Distributed Weighted Vertex Cover via Maximal Matchings[⋆]

Fabrizio Grandoni[1], Jochen Könemann[2], and Alessandro Panconesi[1]

[1] Dipartimento di Informatica, Università di Roma “La Sapienza”, Via Salaria 113, 00198 Roma, Italy, {grandoni,ale}@di.uniroma1.it
[2] Department of Combinatorics and Optimization, University of Waterloo, 200 University Avenue West, Waterloo, ONN2L 3G1, Canada, jochen@math.uwaterloo.ca

Abstract. In this paper we consider the problem of computing a minimum-weight vertex-cover in an n-node, weighted, undirected graph $G = (V,E)$. We present a fully distributed algorithm for computing vertex covers of weight at most twice the optimum, in the case of integer weights. Our algorithm runs in an expected number of $O(\log n + \log \hat{W})$ communication rounds, where $\hat{W}$ is the average vertex-weight. The previous best algorithm for this problem requires $O(\log n(\log n + \log \hat{W}))$ rounds and it is not fully distributed.
For a maximal matching M in G it is a well-known fact that any vertex-cover in G needs to have at least $|M|$ vertices. Our algorithm is based on a generalization of this combinatorial lower-bound to the weighted setting.

1 Introduction

We are given an undirected graph $G = (V,E)$ and non-negative vertex weights $w_v \leq W$ for all vertices $v \in V$. A *vertex cover* is a subset $C \subseteq V$ such that each edge $e \in E$ has at least one end-point in C. In the *minimum-weight vertex-cover* problem we want to compute a vertex-cover of smallest total weight.

Computing minimum-weight vertex-covers is NP-hard [4]. Papadimitriou and Yannakakis [14] show that the problem is APX-hard. More recently, Håstad [8] proves that there is no $(7/6 - \varepsilon)$-approximation algorithm for the vertex-cover problem for any $\varepsilon > 0$ unless $\mathsf{P} = \mathsf{NP}$.

On the positive side, the best known algorithms for the vertex cover problem are due to Bar-Yehuda and Even [1], and to Monien and Speckenmeyer [12]. These algorithms achieve an approximation ratio of $(2 - \frac{\log \log n}{2 \log n})$. In graphs with maximum degree Δ, Hochbaum [9] gives a $(2 - 1/\Delta)$-approximation algorithm for the problem. This was subsequently improved by Halldórsson and Radhakrishnan [5] who present a $(2 - \frac{\log(\Delta) + O(1)}{\Delta})$-approximation algorithm. Finally, Halperin [6] presents the currently best algorithm for the problem with a performance ratio of $(2 - (1 - o(1))\frac{2 \ln \ln \Delta}{\ln \Delta})$.

In the distributed setting, it is known how to compute a 2-approximate vertex cover in the unweighted case. This can be achieved by computing a maximal matching in the graph and by including the matched nodes in the cover. A maximal matching can

[⋆] The first and third authors were supported with funding from EC Projects DELIS and EYES, and project WebMinds of the Italian Ministry of University and Research (MIUR).

be computed in $O(\log^4 n)$ rounds via the algorithm of Hanckowiack et al. [7], and in $O(\Delta + \log^* n)$ rounds via the algorithm of Panconesi and Rizzi [13]. Both algorithms are deterministic. Maximal matchings can also be computed in an expected number of $O(\log n)$ rounds via the randomized algorithm of Israeli and Itai [10].

For the weighted case a $(2+\varepsilon)$-approximation can be computed deterministically in $O(\log n \log \frac{1}{\varepsilon})$ many rounds by using the algorithm of Khuller et al. [11]. Their algorithm is stated as a PRAM algorithm, but it is readily seen to be a bona fide distributed algorithm. Let $\hat{W}$ be the average weight. Then, by setting $\varepsilon = 1/(n\hat{W}+1)$, the latter algorithm computes a 2-approximate vertex cover in $O(\log n(\log n + \log \hat{W}))$ communication rounds. Note that the above choice of ε requires global knowledge of the quantity $n\hat{W}$. This assumption may not be realistic in all scenarios.

In this paper we present an improved fully-distributed algorithm to compute a 2-approximate weighted vertex cover, in the case of integer weights. Our main result can be stated as follows:

Theorem 1. *There is a fully distributed algorithm which computes a* 2*-approximate weighted vertex cover in an expected number of* $O(\log n + \log \hat{W})$ *communication rounds. The message size is* $O(\log W)$ *and the local computation done in each round is* $O(\Delta \log(\Delta W))$ *in expectation.*

Our algorithm can be viewed as a generalization of the reduction from unweighted vertex cover to maximal matching. The basic idea is to expand each node v of weight w_v into w_v *micro-nodes* $v(1), v(2) \ldots, v(w_v)$, and connect each $v(i)$ to every $u(j)$ whenever vu is an edge of the network. Then a maximal matching in the auxiliary graph is computed. The vertex cover is given by the nodes such that all the corresponding micro-nodes are matched. If the maximal matching is computed via the fully-distributed algorithm of Israeli and Itai, the algorithm halts in an expected number of $O(\log n + \log \hat{W})$ rounds.

A naive implementation of the matching algorithm by Israeli and Itai leads to pseudo-polynomial message and time complexity in each round. The main insight leading to the bounds on message-size and local computation time in Theorem 1 is to keep an implicit representation of the auxiliary graph and a maximal matching in it.

The rest of this paper is organized as follows. In Section 2 we introduce some preliminaries. Our algorithm relies on a careful adaptation of the matching algorithm by Israeli and Itai. We present this adaptation in Section 3. Finally, Sections 4 and 5 deal with the naive and refined versions of our weighted vertex cover algorithm, respectively.

2 Preliminaries

The minimum-weight vertex cover problem can be formulated as an *integer linear program* (ILP):

$$
\begin{array}{lll}
\min & \sum_{v \in V} w_v x_v & \\
\text{s.t.} & & \\
x_v + x_u \geq 1, & & \forall vu \in E; \\
x_v \in \{0,1\}, & & \forall v \in V.
\end{array}
$$

Each assignment of the variables which satisfies the constraints (*feasible solution*) corresponds to a vertex cover containing exactly the nodes v with $x_v = 1$. By (LP) we denote the natural *linear programming relaxation* of (ILP).

Let $N(v)$ be the set of neighbors of v. The *linear programming dual* (D) of (LP) is:

$$\begin{array}{ll}
\max & \sum_{vu \in E} y_{vu} \\
\text{s.t.} & \\
\sum_{u \in N(v)} y_{vu} \leq w_v, & \forall v \in V; \\
y_{vu} \geq 0, & \forall vu \in E.
\end{array}$$

By *weak duality* (e.g., see [2]), the value of each feasible solution of (D) is a lower bound for the value of every feasible solution of (LP) and hence (IPL).

In this paper we consider a synchronous message-passing model of computation. The computation proceeds in rounds. In each round, a node can send/receive a message (of unbounded size) to/from each one of its neighbors, and execute an unbounded amount of computation. No global knowledge is available (including the number n of nodes in the graph). The algorithms presented can be easily modified so as to work in a (non-faulty) asynchronous system also.

By $B(p)$, $p \in [0,1]$, we denote a 0-1 *Bernoulli* random variable, which takes value one with probability p. A *random bit* is a Bernoulli variable $B(0.5)$.

3 Distributed Maximal Matching

A *matching* of a graph $G = (V,E)$ is a subset $M \subseteq E$ such that no two edges of M are incident to the same node. The results of the next sections are based on the following simplified version $\mathcal{M}$ of the distributed maximal-matching algorithm of Israeli and Itai [10].

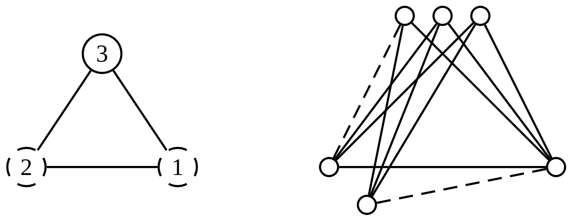
Algorithm $\mathcal{M}$ works in phases, each one consisting of a constant number of rounds. In each phase, a matching is computed and the edges incident on matched nodes are removed. The algorithm halts when no edge is left. The maximal matching is given by the union of the matchings found in the different phases.

In a given phase a matching is computed in the following way. Let $G' = (V',E')$ be the current graph. By $N'(v)$ and δ'_v we denote the set of neighbors of v and the degree of v in G', respectively. Each node v randomly decides to be a *sender* or a *receiver* with probability one half. Note that the same node may play a different role in different phases. Each sender u selects one neighbor $v \in N'(u)$ uniformly at random and makes a *proposal* to v. Each receiver v which receives at least one proposal, selects one of the proponents (arbitrarily) and *accepts* its proposal. The matching is given by the edges corresponding to accepted proposals.

Let a node v be *good* if at least one third of its neighbors u have degree $\delta_u \leq \delta_v$. To prove the bound on the number of rounds, we use the following simple combinatorial result [10]:

Lemma 1. *At least one half of the edges of a graph are incident to good nodes.*

Figure 1 A weighted graph G (on the left) with the corresponding auxiliary graph $\widetilde{G}$. A maximal matching M of $\widetilde{G}$ is indicated via dashed lines. The dashed nodes of G form a vertex cover.



Theorem 2. *Algorithm $\mathcal{M}$ computes a maximal matching in* $\mathrm{O}(\log n)$ *expected rounds.*

Proof. The correctness of the algorithm is trivial.

We show that in each phase at least a constant fraction of the edges is removed in expectation. This implies that the expected number of rounds is $\mathrm{O}(\log(n^2)) = \mathrm{O}(\log n)$. Consider a good node v of G' in a given phase. The probability P'_v that v accepts a proposal is lower bounded by:

$$P'_v \geq \frac{1}{2}\left(1 - \prod_{u \in N'(v)} \left(1 - \frac{1}{2\delta'_u}\right)\right).$$

¿From the definition of good nodes:

$$\prod_{u \in N'(v)} \left(1 - \frac{1}{2\delta'_u}\right) \leq \prod_{u \in N'(v): \delta'_u \leq \delta'_v} \left(1 - \frac{1}{2\delta'_v}\right) \leq \left(1 - \frac{1}{2\delta'_v}\right)^{\delta'_v/3} \leq e^{-1/6}.$$

Thus $P'_v \geq (1 - e^{-1/6})/2$ and the expected number of edges removed is at least a fraction $(1 - e^{-1/6})/4$ of the total. □

4 Distributed Vertex Cover via Maximal Matchings

In this section we present a simple pseudo-polynomial reduction from the problem of computing a 2-approximate vertex cover to the problem of computing a maximal matching in an auxiliary graph. Thanks to this reduction, a 2-approximate vertex cover can be computed in $\mathrm{O}(\log n + \log \hat{W})$ expected rounds via algorithm $\mathcal{M}$ of section 3.

Consider the following auxiliary graph $\widetilde{G}$. For each node v of G, $\widetilde{G}$ contains w_v *micro-nodes* $v(1), v(2) \ldots v(w_v)$. Two micro-nodes $v(i)$ and $u(j)$ are adjacent if and only if vu is an edge of G. In Figure 1 an example of the reduction is given.

Let M be a maximal matching in $\widetilde{G}$. By $V(M)$ we denote the set of nodes v of G such that all the corresponding micro-nodes $v(i)$ are matched by M.

Lemma 2. *Set $V(M)$ is a 2-approximate vertex cover of G.*

Proof. Assume by contradiction that $V(M)$ is not a vertex cover. Thus there are two adjacent nodes v and u in G which do not belong to $V(M)$. This implies that there are two adjacent micro-nodes $v(i)$ and $u(j)$ in $\widetilde{G}$ which are not matched by M. Then the set $M' = M \cup \{v(i)u(j)\}$ is a matching, which contradicts the maximality of M.

Let `apx` and `opt` denote the weight of the vertex cover found and that of a minimum weight vertex cover, respectively. Moreover, let z_v be the number of micro-nodes in $\{v(1), v(2) \dots v(w_v)\}$ that are matched by M. A feasible solution of (D) is obtained by assigning to each dual variable y_{vu} the number of edges of the kind $v(i)u(j) \in M$. This solution is feasible since, for every $v \in V$: $\sum_{u \in N(v)} y_{vu} = z_v \leq w_v$. By weak duality we obtain $apx \leq \sum_{v \in V} z_v \leq 2 \sum_{vu \in E} y_{vu} \leq 2\,opt$ and hence $V(M)$ is 2-approximate. □

Lemma 2 suggests a strategy to compute a 2-approximate vertex cover distributively. The idea is to simulate the behavior of algorithm $\mathcal{M}$ on a virtual auxiliary graph $\widetilde{G}$, and then to select the nodes in the vertex cover as suggested by Lemma 2.

Specifically, each node simulates the execution of the algorithm on the corresponding micro-nodes $v(i)$ in $\widetilde{G}$. Whenever two micro-nodes $v(i)$ and $u(j)$ of $\widetilde{G}$ need to communicate, nodes v and u are responsible for allowing such communication. The vertex cover is given by the nodes v such that all the corresponding micro-nodes $v(i)$ are matched by the maximal matching computed. Since the virtual auxiliary graph contains $O(n\hat{W})$ nodes, the total number of rounds is $O(\log(n\hat{W})) = O(\log n + \log \hat{W})$.

This naive application of Lemma 2 has two major drawbacks. The first problem is the large message size. In fact, in each phase all the micro-nodes of v may send a proposal to some micro-node of u. Thus the message size is $\Omega(W)$.

A second problem is the time complexity of the algorithm: consider a node v in a given phase. Each micro-node $v(i)$ of v, with probability one half, needs to select one neighbor out of $\Theta(\Delta W)$ uniformly at random. This random selection can be performed in $\Theta(\log(\Delta W))$ expected time, assuming that the cost of generating a random bit is $O(1)$ (e.g., see [3]). Thus the expected time complexity of each phase is $\Omega(W \log(\Delta W))$.

In next section we show how to solve both problems by creating the matchings *implicitly*.

5 An Improved Algorithm

In this section we present an improved fully distributed algorithm $\mathcal{A}$ for computing a 2-approximate vertex cover. Algorithm $\mathcal{A}$ still requires $O(\log n + \log \hat{W})$ expected rounds, but it reduces the size of the messages to $O(\log W)$ and the expected time complexity of each phase to $O(\Delta \log(\Delta W))$.

The basic structure of algorithm $\mathcal{A}$ is analogous to the structure of the naive algorithm described in previous section: in each phase, a matching in the current auxiliary graph $\widetilde{G}'$ is computed, and the matched nodes are removed from $\widetilde{G}'$ (together with all the edges incident to them). The algorithm halts when no edge is left. The vertex cover is given by the nodes v such that all the corresponding micro-nodes $v(i)$ are matched by one of the matchings computed. The main novelty in Algorithm $\mathcal{A}$ is that matchings are

Figure 2 Protocol for node v for 2-approximate vertex cover.

```
$w'_v = w_v$; $N'(v) = N(v)$, $s_v$ = *active*;
**while** ($s_v$ = *active*) {
    **send** $w'_v$ and **receive** $w'_u$ **to**/ **from** all $u \in N'(v)$;
    $N'(v) = \{u \in N(v) : w'_u > 0\}$;
    **if** ($|N'(v)| = 0$)
        $s_v$ = *outside*;
    **else** {
        compute the proposals $p_v(u)$;
        **send** $p_v(u)$ and **receive** $p_u(v)$ **to**/ **from** all $u \in N'(v)$;
        compute the counter-proposals $c_v(u)$;
        **send** $c_v(u)$ and **receive** $c_u(v)$ **to**/ **from** all $u \in N'(v)$;
        **for** (all $u \in N'(v)$)
            $w'_v = w'_v - c_v(u) - c_u(v)$;
        **if** ($w'_v = 0$) {
            **send** $w'_v$ **to** all $u \in N'(v)$;
            $s_v$ = *inside*;
        }
    }
}
```

created *implicitly*: in each phase each node only knows the number of the corresponding matched micro-nodes. Intuitively, this simplification is allowed by the symmetry properties of $\widetilde{G}$: all the micro-nodes corresponding to a node v have the same degree and share the same neighborhood. This invariant is kept by all the induced subgraphs of $\widetilde{G}$.

Algorithm $\mathcal{A}$, which is described in Figure 2, works in phases. Each phase consists of a constant number of communication rounds. Each node v has an associated *state* s_v, which is initially *active*. In each phase, part of the *active* nodes switch to the state *inside* or *outside*, and the algorithm terminates when no *active* node is left. When a node leaves the *active* state, it halts. At the end of the algorithm, the *inside* nodes form a vertex cover.

In more details, each node v has an associated *residual weight* w'_v, which is initially w_v. The residual weight w'_v can be interpreted as the number of micro-nodes $v(i)$ of v in the current auxiliary graph $\widetilde{G}'$. Note that all the micro-nodes $v(i)$ have the same degree $W'_v = \sum_{u \in N(v)} w'_u$. In each phase, the expected residual weight of active nodes decreases. The decrease of w'_v in a given phase reflects the number of micro-nodes of v that have been matched in that phase.

At the beginning of each phase, each *active* node v sends w'_v to all its currently active neighbors $N'(v)$. The neighbors with $w'_u = 0$ are removed from $N'(v)$. If $N'(v)$ becomes empty, node v switches to the *outside* state. In fact, in this case the degree W'_v of the micro-nodes $v(i)$ is zero, and thus they will never be matched.

Otherwise, v sends a *proposal* $p_v(u)$ to each *active* neighbor $u \in N'(v)$. The value of $p_v(u)$ can be interpreted as the number of proposals directed from the micro-nodes of v to the micro-nodes of u. Let p'_v be the sum of the proposals $p_v(u)$:

$$p'_v = \sum_{u \in N'(v)} p_v(u).$$

This quantity can be viewed as the number of *micro-senders* among $v(1), v(2)\dots v(w_v)$. We postpone a detailed description of how proposals are fixed until later.

For each proposal $p_u(v)$ received, node v replies with a *counter-proposal* $c_v(u)$. The counter-proposal $c_v(u)$ can be interpreted as the number of micro-nodes of v which accept proposals of micro-nodes of u. Let $c'_v = w'_v - p'_v$ be the number of *micro-receivers* of v. The sum of the counter-proposals for node v then needs to be at most c'_v. At the same time each counter-proposal $c_v(u)$ must not exceed the corresponding proposal $p_u(v)$. Given these restrictions we choose a feasible set of counter-proposals $\{c_v(u)\}_{u\in N'(v)}$ arbitrarily such that their sum is maximum, i.e.

$$\sum_{u\in N'(v)} c_v(u) = \min\left\{c'_v, \sum_{u\in N'(v)} p_u(v)\right\}.$$

Eventually, node v decrements w'_v by the sum of all the counter-proposals $c_v(u)$ and $c_u(v)$ which have been sent and received by v, respectively: $w'_v = w'_v - \sum_{u\in N'(v)} (c_v(u) + c_u(v))$. This decrement reflects the number of micro-nodes of v which are matched in the considered phase.

If w'_v becomes zero, node v sends w'_v to all its neighbors (for the last time) and switches to the *inside* state (since all the corresponding micro-nodes are matched).

We now show how proposals are fixed by each node v. If the number w'_v of micro-nodes $v(i)$ is "sufficiently" small, the proposals $p_v(u)$ are fixed according to algorithm $\mathcal{M}$. Otherwise, they are fixed in a more efficient way, while keeping the same expected value. In more details, there are two different strategies, depending on whether $w'_v < 2\delta'_v$ or not, where $\delta'_v = |N'(v)|$ is the number of currently *active* neighbors of v. If $w'_v < 2\delta'_v$, the proposals $p_v(u)$ are initially set to zero. Then, for w'_v times, an active neighbor $u \in N'(v)$ is selected at random with probability proportional to w'_u, and the corresponding proposal $p_v(u)$ is incremented by one with probability one half. Note that each $p_v(u)$, considered separately, is the sum of w'_v i.i.d. Bernoulli variables $B(w'_u/(2W'_v))$:

$$p_v(u) = \sum_{i=1}^{w'_v} B\left(\frac{w'_u}{2W'_v}\right). \tag{1}$$

Otherwise ($w'_v \geq 2\delta'_v$), the value of each $p_v(u)$ is independently set to:

$$p_v(u) = \left\lfloor \frac{w'_v w'_u}{2W'_v} \right\rfloor + B\left(\frac{w'_v w'_u}{2W'_v} - \left\lfloor \frac{w'_v w'_u}{2W'_v} \right\rfloor\right). \tag{2}$$

Note that, in both cases, the sum p'_v of the proposals is upper bounded by w'_v. In the first case this is trivially true. In the second case, this is a consequence of the small value of δ'_v:

$$\sum_{u\in N'(v)} p_v(u) \leq \sum_{u\in N'(v)} \left(\frac{w'_v w'_u}{2W'_v} + 1\right) = \frac{w'_v}{2} + \delta'_v \leq w'_v.$$

Moreover, in both cases $E[p'_v] = w'_v/2$. The following technical property of proposals will be useful in later parts of the analysis.

Lemma 3. *For any two given nodes v and $u \in N'(v)$ we have*

$$E_{u,v} = E\left[\left(1 - \frac{1}{w'_v}\right)^{p_u(v)}\right] \leq \left(1 - \frac{1}{W'_u}\right)^{w'_u/4}.$$

Proof. If $w'_u < 2\delta'_u$, by Equation (1):

$$E_{u,v} = E\left[\left(1 - \frac{1}{w'_v}\right)^{\sum_{i=1}^{w'_u} B\left(\frac{w'_v}{2W'_u}\right)}\right] = \left(E\left[\left(1 - \frac{1}{w'_v}\right)^{B\left(\frac{w'_v}{2W'_u}\right)}\right]\right)^{w'_u} =$$

$$= \left(1 - \frac{w'_v}{2W'_u} + \frac{w'_v}{2W'_u}\left(1 - \frac{1}{w'_v}\right)\right)^{w'_u} \leq \left(1 - \frac{1}{W'_u}\right)^{w'_u/2}.$$

Consider now the case $w'_u \geq 2\delta'_u$. By Equation (2), if $w'_u w'_v/(2W'_u) \geq 1$:

$$E_{u,v} \leq E\left[\left(1 - \frac{1}{w'_v}\right)^{\lfloor w'_u w'_v/(2W'_u)\rfloor}\right] \leq \left(1 - \frac{1}{w'_v}\right)^{w'_u w'_v/(4W'_u)} \leq \left(1 - \frac{1}{W'_u}\right)^{w'_u/4}.$$

Otherwise ($w'_u w'_v/(2W'_u) < 1$):

$$E_{u,v} = E\left[\left(1 - \frac{1}{w'_v}\right)^{B\left(\frac{w'_u w'_v}{2W'_u}\right)}\right] = 1 - \frac{w'_u}{2W'_u} \leq \left(1 - \frac{1}{W'_u}\right)^{w'_u/2}.$$

□

Lemma 4. *Algorithm $\mathcal{A}$ computes a 2-approximate vertex cover.*

Proof. The algorithm halts. In fact, the residual weight of each *active* node decreases by at least one in each round with positive probability. It follows that the nodes which do not switch to the *outside* state, switch to the *inside* state in a finite expected number of rounds. Assume by contradiction that, at the end of the algorithm, the *inside* nodes do not form a vertex cover. This implies that there is an *outside* node v which has at least one *outside* neighbor. Let v switch to the state *outside* in phase p. At the beginning of phase $(p-1)$, all the neighbors of v are either *inside* or *active* nodes. Consider the *active* neighbors of v in phase $(p-1)$. These nodes are not active any more when phase p starts. But they cannot switch to the state *outside* in phase $(p-1)$, since their active degree is greater than zero in that phase. Thus they all switch to the state *inside*, which is a contradiction. Let z_v be the difference between w_v and the final residual weight w'_v. A feasible solution of (D) is obtained by assigning to each dual variable y_{vu} the sum of all the counter-proposals of the kind $c_v(u)$ and $c_u(v)$. Let `apx` and `opt` be the weight of the vertex cover found and that of a minimum vertex cover, respectively. By weak duality: $apx \leq \sum_{v\in V} z_v \leq 2\sum_{vu\in E} y_{vu} \leq 2\,opt$. Thus the vertex cover found is 2-approximate. □

Lemma 5. *Algorithm $\mathcal{A}$ sends messages of size* $\mathrm{O}(\log W)$. *Each phase of algorithm $\mathcal{A}$ has time complexity* $\mathrm{O}(\Delta \log(\Delta W))$ *in expectation.*

Proof. Both proposals and counter-proposals can be packed in messages of size $\mathrm{O}(\log W)$. The time complexity of each phase is upper bounded by the cost of computing the proposals. Computing the proposals is as expensive as selecting $\mathrm{O}(\Delta)$ times an element out of $\mathrm{O}(\Delta W)$ ones uniformly at random. Each random selection can be performed by generating $\mathrm{O}(\log(\Delta W))$ random bits in expectation. By assuming a $\mathrm{O}(1)$ cost for generating a random bit, the total expected cost of each phase is $\mathrm{O}(\Delta \log(\Delta W))$. □

Recall that a node is *good* if at least one third of its neighbors have degree smaller or equal than its own degree. Consider a node v in G. The degree of all the micro-nodes corresponding to v in $\widetilde{G}$ is $W_v = \sum_{u\in N(v)} w_u$. Thus a micro-node $v(i)$ is good if and only if:

$$\sum_{u\in N(v): W_u \le W_v} w_u \ge \frac{W_v}{3}.$$

Note that, if a micro-node $v(i)$ is good, all the micro-nodes $v(j),\ j \in \{1,2\dots w_v\}$, are good and vice-versa. We call a node of G *heavy* if all its micro-nodes in $\widetilde{G}$ are good. The next observation can be seen as the weighted analogue of Lemma 1.

Lemma 6. *Let $E_H \subseteq E$ be the subset of edges incident to heavy nodes. Then $\sum_{vu\in E_H} w_v w_u \ge \frac{1}{2}\sum_{vu\in E} w_v w_u$.*

Proof. Consider the auxiliary graph $\widetilde{G}$. The number of edges of $\widetilde{G}$ that are incident to good nodes is $\sum_{\{v,u\}\in E_H} w_v w_u$. Since the number of edges of $\widetilde{G}$ is $\sum_{\{v,u\}\in E} w_v w_u$, the claim follows from Lemma 1. □

We use the properties of heavy nodes to prove the following bound on the number of rounds.

Lemma 7. *Algorithm $\mathcal{A}$ halts in $\mathrm{O}(\log n + \log \hat{W})$ expected rounds.*

Proof. We show that the residual weight of heavy nodes decreases by at least a positive constant factor in expectation in each phase. It follows from Lemma 6 that the same holds for the potential function: $0 \le \sum_{vu\in E} w'_v w'_u < (n\hat{W})^2$, thus implying the claim.

Consider a heavy node v in a given phase. Let w''_v be the values of w'_v at the end of the phase. The residual weight of v decreases by at least the sum of the counter-proposals $c_v(u)$ sent by v:

$$\begin{aligned} w''_v \le w'_v - \sum_{u\in N'(v)} c_v(u) &= p'_v + c'_v \left(1 - \frac{1}{c'_v}\min\{c'_v, \sum_{u\in N'(v)} p_u(v)\}\right) \\ &= p'_v + (w'_v - p'_v)\left(1 - \frac{1}{c'_v}\min\{c'_v, \sum_{u\in N'(v)} p_u(v)\}\right). \end{aligned}$$

Note that:

$$\left(1 - \frac{1}{c'_v}\min\{c'_v, \sum_{u\in N'(v)} p_u(v)\}\right) \le \left(1 - \frac{1}{c'_v}\right)^{\sum_{u\in N'(v)} p_u(v)} \le \prod_{u\in N'(v)} \left(1 - \frac{1}{w'_v}\right)^{p_u(v)}.$$

Thus:

$$E[w''_v] \leq \frac{w'_v}{2} + \frac{w'_v}{2} \prod_{u \in N'(v)} E\left[\left(1 - \frac{1}{w'_v}\right)^{p_u(v)}\right],$$

where we used $E[p'_v] = w'_v/2$. By Lemma 3 and the definition of heavy nodes:

$$\prod_{u \in N'(v)} E\left[\left(1 - \frac{1}{w'_v}\right)^{p_u(v)}\right] \leq \prod_{u \in N'(v)} \left(1 - \frac{1}{W'_u}\right)^{\frac{w'_u}{4}}$$

$$\leq \prod_{\substack{u \in N'(v) \\ W'_u \leq W'_v}} \left(1 - \frac{1}{W'_v}\right)^{\frac{w'_u}{4}} \leq \left(1 - \frac{1}{W'_v}\right)^{\frac{W'_v}{12}}.$$

The right-hand side is at most $e^{-1/12}$ and it follows that $E[w''_v] \leq w'_v(1 + e^{-1/12})/2$. $\square$

Lemmas 4, 5, and 7 together imply Theorem 1.

References

1. R. Bar-Yehuda and S. Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.
2. V. Chvátal. *Linear programming*. Freeman, 1983.
3. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press and McGraw-Hill Book Company, 6th edition, 1992.
4. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, 1979.
5. M. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. In *ACM Symposium on the Theory of Computing*, pages 439–448, 23–25 1994.
6. E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, Oct. 2002.
7. M. Hańćkowiak, M. Karoński, and A. Panconesi. On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics*, 15(1):41–57, 2001.
8. J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, July 2001.
9. D. S. Hochbaum. Efficient bounds for the stable set, vertex cover, and set packing problems. *Discrete Applied Mathematics*, 6:243–254, 1983.
10. A. Israeli and A. Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22:77–80, 1986.
11. S. Khuller, U. Vishkin, and N. Young. A primal-dual parallel approximation technique applied to weighted set and vertex cover. *Journal of Algorithms*, 17(2):280–289, 1994.
12. B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22:115–123, 1985.
13. A. Panconesi and R. Rizzi. Some simple distributed algorithms for sparse networks. *DISTCOMP: Distributed Computing*, 14, 2001.
14. C. Papadimitriou and M. Yannakakis. Optimization, approximization and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.