A Self-Stabilizing Distributed Approximation Algorithm for the Minimum Connected Dominating Set

Sayaka Kamei[1] and Hirotsugu Kakugawa[2]

[1]Tottori University of Environmental Studies
Dept. of Information Systems
1-1-1 Wakabadai-kita, Tottori, Japan
s-kamei@kankyo-u.ac.jp

[2]Osaka University
Dept. of Computer Science
1-3 Machikaneyama, Toyonaka, Osaka, Japan
kakugawa@ist.osaka-u.ac.jp

Abstract

Self-stabilization is a theoretical framework of nonmasking fault-tolerant distributed algorithms. A self-stabilizing system tolerates any kind and any finite number of transient faults, such as message loss, memory corruption, and topology change. Because such transient faults occur so frequently in mobile ad hoc networks, distributed algorithms on them should tolerate such events. In this paper, we propose a self-stabilizing distributed approximation algorithm for the minimum connected dominating set, which can be used, for example, as a virtual backbone or routing in mobile ad hoc networks. The size of the solution by our algorithm is at most $8|D_{opt}|+1$*, where* D_{opt} *is a minimum connected dominating set. The time complexity is* $O(n^2)$ *steps.*

1. Introduction

1.1. Self-stabilization

A distributed system is a set of processes and a set of communication links between the processes. The distributed systems are subject to fail by their very nature. Therefore, *fault-tolerance* is a major concern in the study of distributed computing.

Self-stabilization [5] is a theoretical framework of nonmasking fault-tolerant distributed algorithms proposed by Dijkstra in 1974. Self-stabilizing algorithms can start execution from an arbitrary (illegitimate) system configuration, and eventually reach a legitimate configuration. That is to say, they do not need all initializations of states of each process and each link. By this property, they tolerate any kind and any finite number of transient faults, such as message loss, memory corruption, and topology change, and can adjust to dynamic changes of the system [6]. In addition, a self-stabilizing system does not need a global and synchronized initialization of each process at all when a system is started, because we can consider that such a starting configuration is a configuration just after transient faults occur. However, by these properties, design of a self-stabilizing algorithm is difficult.

1-4244-0910-1/07/$20.00 ©2007 IEEE.

1.2. Connected dominating set

In 1979, Sampathkumar et al. introduced the *connected dominating set* (CDS) problem in graph theory [13]. Let $G=(V,E)$ be an undirected connected graph, where V is a set of nodes and E is a set of edges. A dominating set of G is a subset $D \subseteq V$ such that each member of $V \setminus D$ is adjacent to at least one member of D. A set $D' \subseteq V$ is a CDS of G if and only if D' is a dominating set and the induced subgraph by D' is connected. The minimum CDS problem is finding a CDS of the minimum size.

A CDS is useful in the computation of message routing and other network problems for mobile ad hoc networks. Ad hoc networks are collection of wireless mobile nodes, and have no physical backbone infrastructure and no centralized administration. Therefore, a CDS formed by processes can be used for virtual backbone, which plays an important role for routing and connectivity management etc. However, it is known that this problem is NP-hard [4] in unit disk graph model that is one of models of mobile ad hoc networks. In a unit disk graph, there is a link between two nodes if and only if their geographical distance is at most one unit. In mobile ad hoc networks, message loss and topology change occur frequently. Thus, distributed algorithms for mobile ad hoc networks should tolerate such events.

We propose a self-stabilizing distributed approximation algorithm to compute a minimum CDS of an underlying network G assuming that a BFS (i.e., breadth-first spanning) tree T of G rooted at a process P_r is given. An approximation algorithm for the minimization problem is an algorithm which guarantees the approximation ratio D_{alg}/D_{opt}, where D_{alg} is the size of the solution of the approximation algorithm in the worst case and D_{opt} is the size of the optimal solution.

1.3. Related works

Because a CDS can be used for virtual backbone for routing messages in ad hoc networks, many algorithms for the CDS have been proposed. The literature [2] is a good survey for this problem in ad hoc networks.

In [8], Guha et al. propose a centralized approximation algorithm for the minimum CDS in unit disk graphs, and its approximation ratio is $O(H(\Delta))$, where Δ is the maximum degree and $H()$ is the harmonic function. In [12], Marathe et al. propose a heuristic approximation algorithm for the minimum CDS in unit disk graphs, and the size of the solution by their algorithm is at most $8|D_{opt}|+1$. .

In [1], Bharghavan et al. propose a distributed approximation algorithm based on the centralized algorithm in [8]. Its approximation ratio is $O(H(\Delta))$, and both its message complexity and time complexity are $O(n^2)$. In [17], Wu et al. propose a different distributed approximation algorithm in unit disk graphs. Its approximation ratio is $O(n)$, and its message complexity and time complexity are $\Theta(m)$ and $O(n^3)$, respectively, where m is the number of the edges in the unit-disk graph.

There exist some distributed approximation algorithms each of which approximation ratio is a constant. In [3], Cheng et al. propose a distributed approximation algorithm in unit disk graphs. The approximation ratio is 8, respectively. Message complexity and time complexity of the both algorithms are $O(n)$ and $O(n\Delta)$, respectively. In [16], Wan et al. propose a distributed approximation in unit disk graphs. The approximation ratio is 8, and message complexity and time complexity are $O(n\log n)$ and $O(n)$, respectively. In[18], Wu et al. propose a distributed approximation with approximation ratio $O(1)$ in unit disk graphs. In [7], Gao et al. also propose a distributed approximation in unit disk graphs. Its approximation ratio is $O(1)$, and both its message complexity and time complexity are $O(n)$. However, it is not easy to convert these algorithms into self-stabilizing distributed algorithms. Unfortunately, none of these algorithms are self-stabilizing.

In [11], Jain et al. propose a self-stabilizing algorithm for finding a CDS in $O(n^2)$ steps. However, their algorithm is not approximation algorithm, i.e., their algorithm does not guarantee its approximation ratio.

Therefore, our algorithm is the first self-stabilizing distributed approximation algorithm for the minimum CDS. Because our algorithm is based on the algorithm in [12] in unit disk graphs, the size of the solution by proposed algorithm is at most $8|D_{opt}|+1$, and the time complexity is $O(n^2)$ steps.

1.4. Organization of this paper

This paper is organized as follows. In section 2, we formally describe system model and the self-stabilizing distributed minimum CDS problem. In section 3, we present an outline of a heuristic algorithm of Marathe et al. [12] on which our algorithm is based. In section 4, we propose a self-stabilizing approximation algorithm for the minimum CDS. We assume that a BFS tree is given. In section 5, we show proof of correctness of the proposed algorithm, and show performance analysis. In section 6, we give conclusion of this paper and discuss future works.

2. Preliminary

2.1. System model

Let $V=(P_1,P_2,...,P_n)$ be a set of processes (nodes) and $E\subseteq V\times V$ be a set of bidirectional communication links in a distributed system. The number of processes is denoted by n. Then, the topology of the distributed system is represented as an undirected graph $G=(V,E)$. We assume that the graph is connected and simple. In this paper, we use "graphs" and "distributed systems", interchangeably. By N_i, we denote a set of neighbor processes of P_i. For each process P_i, process identifier and a set N_i are given as constants. Let the *distance* between P_i and P_j be the number of the edges on the shortest path between them.

As communication model, we assume that each process can read local state of neighbor processes without delay. This model is called the *state reading model*. Each process can update its own local state only, but each process can read local state of neighbor processes.

A set of local variables defines local state of a process. By Q_i, we denote local state of each process $P_i\in V$. A tuple of local state of each process $(Q_1,Q_2,...,Q_n)$ forms a *configuration* of a distributed system. Let Γ be a set of all configurations.

An algorithm of each process P_i is given as a set of guarded commands:

$$*[\ \mathrm{Grd}_1\rightarrow \mathrm{Act}_1\ \square\ \mathrm{Grd}_2\rightarrow \mathrm{Act}_2\ \cdots\ ]$$

Each Grd_j $(j=1,2,...)$ is called a *guard*, and it is a predicate on P_i's local state and local states of its neighbors. Each

Marathe et al. proved the approximation ratio of their algorithm was 10 in [12]. However, we proof that later.

Act_j is called an *action*, and it updates local state of P_i; next local state is computed from current local states of P_i and its neighbors. We say that P_i is *privileged* in a configuration γ if and only if at least one guard of P_i is true in γ. An atomic step of each process P_i consists of the following three substeps: (1) read local states of neighbor processes and evaluate guards, (2) execute a command that is associated to a true guard, and (3) update its local state.

Executions of processes are scheduled by an external (virtual) scheduler. A scheduler decides which process to execute in the next step. At each step, a scheduler selects only one privileged process arbitrarily, and a selected process executes an atomic step. This type of scheduler is known as the *central daemon*. A scheduler is *fair* if privileged process is eventually selected to execute by a scheduler, otherwise, it is *unfair*. We assume the unfair centralized daemon in this paper. Because a privileged process may not be executed forever under an unfair scheduler, an algorithm must be correct with respect to every execution scheduling. Thus, a scheduler is an adversary against an algorithm.

2.2. Self-stabilization

For any configuration γ, let γ' be any configuration that follows γ. Then, we denote this transition relation by $\gamma \rightarrow \gamma'$. For any configuration γ_0, a *computation* E starting from γ_0 is a maximal (possibly infinite) sequence of configurations $E = \gamma_0, \gamma_1, \gamma_2, \ldots$ such that $\gamma_t \rightarrow \gamma_{t+1}$ for each $t \geq 0$.

Definition 1 (Self-Stabilization) *Let Γ be a set of all configurations. A system S is* self-stabilizing *with respect to Λ such that $\Lambda \subseteq \Gamma$ if and only if it satisfies the following two conditions:*

- *Convergence: Starting from an arbitrary configuration, a configuration eventually becomes one in Λ, and*
- *Closure: For any configuration $\lambda \in \Lambda$, any configuration γ that follows λ is also in Λ as long as the system does not fail.*

Each $\gamma \in \Lambda$ is called a legitimate *configuration.* □

2.3. Formal definition of the problems

Definition 2 *A* dominating set *of a graph $G = (V,E)$ is a subset $V' \subseteq V$ such that there exists $v \in V'$ and $(u,v) \in E$ for any $u \in V \backslash V'$. A dominating set V' of G is* minimal *if no proper subset of V' is a dominating set of G. A dominating set V' of G is* minimum *if $|V'| \leq |V''|$ for any dominating set V'' of G.* □

Definition 3 *An* independent set *of a graph $G = (V,E)$ is a subset $V' \subseteq V$ such that $(u,v) \notin E$ for any $u,v \in V'$. An independent set V' of G is* maximal *if no proper superset of V' is an independent set of G.* □

We call maximal independent sets MISs. It is clear that a subset of V is a minimal dominating set if it is an MIS.

Definition 4 *A* connected dominating set *of a graph $G = (V,E)$ is a dominating set $V' \subseteq V$ such that an induced subgraph by V' is connected. A connected dominating set V' of G is* minimum *if $|V'| \leq |V''|$ for any connected dominating set V'' of G.*

We call the members of the CDS *dominators*, and others *dominatees*. Each dominatee is *dominated* by a dominator.

We consider solving the minimum CDS problem in distributed systems in this paper. We assume that each process P_i does not know global information of the network, and they know local information N_i which is a set of neighbors of P_i. Under such assumption, we defined the distributed minimum CDS problem as follows.

Definition 5 *Let $G = (V,E)$ be a graph that represents a distributed system, let $dom(P_i)$ be a variable that represents whether P_i is in a minimum connected dominating set. The* distributed minimum connected dominating set problem *is a problem defined as follows.*

- *Each process $P_i \in V$ must decide its decision $dom(P_i) \in \{\textbf{true}, \textbf{false}\}$ as output of P_i, and*
- *The set $\{P_i \in V \mid dom(P_i) = \textbf{true}\}$ is a minimum connected dominating set of G.* □

We assume that each process P_i has a local variable $dom(P_i)$.

3. Marathe et al.'s algorithm

Marathe et al. proposed a sequential heuristic algorithm for the minimum CDS in unit disk graphs [12], and we present outline of their algorithm.

The outline is described more formally in Figure 1. By $G(C)$, we denote an induced subgraph of G by a subset C of V.

Figure 2 illustrates an execution example of Marathe et al.'s algorithm. First, their algorithm selects an arbitrary node v from G, and constructs a BFS tree T of G rooted at v. Figure 2(a) illustrates a BFS tree. For any node v_i, let $L(v_i)$ denote the distance from v to v_i. Let k denote the depth of T (i.e., the maximum distance) in G, and let L_d be the set of nodes which have the distance d from the root ($0 \leq d \leq k$). In Figure 2(a)-Figure 2(d), the number in each node v_i is the

1 Arbitrarily pick a node $v \in V$.
2 Construct a BFS tree T of G rooted at v.
3 Let k be the depth of T.
4 For each $0 \le d \le k$, let L_d denote the set of nodes at distance d from the root in T.
5 Set $\mathrm{I}_0 := \{v\}$; $\mathrm{S}_0 := \emptyset$.
6 **for** $d = 1$ **to** k **do begin**
7 $\quad \mathrm{D}_d := \{u \mid u \in L_d$ and u is dominated by some node in $\mathrm{I}_{d-1}\}$.
8 $\quad$ Pick an MIS I_d in $G(L_d \setminus \mathrm{D}_d)$.
9 $\quad \mathrm{S}_d := \{u_f \mid u_f$ is the father in T of some $v_i \in \mathrm{I}_d\}$.
10 **end**
11 **output** $(\cup_{d=0}^{k} \mathrm{I}_d) \cup (\cup_{d=0}^{k} \mathrm{S}_d)$ as the CDS.

Figure 1. Marathe et al.'s Algorithm

distance $L(v_i)$ from the root on the BFS tree, and each arrow represents a father of the node. Figure 2(b)(resp. 2(c), 2(d)) illustrates the configuration after execution of a for-loop of Figure 1 with $d = 1$ (resp. 2, 3), and thick nodes represent that the algorithm has processed the nodes.

The CDS by the heuristic is the union of two subsets of nodes, i.e., $(\cup_{d=0}^{k} I_d) \cup (\cup_{d=0}^{k} S_d)$.

- The first subset $\cup_{d=0}^{k} I_d$ is an MIS for G. The root v definitely joins a set I_0. Let D_d be a set of nodes $v_i \in L_d$ each of which is dominated by some node in I_{d-1}. For each $1 \le d \le k$, a set I_d is an MIS in an induced subgraph by $L_d \setminus D_d$. That is to say, the heuristic paves the field dominated by the members of I in order of increasing of $L(v_i)$ from v. (In Figure 2(a)-Figure 2(d), black nodes are in I_0 or I_2.)
- The second subset is $\cup_{d=0}^{k} S_d$, where a set S_d of nodes which are fathers of the members of I_d for each $1 \le d \le k$. Note that $S_d \subseteq L_{d-1}$. (In Figure 2(c)-Figure 2(d), gray nodes are in S_2.)

We call the above way of construction of an MIS $\cup_{d=0}^{k} I_d$ "*paving* on a BFS tree". For any set $C \subset V$ and any node $v_i \notin C$, let the distance between v_i and C be the minimum distance between v_i and any $v_j \in C$. On the MIS constructed by paving on a BFS tree, the set satisfies the following property.

Theorem 1 *Let I' be the MIS constructed by paving on a BFS tree T. For any v_i in I', the distance between v_i and $I' \setminus \{v_i\}$ is exactly two hops.*

The proof of this theorem is the same as one in [16]. Because of limitation of space, proof of this theorem is omitted.

By Theorem 1, the connectivity of the CDS is ensured. By the proof of Theorem 1, in the MIS constructed by paving on T, each member $v_i \in L_d$ of the MIS has a father on T which is neighbor to at least one member of MIS in L_{d-1} or L_{d-2}. Therefore, the union of the MIS and a set of fathers of the members of the MIS are connected. Because the MIS is also a minimal dominating set, the union is a CDS. It is clear that, if each member $v_i \in L_d$ of any MIS has a father which is neighbor to at least one member of the MIS in L_{d-1} or L_{d-2} on T, then it can be said that the MIS is constructed by paving on T. The CDS is illustrated in Figure 2(e).

We define such a CDS as *CDS-tree* formally as follows:

Definition 6 *Let I' be any MIS constructed by paving on a BFS tree T for G. Let $S'(\neq \emptyset)$ be a set of nodes each of which is the father of a member in I' on T. A set of nodes $I' \cup S'$ is a* CDS-tree *of T.* □

Theorem 2 *Let D_{opt} is the minimum CDS. Any CDS-tree is an approximation for the minimum CDS which size is at most $8|D_{opt}| + 1$ in unit disk graphs.*

The proof of the approximation ratio is similar to the one for Wan et al.'s algorithm in [16]. We give the proof follow.

Proof: We consider a CDS-tree $I' \cup S'$, where I' be a maximal independent set constructed by paving on a BFS tree and S' be a set of nodes each of which is the father of a member in I' on a BFS tree. Let D_{opt} be any minimum CDS.

First, we consider the size of I'. Let v_1 be an arbitrary node in D_{opt}. Let I'_1 be the subset of I' such that each member of it is neighbor to v_1. Then, the size of I'_1 is at most 5. After we count $I'_1, I'_2, \cdots, I'_{i-1}$, for $2 \le i \le |D_{opt}|$, let $v_i \in D_{opt}$ be an arbitrary neighbor of $v_1, v_2, \cdots, v_{i-2}$ or v_{i-1}. Let I'_i be a subset of $I' \setminus \{I'_1 \cup I'_2 \cup \cdots I'_{i-1}\}$ such that each member of it is neighbor to v_i. Because v_i is neighbor to a node v' where v' is $v_1, v_2, \cdots, v_{i-2}$ or v_{i-1}, the members of I'_i are in a part of a unit cycle around v_i such that it does not overlap with a unit cycle around v'. That is, the members of I'_i are only in a field which is smaller than a 240 degree sectoral field around v_i. (See figure 3.) Therefore, the size of I'_i is at most 4 for $2 \le i \le |D_{opt}|$, let $v_i \in D_{opt}$. Therefore, the size of I' is

$$\begin{aligned} |I'| &= \sum_{i=1}^{|D_{opt}|} |I'_i| = |I'_1| + \sum_{i=2}^{|D_{opt}|} |I'_i| \\ &\le 5 + 4(|D_{opt}| - 1) \\ &= 4|D_{opt}| + 1. \end{aligned}$$

Next, we consider the size of S'. Because each member of I' which is not the root of the BFS tree has a father in S', the size of S' is at most $4|D_{opt}|$.

Therefore, the size of the CDS-tree is

$$|I' \cup S'| = |I'| + |S'| \le 8|D_{opt}| + 1.$$

□

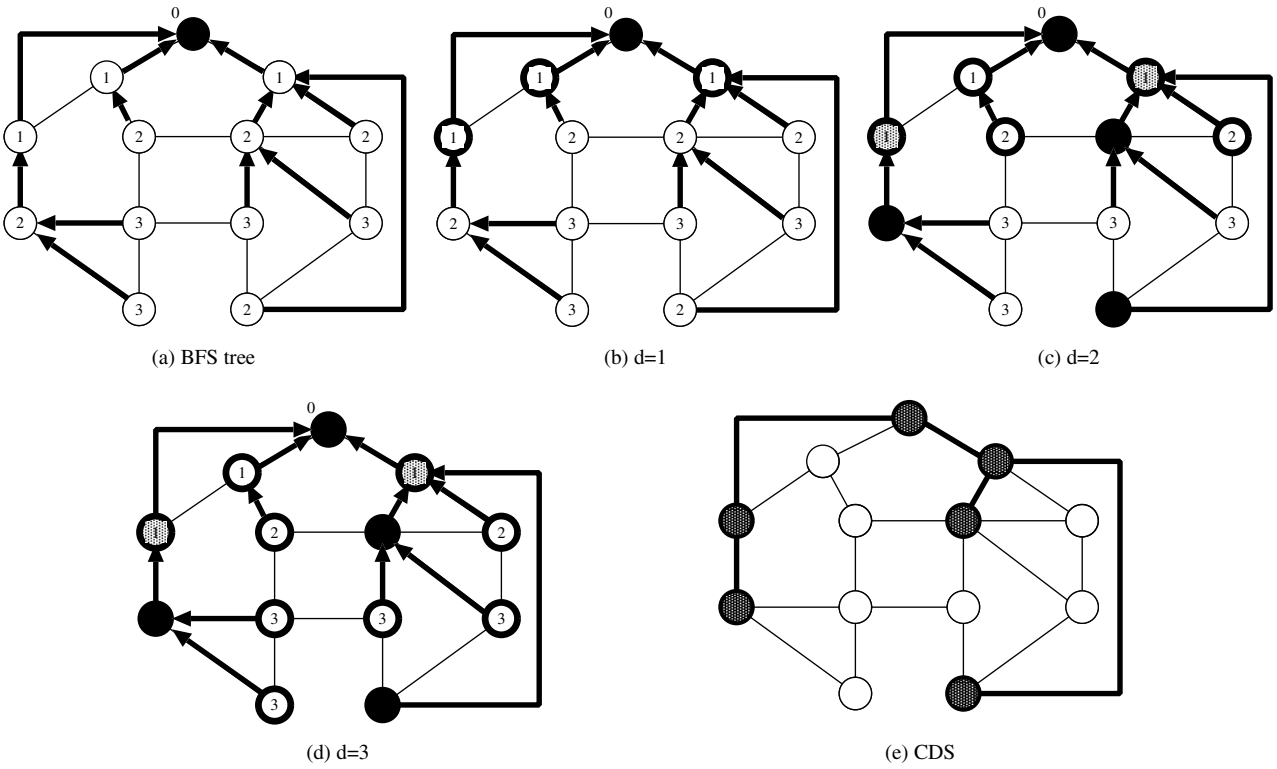

Figure 2. Outline of Marathe et al.'s algorithm

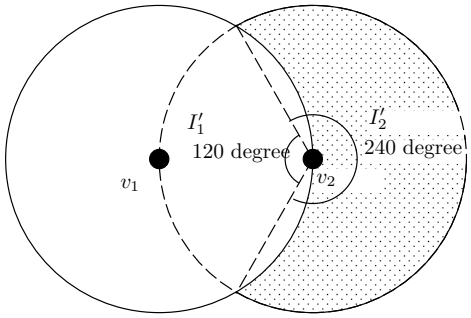

Figure 3. The relationship between I'_1 and I'_2.

4. Proposed algorithm

We propose a self-stabilizing algorithm SS-CDS to find a CDS-tree. Our algorithm computes a BFS tree rooted at P_r for G. Because algorithms for computing a BFS tree have been proposed so far, for example [10, 15], we simply adopt one of them. We assume, without loss of generality, that each process P_i has the following two variables as output of the BFS tree T, and as input of SS-CDS.

- $F(P_i)$ — a process id of the father of P_i on T.
- $L(P_i)$ — the distance from the root P_r to P_i on T.

The output of each process is

- $dom(P_i)$ — a boolean if P_i is a member of the CDS.

First, SS-CDS computes an MIS. For constructing an MIS, there exist many self-stabilizing algorithms, for example [14, 9]. However, these algorithms do not ensure that a computed MIS satisfies Theorem 1 on T.

Formal description of proposed algorithm SS-CDS is shown in Figure 4. There are five guarded-commands (GC1-GC5) in the algorithm.

- By GC1, the root P_r of T with $L(P_r) = 0$ joins both the MIS and the CDS.
- By GC2 and GC3, process P_i ($\neq P_r$) decides if P_i joins the MIS.
  - If each neighbor $P_j \in N_i$ at which $L(P_i) \geq L(P_j)$ holds is not a member of the MIS, then P_i joins the MIS by GC2.
  - If there exists a neighbor P_j such that $L(P_i) \geq L(P_j)$ holds and P_j is a member of the MIS, then P_i leaves the MIS by GC3.

- By GC4 and GC5, process P_i ($\neq P_r$) decides if P_i joins the CDS.
  - If P_i or its at least one child on T is a member of the MIS, then P_i joins the CDS by GC4.
  - Otherwise, P_i leaves the CDS by GC5.

By Γ, we denote a set of all configurations of SS-CDS. A set of legitimate configurations is defined as follows.

Definition 7 *A configuration γ is* legitimate *iff the following three conditions are satisfied.*

- *Condition 1: γ is in Γ_I, where $\Gamma_I \subset \Gamma$ is a set of configurations such that a set of processes $\{P_i \mid ind(P_i) = \textbf{true}\}$ is an MIS of G which satisfies Theorem 1 on T,*
- *Condition 2: $ind(P_i) = \textbf{true}$ implies $dom(P_i) = \textbf{true}$ for each P_i, and*
- *Condition 3: a set $\{P_i \mid dom(P_i) = \textbf{true}\}$ is a CDS-tree of G.*

By Λ_C, we denote a set of legitimate configurations of SS-CDS. □

5. Proof of correctness

Lemma 1 *No process is privileged in configuration γ if and only if $\gamma \in \Lambda_C$.*

Proof: Let γ be a configuration in which no process is privileged. Assume that $\gamma \notin \Lambda_C$.

- Suppose that Condition 1 is false, i.e., $\{P_i \mid ind(P_i) = \textbf{true}\}$ is not an MIS which satisfies Theorem 1 on T. That is, the set $\{P_i \mid ind(P_i) = \textbf{true}\}$ is not an independent set, is not maximal, or does not satisfy Theorem 1 on T.
  - We assume that the set is not an independent set, i.e., there exists P_i and P_j in the set such that they are neighbor each other. If $L(P_i) = L(P_j) \neq 0$, the guard of GC3 is true at P_i and P_j because $ind(P_i) = ind(P_j) = \textbf{true}$. Similarly, if $0 \leq L(P_i) < L(P_j)$ (resp. $L(P_i) > L(P_j) \geq 0$), then the guard of GC3 is true at P_j (resp. P_i). This is a contradiction for the assumption that no process is privileged in γ. Therefore, the set $\{P_i \mid ind(P_i) = \textbf{true}\}$ is an independent set.
  - We assume that the set is not maximal. That is, there exists a process P_i with $ind(P_i) = \textbf{false}$ which has no neighbor P_j with $ind(P_j) = \textbf{true}$. If $L(P_i) = 0$, then the guard of GC1 is true at P_i. If $L(P_i) \neq 0$, then $\forall P_j \in N_i[L(P_j) > L(P_i) \vee ind(P_j) = \textbf{false}]$ holds at P_i, and the guard of GC2 is true at P_i. This is a contradiction for the assumption that no process is privileged in γ. Therefore, the set $\{P_i \mid ind(P_i) = \textbf{true}\}$ is an MIS.
  - We assume that the set does not satisfy Theorem 1 on T. By proof of Theorem 1, this assumption means that there exists a process $P_i \neq P_r$ with $ind(P_i) = \textbf{true}$ whose father P_f ($= F(P_i) \in N_i$) with $ind(P_f) = \textbf{false}$ on T is not adjacent to another process P_j with $ind(P_j) = \textbf{true}$ and $L(P_j) \leq L(P_f)$. Because $L(P_f) = L(P_i) - 1$, this assumption implies that $\forall P_j \in N_f[L(P_j) > L(P_f) \vee ind(P_j) = \textbf{false}]$. Then, the guard of GC2 is true at P_f. This is a contradiction for the assumption that no process is privileged in γ.

Constants— read only and automatically updated
N_i: The set of neighbors of P_i in underlying network G.
$F(P_i)$: A father of P_i in T.
$L(P_i)$: The distance from the root in T.
Variables
$ind(P_i)$: True iff P_i is a member of an MIS.
$dom(P_i)$: True iff P_i is a member of a CDS.
Macro
$\mathrm{Grd}_k (k = 2,3)$: The guard of GCk
A Set of Guarded-Commands:

$*[$
/* GC1: Root joins an MIS and a CDS. */
$L(P_i) = 0 \wedge \{ind(P_i) = \textbf{false} \vee dom(P_i) = \textbf{false}\}$
$\rightarrow ind(P_i) := \textbf{true}$; $dom(P_i) := \textbf{true}$;
/* GC2: Join an MIS. */
$L(P_i) \neq 0 \wedge ind(P_i) = \textbf{false} \wedge$
$\forall P_j \in N_i[L(P_j) > L(P_i) \vee ind(P_j) = \textbf{false}]$
$\rightarrow ind(P_i) := \textbf{true}$;
/* GC3: Leave an MIS. */
$L(P_i) \neq 0 \wedge ind(P_i) = \textbf{true} \wedge$
$\exists P_j \in N_i[L(P_j) \leq L(P_i) \wedge ind(P_j) = \textbf{true}]$
$\rightarrow ind(P_i) := \textbf{false}$;
/* GC4: Join a CDS. */
$L(P_i) \neq 0 \wedge \neg \mathrm{Grd}_2 \wedge \neg \mathrm{Grd}_3 \wedge dom(P_i) = \textbf{false} \wedge$
$\{\exists P_j \in N_i[F(P_j) = P_i \wedge ind(P_j) = \textbf{true}] \vee$
$ind(P_i) = \textbf{true}\}$
$\rightarrow dom(P_i) := \textbf{true}$;
/* GC5: Leave a CDS. */
$L(P_i) \neq 0 \wedge \neg \mathrm{Grd}_2 \wedge \neg \mathrm{Grd}_3 \wedge dom(P_i) = \textbf{true} \wedge$
$\{\forall P_j \in N_i[F(P_j) \neq P_i \vee ind(P_j) = \textbf{false}] \wedge$
$ind(P_i) = \textbf{false}\}$
$\rightarrow dom(P_i) := \textbf{false}$;
$]$

Figure 4. SS-CDS: A self-stabilizing approximation algorithm for the minimum CDS

Therefore, the set $\{P_i \mid ind(P_i) = \textbf{true}\}$ satisfies Theorem 1 on T.

Therefore, the Condition 1 is true in γ.

- Suppose that Condition 2 is false, i.e., there exists a process P_i with $ind(P_i) = \textbf{true}$, but $dom(P_i) = \textbf{false}$. If $L(P_i) = 0$, then the guard of GC1 is true at P_i. If $L(P_i) \neq 0$, then the guard of GC4 is true at P_i. This is a contradiction for the assumption that no process is privileged in γ. Therefore, the Condition 2 is true in γ.
- Suppose that Condition 3 is false. That is, a set $\{P_i \mid dom(P_i) = \textbf{true}\}$ is not a CDS-tree of G.
  - Assume that the set is not a CDS. By the Condition 2, the set is a dominating set because it is clear that an MIS is a dominating set. Therefore, the set is not connected. By Definition 6, there exists a process P_i with $dom(P_i) = \textbf{false}$ which is a father of P_j with $ind(P_j) = \textbf{true}$ for the assumption. Then, the guard of GC4 is true at P_i. This is a contradiction for the assumption that no process is privileged in γ. Therefore, the set is a CDS.
  - Assume that the set is a CDS, but not a CDS-tree. Then, by Definition 6, there exists a process P_i with $dom(P_i) = \textbf{true}$, but neither P_i nor its children are the members of the MIS. Then, the guard of GC5 is true at P_i. This is a contradiction for the assumption that no process is privileged in γ.

  Therefore, the Condition 3 is true in γ.

Therefore, $\gamma \in \Lambda_C$ if no process is privileged.

It is clear that no process is privileged if the configuration is legitimate. □

Lemma 2 *For any configuration γ_0 and any computation starting from γ_0, eventually no process is privileged.*

Proof: First, the root process P_r with $L(P_r) = 0$ executes GC1 at most once, and decides the values of $ind(P_r) = \textbf{true}$ and $dom(P_r) = \textbf{true}$. These values never change after that, because they are not changed by other guarded-commands. Therefore, we suppose below that the values of them are correct at P_r.

By the definition of SS-CDS, at $P_i \neq P_r$, the value of $ind(P_i)$ is decided by GC2 or GC3, and these guarded-commands do not refer to the values of $dom(P_j)$ of any processes P_j. Therefore, the value of $ind(P_i)$ is decided before the value of $dom(P_i)$ is decided at each process P_i.

Suppose that there exists an infinite (non-converging) computation starting from γ_0. Then, there is a process P_i such that $L(P_i) \neq 0$ which executes infinitely often.

- Suppose that P_i with $L(P_i) = 1$ changes the value of $ind(P_i)$ infinitely often, i.e., P_i executes GC2 and GC3 alternately infinitely often. However, because $ind(P_r) = \textbf{true}$ holds at P_r and the value never change, P_i cannot execute GC2 by the definition of algorithm. That is, P_i executes GC3 at most once. Therefore, P_i with $L(P_i) = 1$ cannot execute infinitely often.
- Suppose that P_i with $L(P_i) = d$ $(d > 1)$ changes the value of $ind(P_i)$ infinitely often, and suppose that P_h with $L(P_h) = d - 1$ never execute. Then, $P_j \in N_i$ with $L(P_j) = d$ must change the value of $ind(P_j)$ infinitely often by the definition of GC2 and GC3. If there exists a process $P_h \in N_i$ with $L(P_h) = d - 1$ and $ind(P_h) = \textbf{true}$, then P_i cannot execute GC2. Then, P_i cannot execute GC3 infinitely often. Therefore, there exist no process $P_h \in N_i$ with $L(P_h) = d - 1$ and $ind(P_h) = \textbf{true}$. However, P_i can change the value of $ind(P_i)$ from **false** to **true** by GC2 only when all its neighbor P_j with $L(P_j) \leq d$ hold $ind(P_j) = \textbf{false}$ by the definition of GC2. After P_i executes GC2, the guard of GC2 cannot become true at all of its neighbor P_j with $L(P_j) = d$. Therefore, P_j cannot change the value of $ind(P_j)$ infinitely often. This is a contradiction for the assumption. Therefore, P_i and P_j cannot execute GC2 and GC3 infinitely often.

There is no process which executes GC2 and GC3 infinitely often. We suppose below that the value of $ind(P_i)$ of each P_i is correct, and never change.

Suppose that P_i changes $dom(P_i)$ infinitely often, i.e., P_i executes GC4 and GC5 alternately infinitely often. However, by the definition of GC4 and GC5, each process P_i decides the value of $dom(P_i)$ based only on the values of $ind(P_i)$ and each $ind(P_j)$ where P_j is a child of P_i on T. This is a contradiction for the assumption that the value of $ind(P_i)$ of each process P_i never change. The value of $dom(P_i)$ changes at most once.

Therefore, P_i cannot execute infinitely often. □

Theorem 3 *The algorithm* SS-CDS *is self-stabilizing with respect to Λ_C.*

Proof: Clear from Lemmas 1 and 2. □

Theorem 4 *Time complexity of* SS-CDS *is $O(n^2)$ steps.*

Because of limitation of space, proof of this theorem is omitted.

Theorem 5 SS-CDS *is a self-stabilizing approximation algorithm for the minimum CDS problem in unit disk graphs. The size of the solution by* SS-CDS *is at most $8|D_{opt}| + 1$, where D_{opt} is a minimum CDS.*

Proof: From Theorem 3, the set $\{P_i \mid dom(P_i) = \textbf{true}\}$ is a CDS-tree. From Theorem 2, the size of the solution by SS-CDS is at most $8|D_{opt}|+1$. □

6. Conclusion

In this paper, we proposed a self-stabilizing distributed approximation algorithm for the minimum CDS in unit disk graphs. As an application of the proposed algorithm, a minimum CDS is a virtual backbone or routing in mobile ad hoc networks. Since our algorithm is self-stabilizing, it is strongly desirable in mobile ad hoc networks.

The size of the solution by our algorithm is at most $8|D_{opt}|+1$, where D_{opt} is a minimum CDS. The time complexity is $O(n^2)$ steps. Development of a self-stabilizing approximation algorithm with better approximation ratio is left for future work.

Acknowledgment

This work is supported in part by JSPS: Grant-in-Aid for Scientific Research ((B)17300020) and Ookawa Foundation Research Grant.

References

[1] V. Bharghavan and B.Das. Routing in ad hoc networks using minimum connected dominating sets. In *Proceedings of International Conference on Communications'97*, pages 376–380, 1997.

[2] J. Blum, M. Ding, A. Thaeler, and X. Cheng. *Connected Dominating Set in Sensor Networks and MANETs*, pages 329–369. Kluwer Academic Publishers, 2004.

[3] X. Cheng and D.-Z. Du. Virtual backbone-based routing in multihop ad hoc wireless networks. Technical report, University of Minnesota, 2002.

[4] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[5] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

[6] S. Dolev. *Self-Stabilization*. The MIT Press, 2000.

[7] B. Gao, Y. Yang, and H. Ma. A new distributed approximation algorithm for constructing minimum connected dominating set in wireless ad hoc networks. *International Journal of Communication System*, 18:743–762, 2005.

[8] S. Guha and S. Khuller. Approximation Algorithms for Connected Dominating Sets. *Algorithmica*, 20:347–387, 1998.

[9] S. M. Hedetniemi, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani. Self-stabilizing algorithms for minimal dominating sets and maximal independent sets. *Computers and Mathematics with Applications*, 46(5-6):805–811, 2003.

[10] S. T. Huang and N.-S. Chen. A self-stabilizing algorithm for constructing breadth first trees. *Information Processing Letters*, 41(2):109–117, 1992.

[11] A. Jain and A. Gupta. A distributed self-stabilizing algorithm for finding a connected dominating set in a graph. In *Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)*, pages 615–619. IEEE Press, 2005.

[12] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.

[13] E. Sampathkumar and H. B. Walikar. The connected domination number of a graph. *Journal of Mathematical Physical Sciences*, 13:607–613, 1979.

[14] S. K. Shukla, D. J. Rosenkrantz, and S.S.Ravi. Observations on self-stabilizing graph algorithms for anonymous networks. In *Proceedings of the 2nd Workshop on Self-Stabilizing Systems (WSS'95)*, pages 7.1–7.15. Chicago Journal of Theoretical Computer Science, 1995.

[15] S. Sur and P. K. Srimani. A self-stabilizing distributed algorithm to construct BFS spanning trees of a symmetric graph. *Parallel Processing Letters*, 2:171–179, 1992.

[16] P.-J. Wan, K. M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. *Mobile Networks and Applications*, 9:141–149, 2004.

[17] J. Wu and H.L. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 7–14, 1999.

[18] J. Wu and W. Lou. Forward-node-set-based broadcast in clustered mobile ad hoc networks. *Wireless Networks and Mobile Computing*, 3(2):155–173, 2003.