No Sublogarithmic-time Approximation Scheme for Bipartite Vertex Cover

Mika Göös and Jukka Suomela

Helsinki Institute for Information Technology HIIT,
Department of Computer Science, University of Helsinki

Abstract

König's theorem states that on bipartite graphs the size of a maximum matching equals the size of a minimum vertex cover. It is known from prior work that for every $\epsilon > 0$ there exists a *constant-time* distributed algorithm that finds a $(1 + \epsilon)$-approximation of a maximum matching on 2-coloured graphs of bounded degree. In this work, we show—somewhat surprisingly—that no *sublogarithmic-time* approximation scheme exists for the dual problem: there is a constant $\delta > 0$ so that no randomised distributed algorithm with running time $o(\log n)$ can find a $(1 + \delta)$-approximation of a minimum vertex cover on 2-coloured graphs of maximum degree 3. In fact, a simple application of the Linial–Saks (1993) decomposition demonstrates that this lower bound is tight.

Our lower-bound construction is simple and, to some extent, independent of previous techniques. Along the way we prove that a certain cut minimisation problem, which might be of independent interest, is hard to approximate locally on expander graphs.

1 Introduction

Many graph optimisation problems do not admit an exact solution by a fast distributed algorithm. This is true not only for most NP-hard optimisation problems, but also for problems that can be solved using sequential polynomial-time algorithms. This work is a contribution to the *distributed approximability* of such a problem: the minimum vertex cover problem on bipartite graphs—we call it 2-VC, for short.

Our focus is on *negative* results: We prove an optimal (up to constants) time lower bound $\Omega(\log n)$ for a randomised distributed algorithm to find a close-to-optimal vertex cover on bipartite 2-coloured graphs of maximum degree $\Delta = 3$. In particular, this rules out the existence of a sublogarithmic-time approximation scheme for 2-VC on sparse graphs.

Our lower bound result exhibits the following features:

– The proof is relatively simple as compared to the strength of the result; this is achieved through an application of *expander graphs* in the lower-bound construction.

– To explain the source of hardness for 2-VC we introduce a certain *distributed cut minimisation problem*, which might have applications elsewhere.

– Many previous distributed inapproximability results are based on the hardness of *local symmetry breaking*. This is not the case here: the difficulty we pinpoint for 2-VC is in the semi-global task of *gluing together* two different types of local solutions.

– Our result states that *König's theorem is non-local*—see Section 1.3.

1.1 The $\mathcal{LOCAL}$ model

We work in the standard $\mathcal{LOCAL}$ model of distributed computing [10, 17]. As input we are given an undirected graph $G = (V, E)$. We interpret G as defining a communication network: the nodes V host processors, and two processors can communicate directly if they are connected by an edge. All nodes run the same distributed algorithm $\mathcal{A}$. The computation of $\mathcal{A}$ on G starts out with every node $v \in V$ knowing an upper bound on $n = |V|$ and possessing a globally unique $O(\log n)$-bit identifier $\mathrm{ID}(v)$; for simplicity, we assume that $V \subseteq \{1, 2, \ldots, \mathrm{poly}(n)\}$ and $\mathrm{ID}(v) = v$. Also, we assume that the processors have access to independent (and unlimited) sources of randomness. The computation proceeds in synchronous communication rounds. In each round, all nodes first perform some local computations and then exchange (unbounded) messages with their neighbours. After some r communication rounds the nodes stop and produce local outputs. Here r is the *running time* of $\mathcal{A}$ and the output of v is denoted $\mathcal{A}(G, v)$.

The fundamental limitation of a distributed algorithm with running time r is that the output $\mathcal{A}(G, v)$ can only depend on the information available in the subgraph $G[v, r] \subseteq G$ induced on the vertices in the radius-r neighbourhood ball

$$B_G(v, r) = \{u \in V : \mathrm{dist}_G(v, u) \leq r\}.$$

Conversely, it is well known that an algorithm $\mathcal{A}$ can essentially discover the structure of $G[v, r]$ in time r. Thus, $\mathcal{A}$ can be thought of as a function mapping r-neighbourhoods $G[v, r]$ (together with the additional input labels and random bits on $B_G(v, r)$) to outputs.

While the $\mathcal{LOCAL}$ model abstracts away issues of network congestion and asynchrony, this only makes our *lower-bound* result stronger.

1.2 Our result

Below, we concentrate on bipartite 2-coloured graphs G. That is, G is not only bipartite (which is a global property), but every node v is informed of the bipartition by an additional input label $c(v)$, where $c \colon V \to \{\mathsf{white}, \mathsf{black}\}$ is a proper 2-colouring of G.

Definition 1. In the 2-VC problem we are given a 2-coloured graph $G = (G, c)$ and the objective is to output a minimum-size vertex cover of G.

A distributed algorithm $\mathcal{A}$ computes a vertex cover by outputting a single bit $\mathcal{A}(G, v) \in \{0, 1\}$ on a node v indicating whether v is included in the solution. This way, $\mathcal{A}$ computes the set $\mathcal{A}(G) := \{v \in V : \mathcal{A}(G, v) = 1\}$. Moreover, we say that $\mathcal{A}$ computes an α-*approximation* of 2-VC if $\mathcal{A}(G)$ is a vertex cover of G and

$$|\mathcal{A}(G)| \leq \alpha \cdot \mathsf{OPT}_G,$$

where OPT_G denotes the size of a minimum vertex cover of G.

Our main result is the following.

Theorem 1. *There exists a $\delta > 0$ such that no randomised distributed algorithm with running time $o(\log n)$ can find a $(1 + \delta)$-approximation of 2-VC on graphs of maximum degree $\Delta = 3$.*

A matching time upper bound follows directly from the well-known network decomposition algorithm due to Linial and Saks [11].

Theorem 2. *For every $\epsilon > 0$ a $(1 + \epsilon)$-approximation of 2-VC can be computed with high probability in time $O(\epsilon^{-1} \log n)$ on graphs of maximum degree $\Delta = O(1)$.*

Proof. The subroutine *Construct_Block* in the algorithm of Linial and Saks [11] computes, in time $r = O(\epsilon^{-1} \log n)$, a set $S \subseteq V$ with the following properties. Each component in the subgraph $G[S]$ induced by S has *weak diameter* at most r, i.e., $\mathrm{dist}_G(u, v) \leq r$ for each pair $u, v \in S$ belonging to the same component of $G[S]$. Moreover, they prove that, w.h.p.,

$$|S| \geq (1 - \epsilon)n.$$

Let C be a component of $G[S]$. Every node of C can discover the structure of C in time $O(r)$ by exploiting its weak diameter. Thus, every node of C can internally compute *the same* optimal solution of 2-VC on C. We can then output as a vertex cover for G the union of the optimal solutions at the components together with the vertices $V \smallsetminus S$. This results in a solution of size at most

$$\mathsf{OPT}_{G[S]} + \epsilon n.$$

But since $\mathsf{OPT}_G \geq |E|/\Delta = \Omega(n)$ for connected G, this is a $(1 + O(\epsilon))$-approximation of 2-VC. $\square$

1.3 König duality

The classic theorem of König (see, e.g., Diestel [3, §2.1]) states that, on bipartite graphs, the size of a maximum matching equals the size of a minimum vertex cover. A modern perspective is to view this result through the lens of linear programming (LP) duality. The LP relaxations of these problems are *the fractional matching problem* (primal) and *the fractional vertex cover problem* (dual):

$$
\begin{array}{ll}
\text{maximise} & \sum_{e \in E} x_e \\
\text{subject to} & \sum_{e:\, v \in e} x_e \leq 1, \quad \forall v \in V \\
& \mathbf{x} \geq \mathbf{0}
\end{array}
\qquad\qquad
\begin{array}{ll}
\text{minimise} & \sum_{v \in V} y_v \\
\text{subject to} & \sum_{v:\, v \in e} y_v \geq 1, \quad \forall e \in E \\
& \mathbf{y} \geq \mathbf{0}
\end{array}
$$

In particular, on bipartite graphs, the above LPs do not have an integrality gap (see, e.g., Papadimitriou and Steiglitz [15, §13.2]): among the optimal feasible solutions are integral vectors $\mathbf{x} \in \{0, 1\}^E$ and $\mathbf{y} \in \{0, 1\}^V$ that correspond to maximum matchings and minimum vertex covers.

In the context of distributed algorithms, the following is known on (bipartite) *bounded degree graphs*:

(1) *Primal LP and dual LP admit local approximation schemes.* As part of their general result, Kuhn et al. [7] give a strictly local $(1 + \epsilon)$-approximation scheme for the above LPs. Their algorithms run in constant time independent of the number of nodes.

(2) *Integral primal admits a local approximation scheme.* Åstrand et al. [1] describe a strictly local $(1 + \epsilon)$-approximation scheme for the maximum matching problem on 2-coloured graphs. Again, the running time is a constant independent of the number of nodes.

(3) *Integral dual does not admit a local approximation scheme.* The present work shows—in contrast to the above positive results—that there is no local approximation scheme for 2-VC even when $\Delta = 3$.

1.4 Related lower bounds

There are relatively few independent methods for obtaining negative results for distributed approximation in the $\mathcal{LOCAL}$ model. We list three main sources.

Local algorithms. Linial's [10] lower bound $\Omega(\log^* n)$ for 3-colouring a cycle together with the Ramsey technique of Naor and Stockmeyer [13] establish basic limitations on finding *exact solutions* strictly locally in constant time. These impossibility results were later extended to finding *approximate solutions* on cycle-like graphs by Lenzen and Wattenhofer [9] and Czygrinow et al. [2]. A recent work [4] generalises these techniques even further and proves that deterministic local algorithms in the $\mathcal{LOCAL}$ model are often no more powerful than algorithms running on anonymous port numbered networks. For more information on this line of research, see the survey of local algorithms [18].

Here, the inapproximability results typically exploit the inability of a local algorithm to break local symmetries. By contrast, in this work, we consider the case where the local symmetry is already broken by a 2-colouring.

KMW-bounds. Kuhn, Moscibroda and Wattenhofer [6–8] prove that any randomised algorithm for computing a constant-factor approximation of minimum vertex cover on general graphs requires time $\Omega(\sqrt{\log n})$ and $\Omega(\log \Delta)$. Roughly speaking, their technique consists of showing that a fast algorithm cannot tell apart two adjacent nodes v and u, even though it is globally more profitable to include v in the vertex cover and exclude u than conversely.

The lower-bound graphs of Kuhn et al. are necessarily of unbounded degree: on bounded degree graphs the set of all non-isolated nodes is a constant factor approximation of a minimum vertex cover. By contrast, our lower-bound graphs are of bounded degree and they forbid close-to-optimal approximation of 2-VC.

Sublinear-time centralised algorithms. Parnas and Ron [16] discuss how a fast distributed algorithm can be used as *solution oracle* to a centralised algorithm that approximates parameters of a sparse graph G in sublinear time given a randomised query access to G. Thus, lower bounds in this model of computation also imply lower bounds for distributed algorithms. In particular, an argument of Trevisan (presented in [16]) implies that computing a $(2 - \epsilon)$-approximation of a minimum vertex cover requires $\Omega(\log n)$ time on d-regular graphs, where $d = d(\epsilon)$ is sufficiently large.

We note that 2-VC is easy to approximate in this model: Nguyen and Onak [14] give a centralised constant-time algorithm to approximate the size of a maximum matching on a graph G. If we are promised that G is bipartite, the same algorithm approximates the size of 2-VC by König duality.

2 Deterministic lower bound

To best explain the basic idea of our lower bound result, we first prove Theorem 1 for a toy model that we define in Section 2.1; in this model, we only consider a certain class of deterministic distributed algorithms in anonymous networks. Later in Section 3 we will show how to implement the same proof technique in a much more general setting: randomised distributed algorithms in networks with unique identifiers.

In the present section, we find a source of hardness for 2-VC as follows. First, we argue that any approximation algorithm for the 2-VC problem also solves a certain cut minimisation problem called RECUT. We then show that RECUT is hard to approximate locally, which implies that 2-VC must also be hard to approximate locally.

2.1 Toy model of deterministic algorithms

Throughout this section we consider deterministic algorithms $\mathcal{A}$ running in time $r = o(\log n)$ that operate on *input-labelled anonymous networks* (G, ℓ), where $G = (V, E)$ and ℓ is a labelling of V. More precisely, we impose the following additional restrictions in the $\mathcal{LOCAL}$ model:

- The nodes of G are not given random bits as input.

- The output of $\mathcal{A}$ is invariant under reassigning node identifiers. That is, if $G' = (V', E')$ is isomorphic to G via a mapping $f \colon V' \to V$, then for $v \in V$,

$$\mathcal{A}(G, \ell, v) = \mathcal{A}(G', \ell \circ f, f^{-1}(v)).$$

Put otherwise, the only symmetry breaking information we supply $\mathcal{A}$ is the radius-r neighbourhood topology together with the input labelling—the nodes are anonymous and do not have unique identifiers.

We will also consider graphs G that are *directed*. In this case, the directions of the edges are merely additional symmetry-breaking information; they do not restrict communication.

2.2 The Recut problem

In the following, we consider partitions of V into red and blue colour classes as determined by a labelling $\ell \colon V \to \{\mathsf{red}, \mathsf{blue}\}$. We write $\partial \ell$ for the fraction of edges crossing the red/blue cut, i.e.,

$$\partial \ell := \frac{e(\ell^{-1}(\mathsf{red}), \ell^{-1}(\mathsf{blue}))}{|E|}.$$

Definition 2. In *the* RECUT *problem* we are given a labelled graph (G, ℓ) as input and the objective is to compute an output labelling (*a recut*) ℓ_{out} that minimises $\partial \ell_{\mathsf{out}}$ subject to the following constraints: (a) If $\ell(V) = \{\mathsf{red}\}$, then $\ell_{\mathsf{out}}(V) = \{\mathsf{red}\}$. (b) If $\ell(V) = \{\mathsf{blue}\}$, then $\ell_{\mathsf{out}}(V) = \{\mathsf{blue}\}$.

In words, if we have an all-red input, we have to produce an all-red output, and if we have an all-blue input, we have to produce an all-blue output. Otherwise the output can be arbitrary. See Figure 1 for an illustration.

Needless to say, the global optimum for an algorithm $\mathcal{A}$ would be to produce a constant output labelling $\ell_{\mathcal{A}}$ (either all red or all blue) having $\partial \ell_{\mathcal{A}} = 0$. However, a distributed algorithm $\mathcal{A}$ can only
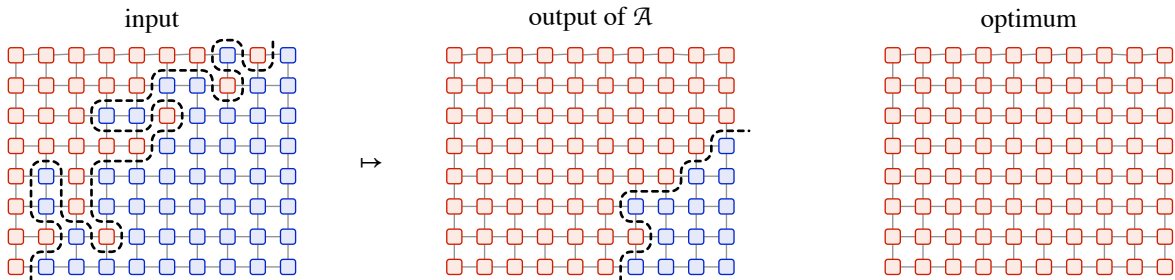

Figure 1: The RECUT problem. In this example, we have used a simple distributed algorithm $\mathcal{A}$ to find a recut ℓ_{out} with a small boundary $\partial \ell_{\mathsf{out}}$: a node outputs red iff there is a red node within distance $r = 3$ in the input. While the solution is not optimal, in a grid graph the boundary will be relatively small. However, our lower bound shows that any fast distributed algorithm—including algorithm $\mathcal{A}$—fails to produce a small boundary in some graph.

5

access the values of the input labelling ℓ in its local radius-r neighbourhood: when encountering a neighbourhood $v \in U \subseteq V$ with $\ell(U) = \{\mathsf{red}\}$, the algorithm is forced to output red at v to guarantee satisfying the global constraint (a), and when encountering a neighbourhood $v \in U \subseteq V$ with $\ell(U) = \{\mathsf{blue}\}$, the algorithm is forced to output blue at v to satisfy (b). Thus, if a connected graph G has two disjoint r-neighbourhoods $U, U' \subseteq V$ with $\ell(U) = \{\mathsf{red}\}$ and $\ell(U') = \{\mathsf{blue}\}$ the algorithm $\mathcal{A}$ cannot avoid producing at least some red/blue edge boundary. Indeed, the best we can hope $\mathcal{A}$ to achieve is a recut $\ell_{\mathcal{A}}$ of size $\partial \ell_{\mathcal{A}} \le \epsilon$ for some small constant $\epsilon > 0$.

Discussion. The RECUT problem models the following abstract high-level challenge in designing distributed algorithms: Each node in a local neighbourhood $U \subseteq V$ can, in principle, internally compute a completely *locally optimal* solution for (the subgraph induced by) U, but difficulties arise when deciding which of these proposed solution are to be used in the final distributed output. In particular, when the *type* of the produced solution changes from one (e.g., red) to another (e.g., blue) across a graph G one might have to introduce suboptimalities to the solution at the (red/blue) boundary in order to glue together the different types of local solutions.

In fact, the RECUT problem captures the first non-trivial case of this phenomenon with only *two* solution types present. One can think of the input labelling ℓ as recording the *initial preferences* of the nodes whereas the output labelling $\ell_{\mathcal{A}}$ records how an algorithm $\mathcal{A}$ decides to combine these preferences into the final unified output. In the end, our lower-bound strategy will be to argue that any $\mathcal{A}$ can be forced into producing too large an edge boundary $\partial \ell_{\mathcal{A}}$ resulting in too many suboptimalities in the produced output.

Next, we show how the above discussion is made concrete in the case of the 2-VC problem.

2.3 The reduction

We call a graph G *tree-like* if all the r-neighbourhoods in G are trees, i.e., G has girth larger than $2r + 1$. Furthermore, if G is directed, we say it is *balanced* if in-degree$(v) =$ out-degree(v) for all vertices v. We note that a deterministic algorithm $\mathcal{A}$ produces the same output on every node of a balanced regular tree-like digraph G, because such a graph is *locally homogeneous*: all the r-neighbourhoods of G are pairwise isomorphic.

Using this terminology we give the following reduction.

Theorem 3. *Suppose $\mathcal{A}$ computes a $(1 + \epsilon)$-approximation of 2-VC on graphs of maximum degree $\Delta = 3$. Then, there is an algorithm (with running time r) that finds a recut $\ell_{\mathcal{A}}$ of size $\partial \ell_{\mathcal{A}} = O(\epsilon)$ on balanced 4-regular tree-like digraphs.*

The proof of Theorem 3 follows the usual route: We give a local reduction (i.e., one that can be computed by a local algorithm) that transforms an instance (G, ℓ) of RECUT into an instance $\Pi(G, \ell)$ of 2-VC. Then we simulate $\mathcal{A}$ on the resulting instance and map the output of $\mathcal{A}$ back to a solution of the RECUT instance (G, ℓ).

Let $G = (V, E)$ be a balanced 4-regular tree-like digraph and let $\ell \colon V \to \{\mathsf{red}, \mathsf{blue}\}$ be a labelling of G. The instance $\Pi(G, \ell)$ is obtained by replacing each vertex $v \in V$ by one of two local gadgets depending on the label $\ell(v)$. We first describe and analyse simple gadgets yielding instances of 2-VC with $\Delta = 4$; the gadgets yielding instances with $\Delta = 3$ are described later.

Red gadget. The red gadget replaces a vertex $v \in V$ by two new vertices w_v (white) and b_v (black) that share a new edge $e_v := \{w_v, b_v\}$. The incoming edges of v are reconnected to w_v, whereas the outgoing edges of v are reconnected to b_v.

Note that the 2-VC instance $\Pi(G, \mathsf{red})$ (where we denote by red the constant labelling $v \mapsto \mathsf{red}$) contains $\{e_v : v \in V\}$ as a perfect matching. Since (G, red) is locally homogeneous, the solutions output by $\mathcal{A}$ on the endpoints of e_v are isomorphic across all v. Assuming $\epsilon < 1$ it follows that the algorithm $\mathcal{A}$ must output either the set of all white nodes or the set of all black nodes on $\Pi(G, \mathsf{red})$. Our reduction branches at this point: we choose the structure of the blue gadget to counteract this white/black decision made by $\mathcal{A}$ on the red gadgets. We describe the case that $\mathcal{A}$ outputs all white nodes on $\Pi(G, \mathsf{red})$; the case of black nodes is symmetric.


red input   blue input

Blue gadget. The blue gadget replacing $v \in V$ is identical to the red gadget with the exception that a third new vertex w'_v (white) is added and connected to b_v.

Similarly as above, we can argue that $\mathcal{A}$ outputs exactly the set of all black nodes on the instance $\Pi(G, \mathsf{blue})$. This completes the description of Π.

Next, we simulate $\mathcal{A}$ on $\Pi(G, \ell)$. The output of $\mathcal{A}$ is then transformed back to a labelling $\ell_{\mathcal{A}} : V \to \{\mathsf{red}, \mathsf{blue}\}$ by setting

$\ell_{\mathcal{A}}(v) = \mathsf{blue} \iff$ the output of $\mathcal{A}$ contains only the black node b_v at the gadget at v.

Note that $\ell_{\mathcal{A}}$ satisfies both feasibility constraints (a) and (b) of RECUT. It remains to bound the size $\partial \ell_{\mathcal{A}}$ of this recut.


red output

blue output
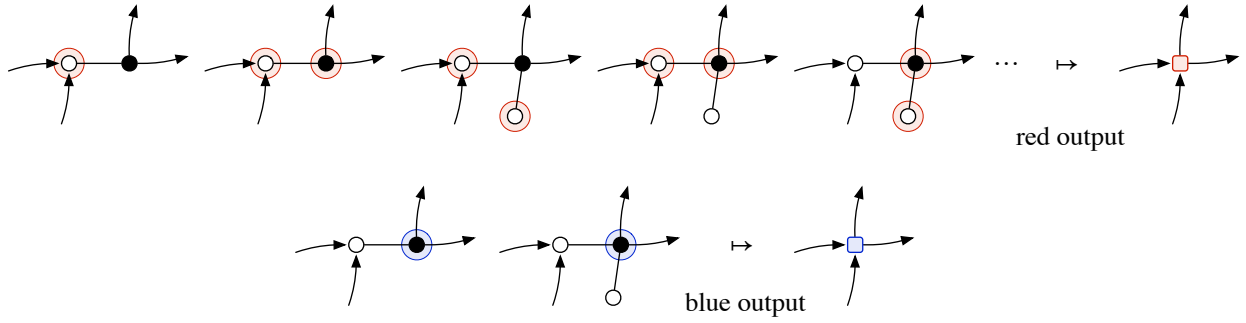
Recut analysis. Call a red vertex v in $(G, \ell_{\mathcal{A}})$ *bad* if v has a blue out-neighbour u. By the definition of "$\ell_{\mathcal{A}}(u) = \mathsf{blue}$", the vertex cover produced by algorithm $\mathcal{A}$ does not contain the white node w_u. Thus to cover the edge (b_v, w_u), the vertex cover has to contain the black node b_v. But by the definition of "$\ell_{\mathcal{A}}(v) = \mathsf{red}$", we must have w_v or w'_v in the solution as well. Hence, at least two nodes are used to cover the gadget at v, which is suboptimal as compared to the minimum vertex cover $\{b_v : v \in V\}$, which uses only one node per gadget. This implies that we must have at most $\epsilon |V|$ bad vertices as $\mathcal{A}$ produces a $(1 + \epsilon)$-approximation of 2-VC on $\Pi(G, \ell)$.

v is bad

On the other hand, exactly half of the edges crossing the cut $\ell_\mathcal{A}$ are oriented from red to blue since G is balanced. Each bad vertex gives rise to at most two of these edges, so we have that $\partial\ell_\mathcal{A} \cdot |E|/2 \leq 2\epsilon|V|$ which gives $\partial\ell_\mathcal{A} \leq 2\epsilon$, as required. This proves Theorem 3 for $\Delta = 4$.

Gadgets for $\Delta = 3$. The maximum degree used in the gadgets can be reduced to 3 by the following modification. The red gadget replaces a vertex $v \in V$ by a path of length 3.

red input   blue input
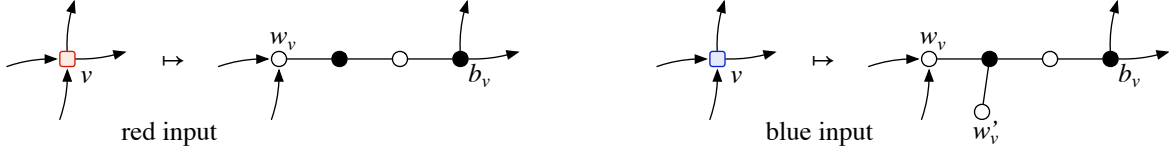
Again, to achieve a 1.499-approximation of 2-VC on $\Pi(G, \mathsf{red})$ the algorithm $\mathcal{A}$ has to make a choice: either leave out the middle black vertex or the middle white vertex from the vertex cover. Supposing $\mathcal{A}$ leaves out the middle black, the blue gadget is defined to be identical to the red gadget with an additional white vertex connected to the middle black one.

After simulating $\mathcal{A}$ on an instance $\Pi(G, \ell)$ we define $\ell_\mathcal{A}(v) = \mathsf{blue}$ iff $\mathcal{A}$ outputs only black nodes at the gadget at v. The recut analysis will then give $\partial\ell_\mathcal{A} \leq 4\epsilon$.

2.4 Recut is hard on expanders

Intuitively, the difficulty in computing a small red/blue cut in the RECUT problem stems from the inability of an algorithm $\mathcal{A}$ to overcome the neighbourhood expansion of an input graph in $r = o(\log n)$ steps—an algorithm cannot hide the red/blue boundary as the radius-r neighbourhoods themselves might have large boundaries.

To formalise this intuition, we use as a basis for our lower-bound construction an infinite family $\mathcal{F}$ of 4-regular δ-*expander graphs*, where each $G = (V, E) \in \mathcal{F}$ satisfies the edge expansion condition

$$e(S, V \smallsetminus S) \geq \delta \cdot |S| \qquad \text{for all } S \subseteq V, \ |S| \leq n/2. \tag{1}$$

Here, $e(S, V \smallsetminus S)$ is the number of edges leaving S and $\delta > 0$ is an absolute constant independent of $n = |V|$. On such graphs it is enough for us to force an algorithm $\mathcal{A}$ to output a *nearly balanced recut* $\ell_\mathcal{A}$ having both colour classes close to $n/2$ in size. This is because if the number of, say, the red nodes is

$$|\ell_\mathcal{A}^{-1}(\mathsf{red})| = n/2 - o(n),$$

then the expansion property (1) implies that

$$\partial\ell_\mathcal{A} \geq \delta/4 - o(1).$$

That is, $\mathcal{A}$ computes a recut of size $\Omega(\delta)$.

Indeed, the following simple fooling trick makes up the very core of our argument.

Lemma 4. *Suppose $\mathcal{A}$ produces a feasible solution for the RECUT problem in time $r = o(\log n)$. Then for each 4-regular graph G there exists an input labelling for which $\mathcal{A}$ computes a nearly balanced recut.*

Proof. Fix an arbitrary ordering $v_1, v_2, \ldots, v_n$ for the vertices of G and define a sequence of labellings $\ell^0, \ell^1, \ldots, \ell^n$ by setting $\ell^i(v_j) = \mathsf{blue}$ iff $j \leq i$. That is, in ℓ^0 all nodes are red, in ℓ^n all nodes are blue, and ℓ^i is obtained from ℓ^{i-1} by changing the colour of v_i from red to blue.

When we switch from the instance (G, ℓ^{i-1}) to (G, ℓ^i) the change of v_i's colour is only registered by nodes in the radius-r neighbourhood of v_i. This neighbourhood has size $|B_G(v_i, r)| \leq 5^r = o(n)$, and so the number of red nodes in the outputs $\ell_\mathcal{A}^{i-1}$ and $\ell_\mathcal{A}^i$ of $\mathcal{A}$ can only differ by $o(n)$. As, by assumption, we have that $\mathcal{A}$ computes the labelling $\ell_\mathcal{A}^0 = \mathsf{red}$ on (G, ℓ^0) and the labelling $\ell_\mathcal{A}^n = \mathsf{blue}$ on (G, ℓ^n), it follows by continuity that some labelling in our sequence must force $\mathcal{A}$ to output $n/2 - o(n)$ red nodes. $\qquad\square$

We now have all the ingredients for the lower-bound proof: We can take $\delta = 2 - \sqrt{3}$ if we choose $\mathcal{F}$ to be the family of 4-regular Ramanujan graphs due to Morgenstern [12]. These graphs are tree-like, as they have girth $\Theta(\log n)$. They can be made into balanced digraphs since a suitable orientation can always be derived from an Euler tour. Thus, $\mathcal{F}$ consists of balanced 4-regular tree-like digraphs. Lemma 4 together with the discussion above imply that every algorithm for RECUT produces a recut of size $\Omega(\delta)$ on some labelled graph in $\mathcal{F}$. Hence, the contrapositive of Theorem 3 proves Theorem 1 for our deterministic toy algorithms.

3 Randomised lower bound

Even though our model of deterministic algorithms in Section 2 is an unusually weak one, we can quickly recover the standard $\mathcal{LOCAL}$ model from it by equipping the nodes with independent sources of randomness. In particular, as is well known, each node can choose an identifier uniformly at random from, e.g., the set $\{1, 2, \ldots, n^3\}$, and this results in the identifiers being globally unique with probability at least $1 - 1/n$.

When discussing randomised algorithms many of the simplifying assumptions made in Section 2 no longer apply. For example, a randomised algorithm need not produce the same output on every node of a locally homogeneous graph. Consequently, the homogeneous feasibility constraints in the RECUT problem do not strictly make sense for randomised algorithms.

However, we can still emulate the same proof strategy as in Section 2: we force the randomised algorithm to output a nearly balanced recut with high probability. Below, we describe this strategy in case of the easy-to-analyse "$\Delta = 4$" gadgets with the understanding that the same analysis can be repeated for the "$\Delta = 3$" gadgets with little difficulty.

3.1 Repeating Section 2 for randomised algorithms

Fix a randomised algorithm $\mathcal{A}$ with running time $r = o(\log n)$ and let $G = (V, E)$, $n = |V|$, be a large 4-regular expander.

Again, we start out with the all-red instance. We denote by W and B the number of black and white nodes output by $\mathcal{A}$ on $\Pi(G, \mathsf{red})$. As each of the edges e_v must be covered, we have that

$$W + B \geq n.$$

Hence, by linearity of expectation, at least one of $\mathbf{E}[W] \geq n/2$ or $\mathbf{E}[B] \geq n/2$ holds. We assume that $\mathbf{E}[W] \geq n/2$; the other case is symmetric.

In reaction to $\mathcal{A}$ preferring white nodes, the blue gadgets are now defined exactly as in Section 2 with an additional white vertex. Furthermore, for any input $\ell \colon V \to \{\mathsf{red}, \mathsf{blue}\}$ we interpret the output of $\mathcal{A}$ on $\Pi(G, \ell)$ as defining an output labelling $\ell_\mathcal{A}$ of V, where, again, $\ell_\mathcal{A}(v) = \mathsf{blue}$ iff $\mathcal{A}$ outputs only the black node at the gadget at v. This definition translates our assumption of $\mathbf{E}[W] \geq n/2$ into

$$\mathbf{E}[R(\mathsf{red})] \geq n/2, \tag{2}$$

where $R(\ell) := |\ell_{\mathcal{A}}^{-1}(\mathsf{red})|$ counts the number of gadgets (i.e., vertices of G) relabelled red by $\mathcal{A}$ on $\Pi(G, \ell)$.

If $\mathcal{A}$ relabels a blue gadget red, it must output at least two nodes at the gadget. This means that the size of the solution output by $\mathcal{A}$ on $\Pi(G, \mathsf{blue})$ is at least $n + R(\mathsf{blue})$. Thus, if $\mathcal{A}$ is to produce a $3/2$-approximation on $\Pi(G, \mathsf{blue})$ in expectation, we must have that

$$\mathbf{E}[R(\mathsf{blue})] \leq n/2. \tag{3}$$

The inequalities (2) and (3) provide the necessary boundary conditions (replacing the feasibility constraints of RECUT) for the argument of Lemma 4: by continuously changing the instance (G, red) into (G, blue) we may find an input labelling ℓ^* achieving

$$\mathbf{E}[R(\ell^*)] = n/2 - o(n). \tag{4}$$

It remains to argue that $\mathcal{A}$ outputs a nearly balanced recut not only "in expectation" but also with high probability.

3.2 Local concentration bound

Focusing on the instance $\Pi(G, \ell^*)$ we write $R = R(\ell^*)$ and

$$R = \sum_{v \in V} X_v, \tag{5}$$

where $X_v \in \{0, 1\}$ indicates whether $\mathcal{A}$ relabels the gadget at v red.

The variables X_v are not *too* dependent: the $2r$th power of G, denoted G^{2r}, where $u, v \in V$ are joined by an edge iff $B_G(v, r) \cap B_G(u, r) \neq \varnothing$, is *a dependency graph* for the variables X_v. Every independent set $I \subseteq V$ in G^{2r} corresponds to a set $\{X_v\}_{v \in I}$ of mutually independent random variables. Since the maximum degree of G^{2r} is at most $\max_v |B_G(v, 2r)| = o(n)$, this graph can always be partitioned into $\chi(G^{2r}) = o(n)$ independent sets.

Indeed, Janson [5] presents large deviation bounds for sums of type (5) by applying Chernoff–Hoeffding bounds for each colour class in a $\chi(G^{2r})$-colouring of G^{2r}. For any $\epsilon > 0$, Theorem 2.1 in Janson [5], as applied to our setting, gives

$$\mathbf{Pr}(R \geq \mathbf{E}[R] + \epsilon n) \leq \exp\left(-2\frac{(\epsilon n)^2}{\chi(G^{2r}) \cdot n}\right) \to 0, \quad \text{as } n \to \infty, \tag{6}$$

and the same bound holds for $\mathbf{Pr}(R \leq \mathbf{E}[R] - \epsilon n)$. That is, R is concentrated around its expectation.

In conclusion, the combination of (4) and (6) implies that, for large n, the algorithm $\mathcal{A}$ outputs a nearly balanced recut on $\Pi(G, \ell^*)$ with high probability. By the discussion in Section 2, this proves Theorem 1.

4 Acknowledgements

References

[1] Matti Åstrand, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. Local algorithms in (weakly) coloured graphs, 2010. Manuscript, arXiv:1002.0125 [cs.DC].

[2] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Proc. 22nd Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *LNCS*, pages 78–92. Springer, Berlin, 2008.

[3] Reinhard Diestel. *Graph Theory*. Springer, Berlin, 3rd edition, 2005.

[4] Mika Göös, Juho Hirvonen, and Jukka Suomela. Lower bounds for local approximation. In *Proc. 31st Symposium on Principles of Distributed Computing (PODC 2012)*. ACM Press, New York, 2012. To appear.

[5] Svante Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24(3):234–248, 2004.

[6] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proc. 23rd Symposium on Principles of Distributed Computing (PODC 2004)*, pages 300–309. ACM Press, New York, 2004.

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

[8] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds, 2010. Manuscript, arXiv:1011.5470 [cs.DC].

[9] Christoph Lenzen and Roger Wattenhofer. Leveraging Linial's locality limit. In *Proc. 22nd Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *LNCS*, pages 394–407. Springer, Berlin, 2008.

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

[11] Nathan Linial and Michael Saks. Low diameter graph decompositions. *Combinatorica*, 13:441–454, 1993.

[12] Moshe Morgenstern. Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q. *Journal of Combinatorial Theory, Series B*, 62(1):44–62, 1994.

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

[14] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Proc. 49th Symposium on Foundations of Computer Science (FOCS 2008)*, pages 327–336. IEEE Computer Society Press, Los Alamitos, 2008.

[15] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., Mineola, NY, USA, 1998.

[16] Michal Parnas and Dana 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] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2000.

[18] Jukka Suomela. Survey of local algorithms. *ACM Computing Surveys*, 2011. To appear.