Complexity of Minimum Certificate Dispersal Problem with Tree Structure

Taisuke IZUMI [†1] Tomoko IZUMI [†2]
Hirotaka ONO [†3] Koichi WADA[†1]

Given an n-vertex graph $G = (V, E)$ and a set $R \subseteq \{\{x, y\} \mid x, y \in V\}$ of requests, we consider to assign a set of edges to each vertex in G so that for every request $\{u, v\}$ in R the union of the edge sets assigned to u and v contains a path from u to v. *The Minimum Certificate Dispersal Problem* (MCD) is defined as one to find an assignment that minimizes the sum of the cardinality of the edge set assigned to each vertex. This problem has been shown to be LOGAPX-complete for the most general setting, and APX-hard and 2-approximable in polynomial time for dense request sets, where R forms a clique. In this paper, we investigate the complexity of MCD with sparse (tree) structures. We first show that MCD is APX-hard when R is a tree, even a star. We then explore the problem from the viewpoint of the *maximum degree* Δ of the tree: MCD for tree request set with constant Δ is solvable in polynomial time, while that with $\Delta = \Omega(n)$ is 2.56-approximable in polynomial time but hard to approximate within 1.01 unless P=NP. As for the structure of G itself, we show that the problem can be solved in polynomial time if G is a tree.

1. Introduction

Background and Motivation. Let $G = (V, E)$ be a graph and $R \subseteq \{\{x, y\} \mid x, y \in V\}$ be a set of pairs of vertices, which represents requests about reachability between two vertices. For given G and R, we consider an assignment of a set of edges to each vertex in G. The assignment satisfies a request $\{u, v\}$ if the union of the edge sets assigned to u and v contains a path from u to v. The *Minimum Certificate Dispersal Problem (MCD)* is the one to find the assignment satisfying all requests in R that minimizes the sum of the cardinality of the edge set assigned to each vertex.

This problem is motivated by a requirement in public-key based security systems, which are known as a major technique for supporting secure communication in a distributed system [2, 5, 6, 8–10, 13, 14]. The main problem of the systems is to make each user's public key available to others in such a way that its authenticity is verifiable. One of well-known approaches to solve this problem is based on public-key certificates. A public-key certificate contains the public key of a user v encrypted by using the private key of another user u. If a user u knows the public key of another user v, user u can issue a certificate from u to v. Any user who knows the public key of u can use it to decrypt the certificate from u to v for obtaining the public key of v. All certificates issued by users in a network can be represented by a certificate graph: Each vertex corresponds to a user and each directed edge corresponds to a certificate. When a user w has communication request to send messages to a user v securely, w needs to know the public key of v to encrypt the messages with it. For satisfying a communication request from a vertex w to v, vertex w needs to get vertex v's public-key. To compute v's public-key, w uses a set of certificates stored in w and v in advance. Therefore, in a certificate graph, if a set of certificates stored in w and v contains a path from w to v, then the communication request from w to v is satisfied. In terms of cost to maintain certificates, the total number of certificates stored in all vertices must be minimized for satisfying all communication requests.

The previous work mainly focuses on directed variants of MCD, in which graph G is directed. Jung et al. discussed MCD with a restriction of available paths in [10] and proved that the problem is NP-hard. In their work, to assign edges to each vertex, only the restricted paths that are given for each request is allowed to be used. MCD with no restriction about available paths was first formulated in [14]. This variant is also proved to be NP-hard even if the input graph is a strongly connected directed graph. On the other hand, MCD for directed graphs with R forming a clique is polynomially solvable for bidirectional trees and rings, and Cartesian products of graphs such as meshes and hypercubes [14].

Based on these work, the (in)approximability of MCD for directed graphs has been

†1 Nagoya Institute of Technology
†2 Ritsumeikan University
†3 Kyushu University

studied from the viewpoint of the topological structure of R (not G) [9]. Since MCD is doubly structured (one is the graph G itself and the other is the request structure R), the hardness of MCD depends not only on the topology of G but also on the one of R. In view of these observation, the (in)approximability of MCD for directed graphs is investigated for general case and R forming a clique, as a typical community structure. It was shown that the former case is $O(\log |V|)$-approximable in polynomial time but has no polynomial time algorithm whose approximation factor is better than $0.2266 \log |V|$ unless P=NP. The latter case is 2-approximable but has no polynomial time algorithm whose approximation factor is better than 1.001, unless P=NP. In [9], the undirected variant of MCD is also considered, and 1.5-approximation algorithm for the case when R forming a clique is presented.

These results naturally raise a new question: For the hardness of approximation or constant-factor approximability, is such a dense structure (i.e., clique) essential? For example, how is the case when R is sparse, e.g., a tree? This paper further investigates the (in)approximability of MCD when R forms a tree, as another typical topology.

Our Contribution. We investigate the complexity of MCD with tree structure. Here, we say "with tree structure" in two senses. One is the case when R forms a tree, and the other is the case when G itself is a tree. One reason of this focus has already been mentioned above. Another reason is that a tree is a minimal connected structure; even if G (resp., R) is not a tree, by solving MCD for G', a spanning tree of G (resp., for a spanning tree R' of R), we can obtain an upper bound on the optimal solution (resp., a lower bound on the optimal solution) of the original MCD problem.

For MCD with tree R, we show that the hardness and approximability depend on the *maximum degree* Δ of tree R: MCD for tree R with constant degree is solvable in polynomial time while that with $\Omega(n)$ degree is APX-hard. As for MCD for tree G, we present a polynomial optimal algorithm. The followings are summary of our contributions:

- *R is an arbitrary tree*: First we consider MCD for the case when R is a *star*. Even in this simplest setting, MCD is shown to be APX-hard: MCD for undirected graph G with sparse R is still APX-hard. Moreover, the reduction *to* the Steiner tree problem for unweighted graphs(STREE) leads to an upper bound 1.28 on approximation ratio for MCD with star request sets. For arbitrary tree R, it is shown that there is a 2.56-approximate algorithm for MCD by utilizing the approximation algorithm for star R.
- *R is a tree with* $\Delta = O(\log |V|)$: By using a similar analysis to arbitrary tree R, the upper bound of approximation ratio for MCD can be reduced to 2. In particular, if R is a star with $\Delta = O(\log n)$ MCD is polynomially solvable.
- *R is a tree with constant degree*: This case is polynomially solvable. These imply that the hardness of MCD for tree R heavily depends on its maximum degree. A key idea is to define normal solutions. Our dynamic programming based algorithm searches not the whole solution space but (much smaller) normal solution space.
- *G is an arbitrary tree*: In this case also, a positive result is shown. For any request set R (not restricted to a tree), our algorithm outputs an optimal solution in polynomial time. The algorithm exploits the polynomial time solvability of VERTEX-COVER for bipartite graphs.

The remainder of the paper is organized as follows. In Section 2, we formally define the Minimum Certificate Dispersal Problem (MCD). Section 3 shows the hardness and approximability of MCD with star request sets, and Section 4 extends it to the approximability of MCD with tree request sets. In Section 5, we present a polynomial time algorithm that optimally solves MCD for tree request with constant degree. Section 6 shows an optimal algorithm for MCD with undirected tree graphs. Section 7 concludes the paper.

2. Minimum Certificate Dispersal Problem

While the Minimum Certificate Dispersal (MCD) Problem is originally defined for directed graphs, we deal with its undirected variant, where the given graph is undirected. The difference between them is the meaning of assignment an edge to a vertex: In the standard MCD, an edge (u, v) means a certificate from u to v. In the undirected variant of MCD, edge means a bidirectional certificate from u to v and v to u which is not separable. Since we treat the undirected variants of MCD throughout this paper, we simply refer those problems as MCD. In the following, we give the formal definition of MCD problem.

Let $G = (V, E)$ be an undirected graph, where V and E are the sets of vertices and edges in G, respectively. An edge in E connects two distinct vertices in V. The edge between vertex u and v is denoted by $\{u, v\}$. The numbers of vertices and edges in G are denoted by n and m, respectively (i.e., $n = |V|, m = |E|$). A sequence of edges $p(v_0, v_k) = \{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$ is called a *path* from v_0 to v_k of length k. A path $p(v_0, v_k)$ can be represented by a sequence of vertices $p(v_0, v_k) = (v_0, v_1, \ldots, v_k)$. For a path $p(v_0, v_k)$, v_0 and v_k are called the endpoints of the path. A shortest path from u to v is the one whose length is the minimum of all paths from u to v, and the distance from u to v is the length of a shortest path from u to v, denoted by $d(u, v)$.

A *dispersal* D of an undirected graph $G = (V, E)$ is a family of sets of edges indexed by V, that is, $D = \{D_v \subseteq E \mid v \in V\}$. We call D_v a local dispersal of v. A local dispersal D_v indicates the set of edges assigned to v. The *cost* of a dispersal D, denoted by $c(D)$, is the sum of the cardinalities of all local dispersals in D (i.e., $c(D) = \Sigma_{v \in V}|D_v|$). A request is a reachable unordered pair of vertices in G. For a request $\{u, v\}$, u and v are called the endpoints of the request. We say a dispersal D of G *satisfies* a set R of requests if a path between u and v is included in $D_u \cup D_v$ for any request $\{u, v\} \in R$. Given two dispersals D and D' of G, the union of two dispersals $\{D_v \cup D'_v \mid v \in V\}$ is denoted by $D \cup D'$.

The *Minimum Certificate Dispersal Problem (MCD)* is defined as follows:

Definition 2.1 (Minimum Certificate Dispersal Problem (MCD))
INPUT: An undirected graph $G = (V, E)$ and a set R of requests.
OUTPUT: A dispersal D of G satisfying R with minimum cost.

The minimum among costs of dispersals of G that satisfy R is denoted by $c_{min}(G, R)$. Let D^{Opt} be an optimal dispersal of G which satisfies R (i.e., D^{Opt} is one such that $c(D^{Opt}) = c_{min}(G, R)$).

Since R is a set of unordered pairs of V, it naturally defines an undirected graph $H_R = (V_R, E_R)$ where $V_R = \{u, v \mid \{u, v\} \in R\}$ and $E_R = R$. The request set R is called *tree* if H_R is a tree, and is also called *star* if it is a tree with exactly one internal vertex. The maximum degree of H_R is denoted by Δ_R. The problem of MCD restricting H_R to tree or star with degree Δ_R is called MCD-tree(Δ_R) and MCD-star(Δ_R). We also denote the problem of MCD restricting H_R to tree (or star) with degree $\Delta_R = O(f(n))$ for some function $f(n)$ as MCD-tree($O(f(n))$) (or MCD-star($O(f(n))$)). When we do not consider any constraint to the maximum degree, the argument Δ_R is omitted.

3. MCD for Star Request Sets

The NP-hardness and inapproximability of directed MCD for strongly-connected graphs are shown in the previous work[14]. In this section, we prove that MCD is APX-hard even if we assume that H_R is a star. The proof is by the reduction from/to the Steiner-tree problem.

Definition 3.1 (Steiner-tree Problem (STREE))
INPUT: An undirected connected graph $G = (V, E)$ and a set $T \subseteq V$ of terminals.
OUTPUT: A minimum-cardinality subset of edges $E' \subseteq E$ that connects all terminals in T.

We often use the notation STREE(t) and STREE($O(f(n))$), which are the Steiner-tree problems for a terminal set with cardinality at most t and $t = O(f(n))$ respectively.

Theorem 3.1
There exists a polynomial time ρ-approximation algorithm for MCD-star(Δ) if and only if there exists a polynomial time ρ-approximation algorithm for STREE($\Delta + 1$).

Proof.
We prove the only-if part and if part can be proved in almost the same way as the proof of the only-if part. Given an instance $(G = (V, E), T)$ of STREE($t + 1$), we construct an instance (G', R) of MCD-star(t) as $G = G'$ and $R = \{\{v_r, u\} \mid u \in T \setminus \{v_r\}\}$, where $t = \Delta_R$ and v_r is an arbitrary vertex in T. To prove the theorem, it suffices to show that any feasible solution of MCD (G', R) (resp. (G, T)) can be transformed to a feasible solution of (G, T) (resp. (G', R)) with no gain of solution cost. Then because (G', R) and (G, T) have the same optimal cost and thus any ρ-approximated solution of (G', R) induces an ρ-approximated solution of (G, T).

From MCD-star(Δ) to STREE($\Delta + 1$): *Given a feasible solution $D = \{D_v \mid v \in V\}$*

of (G', R), we can construct a feasible solution $S = \cup_{v \in V} D_v$ of STREE. Since S necessarily includes a path between any pair in R, its induced graph is connected and contains all vertices in $T = V_R$. Thus, S is a feasible solution for STREE and its cost is at most $\sum_{v \in V} |D_i|$.

From STREE($\Delta + 1$) to MCD-star(Δ): *Given a feasible solution S of (G, T), we obtain the solution of MCD-star by assigning all edges in $S (\subseteq E)$ to the internal vertex v_r of H_R. Since D_{v_r} connects all vertices in V_R, any request in R is satisfied. Thus $D = \{D_{v_r} = S\} \cup \{D_v = \emptyset \mid v \in V, v \neq v_r\}$ is a feasible solution of (G, R) and its cost is equal to $|S|$.*

Then since MCD-star(Δ) and STREE($\Delta+1$) have the same optimal cost, the theorem is proved. □

Since STREE is APX-hard [1] and its known upper and lower bounds for the approximation factor are 1.28 and 1.01, respectively [3, 12], we can obtain the following corollary.

Corollary 3.1

MCD-star is APX-hard, has a polynomial time 1.28-approximation algorithm, and has no polynomial time algorithm with an approximation factor less than 1.01 unless $P = NP$.

4. MCD for Tree Request Sets

4.1 Tree Structure with Arbitrary Degree

The general approximability of MCD-tree can be shown by the following theorem:

Theorem 4.1

Provided any ρ-approximation algorithm for MCD-star, there is a polynomial time 2ρ-approximation algorithm for MCD-tree.

We first introduce the construction of the algorithm: Given an instance $(G = (V, E), R)$ of MCD-tree, we regard H_R as a rooted tree by picking up an arbitrary vertex as its root. Letting $depth(v)$ $(v \in V_R)$ be the distance from the root to v on H_R, we partition the request set R into two disjoint subsets R^i $(i \in \{0, 1\})$ as $R^i = \{\{u, v\} \mid depth(u) < depth(v) \text{ and } depth(u) \bmod 2 = i\}$. Note that both R^1 and R^0 respectively form two forests where each connected component is a star. Thus, using any algorithm for MCD-star (denoted by $\mathcal{A}$), we can obtain two solutions of (G, R^1) and (G, R^0) by independently solving the problems associated with each connected component. Letting D^j be the solution of instance (G, R^j), the union $D^1 \cup D^0$ is the final solution of our algorithm.

It is obvious that the returned solution is feasible. Since both of $c(D^1)$ and $c(D^0)$ are the lower bound of the optimal cost for (G, R), the algorithm achieves approximation ratio 2ρ. In the following, we show the proof details of Theorem 4.1.

Proof.

Let $Opt(G, R)$ be an optimal solution of (G, R), and $\mathcal{A}(G, R)$ be the solution of (G, R) returned by algorithm $\mathcal{A}$. Installing ρ-approximation algorithm of MCD-star into $\mathcal{A}$, we can obtain ρ-approximated solutions of (G, R^1) and (G, R^0) because each connected component of V_R^1 and V_R^0 is a star (trivially, the set of ρ-approximated solutions corresponding to each connected components induces an ρ-approximated solution of the whole instance). Thus, we have $c(\mathcal{A}(G, R^j)) \leq \rho c(Opt(G, R^j))$ $(j \in \{0, 1\})$. Furthermore, since $R^j \subseteq R$ holds for any $j \in \{0, 1\}$, we also have $c(Opt(G, R^j)) \leq c(Opt(G, R))$. Letting S be the solution of (G, R) finally returned and $c_{\max} = \max\{c(Opt(G, R^1)), c(Opt(G, R^0))\}$, we finally obtain $c(S) \leq c(\mathcal{A}(G, R^1))) + c(\mathcal{A}(G, R^0)) \leq 2\rho c_{\max} \leq 2\rho Opt(G, R)$. The theorem is proved. □

The above theorem and Corollary 3.1 leads the following corollary:

Corollary 4.1

MCD-tree has a polynomial time 2.56-approximation algorithm.

4.2 Tree Structures with $O(\log n)$ Degree

In the proof of Theorem 4.1, we have shown that the approximated solution for instance (G, R) of MCD-tree can be constructed by solving several MCD-star instances. Thus, if $\Delta_R = O(\log n)$, each decomposed star has $O(\log n)$ vertices (that is, an instance of MCD-star($O(\log n)$)). By Theorem 3.1, MCD-star($O(\log n)$) and STREE($O(\log n)$)

have the same complexity and STREE($O(\log n)$) is optimally solved in polynomial time [4]. Therefore, Theorem4.1 leads the following corollary.

Corollary 4.2
There is an optimal algorithm to solve MCD-star($O(\log n)$) in polynomial time and there is an approximation factor 2 polynomial time algorithm for MCD-tree($O(\log n)$).

5. Tree Structures with Constant Degree

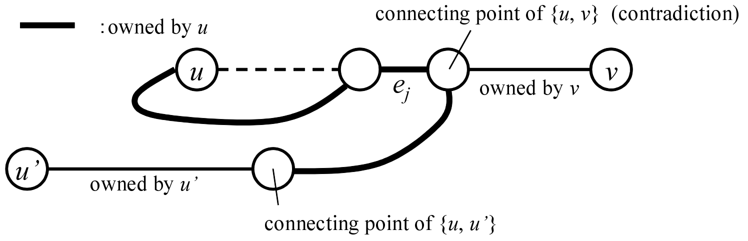
In this section, we provide an algorithm that returns the optimal dispersal for any instance of MCD-tree($O(1)$). Throughout this section, we regard H_R as a rooted tree by picking up an arbitrary vertex r in V_R as its root. Given a vertex $u \in V_R$, let $par(u)$ be the parent of u, and let $Child(u)$ be the set of u's children.

A request $\{u, v\}$ is *well-satisfied* by a feasible D if there exists a vertex $\alpha_{u,v}$ such that D_u contains a path from u to $\alpha_{u,v}$ and D_v contains a path from $\alpha_{u,v}$ to v. Then, vertex $\alpha_{u,v}$ is called the *connecting point* of request $\{u, v\}$ in D.

We begin with the following fundamental property:

Lemma 5.1
For any instance (G, R) of MCD-tree, there is an optimal solution that well-satisfies all requests in R.

Proof.
The proof is done in a constructive way. That is, we show that it is possible to transform any optimal solution to one well-satisfying all requests with no extra cost. Let D be an optimal solution, U be the set of vertices having at least one request not well-satisfied, and u be the vertex farthest from r in U. Since u is the farthest, only the request between u and its parent is not well-satisfied in all requests related to u. Let $v = par(u)$ for short. To prove the lemma, it suffices to show that we can obtain a solution D' where $c(D) = c(D')$ holds, any request well-satisfied in D is also done in D', and $\{u, v\}$ is well-satisfied. Let $e_0, e_1, \cdots e_k$ be the sequence of edges in G organizing a path from u to v. From the fact that $\{u, v\}$ is not well-satisfied, there exists an edge $e_j \in D_u$ such that $e_p \in D_v$ for some $p < j$ and $e_q \in D_v$ for any $q > j$. Since request $\{u, u'\}$ is


:owned by u
connecting point of $\{u, v\}$ (contradiction)
u
e_j
owned by v
v
u'
owned by u'
connecting point of $\{u, u'\}$

1 Illustration of the proof of Lemma 5.1: If $e_j \in D_u$ is used to satisfy request $\{u, u'\}$, D_u contains a path terminating with e_j because $\{u, u'\}$ is well-satisfied. It follows that request $\{u, v\}$ becomes well-satisfied, which is a contradiction.

well-satisfied for any $u' \in Child(u)$, there is a path $P_{u'}$ in D_u from u to $\alpha_{u,u'}$ in D_u. Then, for any $u' \in Child(u)$, each $P_{u'}$ does not contain e_j because $\{u, v\}$ becomes well-satisfied if $e_j \in P_{u'}$ holds for some u' (see Figure 1). Thus, we can construct a dispersal D' as $D'_x = D_x$ for any $x \neq u, v$, $D'_u = D_u \setminus \{e_j\}$ and $D'_v = D_v \cup \{e_j\}$, which is feasible and has the same cost as D. Repeating the construction, we can make request $\{u, v\}$ well-satisfied. Since this procedure does not break the well-satisfied property of any other request, we can eventually obtain a feasible solution well-satisfying all requests without extra cost. The lemma is proved. □

By the above lemma, we can reduce the search space to one where each feasible solution well-satisfies all requests. In the following argument, we assume that every request has a connecting point in the optimal dispersal. The principle of our algorithm is to determine the connecting points recursively from the leaf side of H_R via dynamic programming. Let $T_R(u) = (V_R(u), E_R(u))$ be the subtree of H_R rooted by u, $D^*(u, \alpha)$ be a dispersal for instance $(G, E_R(u))$ with the smallest cost such that D_u contains a path to from u to α. Note that $D^*(r, r)$ is an optimal solution of (G, R). We define $\gamma(u) = |Child(u)|$ for short. The key recurrence of the dynamic programming can be stated by the following lemma:

Lemma 5.2
Let u and α be vertices in V and let $A = (\alpha_1, ..., \alpha_{\gamma(u)}) \in V^{\gamma(u)}$. Then the following equality holds:

$$c(D^*(u,\alpha)) = \min_{A \in V^{|\gamma(u)|}} \{c(D^{Opt}(G, E_A \cup \{\{u,\alpha\}\})) + \sum_{u_k \in Child(u)} c(D^*(u_k, \alpha_k))\}$$

where $E_A = \{\{u,\alpha_1\},\{u,\alpha_2\},\cdots,\{u,\alpha_{\gamma(u)}\}\}$.

Proof.
Let α'_k *be the connecting point of* $\{u, u_k\}$ *in* $D^*(u,\alpha)$*, and* D^*_u *be u's local dispersal in* $D^*(u,\alpha)$*. To prove the lemma, it suffices to show that the right-hand expression is equal to the left for* $A = (\alpha'_1, \alpha'_2, \cdots, \alpha'_{\gamma(u)})$*. Since* D^*_u *has a path to any vertex* $\alpha'_k \in A$*, the edge-induced subgraph by* D^*_u *is one connecting all vertices in* $A \cup \{u,\alpha\}$*. That is, it is a feasible solution for instance* $(G, E_A \cup \{\{u,\alpha\}\})$*, and thus we have* $|D^*_u| \geq c(D^{Opt}(G, E_A \cup \{\{u,\alpha\}\}))$*. Combining the optimality of* $D^*(u_k, \alpha'_k)$ *for any* $u_k \in Child(u)$*, we can conclude that the right-hand is equal to the optimal cost* $c(D^*(u,\alpha))$*.* □

This recurrence naturally induces a polynomial time algorithm for MCD-tree($O(1)$). The pseudo-code of the algorithm is shown in Algorithm 1. The algorithm maintains a table D^*, where each entry $D^*[u][\alpha]$ stores the solution $D^*(u,\alpha)$. The core of the algorithm is to fill the table following the recurrence of Lemma 5.2: Assume an arbitrary ordering $\sigma = u_1, u_2, \cdots u_{|V_R|}$ of vertices in V_R where any vertex appears after all of its descendants have appeared. To compute the solution to be stored in $D^*[u_i][\alpha]$, the algorithm considers all possible choices of connecting points to u_i's children. Let $q_1, q_2, \cdots q_{\gamma(u_i)}$ be the children of u_i. Fixing a choice $A = (\alpha_{u_i,q_1}, \alpha_{u_i,q_2}, \cdots \alpha_{u_i,q_{\gamma(u_i)}})$ of connecting points (in the pseudo-code, α_k corresponds to α_{u_i,q_k}), the algorithm determines the local dispersal to u by computing the optimal solution for $(G, E_A \cup \{\{u_i,\alpha\}\})$. Note that this can be computed in polynomial time because the request set forms a constant-degree star. By Theorem 3.1, it is equivalent to STREE($O(1)$). Letting D' be the computed solution for $(G, E_A \cup \{\{u_i,\alpha\}\})$. we obtain $D = D' \cup D^*[q_1][\alpha_{u_i,q_1}] \cup D^*[q_2][\alpha_{u_i,q_2}] \cup \cdots \cup D^*[q_{\gamma(u)}][\alpha_{u_i,q_{\gamma(u)}}]$. Importantly, we can assume that only D'_u is nonempty in D' (recall the construction of MCD-star solutions from STREE solutions), which implies that D_{u_i} has a path to any connecting point α_{u_i,q_j} in A. Since it D_{q_j} has a path from q_j to α_{u_i,q_j} from the definition of

Algorithm 1 Polynomial Time Algorithm for MCD-tree($O(1)$)

```
1:  $D^*[V_R][V]$ : the array storing the computed solutions
2:     (All entries are initialized by a dummy solution with cost $\infty$)
3:  $\sigma = u_1, u_2, \cdots u_{|V_R|}$ : an ordering of $V_R$
4:     containing parent-child relationship on $H_R$ (children come earlier).

5:  for each $u_i \in V_R$ in order of $\sigma$ do
6:     Let $Q = (q_1, q_2, \cdots q_{\gamma(u_i)})$ be an arbitrary ordering of $Child(u_i)$
7:     for each $(A, \alpha) = ((\alpha_1, \alpha_2, \cdots, \alpha_{\gamma(u_i)}), \alpha) \in V^{\gamma(u_i)} \times V$ do
8:        $D' \leftarrow$ the optimal solution of $(G, E_A \cup \{\{u,\alpha\}\})$ s.t. only $D'_{u_i}$ is nonempty.
9:        /* $E_A = \{\{u_i,\alpha_1\},\{u_i,\alpha_2\},\cdots\{u_i,\alpha_{\gamma(u_i)}\}\}$ */
10:       $D \leftarrow D' \cup \left(\bigcup_{j \in [1,\gamma(u_i)]} D^*[q_j][\alpha_j]\right)$
11:       if $c(D^*[u_i][\alpha]) > c(D)$ then $D^*[u_i][\alpha] \leftarrow D$
12:    endfor
13: endfor
14: return $D^*[u_{|V_R|}][u_{|V_R|}]$
```

$^*[q_i][\alpha_{u_i,q_j}]$, $D_{u_i} \cup D_{q_j}$ necessarily has the path between u_i and q_j, That is the feasibility of D is guaranteed. If D is better than the solution already computed (for other choice of A), $D^*[u_i][\alpha]$ is updated by D. After the computation for all possible choices of A, $D^*[u_i][\alpha]$ stores the optimal solution. Finally, after filling all entries of the table, the algorithm returns $D^*[u_{|V_R|}][u_{|V_R|}]$, which is the optimal solution for instance (G, R).

Lemma 5.2 obviously derives the correctness of Algorithm 1. Since we assume that the maximum degree of tree H_R is a constant, the number of tuples of A is also a constant. Thus the number of possible choices about A is bounded by a polynomial of n. It follows that the running time of Algorithm 1 is bounded by a polynomial of n. We can have the following theorem:

Theorem 5.1
There is a polynomial time algorithm solving MCD-tree($O(1)$*).*

6. MCD for Tree Graphs

While the previous sections focus on the structure of H_R, in this section, we look at the structure of graph G: We show that MCD is solvable in polynomial time if G is a tree. In the algorithm, we compute for each edge $e \in E$ which D_u should contain e; for each $e \in E$, we decide $\{u \in V \mid e \in D_u\}$. For this decision about $e \in E$, we utilize a bipartite graph that represents whether a request $\{u, v\}$ should use e in its path.

Let $T = (V, E)$ be a tree and R be a request set. Now we consider to decide $\{u \in V \mid e \in D_u\}$ for an edge $e = \{u, v\} \in E$. By deleting $e = \{u, v\}$ from T, we obtain two subtrees $T_u = (V_u, E_u)$ and $T_v = (V_v, E_v)$ of T, where T_u and T_v contain u and v, respectively. Note that $V_u \cap V_v = \emptyset$ and $V = V_u \cup V_v$. From these two subtrees T_u and T_v, we construct a bipartite graph $B_{uv} = (V_u \cup V_v, E_{uv})$, where $E_{uv} = \{\{a, b\} \in R \mid a \in V_u, b \in V_v\}$. It should be noted that $\{e\}$ is an a-b cut for every $\{a, b\} \in E_{uv}$, since T is a tree. Thus, this bipartite graph represents that if an edge $\{w_i, w_j\} \in E_{uv}$, at least one of w_i or w_j should have $e = \{u, v\}$ in its local dispersal, i.e., $e \in D_u \cup D_v$, otherwise D does not satisfy request $\{w_i, w_j\}$ due to cut $\{e\}$.

This condition is interpreted as a vertex cover of B_{uv}. A *vertex cover* C of a graph is a set of vertices such that each edge in its edge set is incident to at least one vertex in C. Namely, a necessary condition of D satisfying R is that for each $e = \{u, v\}$, $C_{uv} = \{w \in V \mid e \in D_w\}$ is a vertex cover of B_{uv}. We call this *vertex cover condition*. It can be shown that the vertex cover condition is also sufficient for D to satisfy R. Suppose that a dispersal D satisfies the vertex cover condition. For a request $\{a_0, a_k\}$ and its unique path $p(a_0, a_k) = (a_0, a_1, \ldots, a_k)$ on T, by the definition of B_{uv}, every $B_{a_i a_{i+1}}$ contains edge $\{a_0, a_k\}$. By the vertex cover condition, $\{a_i, a_{i+1}\} \in D_{a_0} \cup D_{a_k}$ holds for $i = 0, \ldots, k-1$, which implies $D_{a_0} \cup D_{a_k}$ contains path $p(a_0, a_k)$; D satisfies request $\{a_0, a_k\}$.

By these arguments, the vertex cover condition is equivalent to the feasibility of D. Also it can be seen that choices of vertex cover of B_{uv} and another $B_{u'v'}$ are independent to each other in terms of the feasibility of D. These imply that the union of the minimum size of vertex cover for B_{uv}'s is an optimal solution of MCD for tree G.

From these, we obtain the following algorithm: For every edge $\{u, v\}$ in T, we first compute a minimum vertex cover C_{uv} of bipartite graph B_{uv}. Then, let $D_w = \{\{u, v\} \in E \mid w \in C_{uv}\}$ and output. Since VERTEX-COVER problem for bipartite graphs can be solved via the *maximum matching problem* [11], whose time complexity is $O(\sqrt{n}m)$ time, where n and m are the numbers of vertices and edges, respectively [7]. Thus, MCD for undirected tree G can be solved in $O(n^{1.5}|R|)$ time.

Theorem 6.1
For an undirected tree graph G and any request R, MCD is solvable in $O(n^{1.5}|R|)$ time.

7. Concluding remarks

We have considered undirected variants of the MCD problem with tree structures and shown that for MCD with tree R, the hardness and approximability depend on the maximum degree of tree R and MCD for any R can be solved in polynomial time if G is a tree.

There are interesting open problems as follows;

- The hardness of MCD-tree($O(\log n)$): Even NP-hardness of that class is not proved yet. Precisely, no hardness result is found for MCD-tree(Δ_R) where $\Delta_R = o(n)$ and $\Delta_R = \omega(1)$.
- The graph class of G allowing any request set R to be tractable: The case of trees (shown in this paper) is only the known class making the problem solvable in polynomial time. We would like to know what sparse graph classes (e.g., rings, series-parallel graphs, and planar graphs) can be solved for any request R in polynomial time. In particular, for MCD of rings with any request R we would like to decide whether it is NP-hard or P.
- Related to the question right above, we would like to extend the DP technique for MCD-tree($O(1)$) presented in Section 5 to other wider classes of H_R. Some sparse and degree-bounded graphs might be its candidates. In fact, the key of polynomial time running time of Algorithm 1 is based only on the following two conditions: (1) There exists an optimal solution that well-satisfies R, (2) There exists an ordering σ on V_R such that every cut $(\{\sigma(1), \ldots, \sigma(i)\}, \{\sigma(i+1), \ldots, \sigma(|V_R|)\})$ on H_R has a constant size.
- The complexity gap between undirected MCD and directed MCD: In general, directed MCD is not easier than undirected MCD in the sense that the latter is a special case of the former. But it is unknown whether it is proper or not. It is not quite trivial to transform any known complexity result for MCD into directed

MCD, and vice versa.

Acknowledgment

This work is supported in part by KAKENHI no. 22700010, 21500013, 21680001 and 22700017, and Foundation for the Fusion of Science and Technology (FOST).

References

1) M.Bern and P.Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, September 1989.
2) S.Capkun, L.Buttyan, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, March 2003.
3) M.Chlebík and J.Chlebíková. The steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207–214, October 2008.
4) S.E. Dreyfus and R.A. Wagner. The steiner problem in graphs. *Networks*, 1:195–207, 1972.
5) M.G. Gouda and E.Jung. Certificate dispersal in ad-hoc networks. In *in Proceeding of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 616–623, March 2004.
6) M.G. Gouda and E.Jung. Stabilizing certificate dispersal. In *in Proceeding of the 7th International Symposium on Self-Stabilizing Systems (SSS'05)*, pages 140–152, October 2005.
7) J.E. Hopcroft and R.M. Karp. An $n^{2.5}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.
8) J.Hubaux, L.Buttyan, and S.Capkun. The quest for security in mobile ad hoc networks. In *in Proceeding of the 2nd ACM international symposium on Mobile ad hoc networking and computing (Mobihoc'01)*, pages 146–155, October 2001.
9) T.Izumi, T.Izumi, H.Ono, and K.Wada. Approximability and inapproximability of the minimum certificate dispersal problem. *Theoretical Computer Science*, 411(31-33):2773–2783, June 2010.
10) E.Jung, E.S. Elmallah, and M.G. Gouda. Optimal dispersal of certificate chains. In *in Proceeding of the 18th International Symposium on Distributed Computing (DISC'04)*, pages 435–449, October 2004.
11) D.Kónig. Graphs and matrices. *Matematikai és Fizikai Lapok*, 38:116–119, 1931. in Hungarian.
12) G.Robin and A.Zelikovsky. Improved steiner tree approximation in graphs. In *in Proceedings of the 11th annual ACM-SIAM Symposium on Discrete Algorithms (SODA'00)*, pages 770–779, 2000.
13) H.Zheng, S.Omura, J.Uchida, and K.Wada. An optimal certificate dispersal algorithm for mobile ad hoc networks. *IEICE Transactions on Fundamentals*, E88-A(5):1258–1266, May 2005.
14) H.Zheng, S.Omura, and K.Wada. An approximation algorithm for minimum certificate dispersal problems. *IEICE Transactions on Fundamentals*, E89-A(2):551–558, February 2006.