A Local 2-Approximation Algorithm for the Vertex Cover Problem

Matti Åstrand, Patrik Floréen, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto

Helsinki Institute for Information Technology HIIT, University of Helsinki
P.O. Box 68, FI-00014 University of Helsinki, Finland
firstname.lastname@cs.helsinki.fi

Abstract. We present a distributed 2-approximation algorithm for the minimum vertex cover problem. The algorithm is deterministic, and it runs in $(\Delta+1)^2$ synchronous communication rounds, where Δ is the maximum degree of the graph. For $\Delta = 3$, we give a 2-approximation algorithm also for the weighted version of the problem.

1 Introduction

The minimum vertex cover is one of the best-known NP-hard graph problems. The decision version was one of Karp's [1] original NP-complete problems, and it is the first problem in Garey and Johnson's [2] list.

In a centralised setting, the polynomial-time approximability of the vertex cover is a long-standing open question. Finding a factor 2 approximation is easy, and there is some evidence that the problem may be hard to approximate within factor $2 - \epsilon$ for any constant $\epsilon > 0$ [3].

In this work, we study the approximability of the vertex cover problem in a *distributed* setting. We present a deterministic distributed algorithm that finds a 2-approximation of a minimum vertex cover in $(\Delta+1)^2$ communication rounds, where Δ is an upper bound on the maximum degree of the graph. To our knowledge, this is the first deterministic distributed 2-approximation algorithm for the vertex cover problem whose running time depends only on Δ and not on the number of nodes in the graph.

Prior Work. Several distributed 2-approximation algorithms are known for the vertex cover problem (see, e.g., Grandoni et al. [4]). In particular, any distributed algorithm that finds a maximal matching also provides a 2-approximation algorithm for the vertex cover problem; for example, Hańćkowiak et al. [5] present a distributed algorithm that finds a maximal matching in $O(\log^4 n)$ rounds, and Panconesi and Rizzi's [6] algorithm finds a maximal matching on $O(\log^* n + \Delta)$ rounds. However, the running time of any such algorithm depends on n, the number of nodes: Linial's [7] seminal result shows that even if we have unique node identifiers, and even if the network topology is an n-cycle, it is not possible to find a maximal matching in $o(\log^* n)$ synchronous communication rounds.

In this work, we focus on *local algorithms* [8,9], in the strict meaning of the term: a local algorithm is a distributed algorithm whose running time is independent of the number of nodes. It is known that finding a constant-factor approximation to the minimum vertex cover requires $\Omega(\log \Delta / \log\log \Delta)$ communication rounds [10], and hence the best that one can hope for is a local approximation algorithm in bounded-degree graphs.

Several such algorithms are known. Kuhn et al. [11] present a local approximation scheme for covering LPs; this scheme, together with deterministic rounding [12], provides a factor $2 + \epsilon$ approximation in $O(\log \Delta/\epsilon^4)$ rounds for any $\epsilon > 0$. Moscibroda [13] gives a $(4 + \epsilon)$-approximation algorithm that uses the primal–dual schema. There is also a simple purely combinatorial 3-approximation algorithm with running time $O(\Delta)$ [14].

On the negative side, the recent work by Czygrinow et al. [15] and Lenzen and Wattenhofer [16] has settled that there is no local algorithm for the minimum vertex cover problem with the approximation factor $2 - \epsilon$ for any $\epsilon > 0$, and this holds even in the case $\Delta = 2$.

Hence for each $\epsilon > 0$, it is known that there is a local $(2 + \epsilon)$-approximation algorithm for vertex cover in bounded-degree graphs, and there is no local $(2 - \epsilon)$-approximation algorithm. However, the existence of a local 2-approximation algorithm for the problem has been open.

Contributions. Our work settles the question of the approximability of the vertex cover problem with distributed constant-time algorithms. We show that there is a local 2-approximation algorithm for the minimum vertex cover problem in bounded-degree graphs; together with the negative result [15,16] for factor $2 - \epsilon$, this provides a complete characterisation of the constant-time approximability of vertex cover.

Our algorithm does not require unique node identifiers. The only piece of symmetry-breaking information that we use is a port numbering, i.e., each node imposes an ordering on the incident edges. Our algorithm is deterministic, it runs in $(\Delta + 1)^2$ communication rounds, and the size of each message is 2 bits.

The algorithm is presented in Sect. 3. In Sect. 5, we give a different algorithm for the *weighted* vertex cover in graphs of maximum degree 3; the need for a different algorithm is justified by a lower bound construction in Sect. 4.

2 Preliminaries

Model of Distributed Computing. Let $\mathcal{G} = (V, E)$ be a simple, undirected graph. Throughout this work, $\mathcal{G}$ is the *communication graph* of a distributed system: each node $v \in V$ is a computational entity, and if $\{u, v\} \in E$ then the nodes u and v can exchange messages on each communication round. Let Δ be an upper bound on the maximum node degree in $\mathcal{G}$.

We assume that $\mathcal{G}$ is an *anonymous network with a port numbering*, i.e., a node $v \in V$ can refer to its neighbours by numbers $1, 2, \ldots, d(v)$, where $d(v)$ is the degree of v. No other symmetry-breaking information is assumed; in particular, we do not need unique node identifiers. We assume that each node $v \in V$ knows $d(v)$ and Δ.

Every node runs the same deterministic synchronous distributed algorithm. In one synchronous *communication round*, the following steps are performed, in this order: (i) each node performs local computation, (ii) each node sends a message to each of its neighbours, and (iii) each node receives a message from each of its neighbours. Finally, after T communication rounds, each node performs local computation and announces an output – in our case, the node announces whether it is part of the vertex cover or not. The number of rounds T is the running time of the algorithm.

We emphasise that the model that we use – deterministic distributed algorithms in anonymous port-numbered networks – is a very weak model of distributed computing. In particular, in this model it is not possible to break the symmetry in a symmetric network. For example, finding a maximal matching in an n-cycle is impossible, regardless of the running time T. The model that we use is strictly weaker than, for example, Linial's [7] model; if we had unique identifiers, we could easily find a port numbering, but the converse is not true. Constant-time distributed algorithms in this model provide efficient algorithms in virtually any conceivable model of distributed computing; to give one example, standard reductions [17] can be used to construct an efficient self-stabilising algorithm that stabilises in constant time.

Vertex Covers. A set of nodes $C \subseteq V$ is a *vertex cover* if each edge $e \in E$ is incident to at least one node in C. A *fractional vertex cover* is a non-negative function $x\colon V \to [0,1]$ such that $x(u) + x(v) \geq 1$ for each edge $\{u, v\} \in E$. A *minimum fractional vertex cover* minimises $\sum_v x(v)$. This is an LP relaxation of the vertex cover problem; a set of nodes $C \subseteq V$ is a vertex cover if and only if the characteristic function $\mathbf{1}_C\colon V \to \{0,1\}$ is a fractional vertex cover.

Edge Packings and Matchings. Throughout this work, we consider non-negative functions $y\colon E \to [0,+\infty)$ that assign a *weight* to each edge. For a node $v \in V$, let us write $s(y,v) = \sum_{e \in E : v \in e} y(e)$ for the total weight assigned to the edges adjacent to v. The function y is an *edge packing* if $s(y,v) \leq 1$ for each node $v \in V$. A *maximum edge packing* maximises $\sum_e y(e)$.

A node $v \in V$ is *saturated* in an edge packing y if $s(y,v) = 1$. An edge $\{u,v\}$ is saturated if u or v is saturated. An edge packing y is *maximal* if each edge is saturated; put otherwise, we cannot increase $y(e)$ for any $e \in E$ without violating a constraint. We write $S(y) = \{v \in V : s(y,v) = 1\}$ for the set of saturated nodes.

Edge packings can be interpreted as fractional matchings: a set of edges $M \subseteq E$ is a *matching* if the characteristic function $\mathbf{1}_M$ is an edge packing, and M is a *maximal matching* if $\mathbf{1}_M$ is a maximal edge packing. We use the shorthand notation $s(X,v) = s(\mathbf{1}_X, v)$ for a set of edges $X \subseteq E$, and we write $\mathcal{G}[X]$ for the subgraph induced by X. In other words, $s(X,v)$ is the degree of v in the graph $\mathcal{G}[X]$.

LP Duality. From the perspective of linear programming, the maximum edge packing problem is the dual of the minimum fractional vertex cover problem. From LP duality, one can obtain the following well-known lemma that forms the basis of our distributed algorithm.

Lemma 1. *If y is a maximal edge packing, then $S(y)$ is a 2-approximation of a minimum vertex cover.*

Lemma 1 is a simplified version of a classical result that dates back to Bar-Yehuda and Even [18]. From a modern perspective, this result can be seen as an application of the primal–dual schema to obtain an approximation algorithm for the minimum vertex cover problem [19,20].

In addition to the algorithm by Bar-Yehuda and Even [18], many other approximation algorithms can be interpreted as applications of Lemma 1 and its various generalisations and special cases. Hochbaum [12] applies Lemma 1 in the case where y is a maximum edge packing. The algorithm generally attributed to Fanica Gavril and Mihalis Yannakakis applies Lemma 1 in the case where y is the characteristic function of a maximal matching; then $S(y)$ consists of the endpoints of the edges in the matching – see, for example, Papadimitriou and Steiglitz [21]. Gonzalez [22] presents a simple algorithm that uses Lemma 1 directly. Khuller et al. [23] present a distributed algorithm that applies a relaxation of Lemma 1.

3 Algorithm

In this section, we present a distributed algorithm that finds a maximal edge packing. The algorithm is purely combinatorial; we do not need to refer to linear programming and duality in the description and analysis of the algorithm. Once we have found a maximal edge packing, we can apply Lemma 1 to find a 2-approximation of a minimum vertex cover.

Our algorithm heavily relies on *half-integral* edge packings. An edge packing y is half-integral if $y(e) \in \{0, 1/2, 1\}$ for each $e \in E$. In a half-integral edge packing we also have $s(y, v) \in \{0, 1/2, 1\}$ for each $v \in V$.

The following definition is central to our work.

Definition 1. *A half-integral edge packing y is* almost saturating *if the following conditions hold: If $s(y, v) = 0$ then $s(y, u) = 1$ for all neighbours u of v. If $s(y, v) = 1/2$ then $s(y, u) = 1$ for at least one neighbour u of v.*

If an edge $e = \{u, v\} \in E$ is not saturated by an almost saturating edge packing y, then $s(y, u) = s(y, v) = 1/2$; we say that e is *half-saturated* in y. Furthermore, both u and v are adjacent to saturated nodes; therefore u and v are incident to saturated edges. See Fig. 1 for an illustration.

Algorithm Overview. Our algorithm begins with the original graph $\mathcal{G}_0 = \mathcal{G}$. In each iteration $i = 0, 1, \ldots, \Delta - 1$, we find an almost saturating edge packing y_i in the graph $\mathcal{G}_i$. Then we form the subgraph $\mathcal{G}_{i+1}$ of $\mathcal{G}_i$ that is induced by the edges that are half-saturated in y_i. See Fig. 2 for an illustration.

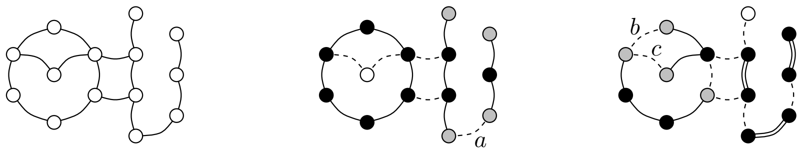


Fig. 1. A graph and two different almost saturating edge packings (see Definition 1). Double lines are edges with weight 1, single lines are edges with weight 1/2, and dashed lines are edges with weight 0. Black circles are saturated nodes, grey circles are nodes with total weight 1/2, and white circles are nodes with total weight 0. Edges a, b and c are half-saturated.

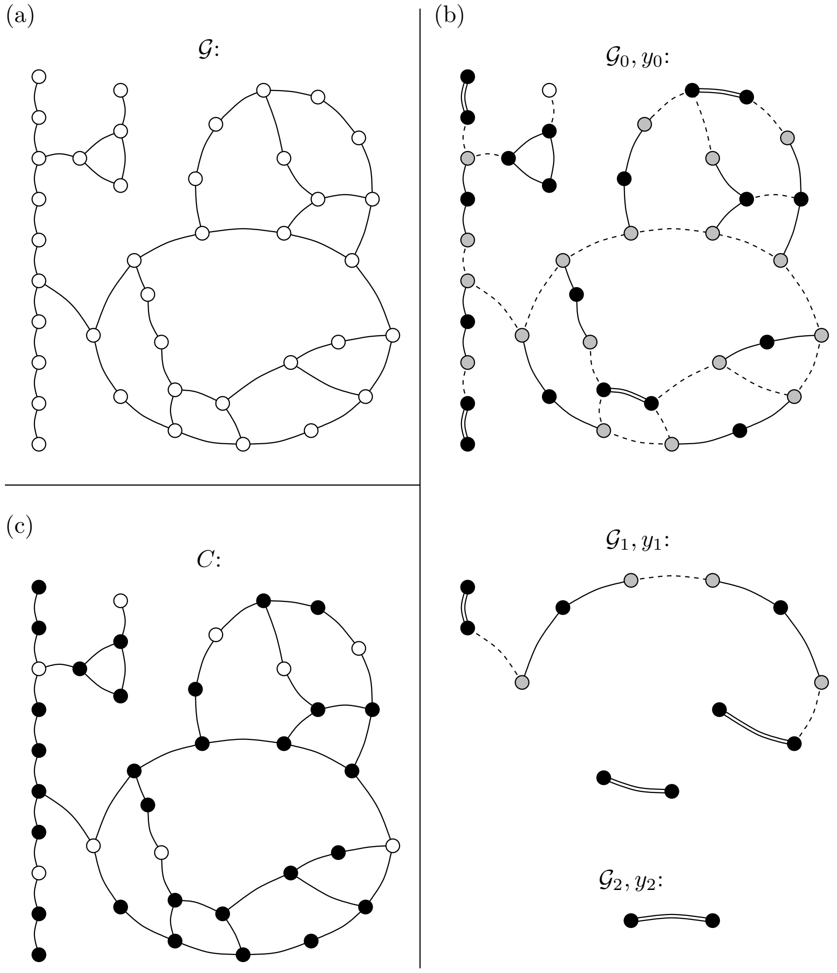


Fig. 2. Overview of the vertex cover algorithm. (a) An input graph $\mathcal{G}$. (b) The edge packings y_0, y_1, and y_2 found by the algorithm; see Fig. 1 for the notation. (c) The resulting vertex cover C.

Both endpoints of a half-saturated edge are incident to a saturated edge; therefore the maximum degree of $\mathcal{G}_{i+1}$ is strictly smaller than the maximum degree of $\mathcal{G}_i$. Since the maximum degree of $\mathcal{G}_0$ is Δ, we conclude that the graph $\mathcal{G}_\Delta$ is empty.

Extend the domain of y_i to E by setting $y_i(e) = 0$ whenever $e \in E$ is not an edge of $\mathcal{G}_i$; now each y_i is an edge packing in $\mathcal{G}$. To find a maximal edge packing in $\mathcal{G}$, construct the function

$$y = \sum_{i=0}^{\Delta-1} 2^{-i} y_i. \tag{1}$$

Lemma 2. *The function y in* (1) *is a maximal edge packing in the graph $\mathcal{G}$.*

Proof. Let us first show that y is an edge packing. Consider a node $v \in V$. Let k be the largest integer such that v is a node in $\mathcal{G}_k$. Then for each $i < k$, the node v is incident to an edge that is half-saturated in y_i, and therefore $s(y_i, v) = 1/2$. Furthermore, $s(y_k, v) \leq 1$ and $s(y_i, v) = 0$ for $i > k$. We conclude that $s(y, v) \leq 1$.

Let us then show that y is maximal. Consider an edge $e \in E$. Let k be the largest integer such that e is an edge in $\mathcal{G}_k$. Then for each $i < k$, the edge e was half-saturated in $\mathcal{G}_i$ by y_i, and finally it was saturated in $\mathcal{G}_k$ by y_k. Let u be an endpoint of e that was saturated in $\mathcal{G}_k$ by y_k. Then $s(y_i, u) = 1/2$ for $i < k$ and $s(y_k, u) = 1$. Therefore $s(y, u) = 1$, and the edge e is saturated in y. □

Thus we have found a maximal edge packing, and the saturated nodes form a 2-approximation of a minimum vertex cover by Lemma 1.

Naturally, in an implementation of the algorithm, we do not need to explicitly compute y. In each iteration i, the nodes with $s(y_i, v) = 0$ are discarded, the nodes with $s(y_i, v) = 1$ join the vertex cover, and the nodes with $s(y_i, v) = 1/2$ get a second chance on iteration $i + 1$.

Finding an Almost Saturating Edge Packing. To complete the description of the algorithm, we have to show how to find an almost saturating edge packing. Our algorithm is based on the idea of forming a maximal matching in the bipartite double cover of $\mathcal{G}_i$. The same idea has been used in prior work [14] to find a 3-approximation of a minimum vertex cover.

To construct the *bipartite double cover* $\mathcal{H} = \mathcal{G}_i \times K_2$ of the graph $\mathcal{G}_i$, replace each node $v \in V$ of $\mathcal{G}_i$ by two copies: a black copy v_1 and a white copy v_2. Replace each edge $\{u, v\} \in E$ by two edges: $\{u_1, v_2\}$ and $\{u_2, v_1\}$. Now $\mathcal{H}$ is a bipartite graph; more importantly, it is 2-coloured, and we can use the colours to break the symmetry in a distributed algorithm. The nodes in the graph $\mathcal{H}$ inherit the port numbering from the graph $\mathcal{G}$.

Now it is easy to find a maximal matching M in $\mathcal{H}$ by a distributed algorithm in 2Δ synchronous communication rounds [24]: For each $j = 1, 2, \ldots, \Delta$, in the round $2j - 1$, unmatched black nodes send proposals to their white neighbour number j, if any. In the round $2j$, all white nodes process the proposals; each

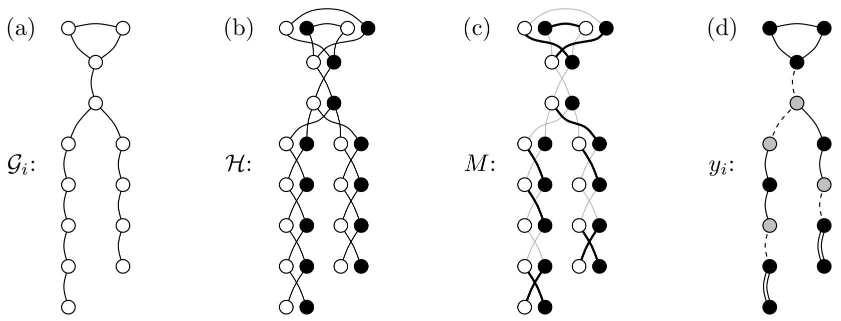


Fig. 3. Using the bipartite double cover to find an almost saturating edge packing. (a) The input graph $\mathcal{G}_i$. (b) The bipartite double cover $\mathcal{H}$ of $\mathcal{G}_i$. (c) A maximal matching M in $\mathcal{H}$. (d) An almost saturating edge packing y_i.

white node accepts the first proposal it gets, breaking the ties with port numbers. See Fig. 3 for an illustration.

Now we can construct an almost saturating edge packing y_i in $\mathcal{G}_i$ by setting

$$y_i(\{u,v\}) = \frac{\mathbf{1}_M\bigl(\{u_1,v_2\}\bigr) + \mathbf{1}_M\bigl(\{u_2,v_1\}\bigr)}{2}. \tag{2}$$

Here $\mathbf{1}_M$ is the characteristic function of M.

Lemma 3. *The function y_i in* (2) *is an almost saturating edge packing in $\mathcal{G}_i$.*

Proof. For each node $v \in V$ we have $s(y_i,v) = (s(M,v_1)+s(M,v_2))/2$. Since $s(M,v_1) \in \{0,1\}$ and $s(M,v_2) \in \{0,1\}$, we have $s(y_i,v) \le 1$, and y_i is an edge packing. Since $\mathbf{1}_M$ is integral, y_i is half-integral.

To show that y_i is almost saturating, first consider a node $v \in V$ with $s(y_i,v) = 0$. Then $s(M,v_1) = s(M,v_2) = 0$. Now let u be a neighbour of v. Since M is maximal, v_1 is not matched, and u_2 is adjacent to v_1 in $\mathcal{H}$, we must have $s(M,u_1) = 1$; similarly, $s(M,u_2) = 1$. Therefore $s(y_i,u) = 1$.

Second, consider a node $v \in V$ with $s(y_i,v) = 1/2$. Assume that $s(M,v_1) = 0$ and $s(M,v_2) = 1$; the other case is symmetric. Then there is a neighbour u of v in $\mathcal{G}_i$ such that $\{v_2,u_1\} \in M$. Furthermore, u_2 is adjacent to v_1 in $\mathcal{H}$ and v_1 is not matched; therefore u_2 must be matched in M. We have $s(M,u_1) = s(M,u_2) = 1$ and $s(y_i,u) = 1$. $\square$

Running Time and Message Complexity. Finding an almost saturating edge packing y_i in $\mathcal{G}_i$ takes $2(\Delta - i)$ synchronous communication rounds [24]. Then, in 1 communication round, each node $v \in V$ can inform its neighbours about $s(y_i,v)$; after that, each node knows its neighbours in the graph $\mathcal{G}_{i+1}$. Therefore iteration i can be completed in $2(\Delta-i)+1$ synchronous communication rounds, and the total running time is bounded by $(\Delta+1)^2$.

The algorithm can be implemented by using 2-bit messages. To find an almost saturating edge packing, the black copies of the nodes send 1-bit messages – ‘proposal’ or ‘no operation’ – and the white copies send 1-bit responses – ‘accept’ or ‘reject’, the latter of which doubles as a ‘no operation’ message. The value of $s(y_i, v)$ can be encoded in 2 bits.

The main result of this section is summarised by the following theorem.

Theorem 1. *A maximal edge packing can be found in* $(\Delta + 1)^2$ *communication rounds, using* 2*-bit messages.*

With Lemma 1, the following corollary is immediate.

Corollary 1. *A* 2*-approximation of a minimum vertex cover can be found in* $(\Delta + 1)^2$ *communication rounds, using* 2*-bit messages.*

4 Weighted Edge Packing

A natural question is whether the results from the previous section can be generalised to the case when each node $v \in V$ has a non-negative weight w_v. The definitions from Sect. 2 have straightforward generalisations: A minimum vertex cover $C \subseteq V$ minimises $\sum_{v \in C} w_v$, and a minimum fractional vertex cover $x \colon V \to [0, 1]$ minimises $\sum_v w_v x(v)$. An edge packing is a function $y \colon E \to [0, +\infty)$ that satisfies $s(y, v) \leq w_v$ for each $v \in V$, and $v \in V$ is a saturated node if $s(y, v) = w_v$.

Lemma 1 holds for the weighted graphs verbatim: if y is a maximal edge packing in a weighted graph, then the set of saturated nodes is a 2-approximation of a minimum vertex cover. Therefore the key question is whether there is a weighted counterpart of Theorem 1.

At first sight, there seems to be some hope. In particular, it is possible to find a maximal edge packing in a weighted 2-coloured graph in 2Δ rounds. However, the trick of using the bipartite double cover can no longer be applied. Indeed, we prove the following impossibility result. It shows that in the weighted case, the running time of *any* distributed algorithm necessarily depends on the number of nodes (or the range of the weights) and not only on the maximum degree.

Theorem 2. *There is no local algorithm that finds a maximal edge packing in weighted cycles.*

Theorem 2 holds even in Linial’s [7] model: we can assume unique node identifiers, and we can allow unbounded local computation and arbitrarily large messages.

Our proof uses ideas that are similar to Czygrinow et al.’s [15] proof of the inapproximability of the maximum independent set problem. Let $\mathcal{A}$ be a local algorithm that finds a feasible edge packing y in any weighted cycle: for each edge $e = \{u, v\}$, both u and v know the value $y(e)$ when the algorithm $\mathcal{A}$ terminates. Let T be the number of synchronous communication rounds that $\mathcal{A}$ takes; w.l.o.g., we assume that T is even.

Let $n \gg T$ be a constant that we fix later; n only depends on the constant T. Let $N = \{1, 2, \dots, n\}$. For any $H \subseteq N$, we define the n-cycle $\mathcal{C}_H = (N, E_H)$ as

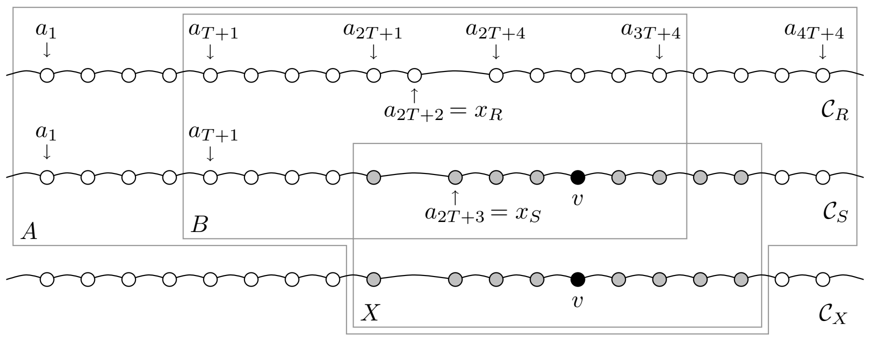
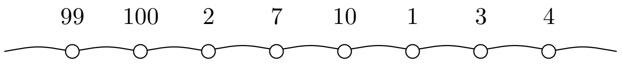


Fig. 4. Construction of $\mathcal{C}_R$ and $\mathcal{C}_S$. In this illustration, $T = 4$. The figure also illustrates the radius-T neighbourhood X of a node $v \in S \cap B$ (see the proof of Lemma 4). The output of the node v in $\mathcal{C}_S$ is identical to the output of the node v in $\mathcal{C}_X$.

follows. Let $k = |H|$. Let $H = \{h_1, h_2, \dots, h_k\}$ and let $N \setminus H = \{j_1, j_2, \dots, j_{n-k}\}$ with $h_1 < h_2 < \dots < h_k$ and $j_i < j_2 < \dots < j_{n-k}$. Then the edges of the cycle $\mathcal{C}_H$ are $E_H = \big\{\{h_1, h_2\}, \{h_2, h_3\}, \dots, \{h_k, j_1\}, \{j_1, j_2\}, \dots, \{j_{n-k}, h_1\}\big\}$. The following figure illustrates C_H in the case $n = 100$ and $H = \{2, 7, 10\}$.



Finally, we assign the unique identifiers and node weights as follows: for each node $v \in N$, the unique identifier of v is v, and the weight of v is also v.

Let us now define a function f that assigns a label 0 or 1 to each subset $X \subset N$ with $|X| = 2T + 1$. Let $X = \{x_1, x_2, \dots, x_{2T+1}\}$ with $x_1 < x_2 < \dots < x_{2T+1}$. Consider the execution of $\mathcal{A}$ in the cycle $\mathcal{C}_X$. If the node x_{T+1} is saturated in the edge packing y produced by $\mathcal{A}$, we set $f(X) = 1$, otherwise we set $f(X) = 0$.

By Ramsey's theorem [25], we can choose the value of n so that the following holds, no matter how we choose the values $f(X)$: there exists a label $\ell \in \{0, 1\}$ and a subset $A \subseteq N$ with $|A| = 4T + 4$ such that $f(X) = \ell$ whenever $X \subset A$ and $|X| = 2T + 1$.

Now let $A = \{a_1, a_2, \dots, a_{4T+4}\}$ with $a_1 < a_2 < \dots < a_{4T+4}$. Let $R = A \setminus \{a_{2T+3}\}$ and $S = A \setminus \{a_{2T+2}\}$; see Fig. 4. Theorem 2 follows from the following lemma.

Lemma 4. *Algorithm $\mathcal{A}$ cannot produce a maximal edge packing in both $\mathcal{C}_R$ and $\mathcal{C}_S$.*

Proof. To reach a contradiction, assume that $\mathcal{A}$ produces a maximal edge packing in both $\mathcal{C}_R$ and $\mathcal{C}_S$. Define $B = \{a_{T+1}, a_{T+2}, \dots, a_{3T+4}\}$. Let $K \in \{R, S\}$, and let y_K be the edge packing computed by $\mathcal{A}$ in $\mathcal{C}_K$.

We first show that each node $v \in K \cap B$ is saturated in by y_K. To see this, let X consist of the nodes that are within distance T from v in $\mathcal{C}_K$. By construction, $X \subseteq A$. We make the following observations. The algorithm $\mathcal{A}$ terminates after

T communication rounds, and in T rounds, information propagates only for T hops in the cycle $\mathcal{C}_K$. Therefore the output of $\mathcal{A}$ at the node v only depends on the identifiers and the weights in the subgraph induced by X. The subgraph induced by X is identical in $\mathcal{C}_K$ and $\mathcal{C}_X$. Therefore the node v produces the same output in $\mathcal{C}_K$ and $\mathcal{C}_X$. In particular, v is saturated in $\mathcal{C}_K$ if and only if $f(X) = 1$. By the choice of A, all nodes in $K \cap B$ are either saturated or non-saturated. Since we assumed that $\mathcal{A}$ produces a maximal edge packing in $\mathcal{C}_K$, they must all be saturated (that is, $\ell = 1$).

Therefore the subset $K \cap B$ in $\mathcal{C}_K$ forms a chain of saturated nodes with strictly increasing weights. Let $x_K = a_{2T+2}$ if $K = R$ and $x_K = a_{2T+3}$ if $K = S$. With this notation, we have

$$\begin{aligned} y_K(\{a_{3T+3}, a_{3T+4}\}) &= a_{3T+3} - y_K(\{a_{3T+2}, a_{3T+3}\}) \\ &= a_{3T+3} - a_{3T+2} + y_K(\{a_{3T+1}, a_{3T+2}\}) = \dots \\ &= a_{3T+3} - a_{3T+2} + \dots \\ &\qquad \dots + a_{2T+5} - a_{2T+4} + x_K - a_{2T+1} + \dots \\ &\qquad \dots + a_{T+2} - y_K(\{a_{T+1}, a_{T+2}\}). \end{aligned} \tag{3}$$

Since the radius-T neighbourhoods of a_{T+1} and a_{3T+4} are identical in $\mathcal{C}_R$ and $\mathcal{C}_S$, we have

$$\begin{aligned} y_R(\{a_{T+1}, a_{T+2}\}) &= y_S(\{a_{T+1}, a_{T+2}\}), \\ y_R(\{a_{3T+3}, a_{3T+4}\}) &= y_S(\{a_{3T+3}, a_{3T+4}\}). \end{aligned}$$

This is a contradiction with (3) and $x_R \neq x_S$. □

Theorems 1 and 2 are one of the few pairs of results where the existence of weights makes a significant difference from the perspective of local algorithms. This is unlike problems such as max-min LPs [26], in which the existence of weights is provably irrelevant as far as local approximability is concerned.

5 Weighted Vertex Cover

Theorem 2 shows that we cannot directly apply the weighted version of Lemma 1 to design a local 2-approximation algorithm for the minimum-weight vertex cover problem in bounded-degree graphs. This setback suggests the possibility that there is no local 2-approximation algorithm for the problem. However, we show that the opposite is the case if $\Delta \leq 3$.

Theorem 3. *There is a local algorithm that finds a factor* 2 *approximation of a minimum-weight vertex cover in graphs with maximum degree* 3.

Let $\mathcal{G} = (V, E)$ be a node-weighted graph with maximum degree 3, see Fig. 5a. Let $w_v \geq 0$ denote the weight of the node $v \in V$; we call w_v the *w-weight* of v. We will now present an algorithm for finding a 2-approximation for weighted vertex cover in $\mathcal{G}$. The algorithm works in three stages. We will construct three sets, C_{I}, C_{II}, and C_{III}, the union of which is the vertex cover C that our algorithm outputs.

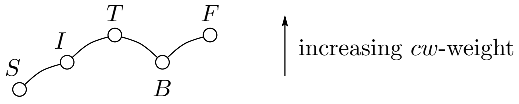
Stage I. Let $E_{\mathrm{I}} = \{\{u,v\} \in E : w_u = w_v\}$ be the edges whose endpoints have the same w-weight, see Fig. 5b. Use the algorithm from Sect. 3 to find a maximal edge packing y_{I} in $\mathcal{G}[E_{\mathrm{I}}]$. Pick the nodes saturated by y_{I} into the set C_{I}; see Fig. 5c.

Stage II. Let $E_{\mathrm{II}} \subseteq E$ be the edges not saturated by y_{I}. Since the endpoints of every edge in E_{II} have different w-weights, the edges in E_{II} can be oriented according to the w-weight; see Fig. 5d. Add a dummy degree-1 neighbour to every node with even degree in $\mathcal{G}[E_{\mathrm{II}}]$, as illustrated in Fig. 5e. The new graph is oriented and every node has an odd degree, so we can run the Naor–Stockmeyer algorithm [8,27] to find a weak 2-colouring of the nodes of the new graph; see black and white nodes in Fig. 5f. Now forget about the dummy nodes and only consider the original nodes of $\mathcal{G}[E_{\mathrm{II}}]$; each node with an odd degree has a neighbour of a different colour.

Consider the edges $E'_{\mathrm{II}} \subseteq E_{\mathrm{II}}$ whose endpoints have different colours (heavy lines in Fig. 5f); the subgraph $\mathcal{G}[E'_{\mathrm{II}}]$ is 2-coloured. The proposal–acceptance procedure for finding a maximal matching in 2-coloured graphs [24] can be modified to find a maximal edge packing: every proposing node proposes its residual weight along an edge, and every accepting node accepts its residual weight; at every proposal–acceptance round each node either saturates at least one adjacent edge or learns that at least one adjacent edge is saturated, so the algorithm completes in 2Δ rounds. Using this procedure, we can find a maximal edge packing y_{II} in $\mathcal{G}[E'_{\mathrm{II}}]$ while giving each node v the weight $r_v = w_v - s(y_{\mathrm{I}}, v)$. Pick the nodes saturated (w.r.t. r-weights) by y_{II} into the set C_{II}; see Fig. 5g. In other words, $C_{\mathrm{I}} \cup C_{\mathrm{II}}$ consists of the nodes saturated (w.r.t. w-weights) by $y_{\mathrm{I}} + y_{\mathrm{II}}$.

Stage III. Let $E_{\mathrm{III}} \subseteq E$ be the edges not saturated by $y_{\mathrm{I}} + y_{\mathrm{II}}$; see Fig. 5h. For each degree-3 node of $\mathcal{G}$, the edge packing $y_{\mathrm{I}} + y_{\mathrm{II}}$ saturates at least one adjacent edge. Thus, $\mathcal{G}[E_{\mathrm{III}}]$ has maximum degree at most 2, i.e., it consists of disjoint paths and cycles. Note that by Theorem 2 we cannot find a maximal edge packing in $\mathcal{G}[E_{\mathrm{III}}]$. Instead, we will find a vertex cover in $\mathcal{G}[E_{\mathrm{III}}]$ directly. To prove that our vertex cover is not too heavy, we exhibit a maximal edge packing with a comparable weight.

Define two new weights for the nodes of $\mathcal{G}[E_{\mathrm{III}}]$. The *c-weight* of $v \in V$ is $c_v = w_v - s(y_{\mathrm{I}} + y_{\mathrm{II}}, v)$. The *cw-weight* of v is a pair (c_v, w_v); see Fig. 5h. Since adjacent nodes always have different w-weights, the endpoints of each edge can be ordered lexicographically according to the cw-weight. Hence we can partition the nodes of $\mathcal{G}[E_{\mathrm{III}}]$ in the five sets illustrated below.



The set B ("bottom") consists of degree-2 nodes that are a local minimum w.r.t. cw-weights, T ("top") consists of degree-2 nodes that are a local maximum

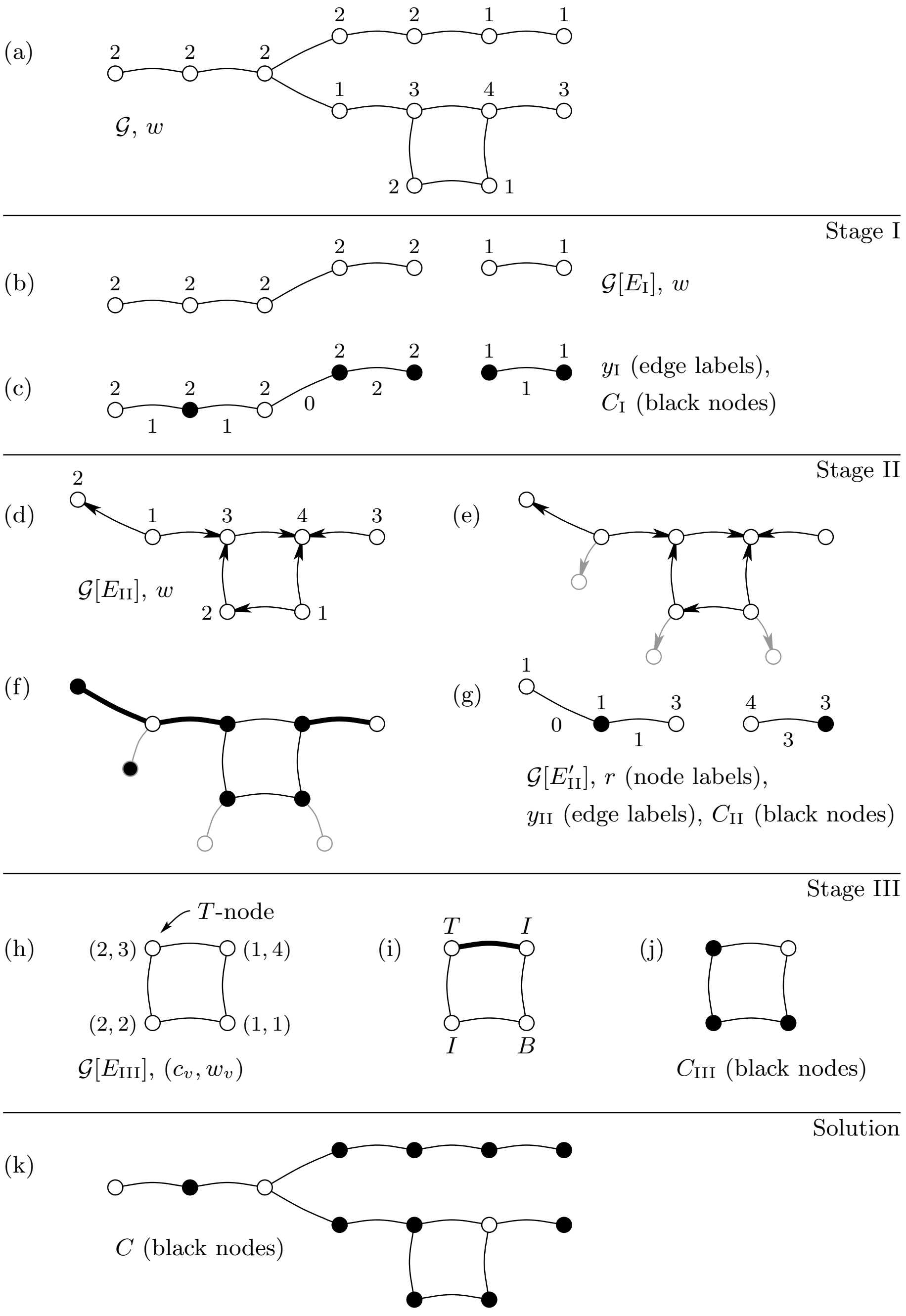
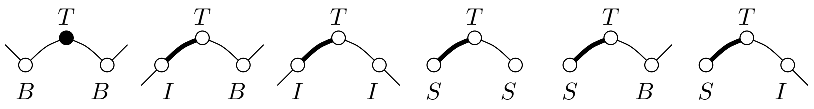

(a)
$\mathcal{G}$, w
Stage I
(b)
$\mathcal{G}[E_{\mathrm{I}}]$, w
(c)
y_{I} (edge labels),
C_{I} (black nodes)
Stage II
(d)
$\mathcal{G}[E_{\mathrm{II}}]$, w
(e)
(f)
(g)
$\mathcal{G}[E'_{\mathrm{II}}]$, r (node labels),
y_{II} (edge labels), C_{II} (black nodes)
Stage III
(h)
T-node
(2, 3)
(1, 4)
(2, 2)
(1, 1)
$\mathcal{G}[E_{\mathrm{III}}]$, (c_v, w_v)
(i)
T
I
I
B
(j)
C_{III} (black nodes)
Solution
(k)
C (black nodes)

Fig. 5. Overview of the algorithm from the proof of Theorem 3

w.r.t. cw-weights, I ("internal") consists of the other degree-2 nodes, S ("start") consists of degree-1 nodes that are a local minimum w.r.t. cw-weights, and F ("finish") consists of degree-1 nodes that are a local maximum w.r.t. cw-weights. We use the terminology such as T-nodes to refer to the nodes in the set T, and TI-edges to refer to the edges that join a T-node and an I-node, etc.

Now let $v \in T$. If both edges incident to v are TB-edges, we say that v is a *BTB-node*. Otherwise v is incident to at least one TI-edge or TS-edge; the node v chooses (arbitrarily, using the port numbers to break the symmetry) one such edge and nominates it as a *hinge edge*. The following figure shows a BTB-node (black) and hinge edges (heavy lines); note that each node is incident to at most one hinge edge.



Finally, we partition the endpoints of the hinge edges into two sets. Let $\{u, v\} \in E_{\mathrm{III}}$ be a hinge edge and let u be the node with the smaller w-weight; then we say that u is a *cheap hinge node* and v is a *costly hinge node*.

We are now ready to construct the set C_{III}. It consists of all nodes of $\mathcal{G}[E_{\mathrm{III}}]$, except the following: (i) F-nodes, (ii) BTB-nodes, and (iii) costly hinge nodes. In the example of Fig. 5i we have chosen one hinge edge. The resulting set C_{III} is illustrated in Fig. 5j.

This completes the construction of our vertex cover. Fig. 5k illustrates the solution $C = C_{\mathrm{I}} \cup C_{\mathrm{II}} \cup C_{\mathrm{III}}$.

Correctness. By construction, all edges in $E \setminus E_{\mathrm{III}}$ are covered by $C_{\mathrm{I}} \cup C_{\mathrm{II}}$. Now consider an edge $e \in E_{\mathrm{III}}$; we show that e is covered by C_{III}.

Clearly, BTB-nodes cannot be adjacent to other BTB-nodes, F-nodes, or hinge nodes. Hence if one of the endpoints of e is a BTB-node, the other endpoint must be in C_{III}. It remains to be shown that both endpoints of $e = \{u, v\}$ cannot be F-nodes or costly hinge nodes. The key observation is that there are no paths of the form (F, F), (F, T), (F, I, T), or (F, S, T). Hence F-nodes are not adjacent to other F-nodes or any kind of hinge nodes. Furthermore, there are no paths of the form (T, T), (T, I, T), (T, S, T), (T, I, I, T), (T, S, I, T), or (T, S, S, T). Hence if both u and v are hinge nodes, then e is a hinge edge and one of the nodes u, v is a cheap hinge node in C_{III}.

Approximation Ratio. We now exhibit a maximal edge packing y of $\mathcal{G}$. The set $S(y)$ of nodes saturated by y is then a 2-approximation of a minimum vertex cover. To complete the proof of Theorem 3, it is enough to show that the total w-weight of the vertex cover C constructed by our algorithm is not larger than the total w-weight of $S(y)$.

More specifically, we construct an edge packing y_{III} of $\mathcal{G}[E_{\mathrm{III}}]$ which is maximal w.r.t. c-weights. Then $y = y_{\mathrm{I}} + y_{\mathrm{II}} + y_{\mathrm{III}}$ is a maximal edge packing of $\mathcal{G}$ w.r.t. w-weights. We have $v \in C_{\mathrm{I}} \cup C_{\mathrm{II}}$ if and only if v is saturated by $y_{\mathrm{I}} + y_{\mathrm{II}}$; hence it is sufficient to show that the set of nodes saturated (w.r.t. c-weights) by y_{III} in $\mathcal{G}[E_{\mathrm{III}}]$ is at least as heavy (w.r.t. w-weights) as C_{III}.

We construct y_{III} by "propagating" c-weights from S-nodes and B-nodes up towards T-nodes and F-nodes. First we process all non-hinge nodes v, from bottom to top in the order of increasing cw-weights:

– If $v \in S$, the incident edge gets the weight c_v.
– If $v \in B$, both incident edges get the weight $c_v/2$.
– If $v \in I$ and we have already assigned the weight to one incident edge, we choose the weight of the other edge so that v is saturated.

Eventually, we have chosen a weight $y_{\mathrm{III}}(e)$ for each non-hinge edge $e \in E_{\mathrm{III}}$. We do not exceed the capacity c_v of any non-hinge node v: if $v \in S \cup B \cup I \cup F$, this follows from the fact that we proceed in the order of non-decreasing c; if $v \in T$, then v must be a BTB-node and both incident edges get weights at most $c_v/2$. Furthermore, the weights y_{III} saturate all non-hinge nodes in $S \cup B \cup I$ (and possibly some F-nodes and BTB-nodes). Hence the total weight of non-hinge nodes in C_{III} is at most the total weight of non-hinge nodes saturated by y_{III}.

Finally, we augment y_{III} so that it saturates all hinge edges as well; now y_{III} is maximal w.r.t. c-weights. Consider a hinge edge $\{u, v\}$ with $w_u < w_v$. The edge packing y_{III} saturates u or v or both, while the set C_{III} constructed by our algorithm contains only u and not v. Hence the total weight of hinge nodes in C_{III} is at most the total weight of hinge nodes saturated by y_{III}. This completes the proof of Theorem 3.

Acknowledgements. We thank anonymous reviewers for their helpful comments. This work was supported in part by the Academy of Finland, Grants 116547 and 118653 (ALGODAN), by Helsinki Graduate School in Computer Science and Engineering (Hecse), and by the Foundation of Nokia Corporation.

References

1. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
3. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2 - \epsilon$. Journal of Computer and System Sciences 74(3), 335–349 (2008)
4. Grandoni, F., Könemann, J., Panconesi, A.: Distributed weighted vertex cover via maximal matchings. ACM Transactions on Algorithms 5(1), 1–12 (2008)
5. Hańćkowiak, M., Karoński, M., Panconesi, A.: On the distributed complexity of computing maximal matchings. SIAM Journal on Discrete Mathematics 15(1), 41–57 (2001)
6. Panconesi, A., Rizzi, R.: Some simple distributed algorithms for sparse networks. Distributed Computing 14(2), 97–100 (2001)
7. Linial, N.: Locality in distributed graph algorithms. SIAM Journal on Computing 21(1), 193–201 (1992)

8. Naor, M., Stockmeyer, L.: What can be computed locally? SIAM Journal on Computing 24(6), 1259–1277 (1995)
9. Suomela, J.: Survey of local algorithms (2009, manuscript), `http://www.iki.fi/jukka.suomela/local-survey`
10. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed locally! In: Proc. 23rd Symposium on Principles of Distributed Computing (PODC), pp. 300–309. ACM Press, New York (2004)
11. Kuhn, F., Moscibroda, T., Wattenhofer, R.: The price of being near-sighted. In: Proc. 17th Symposium on Discrete Algorithms (SODA), pp. 980–989. ACM Press, New York (2006)
12. Hochbaum, D.S.: Approximation algorithms for the set covering and vertex cover problems. SIAM Journal on Computing 11(3), 555–556 (1982)
13. Moscibroda, T.: Locality, Scheduling, and Selfishness: Algorithmic Foundations of Highly Decentralized Networks. PhD thesis, ETH Zürich (2006)
14. Polishchuk, V., Suomela, J.: A simple local 3-approximation algorithm for vertex cover. Information Processing Letters 109(12), 642–645 (2009)
15. Czygrinow, A., Hańćkowiak, M., Wawrzyniak, W.: Fast distributed approximations in planar graphs. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 78–92. Springer, Heidelberg (2008)
16. Lenzen, C., Wattenhofer, R.: Leveraging Linial's locality limit. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 394–407. Springer, Heidelberg (2008)
17. Awerbuch, B., Varghese, G.: Distributed program checking: a paradigm for building self-stabilizing distributed protocols. In: Proc. 32nd Symposium on Foundations of Computer Science (FOCS), pp. 258–267. IEEE Computer Society Press, Los Alamitos (1991)
18. Bar-Yehuda, R., Even, S.: A linear-time approximation algorithm for the weighted vertex cover problem. Journal of Algorithms 2(2), 198–203 (1981)
19. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, Heidelberg (2003)
20. Vazirani, V.V.: Approximation Algorithms. Springer, Heidelberg (2001)
21. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications, New York (1998)
22. Gonzalez, T.F.: A simple LP-free approximation algorithm for the minimum weight vertex cover problem. Information Processing Letters 54(3), 129–131 (1995)
23. Khuller, S., Vishkin, U., Young, N.: A primal-dual parallel approximation technique applied to weighted set and vertex covers. Journal of Algorithms 17(2), 280–289 (1994)
24. Hańćkowiak, M., Karoński, M., Panconesi, A.: On the distributed complexity of computing maximal matchings. In: Proc. 9th Symposium on Discrete Algorithms (SODA), pp. 219–225. SIAM, Philadelphia (1998)
25. Ramsey, F.P.: On a problem of formal logic. Proceedings of the London Mathematical Society 30, 264–286 (1930)
26. Floréen, P., Kaasinen, J., Kaski, P., Suomela, J.: An optimal local approximation algorithm for max-min linear programs. In: Proc. 21st Symposium on Parallelism in Algorithms and Architectures (SPAA). ACM Press, New York (2009)
27. Mayer, A., Naor, M., Stockmeyer, L.: Local computations on static and dynamic graphs. In: Proc. 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS 1995), pp. 268–278. IEEE Computer Society Press, Los Alamitos (1995)