No Sublogarithmic-Time Approximation Scheme for Bipartite Vertex Cover

Mika Göös · **Jukka Suomela**

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 $\varepsilon > 0$ there exists a *constant-time* distributed algorithm that finds a $(1+\varepsilon)$-approximation of a maximum matching on bounded-degree graphs. 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 run-time 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.

Keywords Distributed graph algorithms · Local approximation · Lower bounds · Vertex cover

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, and it implies that *König's theorem is non-local*—see Sect. 1.4.

On a technical level our lower bound result exhibits the following features:

- **Expanders:** Our 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.
- **A new source of hardness:** 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 task of *gluing together* two different types of local solutions.
- **The RECUT problem:** To formalise this new source of hardness we introduce *a new distributed cut minimisation problem* called RECUT, which might have applications elsewhere.

1.1 The $\mathcal{LOCAL}$ Model

We work in the standard $\mathcal{LOCAL}$ model of distributed computing [13,20]. 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

A preliminary version of this work [5] appeared in DISC 2012.

M. Göös
Department of Computer Science, University of Toronto, 3302-10 King's College Road, Toronto, Ontario M5S 3G4, Canada.
E-mail: mika.goos@mail.utoronto.ca

J. 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: jukka.suomela@cs.helsinki.fi

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 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 Bipartite Vertex Cover

Distributed algorithms for the bipartite vertex cover problem are run on *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.

1.3 Our Result

Our main result is the following.

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

Our proof of Theorem 1 permits one to take $\delta = 0.01$, but since our argument does not seem to yield the best possible value for δ, we do not try to optimise it.

The run-time lower bound in Theorem 1 is tight: a matching run-time upper bound is given by the well-known network decomposition algorithm due to Linial and Saks [14].

Theorem 2 (Consequence of Linial–Saks) *Let $\varepsilon > 0$. An expected $(1 + \varepsilon)$-approximation of 2-VC can be computed in time $O(\varepsilon^{-1} \log n)$ on graphs of maximum degree $\Delta = O(1)$.*

1.4 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):

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

It is known from general LP theory (see, e.g., Papadimitriou and Steiglitz [18, §13.2]) that on bipartite graphs the above LPs do not have an integrality gap: 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, respectively.

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

1. *The primal and dual LPs admit local approximation schemes.* As part of their general result, Kuhn et al. [10] give constant-time algorithms for computing $(1 + \varepsilon)$-approximations for the above LPs. That is, their algorithms run in time $O_\varepsilon(1)$ that is dependent on ε, but independent of the number of nodes.
2. *The integral primal problem admits a local approximation scheme.* Nguyen and Onak [17] describe a randomised constant-time approximation scheme for the maximum matching problem on bounded-degree graphs. In case the input graph is 2-coloured, there is also a deterministic constant-time approximation scheme [1].
3. *The integral dual problem 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 $\Delta = 3$.

These run-time bounds are summarised in the following table where our new lower bound is highlighted:

	Primal	Dual
Integral	$O_\varepsilon(1)$	$\Omega(\log n)$
LP	$O_\varepsilon(1)$	$O_\varepsilon(1)$

1.5 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 [13] lower bound $\Omega(\log^* n)$ for 3-colouring a cycle together with the Ramsey technique of Naor and Stockmeyer [16] 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 [12] and Czygrinow et al. [2]. A recent work [4] generalised these techniques even further to show 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 [21].

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 [9,10,11] 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 $\Omega(\log \Delta)$. Their technique consists of showing that a fast algorithm cannot locally 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. Their lower-bound graphs have $\Delta = \omega(1)$, as this is necessary in order to prove an $\omega(1)$ run-time lower bound; if their construction is halted at $\Delta = O(1)$, it becomes possible to locally distinguish v and u topology-wise.

By contrast, in this work, we need a new source of hardness that is present even in case $\Delta = O(1)$.

Sublinear-time centralised algorithms. Parnas and Ron [19] discuss how a fast distributed algorithm can be used as a *solution oracle* in a centralised algorithm that approximates parameters of a sparse graph G via a randomised query access to G. Thus, query complexity lower bounds in this model imply time complexity lower bounds for distributed algorithms. In particular, an argument of Trevisan (presented in [19]) implies that computing a $(2-\varepsilon)$-approximation of a minimum vertex cover requires $\Omega(\log n)$ time on d-regular graphs, where $d = d(\varepsilon)$ is sufficiently large.

We note that (the size of) 2-VC is easy to approximate in the Parnas–Ron model: if we are promised that G is bipartite, König duality applies and we can use any of the constant-time algorithms mentioned in Sect. 1.4 as a solution oracle to obtain an estimate for the common size of a minimum vertex cover and a maximum matching.

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 Sect. 2.1; in this model, we only consider a certain class of deterministic distributed algorithms in anonymous networks. Later in Sect. 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.

We emphasise that the deterministic lower bound given in the present section contains all the important ideas; the subsequent extension to randomised algorithms is somewhat of a technicality.

Proof overview. 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. More formally, we give an approximation-preserving local reduction

$$\text{RECUT} \leq \text{2-VC}. \tag{1}$$

We then show that RECUT is hard to approximate locally, which, by (1), 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 *anonymous networks* $G = (V,E)$. More precisely, we impose the following additional restrictions in the $\mathcal{LOCAL}$ model:

- **Determinism:** The nodes of G are not given random bits as input.
- **Anonymity:** The output of $\mathcal{A}$ is invariant under reassigning node identifiers. That is, if G is isomorphic to $G' = (V',E')$ via a mapping $f\colon V \to V'$, then the output of a node $v \in V$ agrees with the output of $f(v) \in V'$:

$$\mathcal{A}(G,v) = \mathcal{A}(G', f(v)). \tag{2}$$

Put otherwise, the only available symmetry-breaking information is the radius-r neighbourhood topology—the nodes do not have unique identifiers.

We will also consider graphs that may be associated with some additional symmetry-breaking structure:

- **Node labels:** A node $v \in V$ is supplied with a label $\ell(v)$. In the context of property (2), we must now require that the isomorphism $f\colon V \to V'$ between labelled graphs (G,ℓ) and (G',ℓ') preserves the labels in the sense that $\ell(v) = \ell'(f(v))$ for all $v \in V$.
- **Directed edges:** The edges E may be directed. Here, the directions are merely additional data; they do not restrict communication.

2.2 Recut Problem

In the RECUT problem we will be interested in 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:

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

where $e(U,U')$ denotes the number of edges with one endpoint in U and another in U'. As usual, we also write $\ell(V) := \{\ell(v) : v \in V\}$ for the image of ℓ.

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 Fig. 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 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 $\ell(U') = \{\mathsf{blue}\}$, algorithm $\mathcal{A}$ cannot avoid producing at least some red/blue edge boundary.

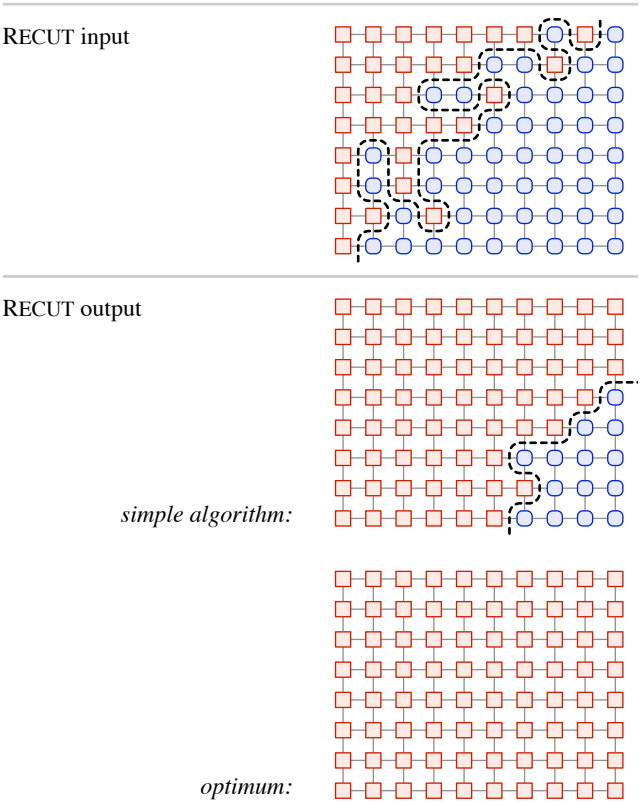

Fig. 1 The RECUT problem. In this example, we have used a simple distributed algorithm $\mathcal{A}$ to find a recut ℓ_{out} with a small boundary $\partial\ell_{\mathsf{out}}$: a node outputs red iff there is a red node within distance $r = 3$ in the input. While the solution is not optimal, in a grid 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.

Indeed, the best we can hope $\mathcal{A}$ to achieve is a recut $\ell_{\mathcal{A}}$ of size $\partial\ell_{\mathcal{A}} \leq \varepsilon$ for some small constant $\varepsilon > 0$. Such recuts can be computed using, e.g., graph decomposition algorithms for G: if we are given a decomposition of G into low-diameter components that is induced by deleting a small fraction of edges, we can simply colour the components monochromatically. Currently, the fastest decomposition algorithms found in the literature are all randomised: the Linial–Saks [14] decomposition algorithm allows one to compute a recut of size $\partial\ell_{\mathcal{A}} \leq \varepsilon$ in time $O_\varepsilon(\log n)$ on any graph G, whereas for some restricted graph families decomposition is possible even in constant-time; see, e.g., Hassidim et al. [6].

Interpretation. 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 Reduction

Terminology. 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 that 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 (unlabelled) digraph G, because such a graph is *locally homogeneous*: all the r-neighbourhoods of G are pairwise isomorphic.

Using this terminology we prove the following.

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

The proof of Theorem 3 follows the usual route. In three steps, we describe a reduction that can be computed by a local algorithm:

1. We start with an instance (G,ℓ) of RECUT and transform it into a white/black-coloured instance $\Pi(G,\ell)$ of 2-VC.
2. Then, we simulate $\mathcal{A}$ on the resulting instance $\Pi(G,\ell)$.
3. Finally, we map the output of $\mathcal{A}$ back to a solution $\ell_{\mathcal{A}}$ of the RECUT instance (G,ℓ).

We now proceed with the details. 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 gadgets. 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. See Fig. 2.

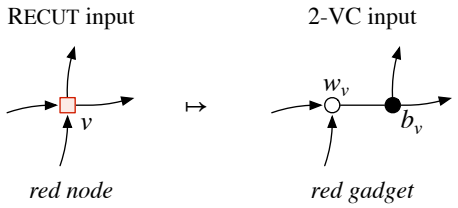

Fig. 2 Red gadget for $\Delta = 4$.

The case of all-red input. 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, in $\Pi(G,\mathsf{red})$ the solutions output by $\mathcal{A}$ on the endpoints of e_v are isomorphic across all v. Assuming $\varepsilon < 1$ it follows that 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.

Blue gadgets. 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. See Fig. 3.

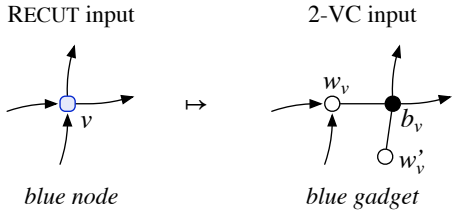

Fig. 3 Blue gadget for $\Delta = 4$ (assuming an all-red input produces an all-white output).

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.

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

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

See Fig. 4. 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.

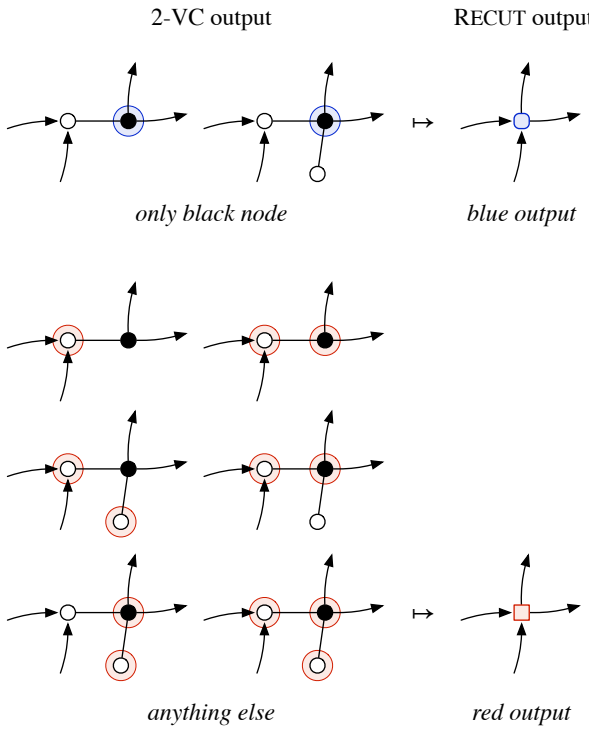

Fig. 4 Mapping the output of $\mathcal{A}$ to a solution of the RECUT problem.

Recut analysis. Call a red vertex v in $(G,\ell_{\mathcal{A}})$ *bad* if v has a blue out-neighbour u; see Fig. 5.

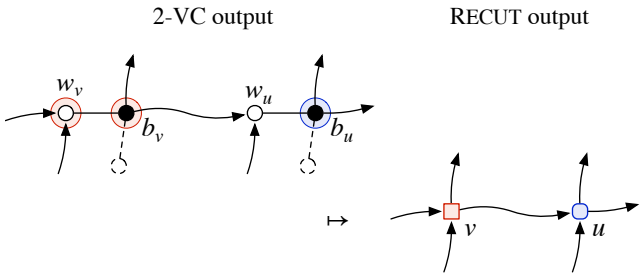

Fig. 5 A bad node: v is red and its out-neighbour u is blue.

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 $\varepsilon|V|$ bad vertices as $\mathcal{A}$ produces a $(1+\varepsilon)$-approximation of 2-VC on $\Pi(G,\ell)$.

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 $\partial\ell_{\mathcal{A}} \cdot |E|/2 \leq 2\varepsilon|V|$ which gives $\partial\ell_{\mathcal{A}} \leq 2\varepsilon$, 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; see Fig. 6.

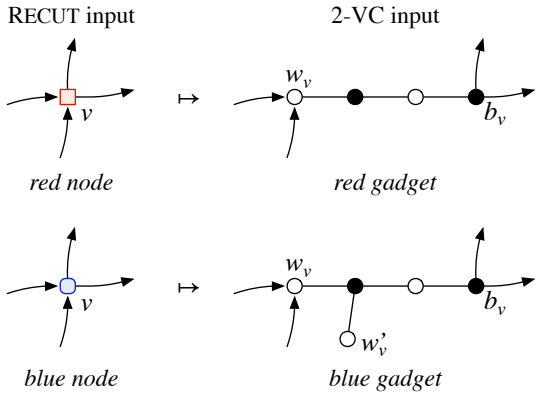

Fig. 6 Gadgets for $\Delta = 3$.

Again, to achieve a 1.499-approximation of 2-VC on $\Pi(G,\mathsf{red})$, 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 $\partial\ell_{\mathcal{A}} \leq 4\varepsilon$.

2.4 Recut Is Hard on Expanders

Intuitively, the difficulty in computing a small recut on general graphs stems from the inability of an algorithm 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 *expander graphs* as a basis for our lower-bound construction.

Definition 3 (Expander graphs) Let $\delta > 0$. A graph $G = (V,E)$, $|V| = n$, is called a δ-*expander* if it satisfies the edge expansion condition

$$e(S, V \setminus S) \geq \delta \cdot |S| \qquad \text{for all } S \subseteq V,\ |S| \leq n/2, \tag{3}$$

where $e(S, V \setminus S)$ is the number of edges leaving S.

See Hoory et al. [7] for a survey on expanders graphs.

For our proof, we will need an infinite family $\mathcal{F}$ of expander graphs, i.e., there is a universal constant $\delta > 0$ so that each $G \in \mathcal{F}$ is a δ-expander. Now, to fool an algorithm $\mathcal{A}$ into producing a large recut on the graphs $G \in \mathcal{F}$ it is enough for us to force $\mathcal{A}$ to output a *nearly balanced recut* $\ell_{\mathcal{A}}$ on G where both colour classes have size $n/2 \pm o(n)$. 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 (3) 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 1 *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 $|B_G(v_i, r)| \leq 4^r + 1 = o(n)$, 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 $o(n)$; see Fig. 7 for an illustration. 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 that some labelling in our sequence must force $\mathcal{A}$ to output $n/2 - o(n)$ red nodes. □

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 [15]. 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 1 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.

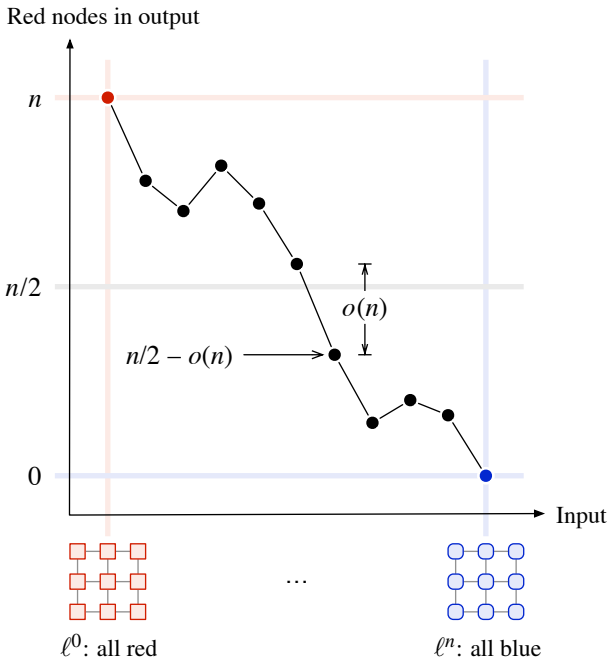

Fig. 7 An illustration for the proof of Lemma 1.

3 Randomised Lower Bound

Model. Even though our model of deterministic algorithms in Sect. 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 $1 - 1/n$.

Simplifying assumptions. Without loss of generality, we may assume the randomised algorithm is of the following form:

(a) **Deterministic run-time:** Each node runs for at most $r = o(\log n)$ steps.
(b) **Las Vegas algorithm:** The algorithm always produces a feasible solution.

Indeed, if we are given an algorithm that has expected running time $r' = o(\log n)$ and covers each edge with probability $1 - o(1)$, we can modify it so that it satisfies the above two properties at a cost of only an additive $o(1)$ term in the expected approximation ratio. This is done as follows:

(a) Choose a slowly growing function t such that $r := tr' = o(\log n)$. If a node v runs longer than r steps, we stop v's computation and output v into the vertex cover. By Markov's inequality, this modification interferes with the computation of only $o(n)$ nodes in expectation.
(b) After r steps we finish by including both endpoints of each uncovered edge in the output.

Overview. When discussing randomised algorithms many of the simplifying assumptions made in Sect. 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 Sect. 2: we force the randomised algorithm to output a nearly balanced recut *with high probability*, i.e., with probability $1-o(1)$. 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 Sect. 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 Sect. 2. 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

$$\mathbf{E}[R(\mathsf{red})] \geq n/2, \tag{4}$$

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

$$\mathbf{E}[R(\mathsf{blue})] \leq n/2. \tag{5}$$

The inequalities (4) and (5) provide the necessary boundary conditions for the argument of Lemma 1: by transforming the instance (G, red) into (G, blue) by changing the node colours one at a time we may find an input labelling ℓ^* achieving

$$\mathbf{E}[R(\ell^*)] = n/2 - o(n). \tag{6}$$

Note that this does not yet prove that $\mathcal{A}$'s output is a bad approximation in expectation; it might be the case that half the time $\mathcal{A}$ outputs all-red and half the time all-blue. To rule out this possibility we need to show 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

$$R = \sum_{v \in V} X_v, \tag{7}$$

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 [8] presents large deviation bounds for sums of type (7) by applying Chernoff–Hoeffding bounds for each colour class in a $\chi(G^{2r})$-colouring of G^{2r}. For any $\varepsilon > 0$, Theorem 2.1 in Janson [8], as applied to our setting, gives

$$\begin{aligned}\mathbf{Pr}(R \geq \mathbf{E}[R] + \varepsilon n) &\leq \exp\left(-2\frac{(\varepsilon n)^2}{\chi(G^{2r}) \cdot n}\right) \\ &\to 0, \quad \text{as } n \to \infty,\end{aligned} \tag{8}$$

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

In conclusion, the combination of (6) and (8) implies that, for large n, algorithm $\mathcal{A}$ outputs a nearly balanced recut on $\Pi(G,\ell^*)$ with high probability. By the discussion in Sect. 2, this proves Theorem 1.

4 Randomised Upper Bound

We now proceed to prove the matching positive result, Theorem 2. The subroutine *Construct_Block* in the algorithm of Linial and Saks [14] computes, in time $r = O(\varepsilon^{-1} \log n)$, a set $S \subseteq V$ with the following two properties.

- Each component in the subgraph $G[S]$ induced by S has a small *weak diameter* in the following sense: for each pair $u, v \in S$ that belong to the same component of $G[S]$, we have that $\mathrm{dist}_G(u,v) \leq r$.
- In expectation, $|S| \geq (1-\varepsilon)n$.

Let C be a component of $G[S]$. Every node of C can discover the structure of C in time $O(r)$ by exploiting the weak diameter property: C is contained in the r-neighbourhood of every node in C. 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 $V \setminus S$. This results in a solution of size at most

$$\mathsf{OPT}_{G[S]} + \varepsilon n \leq \mathsf{OPT}_G + \varepsilon n.$$

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

5 Conclusions

In this work we have shown that there is a positive constant δ such that no distributed algorithm can find a $(1+\delta)$-approximation of a minimum vertex cover in sublogarithmic time. While prior lower bound constructions are based on high-degree graphs, our negative result holds in the simplest possible case of bipartite graphs of maximum degree 3.

To prove the lower bound result, we introduced a very simple graph problem—the RECUT problem. In a sense, the RECUT problem resembles the binary consensus problem: if all inputs are red, all outputs must be red, and if all inputs are blue, all outputs must be blue. In the general case of mixed inputs, we do not require that all nodes agree on the same output. Nevertheless, we would prefer to have few pairs of adjacent nodes that disagree on their outputs. This way we have turned the binary consensus problem into an optimisation problem that can be studied from the perspective of distributed graph algorithms.

We show that RECUT is difficult to solve well on expanded graphs in sublogarithmic time, even if we can use randomness. The lower bound for the vertex cover problem follows by simple local reductions. We believe similar techniques could be used to prove lower bounds for other graph problems as well.

Acknowledgements Many thanks to Valentin Polishchuk for discussions, and to anonymous reviewers for their helpful comments and suggestions. This work was supported in part by the Academy of Finland, Grants 132380 and 252018.

References

1. Åstrand, M., Polishchuk, V., Rybicki, J., Suomela, J., Uitto, J.: Local algorithms in (weakly) coloured graphs (2010). Manuscript, arXiv:1002.0125 [cs.DC]
2. Czygrinow, A., Hańćkowiak, M., Wawrzyniak, W.: Fast distributed approximations in planar graphs. In: Proc. 22nd Symposium on Distributed Computing (DISC 2008), *LNCS*, vol. 5218, pp. 78–92. Springer, Berlin (2008). DOI 10.1007/978-3-540-87779-0_6
3. Diestel, R.: Graph Theory, 3rd edn. Springer, Berlin (2005). URL `http://diestel-graph-theory.com/`
4. Göös, M., Hirvonen, J., Suomela, J.: Lower bounds for local approximation. In: Proc. 31st Symposium on Principles of Distributed Computing (PODC 2012), pp. 175–184. ACM Press, New York (2012). DOI 10.1145/2332432.2332465
5. Göös, M., Suomela, J.: No sublogarithmic-time approximation scheme for bipartite vertex cover. In: Proc. 26th Symposium on Distributed Computing (DISC 2012), *LNCS*, vol. 7611, pp. 181–194. Springer, Berlin (2012). DOI 10.1007/978-3-642-33651-5_13
6. Hassidim, A., Kelner, J.A., Nguyen, H.N., Onak, K.: Local graph partitions for approximation and testing. In: Proc. 50th Symposium on Foundations of Computer Science (FOCS 2009), pp. 22–31. IEEE Computer Society Press, Los Alamitos (2009). DOI 10.1109/FOCS.2009.77
7. Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bulletin of the American Mathematical Society **43**(4), 439–561 (2006). DOI 10.1090/S0273-0979-06-01126-8
8. Janson, S.: Large deviations for sums of partly dependent random variables. Random Structures & Algorithms **24**(3), 234–248 (2004). DOI 10.1002/rsa.v24:3
9. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed locally! In: Proc. 23rd Symposium on Principles of Distributed Computing (PODC 2004), pp. 300–309. ACM Press, New York (2004). DOI 10.1145/1011767.1011811
10. Kuhn, F., Moscibroda, T., Wattenhofer, R.: The price of being near-sighted. In: Proc. 17th Symposium on Discrete Algorithms (SODA 2006), pp. 980–989. ACM Press, New York (2006). DOI 10.1145/1109557.1109666
11. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local computation: Lower and upper bounds (2010). Manuscript, arXiv:1011.5470 [cs.DC]
12. Lenzen, C., Wattenhofer, R.: Leveraging Linial's locality limit. In: Proc. 22nd Symposium on Distributed Computing (DISC 2008), *LNCS*, vol. 5218, pp. 394–407. Springer, Berlin (2008). DOI 10.1007/978-3-540-87779-0_27
13. Linial, N.: Locality in distributed graph algorithms. SIAM Journal on Computing **21**(1), 193–201 (1992). DOI 10.1137/0221015
14. Linial, N., Saks, M.: Low diameter graph decompositions. Combinatorica **13**, 441–454 (1993). DOI 10.1007/BF01303516
15. Morgenstern, M.: 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). DOI 10.1006/jctb.1994.1054
16. Naor, M., Stockmeyer, L.: What can be computed locally? SIAM Journal on Computing **24**(6), 1259–1277 (1995). DOI 10.1137/S0097539793254571
17. Nguyen, H.N., Onak, K.: Constant-time approximation algorithms via local improvements. In: Proc. 49th Symposium on Foundations of Computer Science (FOCS 2008), pp. 327–336. IEEE Computer Society Press, Los Alamitos (2008). DOI 10.1109/FOCS.2008.81
18. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications, Inc., Mineola, NY, USA (1998)
19. Parnas, M., Ron, D.: Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. Theoretical Computer Science **381**(1–3), 183–196 (2007). DOI 10.1016/j.tcs.2007.04.040
20. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia (2000)
21. Suomela, J.: Survey of local algorithms. ACM Computing Surveys **45**(2), 24:1–40 (2013). DOI 10.1145/2431211.2431223. URL `http://www.cs.helsinki.fi/local-survey/`