a unique edge of $\mathcal{Q}$ incident with the vertex q. Each edge of $\mathcal{Q}$ is now associated with exactly two leaf nodes; by construction, these leaf nodes always occur in different hypertrees $\mathcal{T}_q$. For a leaf $v \in \bigcup_q L_q$, let $f(v)$ be the other leaf associated with the same edge of $\mathcal{Q}$. Observe that $f(f(v)) = v$ holds for all $v \in \bigcup_q L_q$; in particular, f is a permutation of $\bigcup_q L_q$. To complete the construction of $\mathcal{H}$, add the hyperedge $\{v, f(v)\}$ to $\mathcal{H}$ for each $v \in \bigcup_q L_q$. Call these hyperedges type III hyperedges.

Let us now define the instance $\mathcal{S}$ of (1) based on the hypergraph $\mathcal{H}$. Let the set of agents V be the node set of $\mathcal{H}$. For each hyperedge e of type I, there is a resource $i \in I$; let $a_{iv} = 1$ if $v \in e$, otherwise $a_{iv} = 0$. For each hyperedge e of type II, there is a beneficiary party $k \in K$; let $c_{kv} = 1/D$ if $v \in e$, otherwise $c_{kv} = 0$. For each hyperedge e of type III, there is a beneficiary party $k \in K$; let $c_{kv} = 1$ if $v \in e$, otherwise $c_{kv} = 0$. The locally unique identifiers of the agents can be chosen in an arbitrary manner. (This proof applies also if the identifiers are globally unique; for example, we can equally well consider the standard definition where the identifiers are a permutation of $1, 2, \ldots, |V|$.) This completes the construction of $\mathcal{S}$. Observe that $\mathcal{S}$ has $\mathcal{H}$ as its underlying hypergraph.

4.3. Construction of $\mathcal{S}'$

Next we construct another instance of (1), called $\mathcal{S}'$, by restricting to a part of $\mathcal{S}$. To select the part, we apply the algorithm $\mathcal{A}$ to the instance $\mathcal{S}$. We do not care what is the optimal solution of $\mathcal{S}$; all that matters at this point is the fact

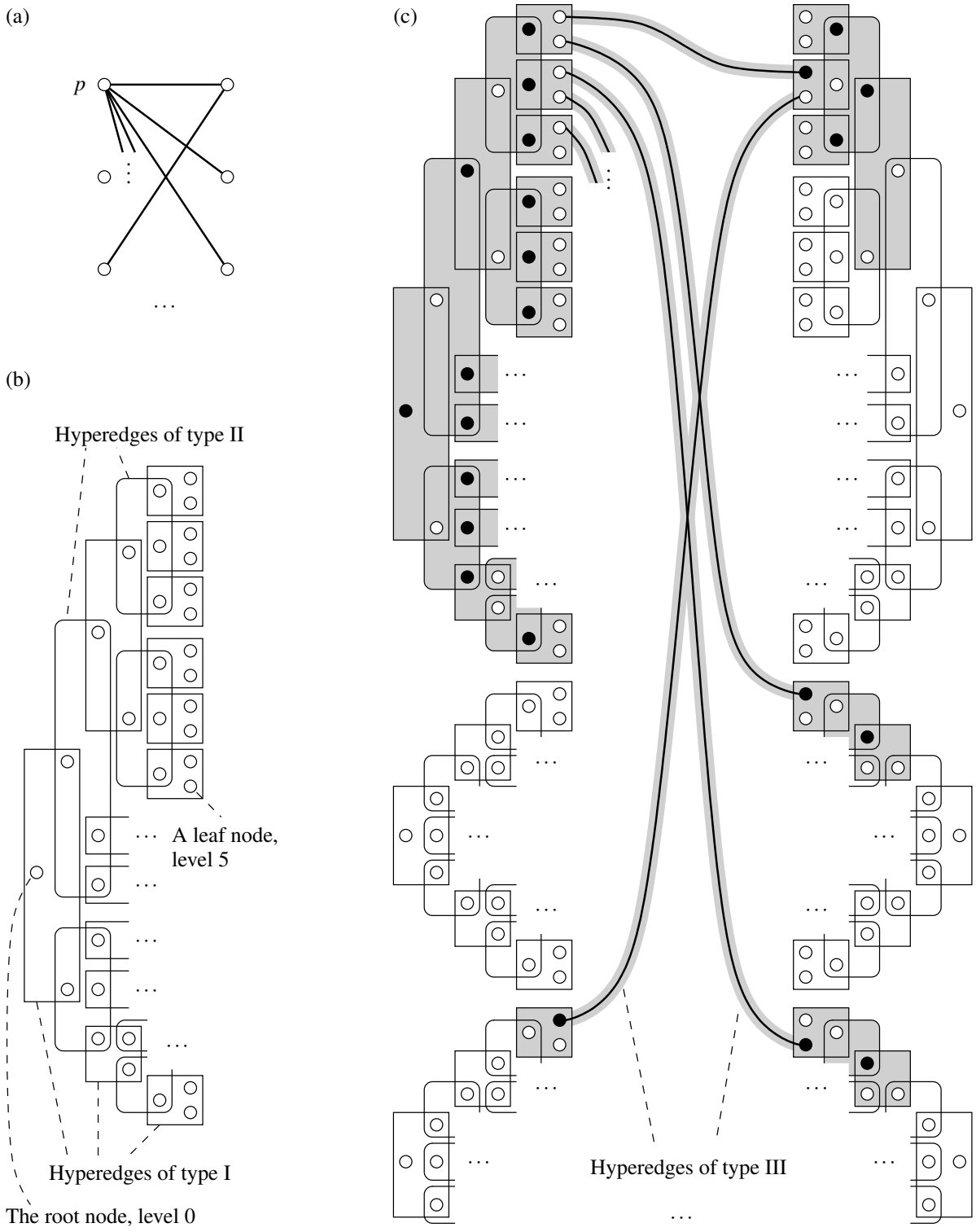

Figure 1. The construction of $\mathcal{S}$, in the case $d = 2$, $D = 3$, $r = 2$, $R = 3$. (a) A small part of the bipartite, 72-regular, high-girth graph Ω. (b) A complete (2,3)-ary hypertree of height 5, with 72 leaves. (c) The underlying hypergraph of $\mathcal{S}$. Grey highlighting indicates the underlying hypergraph of $\mathcal{S}'$; black circles are the variables of $\mathcal{S}'$ which we set to 1 in Section 4.5.

that each agent $v \in V$ must choose some value $x_v \geq 0$. In particular, we pay attention to the values x_v at the leaf nodes $v \in \cup_q L_q$.

For all $q \in Q$, let

$$\delta(q) = \sum_{v \in L_q} \left(x_v - x_{f(v)} \right). \tag{3}$$

For all $P \subseteq Q$, let $\delta(P) = \sum_{q \in P} \delta(q)$. Because f is a permutation of $\cup_q L_q$ with $f(f(v)) = v$ for all $v \in \cup_q L_q$, we have $\delta(Q) = 0$. Thus, there exists a $p \in Q$ with $\delta(p) \geq 0$.

The instance $\mathcal{S}'$ is now constructed based on p. The set of agents in $\mathcal{S}'$ is

$$V' = T_p \cup \bigcup_{u \in L_p} B_{\mathcal{H}}(u, 2r),$$

the set of resources is

$$I' = \{ i \in I : V_i \subseteq V' \},$$

and the set of beneficiary parties is

$$K' = \{ k \in K : V_k \subseteq V' \}.$$

The coefficients a_{iv} and c_{kv} for $i \in I'$, $k \in K'$, and $v \in V'$ are the same as in the instance $\mathcal{S}$. The locally unique identifiers of the agents $v \in V'$ are the same as in the instance $\mathcal{S}$. (If we prefer globally unique identifiers which are a permutation of $1, 2, \ldots, |V'|$, we can add redundant variables to V'.)

4.4. The structure of $\mathcal{S}'$

Next we show that the structure of $\mathcal{S}'$ is tree-like, that is, there are no cycles in the hypergraph $\mathcal{H}'$ defined by the instance $\mathcal{S}'$; by construction, $\mathcal{H}'$ is a subgraph of $\mathcal{H}$.

For each $q \in Q$, the subgraph induced by T_q in $\mathcal{H}$ is a hypertree. Furthermore, the subsets T_q form a partition of V. Therefore any cycle in $\mathcal{H}$ and, therefore, any cycle in $\mathcal{H}'$ must involve hyperedges which cross between the subsets T_q and, finally, return back to the same subset.

The only hyperedges which connect nodes in T_q and T_w for distinct $q, w \in Q$ are the hyperedges of type III. There is at most one such hyperedge for any fixed $q \neq w$; this hyperedge corresponds to the edge $\{q, w\}$ in the graph $\mathcal{Q}$. Therefore a cycle in $\mathcal{H}'$ implies a cycle in $B_{\mathcal{Q}}(p, 2r)$; this implies a cycle of length at most $4r+1$ in $\mathcal{Q}$; by the choice of $\mathcal{Q}$, no such cycle exists.

4.5. A feasible solution of $\mathcal{S}'$

Next we show that there is a feasible solution $\hat{x}$ of $\mathcal{S}'$ with $\omega = 1$. Let u be the root node in $\mathcal{T}_p$. By construction, $u \in T_p \subseteq V'$. For each $v \in V'$, let $\hat{x}_v = 1$ if $d_{\mathcal{H}'}(u, v)$ is even; otherwise, let $\hat{x}_v = 0$. See Figure 1 for an illustration.

Because $\mathcal{S}'$ is tree-like, there is a unique path connecting u to v in $\mathcal{H}'$ for each $v \in V'$. In particular, this path is a shortest path and has length $d_{\mathcal{H}'}(u, v)$. Observe that hyperedges of resources and beneficiary parties alternate in paths from u. By the structure of $\mathcal{S}$ and $\mathcal{S}'$, it follows that the hyperedges of resources (type I) have a unique node with even distance to u. Therefore, $\sum_{v \in V'} a_{iv} \hat{x}_v = 1$ for each $i \in I'$; the solution is feasible. Analogously, the hyperedges of beneficiary parties (types II and III) have a unique node with odd distance to u. Therefore, $\sum_{v \in V'} c_{kv} \hat{x}_v = 1$ for each $k \in K'$, implying $\omega = 1$.

4.6. The solution achieved by $\mathcal{A}$ in $\mathcal{S}'$

Now we apply $\mathcal{A}$ to $\mathcal{S}'$. The local radius-r view of the nodes $v \in T_p$ is identical in both $\mathcal{S}$ and $\mathcal{S}'$. In particular, the deterministic local algorithm $\mathcal{A}$ must make the same choices x_v for $v \in T_p$ in both instances.

As there is a feasible solution with $\omega = 1$, the approximation algorithm $\mathcal{A}$ must choose a solution x with

$$\sum_{v \in V'} c_{kv} x_v \geq \frac{1}{\alpha} \quad \text{for all } k \in K'.$$

We proceed in levels $\ell = 0, 1, \ldots, 2R-1$ of $\mathcal{T}_p$. We study the *total* value assigned to the variables at level ℓ, defined by

$$S(\ell) = \sum_{v \in T_p(\ell)} x_v.$$

Recall that $|T_p(\ell)| = (dD)^{l/2}$ for ℓ even, and $|T_p(\ell)| = (dD)^{(l-1)/2} d$ for ℓ odd.

Let us start with level $\ell = 2R-1$, that is, the leaf nodes in $\mathcal{T}_p$. For each $v \in L_p$, there is a $k \in K'$ such that $V'_k = \{v, f(v)\}$ and $c_{kv} = c_{kf(v)} = 1$. Therefore, by (3) and the fact that $\delta(p) \geq 0$,

$$\begin{aligned} S(2R-1) &= \sum_{v \in L_p} x_v \\ &= \frac{1}{2}\delta(p) + \frac{1}{2} \sum_{v \in L_p} \left(x_v + x_{f(v)} \right) \\ &\geq \frac{d^R D^{R-1}}{2\alpha}. \end{aligned} \tag{4}$$

Next, we study the remaining odd levels $\ell = 2j-1$ for $j = 1, 2, \ldots, R-1$. Consider the set

$$F_p(2j-1) = T_p(2j-1) \cup T_p(2j).$$

Observe that the hyperedges of type II which occur in $F_p(2j-1)$ form a partition of $F_p(2j-1)$. Each of the $d^j D^{j-1}$ hyperedges in the partition has exactly one node in $T_p(2j-1)$ and exactly D nodes in $T_p(2j)$. The coefficients

c_{kv} of each beneficiary party $k \in K'$ associated with these hyperedges are $1/D$ for all $v \in V'_k$. Thus, by the approximation ratio, we obtain the bound

$$\begin{aligned} S(2j-1)+S(2j) &= \sum_{k \in K' \colon V'_k \subseteq F_p(2j-1)} D \sum_{v \in V'_k} c_{kv} x_v \\ &\geq d^j D^j / \alpha. \end{aligned} \tag{5}$$

Let us finally study the even levels $\ell = 2j$ for $j = 0, 1, 2, \ldots, R-1$. Observe that the hyperedges of type I occurring in

$$F_p(2j) = T_p(2j) \cup T_p(2j+1)$$

partition $F_p(2j)$. Each of the $d^j D^j$ hyperedges in the partition has exactly one node in $T_p(2j)$ and exactly d nodes in $T_p(2j+1)$. The coefficients a_{iv} of the resources $i \in I'$ associated with these hyperedges are 1 for all $v \in V'_i$. Thus, by the feasibility of x, we obtain the bound

$$\begin{aligned} S(2j)+S(2j+1) &= \sum_{i \in I' \colon V'_i \subseteq F_p(2j)} \sum_{v \in V'_i} a_{iv} x_v \\ &\leq d^j D^j. \end{aligned} \tag{6}$$

Put together, we have, for $j = 1, 2, \ldots, R-1$,

$$S(1) \leq S(0)+S(1) \leq 1, \tag{7}$$

$$\begin{aligned} S(2j-1) &\overset{(5)}{\geq} \frac{d^j D^j}{\alpha} - S(2j) \\ &\overset{(6)}{\geq} S(2j+1) - \left(1 - \frac{1}{\alpha}\right) d^j D^j \end{aligned} \tag{8}$$

which, together with the assumption $dD > 1$, implies

$$\begin{aligned} 1 \overset{(7)}{\geq} S(1) &\overset{(8)}{\geq} S(2R-1) - \left(1 - \frac{1}{\alpha}\right) \sum_{j=1}^{R-1} d^j D^j \\ &\overset{(4)}{\geq} \frac{d^R D^{R-1}}{2\alpha} - \left(1 - \frac{1}{\alpha}\right) \frac{d^R D^R - dD}{dD - 1}. \end{aligned}$$

Therefore

$$\alpha \geq \frac{d}{2} + 1 - \frac{1}{2D} + \frac{d + 2 - 2dD - 1/D}{2d^R D^R - 2}.$$

Should we have $\alpha < d/2 + 1 - 1/(2D)$, we would obtain a contradiction by choosing a large enough R. This concludes the proof of Theorem 1.

The same proof with $D = 1$ gives the following corollary which shows inapproximability even if both $a_{iv} \in \{0, 1\}$ and $c_{kv} \in \{0, 1\}$.

Corollary 2. *Let $\Delta^V_I > 2$ be given. There is no local approximation algorithm for* (1) *with the approximation ratio less than $\Delta^V_I / 2$. This holds even if we make the following restrictions: $a_{iv} \in \{0, 1\}$, $c_{kv} \in \{0, 1\}$, $\Delta^V_K = 2$, $\Delta^I_V = 1$ and $\Delta^K_V = 1$.*

5. Approximability

We have seen that the approximation ratio provided by the safe algorithm is within factor 2 of the best possible in general graphs; there is no local approximation scheme if $\Delta^V_I > 2$ or $\Delta^V_K > 2$.

However, the graph in our construction is very particular: it is tree-like, and the number of nodes in a radius-r neighbourhood grows exponentially as the radius r increases. Such properties are hardly realistic in practical applications such as sensor networks; if nodes are embedded in a low-dimensional physical space, the length of each communication link is bounded by the limited range of the radio, and the distribution of the nodes and the network topology are not particularly pathological, we expect that the number of nodes grows only polynomially as the radius r increases. We shall see that better approximation ratios may be achieved in such cases.

Formally, we define the relative growth of neighbourhoods by

$$\gamma(r) = \max_{v \in V} \frac{|B_{\mathcal{H}}(v, r+1)|}{|B_{\mathcal{H}}(v, r)|}.$$

We prove the following theorem.

Theorem 3. *For any R, there is a local approximation algorithm for* (1) *with the approximation ratio $\gamma(R-1)\,\gamma(R)$ and local horizon $\Theta(R)$.*

To illustrate this result, consider the case where $\mathcal{H}$ is a d-dimensional grid. In such a graph,

$$\begin{aligned} |B_{\mathcal{H}}(v, r)| &= \Theta(r^d), \\ |B_{\mathcal{H}}(v, r+1)| &= |B_{\mathcal{H}}(v, r)| + \Theta(r^{d-1}). \end{aligned}$$

Therefore $\gamma(r) = 1 + \Theta(1/r)$ and our algorithm is a local approximation scheme in this family of graphs.

We emphasise that the algorithm does not need to know any bound for $\gamma(r)$. We can use the same algorithm in any graph. The algorithm achieves a good approximation ratio if such bounds happen to exist, and it still produces a feasible solution if such bounds do not exist. Furthermore, due to the local nature of the algorithm, if the graph fails to meet such bounds in a particular area, this only affects the optimality of the beneficiary parties that are close to this area.

5.1. Algorithm

The algorithm is based on the idea of averaging local solutions of local LPs; similar ideas have been used in earlier work to derive distributed and local approximation algorithms for LPs [5, 7, 9].

Fix a radius $R = 1, 2, \ldots$; the local horizon of the algorithm will be $\Theta(R)$. For each agent $u \in V$, define the set of

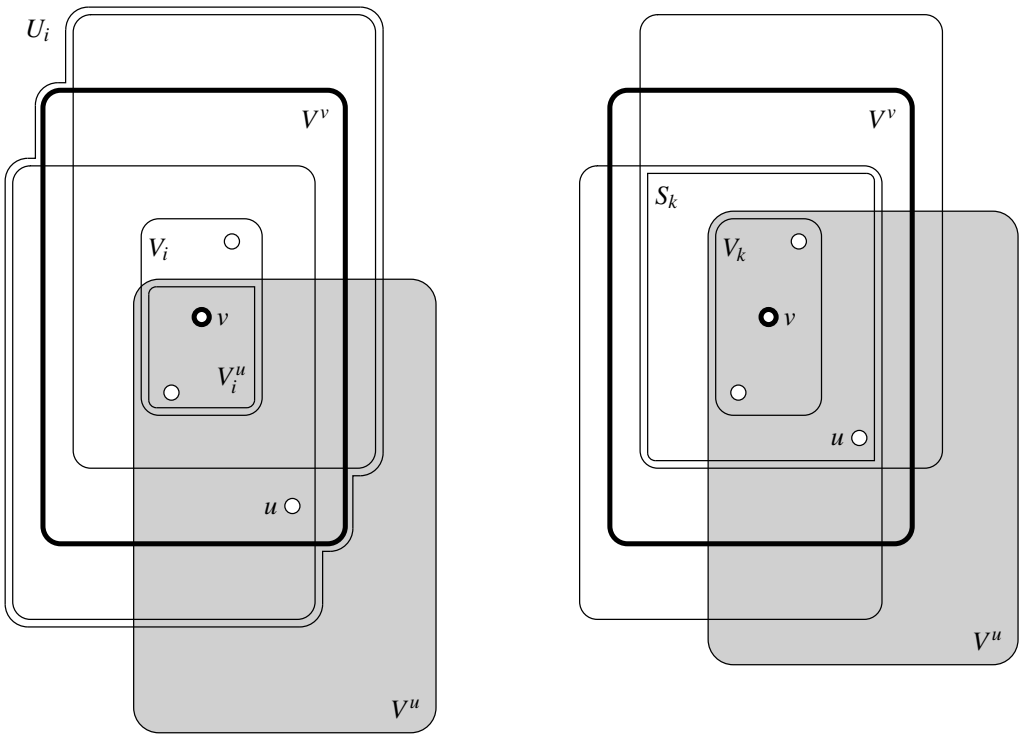

Figure 2. Definitions used in the algorithm.

the agents that are close to u:

$$V^u = B_{\mathcal{H}}(u,R),$$

the set of the agents that are close to u and consume the resource i:

$$V_i^u = V_i \cap V^u,$$

the set of the parties whose benefit is determined by V^u:

$$K^u = \{k \in K : V_k \subseteq V^u\},$$

and the set of the resources consumed by the agents in V^u:

$$I^u = \{i \in I : V_i^u \neq \emptyset\}.$$

For each $k \in K$ and $i \in I$, define the set of agents that are close to all agents who benefit the party k:

$$S_k = \bigcap_{v \in V_k} V^v,$$

and the set of the agents that are close to at least one agent who consumes the resource i:

$$U_i = \bigcup_{v \in V_i} V^v.$$

See Figure 2 for an illustration.

Finally, let

$$\begin{aligned} m_k &= |S_k|, \\ N_i &= |U_i|, \\ M_k &= \max\{|V^v| : v \in V_k\}, \\ n_i &= \min\{|V^v| : v \in V_i\}. \end{aligned}$$

For each $u \in V$, let x^u be an optimal solution of the following problem:

$$\begin{aligned} \text{maximise} \quad & \omega^u = \min_{k \in K^u} \sum_{v \in V_k} c_{kv} x_v^u \\ \text{subject to} \quad & \sum_{v \in V_i^u} a_{iv} x_v^u \leq 1 \quad \forall i \in I^u, \\ & x_v^u \geq 0 \quad \forall v \in V^u. \end{aligned} \tag{9}$$

The solution x^u can be computed by the agent u; or it can be computed separately by each agent $v \in V^u$ which needs x^u, by using the same deterministic algorithm.

The agent $v \in V$ makes the following choice, which depends only on its radius $2R+1$ neighbourhood:

$$\begin{aligned} \beta_v &= \min_{i \in I_v} \frac{n_i}{N_i}, \\ \tilde{x}_v &= \frac{\beta_v}{|V^v|} \sum_{u \in V^v} x_v^u. \end{aligned} \tag{10}$$

5.2. Constraints

Consider a resource $i \in I$. We note that

$$\begin{aligned} & v \in V_i \text{ and } u \in V^v \\ \iff & u \in U_i \text{ and } v \in V_i \text{ and } u \in V^v \\ \iff & u \in U_i \text{ and } v \in V_i \text{ and } v \in V^u \\ \iff & u \in U_i \text{ and } v \in V_i^u \end{aligned} \tag{11}$$

and

$$\begin{aligned} u \in U_i &\iff \exists v \in V_i : u \in V^v \\ &\iff \exists v \in V_i : v \in V^u \\ &\iff V_i^u \neq \emptyset \\ &\iff i \in I^u \\ &\overset{(9)}{\implies} \sum_{v \in V_i^u} a_{iv} x_v^u \leq 1. \end{aligned} \tag{12}$$

By definition, $\beta_v \leq n_i/N_i$ for all $i \in I_v$, that is, for all $v \in V_i$. Combining these observations, we obtain

$$\begin{aligned} \sum_{v \in V_i} a_{iv} \tilde{x}_v &\overset{(10)}{=} \sum_{v \in V_i} a_{iv} \frac{\beta_v}{|V^v|} \sum_{u \in V^v} x_v^u \\ &\leq \frac{1}{n_i} \frac{n_i}{N_i} \sum_{v \in V_i} \sum_{u \in V^v} a_{iv} x_v^u \\ &\overset{(11)}{=} \frac{1}{N_i} \sum_{u \in U_i} \sum_{v \in V_i^u} a_{iv} x_v^u \\ &\overset{(12)}{\leq} \frac{1}{N_i} \sum_{u \in U_i} 1 \\ &= 1. \end{aligned}$$

Therefore $\tilde{x}$ is a feasible solution of (1).

5.3. Benefit

Let x^* be an optimal solution of (1), with $\omega = \omega^*$. Then x^* is a feasible solution of (9), with $\omega^u \geq \omega^*$. Therefore the optimal solution x^u of (9) satisfies

$$\sum_{v \in V_k} c_{kv} x_v^u \geq \omega^* \tag{13}$$

for all $k \in K^u$. Let

$$\beta = \min_{v \in V} \beta_v = \min_{i \in I} \frac{n_i}{N_i}.$$

Consider a beneficiary party $k \in K$. We note that

$$u \in S_k \text{ and } v \in V_k \implies v \in V_k \text{ and } u \in V^v \tag{14}$$

and

$$\begin{aligned} u \in S_k &\iff u \in V^v \text{ for all } v \in V_k \\ &\iff v \in V^u \text{ for all } v \in V_k \\ &\iff V_k \subseteq V^u \\ &\iff k \in K^u \\ &\overset{(13)}{\implies} \sum_{v \in V_k} c_{kv} x_v^u \geq \omega^*. \end{aligned} \tag{15}$$

Combining these observations, we obtain

$$\begin{aligned} \sum_{v \in V_k} c_{kv} \tilde{x}_v &\overset{(10)}{=} \sum_{v \in V_k} c_{kv} \frac{\beta_v}{|V^v|} \sum_{u \in V^v} x_v^u \\ &\geq \frac{\beta}{M_k} \sum_{v \in V_k} \sum_{u \in V^v} c_{kv} x_v^u \\ &\overset{(14)}{\geq} \frac{\beta}{M_k} \sum_{u \in S_k} \sum_{v \in V_k} c_{kv} x_v^u \\ &\overset{(15)}{\geq} \frac{\beta}{M_k} \sum_{u \in S_k} \omega^* \\ &= \beta \frac{m_k}{M_k} \omega^*. \end{aligned}$$

In summary, the solution $\tilde{x}$ approximates (1) within the approximation ratio

$$\max_{k \in K} \frac{M_k}{m_k} \cdot \max_{i \in I} \frac{N_i}{n_i}.$$

We proceed to derive an upper bound for this ratio. Fix a $k \in K$. Let $v \in V_k$. Now $d_{\mathcal{H}}(u,v) \leq 1$ for any $u \in V_k$ and therefore $B_{\mathcal{H}}(v, R-1) \subseteq B_{\mathcal{H}}(u,R)$ for any $u \in V_k$. This implies $B_{\mathcal{H}}(v,R-1) \subseteq S_k$. Therefore $m_k \geq |B_{\mathcal{H}}(v,R-1)|$ for each $v \in V_k$. It follows that

$$\frac{M_k}{m_k} = \max_{v \in V_k} \frac{|B_{\mathcal{H}}(v,R)|}{m_k} \leq \max_{v \in V_k} \frac{|B_{\mathcal{H}}(v,R)|}{|B_{\mathcal{H}}(v,R-1)|}$$

and

$$\max_{k \in K} \frac{M_k}{m_k} \leq \max_{v \in V} \frac{|B_{\mathcal{H}}(v,R)|}{|B_{\mathcal{H}}(v,R-1)|} = \gamma(R-1).$$

Then fix an $i \in I$. Let $v \in V_i$. Now $d_{\mathcal{H}}(u,v) \leq 1$ for any $u \in V_i$ and therefore $B_{\mathcal{H}}(u,R) \subseteq B_{\mathcal{H}}(v,R+1)$ for any $u \in V_i$. This implies $U_i \subseteq B_{\mathcal{H}}(v,R+1)$. Therefore $N_i \leq |B_{\mathcal{H}}(v,R+1)|$ for each $v \in V_i$. It follows that

$$\frac{N_i}{n_i} = \max_{v \in V_i} \frac{N_i}{|B_{\mathcal{H}}(v,R)|} \leq \max_{v \in V_i} \frac{|B_{\mathcal{H}}(v,R+1)|}{|B_{\mathcal{H}}(v,R)|}$$

and

$$\max_{i \in I} \frac{N_i}{n_i} \leq \max_{v \in V} \frac{|B_{\mathcal{H}}(v,R+1)|}{|B_{\mathcal{H}}(v,R)|} = \gamma(R).$$

This completes the proof of Theorem 3.

Acknowledgments

We thank Marja Hassinen for discussions and comments. 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] B. Awerbuch and G. 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, Piscataway, NJ, USA, 1991. IEEE.

[2] Y. Bartal, J. W. Byers, and D. Raz. Global optimization using local information with applications to flow control. In *Proc. 38th Annual Symposium on Foundations of Computer Science (FOCS, Miami Beach, FL, USA, October 1997)*, pages 303–312, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[3] S. Dolev. *Self-Stabilization*. The MIT Press, Cambridge, MA, USA, 2000.

[4] P. Floréen, P. Kaski, T. Musto, and J. Suomela. Local approximation algorithms for scheduling problems in sensor networks. In *Proc. 3rd International Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors, Wrocław, Poland, July 2007)*, volume 4837 of *Lecture Notes in Computer Science*, pages 99–113, Berlin, Germany, 2008. Springer-Verlag. To appear.

[5] F. Kuhn and T. Moscibroda. Distributed approximation of capacitated dominating sets. In *Proc. 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA, San Diego, CA, USA, June 2007)*, pages 161–170, New York, NY, USA, 2007. ACM Press.

[6] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC, St. John's, Newfoundland, Canada, July 2004)*, pages 300–309, New York, NY, USA, 2004. ACM Press.

[7] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the locality of bounded growth. In *Proc. 24th Annual ACM Symposium on Principles of Distributed Computing (PODC, Las Vegas, NV, USA, July 2005)*, pages 60–68, New York, NY, USA, 2005. ACM Press.

[8] F. Kuhn, T. Moscibroda, and R. 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)*, Los Alamitos, CA, USA, 2006. IEEE Computer Society Press.

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

[10] F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. *Distributed Computing*, 17(4):303–310, 2005.

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

[12] B. D. McKay, N. C. Wormald, and B. Wysocka. Short cycles in random regular graphs. *Electronic Journal of Combinatorics*, 11(1):#R66, 2004.

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

[14] C. H. Papadimitriou and M. Yannakakis. On the value of information in distributed decision-making. In *Proc. 10th Annual ACM Symposium on Principles of Distributed Computing (PODC, Montreal, Quebec, Canada, August 1991)*, pages 61–64, New York, NY, USA, 1991. ACM Press.

[15] C. H. Papadimitriou and M. 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, New York, NY, USA, 1993. ACM Press.

[16] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1–3):183–196, 2007.

[17] J. Urrutia. Local solutions for global problems in wireless networks. *Journal of Discrete Algorithms*, 5(3):395–407, 2007.

[18] J. S. Vitter. External memory algorithms and data structures: dealing with massive data. *ACM Computing Surveys*, 33(2):209–271, 2001.