Approximated Distributed Minimum Vertex Cover Algorithms for Bounded Degree Graphs

Yong Zhang[1.2], Francis Y.L. Chin[2⋆], and Hing-Fung Ting[2⋆⋆]

[1] College of Mathematics and Computer Science, Hebei University, China.
[2] Department of Computer Science, The University of Hong Kong, Hong Kong
{yzhang,chin,hfting}@cs.hku.hk

Abstract. In this paper, two distributed algorithms for the minimum vertex cover problem are given. In the unweighted case, we propose a 2.5-approximation algorithm with round complexity $O(\Delta)$, where Δ is the maximal degree of G, improving the previous 3-approximation result with the same round complexity $O(\Delta)$. For the weighted case, we give a 4-approximation algorithm with round complexity $O(\Delta)$.

1 Introduction

Minimum vertex cover (MVC in short) is a fundamental and classical problem in computer science. For a graph $G = (V, E)$, we say a subset $S \subseteq V$ is a *vertex cover* if any edge $e \in E$ is incident to at least one vertex in S. The problem of MVC is to find a vertex cover S such that the number of vertices in S is minimum. If each vertex $u \in V$ has a weight $w(u)$, this problem is turned to be the weighted version of minimum vertex cover (WMVC in short). The objective of the WMVC is to find a vertex cover $S \subseteq V$ such that the total weight of vertices in S is minimized, i.e., $\min w(S)$, where $w(S) = \sum_{u \in S} w(u)$.

In this paper, we study the distributed algorithms for both the unweighted and weighted versions of the minimum vertex cover problem. In the distributed model, each vertex is a server and can do some computation on its local information. Each vertex only knows its local information, i.e., the property of itself (e.g., weight) and which vertices are its neighbors. Each vertex can exchange information with its neighbors. We assume the exchange of information is done synchronously. At each round, a vertex can send/receive messages to/from its neighbors. In this model, each vertex does not know the topology of G and the number of vertices in V. The complexity of distributed algorithm is measured by the the number of rounds to achieve the expected result (e.g., a vertex cover).

Related Works:
Minimum vertex cover is a very well studied problem in theoretical computer science. Both the unweighted and weighted versions are NP-hard [3]. There are

⋆ Research supported by HK RGC grant HKU-7113/07E and the William M.W. Mong Engineering Research Fund
⋆⋆ Research supported by HK RGC grant HKU-7171/08E

no polynomial algorithm can achieve the minimum vertex cover unless $P = NP$, thus, most studies focus on the approximation algorithms for this problem. In 2005, Karakostas [6] gave a $(2 - \Theta(1/\sqrt{\log n}))$-approximation algorithm, which is the best known upper bound. The current lower bound of the approximation ratio is $10\sqrt{5} - 21 > 1.36$ [2].

Distributed computing for finding a vertex cover is also well studied during these years. In the unweighted version, a 2-approximation algorithm is a straightforward extension of the algorithm in [5] for finding the maximal matching in $O(\log^4 n)$ rounds. Note that all vertices in the maximal matching 2-approximate the minimum vertex cover. Recently, another 2-approximation algorithm was proposed in [1] with round complexity $O(\Delta^2)$, where Δ is the maximum degree of the graph. With less number of rounds, a 3-approximation $O(\Delta)$-rounds algorithm was given in [8]. For the weighted version, Khuller et al [7] gave a $(2 + \varepsilon)$-approximation in $O(\log n \log \frac{1}{\varepsilon})$ rounds. A randomized 2-approximation algorithm was given by Grandoni et al [4], the expected number of rounds is $O(\log n + \log \hat{W})$, where $\hat{W}$ is the average weight of all vertices.

Our Contributions:
In this paper, we study the distributed algorithms for both the unweighted and weighted versions of the minimum vertex cover problem. In the unweighted case, we improve the previous 3-approximation result and give a 2.5-approximation algorithm with the complexity $O(\Delta)$ (Section 2); in the weighted case, we give a 4-approximation algorithm with $O(\Delta)$ rounds (Section 3).

2 Algorithm for MVC

In this section, we give a distributed 2.5-approximation algorithm for finding the minimum vertex cover in graph $G = (V, E)$. The high level idea is described as follows. Find a subgraph $\hat{G} = (\hat{V}, \hat{E})$ of G, such that (1) the degree of each vertex in $\hat{V}$ is at least one and at most two, thus, each connected component in $\hat{G}$ is either a cycle or a path; and (2) each edge in G is incident to at least one vertex in $\hat{V}$, thus, $\hat{V}$ is a vertex cover of G. Let $\{P_1, P_2, ..., P_k\}$ denote a partition of the connected components in $\hat{G}$, and any two connected components in different partitions are disjoint. In each partition, define the *local approximation ratio* to be the ratio between the number of vertices in the partition and the number of vertices selected into the minimum vertex cover. In partition P_i, let p_i and o_i be the number of vertices selected into the vertex cover by our algorithm (all vertices in this partition) and by the optimum algorithm, respectively. Since the partition of connected components are disjoint each other, the size of the minimum vertex cover for G is at least $\sum_{i=1}^{k} o_i$. Thus, the approximation ratio of our algorithm is at most $\sum_{i=1}^{k} p_i / \sum_{i=1}^{k} o_i \leq \max p_i/o_i$.

Our algorithm has two phases: the *partition phase* and the *modification phase*. In the partition phase, similar to the above description, we find a subgraph

$G' = (V', E')$ of G such that each connected component in G' contains only one path or cycle. Consider a connected component P_i with p_i vertices,

- if P_i is a cycle
  P_i contains p_i edges, to cover these edges, any vertex cover, including the minimum vertex cover, must select at least $\lceil p_i/2 \rceil$ vertices from P_i. Since V' contains all vertices in P_i, the local approximation ratio for P_i w.r.t. V' is $p_i/\lceil p_i/2 \rceil$, which is at most 2.
- If P_i is a path
  P_i contains $p_i - 1$ edges, to cover these edges, any vertex cover must select at least $\lceil (p_i - 1)/2 \rceil$ vertices. Since V' contains all vertices in P_i, the local approximation ratio for P_i w.r.t. V' is $p_i/\lceil (p_i - 1)/2 \rceil$.

Combine the above two cases, the approximation ratio of such algorithm is at most $\max_i p_i/\lceil (p_i - 1)/2 \rceil$. In the worst case, consider a path with two edges (3 vertices), the minimum vertex cover may contain only one vertex, but the algorithm may select all these three vertices, the local approximation ratio is 3. Intuitively, the longer the path, the lower the ratio. Therefore, we have a modification phase for handling the length-2 paths in G'.

2.1 the partition phase

This part is similar to the algorithm in [8], and is the background of the modification phase, we describe this phase briefly.

- Each vertex sends requests to its neighbors one by one, until it receives an accept message, or all neighbors reject its requests.
- Each vertex must accept one incoming request if it has not received any requests before. After that, this vertex will reject all the other incoming requests.

From the above request-accept algorithm, each vertex accepts at most one neighbor's request, and receives at most one neighbor's accept message. We can construct the graph $G' = (V', E')$ as follows: V' contains such vertices that either accepts the request from one of its neighbors, or receives an accept message from one of its neighbors. An edge (u, v) is in E' if u accepts an request from v, or vise versa. From the definition of G', we can see that G' is a subgraph of G. Polishchuk and Suomela proved the following theorem in [8].

Theorem 1 *V' is a vertex cover of G and the size of V' is at most 3 times to the size of the minimum vertex cover. The round complexity of the above algorithm is $O(\Delta)$.*

Now we give a tight example for the partition phase. Consider the graph G is a combination of some length-2 paths, as shown in Figure 1. According to the above algorithm, the middle vertex of each length-2 path may accept the request from upper vertex, and the lower vertex may accept the request from middle vertex. Thus, all vertices are selected into the vertex cover, but the minimum vertex cover only contains the middle vertex of each path.

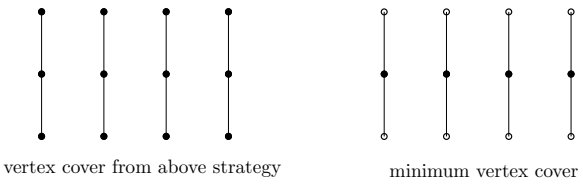

Fig. 1. Tight example of the partition phase (dark vertices are selected into the vertex cover).

2.2 the modification phase

From last subsection, if all vertices in V' are selected into the vertex cover, the approximation ratio is 3.

In $G' = (V', E')$, each vertex receives at most one accept message and accepts at most one of its neighbors' requests, the degree of each vertex in G' is at most 2. Thus, each connected component in G' is either a path or a cycle. Note that V' is a vertex cover of G, any edge in E contains at least one vertex in V'.

From the previous analysis, the approximation ratio after the partition phase is at most $p_i/\lceil (p_i-1)/2 \rceil$, which is at most 2.5 if $p_i \neq 3$. This gives us an heuristic to improve the approximation ratio by modifying the length-2 paths in G'.

In the modification phase, the components with lengths not equal to 2 remain (Step 1 of the modification phase); some length-2 paths are extended to length-3 paths (Step 2 of the modification phase); some terminals of some length-2 paths are removed from G' (Step 3 of the modification phase). After the modification phase, the selected vertices is still a vertex cover.

Now we describe the modification phase step by step.

Step 1: Select all vertices in some connected component with length 1, or more than 2 into V'', then remove all these vertices and edges incident to them from G.

Fact 1 *For any vertex in graph G', checking whether it is in a connected component of length not equal to 2 can be done in constant rounds.*

Proof. In G', each vertex knows its degree. The checking procedure starts from each vertex with degree one. For example, the degree of u is one, u sends a message $(u, 0)$ to its neighbor. When vertex v receives a message (u, i) from one of its neighbors, if v's degree is 2, v sends the message $(u, i+1)$ to its another neighbor; otherwise, v's degree is one, it sends the message $(u, i+1)$ back to its only neighbor.

For vertex u with degree one, if it receives a message $(u, 1)$, or it does not receive any feedback message after 4 rounds, u can confirm that it is not in a length-2 path. For any vertex with degree 2, if it receives messages $(u, 0)$ and $(u, 2)$, this vertex must be in a length-2 path; and must not otherwise. □

After Step 1, all the remaining components in G' are length-2 paths.

Step 2: Each terminal of the remaining (length-2) paths sends requests to its neighbors in G one by one.

- Vertex in V' rejects all requests.
- Vertex in $V - V'$ must accept one incoming request if it has not received any requests before. After that, this vertex will reject all other requests.
- Each terminal will not sends requests if it gets an accept message, i.e., each terminal can be accepted at most once.

If terminal u is accepted by vertex v, add (u, v) to this connected component, the length of this path is increased to be 3. Such component must be disjoint with other components because each vertex in $V - V'$ can accept at most one request. After such request-accept strategy,

- select all vertices in such components with extended length into V''.
- remove all these vertices and edges incident to them from G.

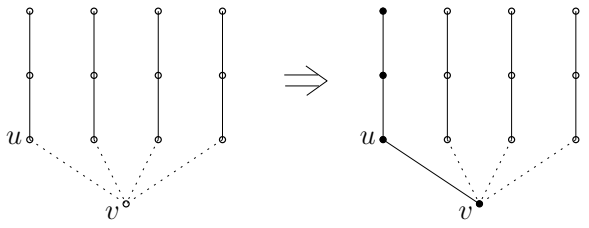

Fig. 2. example of step 2.

For example, consider the graphs as shown in Figure 2. The left part shows the structure before Step 2. In G, there are several edges connecting a vertex v and some terminals of length-2 paths in G', such that v does not belong to any component in G'. According to step 2 of this strategy, these terminals may send requests to v, and v must accept one of them. W.l.o.g., v accepts the request from u and rejects all the other requests. Thus, the length-2 path including vertex u can be extended to v. By such modification, the extended path is still disjoint with other components in G', and the length of this path is increased to be 3.

Since each terminal sends at most Δ requests, the total number of rounds in this step is at most $O(\Delta)$.

Step 3: Each terminal of the remaining (length-2) paths sends requests to its neighbors in G one by one.

- Each vertex in V'' and each non-terminal in the remaining length-2 paths must reject all requests.
- Each terminal in the remaining length-2 paths must accept one incoming request if it does not receive any requests before. After that, this vertex will reject all the other requests.

- Each terminal will not send any request if it gets an accept message, i.e., each terminal can be accepted at most once.

Select all vertices, except those terminals rejected by all its neighbors, into V''.

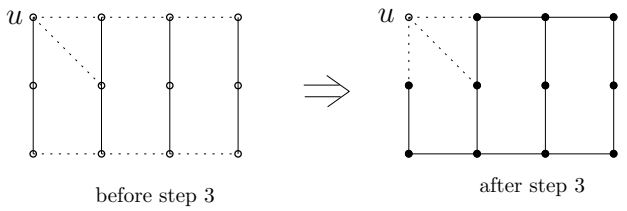

Fig. 3. example of step 3.

For example, consider the graphs as shown in Figure 3. The left part shows the structure before Step 3, there are several length-2 paths remained after step 2. After processing step 3, some terminals accept a request from one of its neighbors, or receive an accept message from one of its neighbor. Some terminals are rejected by all its neighbors, e.g., vertex u. In this example, we select all vertices except u since any edge adjacent to u must be adjacent to some selected vertex.

For each vertex pair (u, v), there exists an edge in G connecting them after step 3 if either u accepts a request from v or vise versa, as shown in Figure 3. Similar to the above analysis, the total number of rounds in this step is at most $O(\Delta)$.

Theorem 2 *The selected vertex set V'' is a vertex cover of G, the size of V'' is at most 2.5 times to the size of the minimum vertex cover. The round complexity of this algorithm is $O(\Delta)$.*

Proof. Consider an edge (u, v), from the partition phase, at least one vertex is selected into V', w.l.o.g., $u \in V'$. In the modification phase, if u is selected in step 1, 2, or 3, the edge (u, v) is covered. Otherwise, suppose u is not selected in any step. From the description of modification phase, u must be a terminal of a length-2 path. We claim that any vertex in $V - V'$ connected to u must be selected in step 2. Note that the vertices in $V - V'$ will not belong to any connected component after the partition phase. If there exists a vertex $v \in V - V'$ and (u, v) is an edge in G, u must have sent a request to v and v should have accepted this request if it does not receive any other requests. Since u is not selected, v must have rejected the request from u and thus v has been selected in step 2. In step 3, since u is not selected, all its requests must have been rejected. That means all its neighbors must have been selected in some step. By considering all cases, for any edge (u, v), either u or v must be contained in the vertex cover. Thus, V'' is a vertex cover.

Now let us consider the local approximation ratio for the partitions of the connected components step by step. In step 1, the first partition contains all components whose sizes are either 1 or strictly larger than 2. Thus, the local approximation ratio of each component in this partition is at most 2.5. In step 2, the length of each connected component with extended length is 3, thus, the local approximation ratios for these components are at most 2. Note that in step 3, some terminals of some length-2 paths are not selected, i.e., the lengths of these paths are reduced to be 1. For these paths with reduced length, the optimum algorithm must select at least one vertex, thus, the local approximation ratio for them are at most 2.

In step 3, consider the last partition includes all remaining length-2 paths. We select all vertices in these paths into the vertex cover. Since each terminal of such paths must either accept one request, or receive one accept message. There exist edges in G corresponding to every pair of request-accept, and such request-accept pair must be between some terminals of some length-2 paths. Therefore, only selecting the middle vertex of each path cannot cover all edges. Suppose there are k length-2 paths in the remaining length-2 paths. The minimum vertex cover contains one vertex in x paths, and contains at least two vertices in $k - x$ paths. Thus, the minimum vertex cover selects at least $x + 2(k - x) = 2k - x$ vertices, while our algorithm selects $3k$ vertices. We say two length-2 paths in this partition are adjacent if some terminals in these two paths form a request-accept pair, i.e., there exists an edge between the terminals of these two paths. For each path with 1 selected vertex in the minimum cover, each of the two terminals must connect to some terminal of its adjacent length-2 path, which must contain at least 2 selected vertices. For each path with at least 2 selected vertices in the minimum vertex cover, since these two selected vertices may be terminals and each terminal may be contained in two request-accept pairs, the number of adjacent length-2 paths is at most 4, which may contain one selected vertex in the minimum vertex cover. Thus, we have $x \leq 4(k - x)$, and in this partition, the local approximation ratio is at most $3k/(2k - x) \leq 2.5$.

By considering all these cases, the local approximation ratio for each component in the partition in each step is at most 2.5. Thus, the size of the selected vertex cover is at most 2.5 times to the size of the minimum vertex cover.

Now we analyze the complexity of the strategy. In the partition phase, the round complexity is $O(\Delta)$. Checking the length of a connected component is not 2 can be done in constant time. Since the degree of each vertex is at most Δ, step 1 can be done in $O(\Delta)$ rounds. From the description of step 2 and 3, the round complexity of these two steps is $O(\Delta)$ too. Therefore, the round complexity of our 2.5-approximation algorithm is $O(\Delta)$. □

3 Algorithm for WMVC

In this section, we propose a distributed algorithm to approximate the weighted vertex cover. The intuition idea can be regarded as a generalization of the distributed 3-approximation algorithm in [8], where each vertex may accept one

request or receive one accept message from one of its neighbors. In the weighted case, we still implement the request-accept strategy. For each vertex u, if it accepts the request from vertex v, the weight of u will be decreased. On the other hand, if vertex u receives an accept message, the weight will be decreased too. During the processing of the algorithm, the weights of vertices are decreased step by step. After the termination of the algorithm, we select vertex u into the vertex cover if $w(u) \leq 0$. At each step, each vertex may send requests or accept/reject messages to its neighbors. To specify the messages, each request from u is $(req, w(u))$; if u rejects the request from v, u will send $(rej, -1)$ to v; otherwise, each accept message is associated with a positive number, e.g., $(acc, x > 0)$. Let $w^o(u)$ be the original weight of vertex u, i.e., in the initial state, $w^o(u) = w(u)$.

Before describing the algorithm, we give a simple example to illustrate the decreasing of weights. Consider the graphs as shown in Figure 4, the left graph contains four vertices with weights 5, 1, 1, and 0.5 respectively. According to the request-accept strategy, the upper vertex sends request to the first lower vertex, and each lower vertex sends request to the upper vertex. All these four requests will be accepted and the weights of vertices will be decreased accordingly. The upper vertex accepts three requests and receives one accept message from vertices with weights 1 and 0.5, the weight of this vertex is decreased by 3.5. Similarly, the final weights of the lower vertices are -1, 0, and 0 respectively.

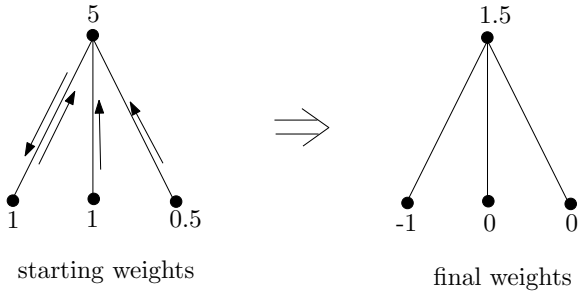

Fig. 4. Weights of vertices are decreased according to the algorithm.

Suppose step 1 is the first step and each round contains two continuous steps. Now we describe the algorithm on the i-th round for each vertex $u \in V$.

At step $2i - 1$:
u receives message x from its $(i-1)$-th neighbor v.

- if $x = (rej, -1)$, u is rejected by v;
- if $x = (acc, y > 0)$, u is accepted by v, then set $w(u) = w(u) - y$;

If $w(u) > 0$, u sends request message $(req, w(u))$ to its i-th neighbor.

At step $2i$:
u receives requests from v_1, v_2, ..., v_k,

- if $w(u) \geq \sum_{i=1}^{k} w(v_i)$, u sends accept message $(acc, w(v_i))$ to each v_i and set $w(u) = w(u) - \sum_{i=1}^{k} w(v_i)$;
- otherwise, assume $\sum_{i=1}^{\ell} w(v_i) \leq w(u)$ and $\sum_{i=1}^{\ell+1} w(v_i) > w(u)$, u sends accept message $(acc, w(v_i))$ to each v_i $(1 \leq i \leq \ell)$; sends accept message (acc, x) to $v_{\ell+1}$, where $x = w(u) - \sum_{i=1}^{\ell} w(v_i)$; sends reject message $(rej, -1)$ to $v_{\ell+2}$, ..., v_k; then set $w(u) = 0$.

Theorem 3 *After the termination of the algorithm, the vertex set $S = \{u \in V | w(u) \leq 0\}$ is a vertex cover.*

Proof. This is proved by contradiction. Assume that after the termination of the algorithm, there exists an edge (u, v) such that $w(u) > 0$ and $w(v) > 0$. From the above algorithm, since $w(u) > 0$, u must send a request to v at some time. If v accepts u's request, either u's or v's weight will be decreased to 0. Otherwise, v rejects u's request, that means v's weight is already decreased to be 0. Contradiction! □

To upper bound the total weights of selected vertices, we can assign weights to edges according to the above algorithm. Firstly, set the initial weights of all edges to be 0. If an accept message (acc, x) is sent through (u, v), set the edge weight $p_{(u,v)} = x$, where x is a positive value contained in the accept message. Note that in each round, there may be two accept messages sent through an edge (see the example in Figure 4), in this case, two positive values are associated with these two messages, set the edge weight to be the higher one. After processing the algorithm, each edge is assigned a non-negative weight. Now we analyze the total weight of edges incident to the vertex in S.

Consider vertex $u \in S$, suppose $w(u)$ is decreased to non-positive in the i-th round. From round 1 to round $i-1$, u may send accept messages to some neighbors, and receive some accept messages from some neighbors. At each round, u may send more than one accept messages but receives at most one accept message. All these accept messages may assign weights to the edges incident to u, and the total assigned weights is no more than the original weight $w^o(u)$. Let $w'(u)$ denote the decreased weight of vertex u in the first $i-1$ rounds, i.e., $w(u) = w^o(u) - w'(u)$ before the i-th round. Then in the i-th round, u may send some accept messages to some of its neighbors, or receives an accept message from one of its neighbors. From the rules of assigning weights on edges, the total assigned weights of those edges through which u sends accept message is at most $w^o(u) - w'(u)$; the weight of the edge through which u receives an accept message is at most $w^o(u) - w'(u)$ too. Note that at most two messages may be sent through an edge incident to u, we have the following lemma:

Lemma 2 *For each vertex $u \in S$, the total assigned weights on edges incident to u is at least $w^o(u)$ and at most $2 \cdot w^o(u)$.*

If we assign weight p_e to each edge $e \in E$ and satisfying $\sum_{(u,v)} p_{(u,v)} \leq w(u)$ for each $u \in V$, we have the following fact.

Fact 3 *$\sum_{e \in E} p_e \leq w(S)$ for any vertex cover S.*

Proof. $\sum_{e \in E} p_e \leq \sum_{u \in S} \sum_{(u,v)} p_{(u,v)} \leq \sum_{u \in S} w(u) = w(S)$. The first inequality holds because S is a vertex cover set. □

Let p_e denote the weight assigned to edge $e \in E$, let $p'_e = p_e/2$ for each $e \in E$, we have

$$w(S) = \sum_{u \in S} w^o(u) \leq \sum_{u \in S} \sum_{(u,v)} p_{(u,v)} = 2 \cdot \sum_{u \in S} \sum_{(u,v)} p'_{(u,v)}.$$

From Lemma 2, we know that for any $u \in V$, $\sum_{(u,v)} p'_{(u,v)} \leq w^o(u)$, then from Fact 3, the total weights of p'_e for all edges is no more than the total weights of any vertex cover, including the minimum vertex cover S^*. Thus,

$$w(S) \leq 2 \cdot \sum_{u \in S} \sum_{(u,v)} p'_{(u,v)} \leq 2 \cdot \sum_{u \in V} \sum_{(u,v)} p'_{(u,v)} = 4 \cdot \sum_{e \in E} p'_e \leq 4 \cdot w(S^*)$$

The third equality holds because each edge is incident to two vertices and counted twice. Therefore, we have the following conclusion:

Theorem 4 *Our algorithm 4-approximates the minimum weighted vertex cover with round complexity $O(\Delta)$.*

Proof. From previous analysis, we know the algorithm is 4-approximation. From the description of algorithm, the round complexity is $O(\Delta)$. □

References

1. Matti Åstrand, Patrik Floréen, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, Jara Uitto. A Local 2-Approximation Algorithm for the Vertex Cover Problem. In Proc. of the , 23rd International Symposium on Distributed Computing, DISC 2009, pp. 191-205.
2. I. Dinur, and S. Safra. The importance of being biased. In Proceedings of the ACM Symposium on the Theory of Computing (STOC 2002). ACM, New York, 33-42.
3. M. R. Garey, and D. S. Johnson. Computers and Intractability. A Guide to the Theory of NP-Completeness. Freeman.
4. Fabrizio Grandoni, Jochen Könemann, and Alessandro Panconesi. Distributed Weighted Vertex Cover via Maximal Matchings. ACM Transactions on Algorithms, Vol. 5, No. 1.
5. M. Hańćkowiak, M. Karoński, and A. Panconesi. On the distributed complexity of computing maximal matchings. SIAM J. Discrete Math. 15, 1, 41-57.
6. G. Karakostas. A better approximation ratio for the vertex cover problem. In Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP 2005). 1043-1050.
7. S. Khuller, U. Vishkin, and N. Young. Aprimal-dual parallel approximation technique applied to weighted set and vertex cover. J. Algorithms 17, 2, 280-289, 1994.
8. Valentin Polishchuk, Jukka Suomela, A simple local 3-approximation algorithm for vertex cover, Information Processing Letters 109(2009) pp. 642-645.