

A Linear-Time Self-Stabilizing Algorithm for the Minimal 2-Dominating Set Problem in General Networks

TETZ C. HUANG, CHIH-YUAN CHEN AND CHENG-PIN WANG

Department of Computer Engineering and Science

Yuan Ze University

Chungli, 320 Taiwan

E-mail: cstetz@saturn.yzu.edu.tw

Kamei and Kakugawa have recently proposed a self-stabilizing algorithm for the minimal k -dominating set problem. Their algorithm is a general form of the maximal-independent-set algorithm proposed by Shukla *et al.* The results in their paper are for any tree network that assumes Dijkstra's central demon model. In particular, the worst-case stabilization time is claimed to be $O(n^2)$, where n is the number of nodes in the system. In this paper, we generalize their results for the case $k = 2$. We show that their algorithm with $k = 2$, when operating in any general network, is self-stabilizing under the central demon model, and solves the minimal 2-dominating set problem. We also derive that the worst-case stabilization time is linear, *i.e.*, $O(n)$. A bounded function technique is employed in obtaining these results.

Keywords: self-stabilizing algorithm, central demon model, maximal independent set, minimal k -dominating set, tree network, general network, bounded function, cut point

1. INTRODUCTION

A *distributed system* consists of a set of loosely connected processors that do not share a common or global memory. Each processor has one or more shared registers and possibly some non-shared local variables, the contents of which specify the *local state* of the processor. Local states of all processors in the system at a certain time constitute the *global configuration* (or, simply, *configuration*) of the system at that time. The main restriction of a distributed system is that each processor in the system can only access the data (*i.e.*, read the shared data) of its neighbors. Since a distributed algorithm is an algorithm that works in a distributed system, it cannot violate this main restriction. Depending on the purpose of a distributed system, a global criterion for the global configuration is defined. Those global configurations satisfying the criterion are called *legitimate configurations*, whereas other global configurations are called *illegitimate configurations*. When the system is in a legitimate configuration, the purpose of the system is fulfilled.

1.1 Dijkstra's Central Demon Model

Dijkstra's *central demon model* of computation [7] of an algorithm in a distributed system has the following features:

Received December 12, 2005; revised March 20, 2006; accepted March 27, 2006.
Communicated by Takeshi Tokuyama.

- (a) The algorithm running on each processor consists of one or more rules. Each rule is of the form

condition part \rightarrow *action part*.

The *condition part* (or *guard*) is a Boolean function over the states of the processor and its neighbors; the action part is an assignment of values to some of the processor's shared registers. If the condition part of a rule in a processor is evaluated as true, we say that the processor is *privileged* to execute the action part (or to *make a move*, or to *write*).

- (b) At the initial configuration, if none of the processors is privileged, then the system is deadlocked. Otherwise, if a privileged processor exists, the *central demon* in the system will randomly select exactly one among all the privileged processors to make a move, in a single atomic step. The local state of the selected processor thus changes, which in the meantime results in the change of the global configuration of the system. The system will then repeat the above process again and again to change the global configuration as long as it does not encounter any deadlock situation. Thus, the behavior of the system under the action of the algorithm can be described by an *execution sequence* $\Gamma = (\gamma_1, \gamma_2, \dots)$ in which for any $i \geq 1$, γ_i represents a global configuration, and γ_{i+1} is obtained from γ_i after exactly one processor in the system makes the i^{th} move $\gamma_i \rightarrow \gamma_{i+1}$.

Under this computational model, Dijkstra introduced the notion of self-stabilization of a distributed system in his classic paper [2] in 1974 (cf. also [3, 4]). According to Dijkstra, an algorithm is *self-stabilizing* if, regardless of any initial configuration of the system, any execution of the algorithm will lead the system to a legitimate configuration, and then let the system stay in the legitimate configuration (or some legitimate configurations) forever, unless the system incurs a transient fault. Many papers have been published, regarding self-stabilizing algorithms under Dijkstra's central demon model.

1.2 Minimal k -dominating Set Problem

Let $G = (V, E)$ be a simple connected undirected graph that models a distributed system, with each node $x \in V$ representing a processor in the system and each edge $\{x, y\} \in E$ representing the bidirectional link connecting processors x and y . Let k be an arbitrary positive integer. A subset D of V is a *k -dominating set* in G if each node not in D has at least k neighbors in D . A k -dominating set D in G is *minimal* if any proper subset of D is not a k -dominating set in G . The so-called minimal k -dominating set problem is to find a minimal k -dominating set in G . A k -dominating set for $k = 1$, *i.e.*, a 1-dominating set, is just an ordinary dominating set. Also, since any maximal independent set in a graph is a minimal dominating set in that graph, a self-stabilizing algorithm for the maximal independent set problem can be viewed as a self-stabilizing algorithm for the minimal dominating set problem. Shukla *et al.* [12], Goddard *et al.* [5], and Ikeda *et al.* [8] have proposed self-stabilizing algorithms for the maximal independent set problem, under the central demon model, the synchronous model, and the distributed demon model, respectively. The worst-case stabilization times for these three algorithms were computed to be

$O(n)$ steps, $O(n)$ rounds, and $O(n^2)$ steps, respectively, where n is the number of nodes in the system. These three algorithms are very similar. In Huang *et al.* [6], an extensive study on the self-stabilization of the algorithms intended for solving the maximal independent set problem is conducted. The self-stabilization of all the above three similar algorithms and another also very similar algorithm are checked against the central demon model, the synchronous model, and the distributed demon model; and the naturally adapted versions from these four algorithms are considered under the Dolev model. A number of results are obtained in that paper. In [11], Lin and Huang proposed a fault-containing self-stabilizing algorithm for the maximal independent set problem, under the central demon model. The worst-case stabilization time for single-fault situations was computed to be $O(\Delta)$ steps, where Δ is the maximum node degree in the system.

All the above works can be regarded as for the minimal k -dominating set problem with $k = 1$. Recently, Kamei and Kakugawa [9] made a step forward to consider the same problem for general k (*i.e.*, for k being an arbitrary positive integer). They proposed an algorithm that appeared to be a general form of the maximal-independent-set-algorithm in [12]. They showed that the algorithm is self-stabilizing under the central demon model, and solves the minimal k -dominating set problem; they also claimed that the time complexity of the algorithm is $O(n^2)$ steps. These results are interesting, although they are restricted to tree networks only. In [10], Kamei and Kakugawa proposed a self-stabilizing approximation algorithm for minimum k -dominating set problem for arbitrary networks. As noted in that paper, if the minimum degree of the system is less than k , the algorithm can find a minimal k -dominating set. Evidently, a natural direction for further investigation would be to relax the above restrictions in [9, 10] and pursue a self-stabilizing algorithm for minimal k -dominating set problem for arbitrary networks; and an even further investigation would be to consider the same problem under the distributed demon model.

1.3 Main Results and the Organization of the Rest of the Paper

In our ongoing project, we are considering the extension problem just mentioned. In this paper, we will solve that extension problem for the case $k = 2$. More explicitly, we will show that the algorithm in [9] with $k = 2$, when operating in any general network, is self-stabilizing under the central demon model, and solves the minimal 2-dominating set problem. We will also derive that the worst-case stabilization time is linear, *i.e.*, $O(n)$. This will generalize the results in [9] for the case $k = 2$, and also extend the results regarding the maximal-independent-set-algorithm in [12].

The rest of this paper is organized as follows: In section 2, we first recall the algorithm in [9], letting $k = 2$; then we show that any legitimate configuration can solve the minimal 2-dominating set problem. An example is given to illustrate the execution of the algorithm in section 3. The correctness proof of the algorithm is provided, and the worst-case stabilization time of the algorithm is computed in section 4. Finally in section 5, some remarks conclude the discussion in this paper.

2. THE ALGORITHM AND THE LEGITIMATE CONFIGURATIONS

We assume that the system in consideration is a simple connected undirected graph

$G = (V, E)$, with each processor x in the system maintaining a shared register d_x , whose value is either 0 or 1. The algorithm to be considered is the algorithm in [9], with every “ k ” in that algorithm replaced by “2”.

Algorithm 1

{For any node x }

R1: $d_x = 0 \wedge |D(x)| \leq 1 \rightarrow d_x := 1$

R2: $d_x = 1 \wedge |D(x)| \geq 2 \rightarrow d_x := 0$

Note that in the above algorithm, $D(x)$ denotes the set $\{y \in N(x) \mid d_y = 1\}$, and $|D(x)|$ denotes the cardinality of $D(x)$. The legitimate configurations are defined to be all those configurations in which

$$\forall x \in V [(d_x = 0 \wedge |D(x)| \geq 2) \vee (d_x = 1 \wedge |D(x)| \leq 1)].$$

It is obvious that the system is in a legitimate configuration if and only if no node in the system is privileged. The following lemma clarifies that in any legitimate configuration, a minimal 2-dominating set can be identified.

Lemma 1 If no node in the system is privileged, then the set $D = \{x \in V \mid d_x = 1\}$ is a minimal 2-dominating set in the system.

Proof: Suppose no node in G is privileged. For any $x \notin D$, since $d_x = 0$ and x is not privileged by R1, we have $|D(x)| \geq 2$. Thus x has at least two neighbors in D and thus, D is a 2-dominating set in G . If D is not minimal, then there is a proper subset D' of D such that D' is also a 2-dominating set. Let y be a node in $D - D'$. Then $d_y = 1$. Since D' is a 2-dominating set and $y \notin D'$, y has at least two neighbors in D' in view of the definition of a 2-dominating set. It follows that y has at least two neighbors in D , i.e., $|D(y)| \geq 2$. With $d_y = 1$ and $|D(y)| \geq 2$, y is privileged by R2, which causes a contradiction. Therefore D is a minimal 2-dominating set. \square

3. AN ILLUSTRATION

The example in Fig. 1 is to illustrate the execution of Algorithm 1. Note that in each configuration, the shaded nodes represent privileged nodes, whereas the shaded node with a darkened circle represents the privileged node selected by the central demon to make a move.

4. CORRECTNESS PROOF AND THE STABILIZATION TIME

In this section, we will provide a correctness proof and a computation of the worst-case stabilization time for Algorithm 1. As mentioned in section 1.1, under the central demon model, the behavior of the system under the action of the algorithm can be described by an execution sequence $\Gamma = (\gamma_1, \gamma_2, \dots)$ in which for any $i \geq 1$, γ_i represents a global configuration. In order to give a rigorous proof for the self-stabilization of

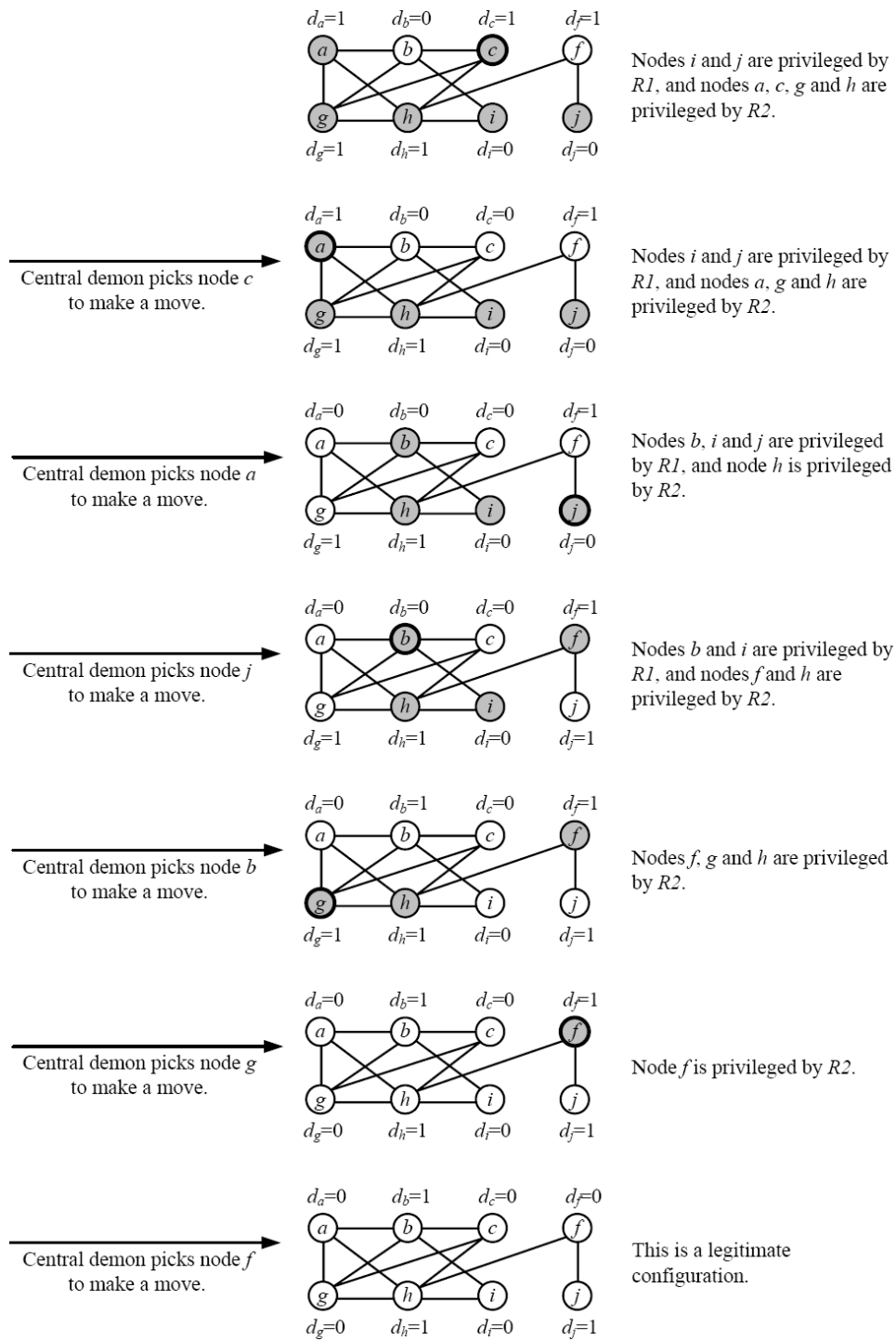


Fig. 1. An example to illustrate the execution of Algorithm 1. (In the last configuration, which is a legitimate configuration, a minimal 2-dominating set $D = \{b, h, j\}$ can be identified.)

Algorithm 1, we need to make the concept of execution sequence more precise. A sequence $\Gamma = (\gamma_1, \gamma_2, \dots)$ is called an *infinite execution (of Algorithm 1 under the central demon model)* if

- (a) Γ is an infinite sequence,
- (b) for any $M \in Z^+$, γ_m is a global configuration, and
- (c) for any $M \in Z^+$, γ_{m+1} is induced from γ_m after a unique privileged processor selected by the central demon makes the move $\gamma_m \rightarrow \gamma_{m+1}$.

A sequence $\Gamma = (\gamma_1, \gamma_2, \dots)$ is called a *finite execution (of Algorithm 1 under the central demon model)* if

- (a) Γ is a finite sequence and $\Gamma = (\gamma_1, \gamma_2, \dots, \gamma_q)$ for some $q \in Z^+$,
- (b) for any $m = 1, 2, \dots, q$, γ_m is a global configuration,
- (c) for any $m = 1, 2, \dots, q-1$, γ_{m+1} is induced from γ_m after a unique privileged processor selected by the central demon makes the move $\gamma_m \rightarrow \gamma_{m+1}$, and
- (d) no node in the system is privileged in the last configuration γ_q .

Thus, unless clearly specified, an execution $\Gamma = (\gamma_1, \gamma_2, \dots)$ may refer to an infinite or a finite execution.

Let γ be any configuration of the system $G = (V, E)$. We define $V_\gamma = \{x \in V \mid d_x = 1 \text{ in } \gamma\}$ and we use $G_\gamma = (V_\gamma, E_\gamma)$ to stand for the induced subgraph from V_γ in G , i.e., $E_\gamma = E \cap \{\{x, y\} \mid x, y \in V_\gamma\}$. If node $i \in V - V_\gamma$, then $G_\gamma \cup \{i\}$ stands for the subgraph of G obtained from G_γ by adding to G_γ node i and all the edges connecting node i and G_γ . If $i \in V_\gamma$, then $G_\gamma - \{i\}$ stands for the subgraph of G obtained from G_γ by deleting from G_γ node i and all the edges incident with i . We classify the nodes in G_γ into four types: isolated nodes, leaf nodes, cut points and miscellaneous nodes. A node i in G_γ is

- (1) an *isolated node* if it has no neighbor in G_γ ,
- (2) a *leaf node* if it has exactly one neighbor in G_γ ,
- (3) a *cut point* if the graph $G_\gamma - \{i\}$ has more connected components than G_γ , and
- (4) a *miscellaneous node* if it is none of the above three types of nodes.

We use $I_\gamma, L_\gamma, C_\gamma$ and M_γ to denote the set of all isolated nodes, the set of all leaf nodes, the set of all cut points and the set of all miscellaneous nodes in G_γ , respectively. We also use α_γ to denote the number of the connected components of G_γ . Then we define $F(\gamma) = 6\alpha_\gamma + |I_\gamma| + |L_\gamma| + |C_\gamma| - |M_\gamma|$, where $|\cdot|$ stands for the cardinality of a set. Thus F can be viewed as a function defined over the set of all configurations of G , and $F(\gamma) \in \{-n, -n+1, -n+2, \dots, 9n-1, 9n\}$, where n is the number of nodes in G . For any x in G , we define $D_\gamma(x) = \{y \in N(x) \mid d_y = 1 \text{ in } \gamma\}$.

We now start our proof. First, consider any R1 move $\gamma \rightarrow \gamma'$. Let i be the node that makes the move $\gamma \rightarrow \gamma'$. Then $|D_\gamma(i)| \leq 1$, and since after the move $\gamma \rightarrow \gamma'$, d_i changes from 0 to 1, we have $V_{\gamma'} = V_\gamma \cup \{i\}$ and $G_{\gamma'} = G_\gamma \cup \{i\}$.

Case 1: $|D_\gamma(i)| = 0$. Then the condition $[\forall j \in N(i), d_j = 0]$ holds in γ , and thus in γ' . Hence i is an isolated node in $G_{\gamma'} = G_\gamma \cup \{i\}$. It follows that $G_{\gamma'}$ has one more connected

component $\{i\}$ than G_γ , $I_{\gamma'} = I_\gamma \cup \{i\}$, $L_{\gamma'} = L_\gamma$, $C_{\gamma'} = C_\gamma$, and $M_{\gamma'} = M_\gamma$. Therefore,

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &= 6(\alpha_\gamma + 1) + (|I_\gamma| + 1) + |L_\gamma| + |C_\gamma| - |M_\gamma| \\ &= F(\gamma) + 7 \\ &> F(\gamma). \end{aligned}$$

Case 2: $|D_\gamma(i)| = 1$. Then the condition $[\exists! u \in N(i) \text{ s.t. } d_u = 1]$ holds in γ , and thus in γ' . Hence, in $G_{\gamma'} = G_\gamma \cup \{i\}$, i is a leaf node adjacent to u . So we can see that node i joins the connected component of G_γ that contains node u , and they together form a slightly larger connected component of $G_{\gamma'}$. All the other connected components of G_γ still remain connected components of $G_{\gamma'}$. Thus $\alpha_{\gamma'} = \alpha_\gamma$. We can also see that for any node in $V_\gamma - \{u\}$, the type it belongs to in G_γ and the type it belongs to in $G_{\gamma'}$ are the same (e.g., if it is an isolated node in G_γ , then it is an isolated node in $G_{\gamma'}$). So it only remains to consider the type changes of node u .

Case 2.1: Node u is an isolated node in G_γ . Then u becomes a leaf node in $G_{\gamma'}$, and we have $I_{\gamma'} = I_\gamma - \{u\}$, $L_{\gamma'} = L_\gamma \cup \{i, u\}$, $C_{\gamma'} = C_\gamma$ and $M_{\gamma'} = M_\gamma$. Thus,

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &= 6\alpha_\gamma + (|I_\gamma| - 1) + (|L_\gamma| + 2) + |C_\gamma| - |M_\gamma| \\ &= F(\gamma) + 1 \\ &> F(\gamma). \end{aligned}$$

Case 2.2: Node u is a leaf node in G_γ . Then u becomes a cut point in $G_{\gamma'}$, and we have $I_{\gamma'} = I_\gamma$, $L_{\gamma'} = (L_\gamma \cup \{i\}) - \{u\}$, $C_{\gamma'} = C_\gamma \cup \{u\}$ and $M_{\gamma'} = M_\gamma$. Thus,

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &= 6\alpha_\gamma + |I_\gamma| + (|L_\gamma| + 1 - 1) + (|C_\gamma| + 1) - |M_\gamma| \\ &= F(\gamma) + 1 \\ &> F(\gamma). \end{aligned}$$

Case 2.3: Node u is a cut point in G_γ . Then u still remains a cut point in $G_{\gamma'}$, and we have $I_{\gamma'} = I_\gamma$, $L_{\gamma'} = L_\gamma \cup \{i\}$, $C_{\gamma'} = C_\gamma$ and $M_{\gamma'} = M_\gamma$. Thus,

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &= 6\alpha_\gamma + |I_\gamma| + (|L_\gamma| + 1) + |C_\gamma| - |M_\gamma| \\ &= F(\gamma) + 1 \\ &> F(\gamma). \end{aligned}$$

Case 2.4: Node u is a miscellaneous node in G_γ . Then u becomes a cut point in $G_{\gamma'}$, and we have $I_{\gamma'} = I_\gamma$, $L_{\gamma'} = L_\gamma \cup \{i\}$, $C_{\gamma'} = C_\gamma \cup \{u\}$ and $M_{\gamma'} = M_\gamma - \{u\}$. Thus,

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &= 6\alpha_\gamma + |I_\gamma| + (|L_\gamma| + 1) + (|C_\gamma| + 1) - (|M_\gamma| - 1) \\ &= F(\gamma) + 3 \\ &> F(\gamma). \end{aligned}$$

From all the above discussion, we obtain the following lemma.

Lemma 2 For any $R1$ move $\gamma \rightarrow \gamma'$, $F(\gamma) < F(\gamma')$.

Next, we consider any $R2$ move $\gamma \rightarrow \gamma'$. Let i be the node that makes the move $\gamma \rightarrow \gamma'$. Then $|D_\gamma(i)| \geq 2$, i.e., $\deg(i) \geq 2$ in G_γ , and thus node i is a cut point or a miscellaneous node in G_γ . Since after the move $\gamma \rightarrow \gamma'$, d_i changes from 1 to 0, we have $V_{\gamma'} = V_\gamma - \{i\}$ and $G_{\gamma'} = G_\gamma - \{i\}$. The following three observations are easy.

Observation 1 $G_{\gamma'} = G_\gamma - \{i\}$ has no fewer connected components than G_γ , i.e., $\alpha_{\gamma'} \geq \alpha_\gamma$.

Observation 2 Any isolated node in G_γ remains an isolated node in $G_{\gamma'}$, i.e., $I_\gamma \subseteq I_{\gamma'}$.

Observation 3 Any leaf node in G_γ either remains a leaf node in $G_{\gamma'}$ or turns into an isolated node in $G_{\gamma'}$, i.e., $L_\gamma \subseteq I_{\gamma'} \cup L_{\gamma'}$.

The following graph-theoretic property can be derived easily from [1, Theorem 1.4].

Property 1 In a simple undirected graph, a node x is a cut point if and only if node x has two distinct neighbors y and z such that the path (y, x, z) is the unique simple path in the graph that connects y and z .

Claim 1 For any node in G_γ , if it is neither node i nor a neighbor of node i , then it cannot turn from a cut point in G_γ into a miscellaneous node in $G_{\gamma'}$.

Proof: Let x be an arbitrary cut point in G_γ that is neither node i nor a neighbor of node i . Then, by Property 1, x has two distinct neighbors y and z such that the path (y, x, z) is the unique simple path in G_γ that connects y and z . Since x is not a neighbor of node i , neither y nor z is i . Hence the path (y, x, z) is still the unique simple path in $G_{\gamma'}$ that connects y and z . By Property 1 again, x remains a cut point in $G_{\gamma'}$. The claim is proved. \square

Claim 2 If node j is a neighbor of node i in G_γ that turns from a cut point in G_γ into a miscellaneous node in $G_{\gamma'}$, then in $G_{\gamma'} = G_\gamma - \{i\}$, node j and any other neighbor of node i cannot be in the same connected component.

Proof: Since j is a cut point in G_γ , j has two distinct neighbors y and z such that the path (y, j, z) is the unique simple path in G_γ that connects y and z (by Property 1). Suppose neither y nor z is i , then the path (y, j, z) is still the unique simple path in $G_{\gamma'}$ that connects y and z . Thus, by Property 1, j remains a cut point in $G_{\gamma'}$, which contradicts the assumption that j is a miscellaneous node in $G_{\gamma'}$. Hence either y or z is i . Without loss of generality, let $y = i$. Now let x be an arbitrary neighbor of node i other than j . Suppose j and x are in the same connected component of $G_{\gamma'}$. Then there exists a path in $G_{\gamma'}$ that connects j and x . This path, the edge $\{z, j\}$ and the edge $\{x, i\}$ together constitute a path P in G_γ that connects $y = i$ and z , and path P does not contain the edge $\{y, j\}$. Hence path P can induce a simple path in G_γ that connects y and z , and the induced simple path is dif-

ferent from the simple path (y, j, z) . This causes a contradiction because (y, j, z) has just been proclaimed to be the unique simple path in G_γ that connects y and z . Hence j and x cannot be in the same connected component of $G_{\gamma'}$. The claim is proved. \square

The following observation follows immediately from Claim 2.

Observation 4 If j_1, j_2, \dots, j_l , $l \geq 2$, are neighbors of node i in G_γ that turn from cut points in G_γ into miscellaneous nodes in $G_{\gamma'}$, then $G_{\gamma'}$ has at least $l - 1$ more connected components than G_γ , i.e., $\alpha_{\gamma'} \geq \alpha_\gamma + (l - 1)$.

Now we are in a position to see the effect of the $R2$ move $\gamma \rightarrow \gamma'$ on the values of the bounded function.

Case 1: Node i is a miscellaneous node in G_γ . Suppose i has a neighbor j that turns from a cut point in G_γ into a miscellaneous node in $G_{\gamma'}$. Then, since $\deg(i) \geq 2$ in G_γ , there is a neighbor x of i such that $x \neq j$. By Claim 2, j and x are not in the same connected component of $G_{\gamma'}$. In other words, the connected component of G_γ that contains i breaks into more than one connected component of $G_{\gamma'}$. This contradicts the assumption that i is a miscellaneous node in G_γ . Hence i has no neighbor that turns from a cut point in G_γ into a miscellaneous node in $G_{\gamma'}$. This, together with Observations 2 and 3 and Claim 1, implies

$$I_\gamma \cup L_\gamma \cup C_\gamma \subseteq I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'},$$

and thus,

$$\begin{aligned} M_\gamma - \{i\} &= (V_\gamma - \{i\}) - I_\gamma \cup L_\gamma \cup C_\gamma \\ &\supseteq (V_\gamma - \{i\}) - I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'} \\ &= M_{\gamma'}. \end{aligned}$$

Hence, in view of Observation 1, we have

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &\geq 6\alpha_{\gamma'} + |I_\gamma| + |L_\gamma| + |C_\gamma| - (|M_\gamma| - 1) \\ &= F(\gamma) + 1 \\ &> F(\gamma). \end{aligned}$$

Case 2: Node i is a cut point in G_γ . Then, in view of the definition of a cut point, $G_{\gamma'} = G_\gamma - \{i\}$ has more connected components than G_γ , i.e., $\alpha_{\gamma'} \geq \alpha_\gamma + 1$.

Case 2.1: Node i has no neighbor in G_γ that turns from a cut point in G_γ into a miscellaneous node in $G_{\gamma'}$. The assumption for this case, together with Observations 2 and 3 and Claim 1, implies

$$I_\gamma \cup L_\gamma \cup C_\gamma - \{i\} \subseteq I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'},$$

and thus,

$$M_\gamma = (V_\gamma - \{i\}) - (I_\gamma \cup L_\gamma \cup C_\gamma - \{i\}) \supseteq (V_\gamma - \{i\}) - I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'} = M_{\gamma'}.$$

Hence we have

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &\geq 6(\alpha_{\gamma} + 1) + (|I_{\gamma}| + |L_{\gamma}| + |C_{\gamma}| - 1) - |M_{\gamma}| \\ &= F(\gamma) + 5 \\ &> F(\gamma). \end{aligned}$$

Case 2.2: Node i has a unique neighbor j in G_{γ} that turns from a cut point in G_{γ} into a miscellaneous node in $G_{\gamma'}$. The assumption for this case, together with Observations 2 and 3 and Claim 1, implies

$$I_{\gamma} \cup L_{\gamma} \cup C_{\gamma} - \{i, j\} \subseteq I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'},$$

and thus,

$$M_{\gamma} \cup \{j\} = (V_{\gamma} - \{i\}) - (I_{\gamma} \cup L_{\gamma} \cup C_{\gamma} - \{i, j\}) \supseteq (V_{\gamma} - \{i\}) - I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'} = M_{\gamma'}.$$

Hence we have

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &\geq 6(\alpha_{\gamma} + 1) + (|I_{\gamma}| + |L_{\gamma}| + |C_{\gamma}| - 2) - (|M_{\gamma}| + 1) \\ &= F(\gamma) + 3 \\ &> F(\gamma). \end{aligned}$$

Case 2.3: Node i has more than one neighbor in G_{γ} that turns from a cut point in G_{γ} into a miscellaneous node in $G_{\gamma'}$. Let j_1, j_2, \dots, j_l be precisely all the neighbors of i that turn from cut points in G_{γ} into miscellaneous nodes in $G_{\gamma'}$. Thus $l \geq 2$ and, in view of Observations 2 and 3 and Claim 1, we have

$$I_{\gamma} \cup L_{\gamma} \cup C_{\gamma} - \{i, j_1, j_2, \dots, j_l\} \subseteq I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'}.$$

Thus,

$$\begin{aligned} M_{\gamma} \cup \{j_1, j_2, \dots, j_l\} &= (V_{\gamma} - \{i\}) - (I_{\gamma} \cup L_{\gamma} \cup C_{\gamma} - \{i, j_1, j_2, \dots, j_l\}) \\ &\supseteq (V_{\gamma} - \{i\}) - I_{\gamma'} \cup L_{\gamma'} \cup C_{\gamma'} \\ &= M_{\gamma'}. \end{aligned}$$

Hence, in view of Observation 4, we have

$$\begin{aligned} F(\gamma') &= 6\alpha_{\gamma'} + |I_{\gamma'}| + |L_{\gamma'}| + |C_{\gamma'}| - |M_{\gamma'}| \\ &\geq 6[\alpha_{\gamma} + (l - 1)] + [|I_{\gamma}| + |L_{\gamma}| + |C_{\gamma}| - (l + 1) - (|M_{\gamma}| + l)] \\ &= F(\gamma) + 4l - 7 \\ &> F(\gamma). \end{aligned}$$

(Note that the last inequality is due to $l \geq 2$.)

From all the above discussion, we obtain the following lemma.

Lemma 3 For any R2 move $\gamma \rightarrow \gamma'$, $F(\gamma) < F(\gamma')$.

Theorem 1 Algorithm 1 is self-stabilizing and the worst-case stabilization time of Algorithm 1 is $O(n)$ steps, where n is the number of nodes in the system.

Proof: From Lemmas 2 and 3, we see that for any execution $\Gamma = (\gamma_1, \gamma_2, \dots)$, $F(\gamma_1) < F(\gamma_2) < \dots$, and thus $F(\gamma_1), F(\gamma_2), \dots$ are all distinct. Since $\{F(\gamma_1), F(\gamma_2), \dots\} \subseteq \{-n, -n+1, \dots, 9n-1, 9n\}$, $\{F(\gamma_1), F(\gamma_2), \dots\}$ is a finite set with at most $10n+1$ elements. Thus $\Gamma = (\gamma_1, \gamma_2, \dots)$ must be a finite execution $(\gamma_1, \gamma_2, \dots, \gamma_q)$ for some $q \leq 10n+1$. By the definition of a finite execution, no node is privileged in γ_q , and hence γ_q is a legitimate configuration. Therefore Algorithm 1 is self-stabilizing and the worst-case stabilization time of Algorithm 1 is $O(n)$ steps. \square

5. CONCLUDING REMARKS

In the above, we have successfully solved the extension problem, “To find a self-stabilizing algorithm for the minimal k -dominating set problem in general networks, under the central demon model”, for the case $k = 2$. Since the minimal k -dominating set problem is more general than the maximal independent set problem (because the latter can be viewed as the minimal 1-dominating set problem), it is much complicated to handle. Kamei and Kakugawa have been the first to attack the above extension problem, although their results in [9] are restricted to tree networks only. In this paper, we relax their restriction and consider general networks. However, we still confine ourselves to the case $k = 2$.

There are two key steps in our above work:

- (1) classifying all nodes in the system into four types, and
- (2) designing the bounded function.

Of course, graph-theoretic properties relating to the four types of nodes have also been employed in the discussion.

REFERENCES

1. F. Buckley and F. Harary, *Distance in Graphs*, Addison-Wesley Publishing Company, California, 1990.
2. E. W. Dijkstra, “Self-stabilizing systems in spite of distributed control,” *Communications of the ACM*, Vol. 17, 1974, pp. 643-644.
3. E. W. Dijkstra, “Self-stabilization in spite of distributed control,” *Selected Writings on Computing: a Personal Perspective*, Springer-Verlag, Berlin-Heidelberg-New York, 1982, pp. 41-46.
4. E. W. Dijkstra, “A belated proof of self-stabilization,” *Distributed Computing*, Vol. 1, 1986, pp. 5-6.
5. W. Goddard, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani, “Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks,” in

- Proceedings of the 17th International Parallel and Distributed Processing Symposium, Workshop on Advances in Parallel and Distributed Computational Models*, 2003, pp. 22-26.
6. T. C. Huang, C. Y. Chen, and C. P. Wang, "On the self-stabilization of distributed algorithms intended for solving the maximal independent problem," submitted.
 7. T. C. Huang, "A self-stabilizing algorithm for the shortest path problem assuming the distributed demon," *Computers and Mathematics with Applications*, Vol. 50, 2005, pp. 671-681.
 8. M. Ikeda, S. Kamei, and H. Kakugawa, "A space-optimal self-stabilizing algorithm for the maximal independent set problem," in *Proceedings of the 3rd International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2002, pp. 70-74.
 9. S. Kamei and H. Kakugawa, "A self-stabilizing algorithm for the distributed minimal k -redundant dominating set problem in tree networks," in *Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003, pp. 720-724.
 10. S. Kamei and H. Kakugawa, "A self-stabilizing approximation algorithm for the distributed minimum k -domination," *IEICE Transactions on Information and Systems*, Vol. E88-A, 2005, pp. 1109-1116.
 11. J. C. Lin and T. C. Huang, "An efficient fault-containing self-stabilizing algorithm for finding a maximal independent set," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, 2003, pp. 742-754.
 12. S. Shukla, D. J. Rosenkrant, and S. S. Ravi, "Observations on self-stabilizing graph algorithms on anonymous networks," in *Proceedings of the 2nd Workshop on Self-Stabilizing Systems*, 1995, pp. 7.1-7.15.



Tetz C. Huang (黃哲志) received his B.S. degree in Mathematics from Fu Jen Catholic University, Taiwan, in 1975. From 1979 to 1986, he studied at the University of Chicago and received his M.S. degree in Mathematics. He was a lecturer in Mathematics at the University of Chicago from 1981 to 1984. Since 1989, he has joined in Yuan Ze University, Taiwan, first as an instructor and later as an associate professor. He has publications on variational inequalities and self-stabilizing systems. His current research interest is focused mainly on self-stabilizing systems.



Chih-Yuan Chen (陳志遠) received his B.S. and M.S. degrees in Mathematics from Chung Yuan Christian University, Taiwan, in 1996 and 1998, respectively. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering, Yuan Ze University, Taiwan. His current research interest is on self-stabilizing algorithms.



Cheng-Pin Wang (王承斌) received his B.S. and M.S. degrees in Mathematics from Chung Yuan Christian University, Taiwan, in 1996 and 1998, respectively. He was a lecturer in mathematics at the Chung Yuan Christian University from 2001 till now. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering, Yuan Ze University, Taiwan. His current research interest is on self-stabilizing algorithms.