Approximation algorithms for the L-distance vertex cover problem

Chen Qiaoyun[1,a)] Zhao Liang[1,b)]

Abstract: Given a graph G and an integer $L \geq 0$, an *L-distance vertex cover* (L-VC) is a set U of vertices such that each edge can be reached from some vertex in U via at most L edges. The L-distance vertex cover problem asks to find an L-VC of the minimum size. This problem generalizes the well-known Vertex Cover problem with $L = 0$ and is known to be NP-hard for any fixed L. This paper proposes centralized and distributed approximation algorithm with guarantees $\Delta(\Delta - 1)^{\frac{L-1}{2}}$ for odd $L \geq 1$ and $2\Delta(\Delta - 1)^{\frac{L}{2}-1}$ for even $L \geq 2$ (or 2 for $L = 0$) respectively, where Δ is the maximum degree of vertices. The running time of the centralized version is $O(m+n)$, where m and n are the number of edges and the number of vertices respectively. On the other hand, the distributed version requires $O(L(\Delta^{L+1} + \log^* n))$ rounds (respectively $O(L(\Delta^{L+2} + \log^* n))$ rounds) for odd L (even L), where $\log^* n$ is the iterated logarithm of n.

1. Introduction

First we give some notations and definitions. Throughout the paper, if a set S consists of only one element s, we may denote S by a single s for ease of notation. Let $G = (V, E)$ be a connected and undirected graph with a set V of n vertices and a set E of m edges. We assume that G has no self-loop and it is simple. Then an edge $e = (u, v) \in E$ may also be treated as a set $e = \{u, v\}$ of vertices, and a set $M \subseteq E$ of edges may also be treated as a set $M = \bigcup_{e\in M} e$ of all the endpoints of the edges in M.

A *path* is a sequence $P = v_1, e_1, v_2, e_2, v_3, \ldots, v_k, e_k, v_{k+1}$ of vertices $v_1, v_2, v_3, \ldots, v_{k+1}$ and edges $e_i = (v_i, v_{i+1})$, $1 \leq i \leq k$, whose length $\ell(P) = k$ is defined by the number of edges in it. For any two subsets $S, T \subseteq V$, define their *distance* $d_G(S, T)$ to be the length of a shortest path such that starts from a vertex in S and ends at a vertex in T. By definition, we have $d_G(S, S) = 0$ and $d_G(S, T) = d_G(T, S)$. For any $S \subseteq V$, let

$$V_L(S) = \{v \in V \mid d_G(S, v) \leq L\} \subseteq V$$

denote the set of vertices that can be reached from (some vertex in) S via at most L edges (e.g., $V_0(S) = S$). And let

$$E_L(S) = \{e \in E \mid d_G(S, e) \leq L\} \subseteq E$$

denote the set of edges that can be reached from (some vertex in) S via at most L edges. We say S *L-covers* an edge e if $e \in E_L(S)$.

An *L-distance vertex cover* (L-VC) is a set $S \subseteq V$ such that all edges in E can be L-covered by S, i.e., $E_L(S) = E$. Given a graph G and an integer $L \geq 0$, the *L-distance vertex cover problem* (Problem L-VC) asks to find an L-VC of the minimum size.

1 Graduate School of Informatics, Kyoto University, Yoshida, Kyoto 606-8501, Japan
a) cqy@amp.i.kyoto-u.ac.jp
b) zhao.liang.7s@kyoto-u.ac.jp

This can be formulated as the following Integer Programming.

$$\text{(IP)} \quad \text{minimize} \ \textstyle\sum_{v\in V} x_v$$

$$\text{subject to} \quad \textstyle\sum_{v\in V_L(e)} x_v \geq 1, \quad \forall e \in E,$$

$$x_v \in \{0, 1\}, \quad \forall v \in V.$$

Problem L-VC arises from Internet link monitoring [5]. It also has applications in facility location problems in road networks and so on. Notice that the classical vertex cover problem is a special case of $L = 0$. For any fixed L, Problem L-VC reduces to a Set Cover problem (SC). In fact, it is NP-hard for any fixed $L \geq 0$ ([5]). Hence we consider approximation algorithms.

Since Problem L-VC can be reduced to SC, any (approximation) algorithm for SC can be employed (with an additional $O(mn)$ reduction time). In particular, the greedy algorithm [4] and the primal-dual algorithm [3] can be applied. This paper proposes better algorithms. Let Δ be the maximum degree of vertices. We show the results of centralized algorithms in **Table 1**.

Table 1 Comparison of centralized approximation algorithms (the detailed implementations for the first two are omitted in this abstract).

Guarantee	Time	Algorithm
$(L+1)\log\Delta$	$O(mn)$	Greedy [4]+reduction
$\frac{2((\Delta-1)^{L+1}-1)}{\Delta-2}$	$O(m+n)$	Primal-dual [3]+reduction
$\Delta(\Delta-1)^{\frac{L-1}{2}}$	$O(m+n)$	This study (for odd $L \geq 1$)
$2\Delta(\Delta-1)^{\frac{L}{2}-1}$	$O(m+n)$	This study (for even $L \geq 2$)

For distributed algorithms, we may employ the algorithm [1] for SC with additional L broadcasts in each round. The number of rounds is independent of n, but each round requires a huge amount of work (though independent of n). This study adopts a different approach, which requires much less work in each round but requires an $L\log^* n$-rounds coloring procedure of the graph,

Table 2 Comparison of distributed approximation algorithms, where the detailed implementation of the first algorithm is omitted here.

Guarantee	Rounds	Algorithm
$\frac{2((\Delta-1)^{L+1}-1)}{\Delta-2}$	$O(\Delta^{4L}L)$	Primal-dual [1]+reduction
$\Delta(\Delta-1)^{\frac{L-1}{2}}$	$O(L(\Delta^{L+1}+\log^* n))$	This study (for odd $L \geq 1$)
$2\Delta(\Delta-1)^{\frac{L}{2}-1}$	$O(L(\Delta^{L+2}+\log^* n))$	This study (for even $L \geq 2$)

where $\log^* n$ is the iterated logarithm of n which is small in real applications (e.g., $\log^* n \leq 5$ for all $n \leq 2^{65536} \approx 10^{19660}$). As a result, our approach also improves the approximation guarantees. See a summary in **Table 2**.

The rest of this paper is organized as follows. The centralized algorithm is given in Section 2 and its distributed version is shown in Section 3. Finally, Section 4 concludes with remarks.

2. A centralized algorithm for Problem *L*-VC

Call an edge set $M \subseteq E$ a *p-matching* if the distance between any two edges in M is at least p. For example, a normal matching is an 1-matching. A p-matching M is said *maximal* if no more edge can be added into M without violating the p-matching property. In other words, the distance between M and any edge $e \in E$ is $p-1$ or less. The next lemma is a direct observation from [3], hence its proof is omitted here.

Lemma 1. *If M is a maximal $(2L+1)$-matching, then the vertex set $C_L(M) = \bigcup_{e \in M} V_L(e)$ is a feasible L-VC and a $\max_{e \in E} |V_L(e)|$-approximation.* □

(Notice that $\max_{e \in E} |V_L(e)| \leq \frac{2((\Delta-1)^{L+1}-1)}{\Delta-2}$.)

In the following we show an improved algorithm. First let us suppose that $L = 2k+1$ ($k \geq 0$) is an odd number. Let M be a maximal L-matching. We choose *one arbitrary* endpoint for each edge $e \in M$. Let C denote the set of the selected endpoints.

Theorem 1. *Vertex set C is a feasible L-VC and a $\Delta(\Delta-1)^{\frac{L-1}{2}}$-approximation solution for Problem L-VC and an odd L.*

Proof. For all edges $e \in E$, since M is a maximal L-matching, we have $d_G(M,e) \leq L-1$. Therefore $d_G(C,e) \leq d_G(M,e)+1 \leq L$. Thus C is a feasible L-VC.

To show the approximation ratio, let opt_i denote the optimum value for Problem i-VC, $i \geq 0$. Notice that no vertex can k-cover two or more edges in M since the distance between any two edges in M is at least $L = 2k+1$. Hence $opt_k \geq |M| = |C|$. Thus to show the claimed guarantee, we only need to prove that $opt_k \leq \Delta(\Delta-1)^k opt_L$.

Let us consider an optimal L-VC V^*. We may finish the proof by showing how to construct a feasible k-VC W (hence $opt_k \leq |W|$) with at most $\Delta(\Delta-1)^k|V^*| = \Delta(\Delta-1)^k opt_L$ vertices.

Let $Q_{k+1}(v) = V_{k+1}(v) - V_k(v)$ be the set of vertices with distance exactly $k+1$ from a vertex v. It is clear that $|Q_{k+1}(v)| \leq \Delta(\Delta-1)^k$ since Δ is the maximum degree of vertices. Notice that $Q_{k+1}(v)$ can k-cover all edges $e \in E_L(v)$ with $d_G(v,e) \geq k+1$ but it may fail to k-cover some edges $e' \in E_L(v)$ with $d_G(v,e') \leq k$ (see an illustration in Fig. 1).

Let $W_v = Q_{k+1}(v)$ if $Q_{k+1}(v)$ can k-cover all the edges in $E_L(v)$, otherwise let $W_v = Q_{k+1}(v) \cup \{v\}$ (notice that v can k-cover all edges that $Q_{k+1}(v)$ cannot). In both cases, W_v can k-cover all edges that v can L-cover, and $|W_v| \leq \Delta(\Delta-1)^k$. Therefore $W = \bigcup_{v \in V^*} W_v$ is a feasible k-VC, and $|W| \leq \Delta(\Delta-1)^k|V^*| = \Delta(\Delta-1)^k opt_L$. This completes the proof. □

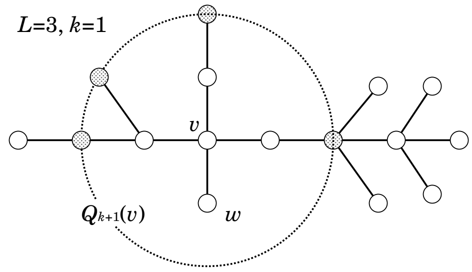

Fig. 1 An illustration to show that $Q_{k+1}(v)$ can k-cover all edges in $E_L(v) - E_k(v)$ but may fail to k-cover some edges in $E_k(v)$ (e.g., edge (v,w)).

On the other hand, if $L = 2k$ ($k \geq 0$) is an even number, let M be a maximal $(L+1)$-matching. We choose *all* the endpoints of edges in M. Let C' be the set of the selected vertices.

Theorem 2. *Vertex set C' is a feasible L-VC and a $2\Delta(\Delta-1)^{\frac{L}{2}-1}$-approximation (respectively 2-approximation) solution for Problem L-VC with an even $L \geq 2$ (respectively $L = 0$).*

Proof. C' is a feasible L-VC since $d_G(C',e) \leq 2k = L$ for any $e \in E$. The approximation guarantee can be proved in a similar way as we did for Theorem 1. We remark that choosing only one endpoint can only give a weaker guarantee in general for $\Delta \geq 3$, whereas the problem is trivial if $\Delta \leq 2$. □

We give a formal description of the above algorithm.

Algorithm 1 A centralized algorithm for Problem *L*-VC

Input: An graph $G(V,E)$ and an integer $L \geq 0$.
Output: An L-VC.

```
E' = E, M = ∅, K = 2⌊L/2⌋ + 1
while E' ≠ ∅ do
    Select an (arbitrary) edge e ∈ E'
    M = M ∪ {e}, E' = E' − E_K(e)
end while
Output the set of one arbitrary endpoint for each e ∈ M if L is odd,
  otherwise output the set of all endpoints of edges in M.
```

Theorem 3. *Algorithm 1 can be implemented in $O(m+n)$ time.*

Proof. The detail is omitted in this abstract. □

3. A distributed version of Algorithm 1

In this section, we extend our idea to the distributed setting, where the difficult part is to find a maximal p-matching ($p \geq 0$). We first describe the distributed computing model.

3.1 Model of a synchronized distributed system

We adopt the port-numbering model in [1]. In the system, each vertex v executes the same algorithm but against its local data. At the beginning, it knows a task-specific local input and it produces a local output. We always assume that a unique identifier, the degree and the neighbours are part of the local input for each vertex. The system operates in a synchronous manner. In each round each vertex v executes the following operations in that order.

(1) Performs local computation.
(2) Sends one message to each neighbour of it.
(3) Receives one message from each neighbour of it.

Finally each vertex $v \in V$ finishes its local computation and announces its local output. The size of a message is unbounded and we are interested in the number of required rounds.

3.2 A 3-phases distributed algorithm

Let $K = 2\lfloor \frac{L}{2} \rfloor + 1$ ($K = L$ for odd L and $K = L + 1$ for even L). We want to find an L-VC by constructing a maximal K-matching. Our algorithm consists of three phases: Phase I colors the graph; Phase II uses the coloring to construct a K-matching; and finally Phase III selects vertices. Let us describe them in the following.

Phase I: In this phase we construct a $(K + 1)$-distance coloring for G, i.e., any two vertices of distance $K + 1$ or less must be assigned different colors (e.g., a normal coloring is a 1-distance coloring). Let G^{K+1} denote the $(K + 1)^{th}$ power graph of G, i.e., the graph consisting of the same vertex set V but having an edge $e = (u, v)$ for each pair $\{u, v\}$ with $d_G(u, v) \leq K + 1$. Since a vertex can directly communicate with its neighbours in graph G, graph G^{K+1} can be obtained by $K + 1$ rounds of broadcasts with radius K. After that, each vertex knows its neighbours in G^{K+1}. Then we can construct a normal coloring for G^{K+1} by using the distributed algorithm of Barenboim et al. [2]. Obviously this coloring is a $(K + 1)$-distance coloring for G.

Phase II: We constructs a maximal K-matching M in Phase II. Let c_v denote the color of vertices v found in Phase I. We use x_v to denote if there is an incident edge of v that could be selected without violating the K-matching property ($x_v = 1$ means yes and $x_v = 0$ means no). Initially $x_v = 1$ for each $v \in V$. For each color $i = 1, 2, \ldots$, we repeat the following steps.

(1) For each vertex v, select an arbitrary incident edge $e = (v, w)$ of v with $c_v = i$ and $x_v = x_w = 1$. Do nothing if there is no such an edge.

(2) For each vertex v, if an edge $e = (v, w)$ was newly selected in (1), do a radius-$(K - 1)$ broadcast and set $x_u = 0$ for all vertices $u \in V_{K-1}(e)$.

Phase III: If L is odd, for each selected edge e, choose the endpoint of e with the smaller identifier; otherwise choose both the two endpoints of e.

Theorem 4. *The above* 3*-phases distributed algorithm correctly implement Algorithm 1 with* $O(L(\Delta^{L+1} + \log^* n))$ (*respectively* $O(L(\Delta^{L+2} + \log^* n))$) *rounds for odd* (*even*) *L.*

Proof. It is easy to see that a maximal matching can be found, hence the algorithm is correct. To estimate the required rounds, the algorithm of Barenboim et al. [2] colors a graph with maximum degree q in $O(q) + o.5 \log^* n$ rounds using $q + 1$ colors. Since for G^{K+1}, $q \leq \Delta + \Delta(\Delta - 1) + \cdots + \Delta(\Delta - 1)^K = O(\Delta^{K+1})$, and we need $O(L)$ broadcasts to communicate with neighbours in G in each round, Phase I takes $O(\Delta^{K+1})$ colors and $O(L(\Delta^{K+1} + \log^* n))$ rounds. Phase II takes $O(\Delta^{K+1} L)$ rounds, and Phase III takes one round. Therefore it takes $O(\Delta^{K+1} L)$ rounds in total. Since $K = 2\lfloor \frac{L}{2} \rfloor + 1$, the number of rounds is $O(L(\Delta^{L+1} + \log^* n))$ for odd L, and $O(L(\Delta^{L+2} + \log^* n))$ for even L. □

4. Summary

In this paper, we proposed novel approximation algorithms for Problem L-VC in both centralized and distributed setting by exploiting the graph structure. The approximation guarantees are $\Delta(\Delta - 1)^{\frac{L-1}{2}}$ for odd $L \geq 1$ and $2\Delta(\Delta - 1)^{\frac{L}{2}-1}$ for even $L \geq 2$ (or 2 for $L = 0$) respectively, where Δ is the maximum degree of vertices. The running time of the centralized version is $O(m + n)$, where m and n are the number of edges and the number of vertices respectively. On the other hand, the distributed version requires $O(L(\Delta^{L+1} + \log^* n))$ rounds (respectively $O(L(\Delta^{L+2} + \log^* n))$ rounds) for odd L (even L), where $\log^* n$ is the iterated logarithm of n. For a future work, we want to finish our work on the weighted case, which has been partially done at the point of writing this paper. Another interesting issue is to find a local algorithm whose number of rounds is independent of n.

References

[1] Astrand, M. and Suomela, J.: Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks, Proc. 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures, 191–205 (2010).

[2] Barenboim, L. and Elkin, M.: Distributed $(\Delta + 1)$-coloring in linear (in Δ) time, Proc. 41st Annual ACM Symposium on Theory of Computing, 111–120 (2009).

[3] Bar-Yehuda, R. and Even, S.: A linear time approximation algorithm for the weighted vertex cover problem, Journal of Algorithms, 198–203 (1981).

[4] Chvatal, V.: A greedy heuristic for the set-covering problem, Mathematics of Operations Research, 4 (3), 233–235 (1979).

[5] Sasaki, M., Zhao, L. and Nagamochi, H.: Security-aware beacon based network monitoring, Proc. IEEE ICCS 2008, 527–531 (2008).

