Self-Stabilizing Distributed Algorithms for Graph Alliances

Zhenyu Xu and Pradip K Srimani
Department of Computer Science, Clemson University, Clemson, SC 29634-0974

Abstract

Graph alliances are recently developed global properties of any symmetric graph. Our purpose in the present paper is to design self-stabilizing fault tolerant distributed algorithms for the global offensive and the global defensive alliance in a given arbitrary graph. We also provide complete analysis of the convergence time of both the algorithms.

1. Introduction

A relatively new concept of alliances in arbitrary network graphs has been recently studied in connection with reliability of networks [1, 2, 3, 4, 5]. Intuitively, a node in a graph is said to be a defender of a neighboring node, if both the nodes are in the alliance, or both the nodes are not in the alliance. Also, a node is considered a defender of itself. A node is said to be an attacker of an adjacent node, if one of them is in the alliance but the other one is not. A node is called defended, if the number of its defenders is greater or equals to the number of attackers. And a node is called to be attacked, if the number of its attackers is greater or equals to the number of defenders. There are two basic types of alliances: offensive alliance and defensive alliance. In an offensive alliance, all the nodes that are not in the alliance set are attacked. In a defensive alliance, all the nodes that are in the alliance set are defended. An alliance set is global if it is also a dominating set. We give the formal definitions of offensive alliance and defensive alliance in the next section.

Most of the essential fundamental services for mobile networked distributed systems (ad hoc, wireless or sensor) involve maintaining a global predicate over the entire network (defined by some invariance relation on the global state of the network) by using local knowledge at each of the participating nodes. Graph theoretic optimization problems remain popular and useful for such dynamic networks; fault tolerant distributed protocols for such problems provide the key resources for designing such wireless, sensor and ad hoc networks and they offer new insight into the fundamental role of discrete distributed algorithms in developing these real life applications. Self-stabilization is a relatively new paradigm for designing such localized distributed algorithms for networks; it is an *optimistic* way of looking at system fault tolerance and scalable coordination, because it provides a built-in safeguard against transient failures that might corrupt the data in a distributed system. The concept was introduced by Dijkstra in 1974 [6], and Lamport [7] showed its relevance to fault tolerance in distributed systems in 1983; a good survey of early self-stabilizing algorithms can be found in [8] and Herman's bibliography [9] also provides a fairly comprehensive listing of most papers in this field. Self-stabilizing algorithms are fault tolerant. Starting from any illegitimate states, the system will converge to legitimate state. Several authors have considered self-stabilizing algorithms for graph problems. For example, matchings are studied in [10, 11], maximal independent sets in [12], and domination in [13]. In some sense, the concept "graph problem" is not very restrictive as it can apply to any problem where there is some static global calculation in the network. In this paper, our purpose is to present two self-stabilizing algorithms for minimal global offensive alliance and 1-minimal global defensive alliance.

A self-stabilizing algorithm is called uniform, if all the vertices run the same set of rules. Both of our two algorithms are uniform in that sense. A self-stabilizing algorithm is anonymous, if the computation involved in the algorithm does not require unique id for each vertex; otherwise the algorithm is called id-based. Our offensive alliance algorithm is anonymous, but our defensive alliance algorithm is id-based. The defensive alliance algorithm uses the technique developed in [14].

2. Alliances

For the purpose of this paper we define a graph G to be a pair (V, E) where V denotes the set of vertices (nodes) and E to be the set of undirected edges in

1-4244-0054-6/06/$20.00

the graph; we assume that the graph G is connected. For any vertex $v \in V$ we define $N(v)$ to be the open neighborhood of the vertex v, i.e., the set of vertices that are adjacent to v in G and $N[v]$ to be the closed neighborhood of the vertex v, i.e., $N[v] = N(v) \cap \{v\}$.

2.1. Global Offensive Alliance

Definition 1 *Given a graph $G = (V, E)$, a* **global offensive alliance** *S is defined to be a subset of V, such that $\forall v \in V - S$, we have $|N[v] \cap S| \geq |N[v] \cap (V - S)|$.*

Remark 1 *A global offensive alliance S is always a dominating set for the graph G, since $\forall v \in V - S, |N[v] \cap S| \geq |N[v] \cap (V - S)| \rightarrow |N[v] \cap S| \geq 0$, i.e., any node of G not included in the set S is adjacent to at least one node in S.*

A global offensive alliance S is called **minimal** when no proper subset of S is a global offensive alliance of the graph G. The following lemma provides an efficient way to check the minimality of a global alliance of a graph.

Lemma 1 *If S is a global offensive alliance of a graph G, but it is not minimal, then there exists a node $i \in S$, such that $|N(i) \cap S| \geq |N(i) \cap (V - S)| + 1$.*

Proof : If S is not minimal, there exists a $S' \subset S$ which is a global offensive alliance. For any node $i \in S - S'$, by definition of global alliance, $|N[i] \cap S'| \geq |N[i] \cap (V - S')|$, or in other words, $|N(i) \cap S'| \geq |N(i) \cap (V - S')| + 1$. Since $S' \subset S$, we have $|N(i) \cap S| \geq |N(i) \cap S'|$, $|N(i) \cap (V - S)| \leq |N(i) \cap (V - S')|$. So $|N(i) \cap S| \geq |N(i) \cap (V - S)| + 1$. □

An example of global offensive alliance in a graph of size 9 is given in figure 1. The black nodes represent the global offensive alliance.

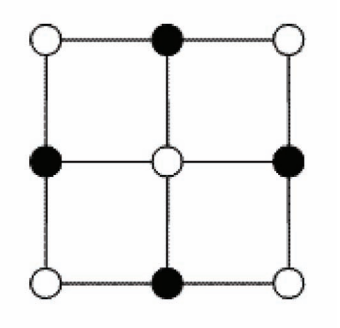

Figure 1. Example: global offensive alliance

2.2. Global Offensive Alliance Algorithm

Each node i maintains a boolean flag F_i to denote whether the node is in the offensive alliance S. When the algorithm terminates, the set of nodes with $F_i = true$ gives the minimal global offensive alliance set S. The algorithm is shown in Figure 2.

The algorithm consists of two rules. A node executes Rule 1 to decide locally if it should enter into the alliance set. Node i goes in the set (by setting flag F_i), if it is not in the set yet and the number of its current attackers is less than the number of its defenders. A node executes Rule 2 to decide if should leave the alliance set. Node i leaves the set (by reseting the flag F_i), if it is currently in the set and the number of its current defenders is greater than or equals to the number of its attackers plus 1, i.e., after i goes out of the set, number of nodes in $N[i]$ belonging to S (i.e., nodes with their flag F set) will be equal to or greater than the others.

2.3. Correctness

Lemma 2 *If the algorithm converges, the set S marked by boolean flag F is a global offensive alliance.*

Proof : When the algorithm converges, for any node i that have $F_i = false$, there must have $|\{j | j \in N[i] \wedge F_j = true\}| \geq |\{j | j \in N[i] \wedge F_j = false)\}|$, otherwise node i will be privileged to move by rule R1. So by definition, S is a global offensive alliance, □

Theorem 1 *When the algorithm converges, the set S marked by boolean flag F is a minimal global offensive alliance.*

Proof : By Lemma 2, S is a global offensive alliance. If S is not minimal, by lemma 1, there exists a node $i \in S$, s.t. $|N(i) \cap S| \geq |N(i) \cap (V - S)| + 1$. Thus this node i will be privileged by rule R2. □

2.4. Complexity

Theorem 2 *The algorithm converges in $2m+n$ steps, where $n = |V|$ and $m = |E|$.*

Proof : In a given system state, call an edge to be an "x-edge" if exactly one of its two ending nodes is in the set S and define an integer X as $X = |\{(u, v) | F_u = true \wedge F_v = false\}|$, i.e., X denotes the number of x-edges.

$$\textbf{R1:} \quad \begin{cases} \textbf{if } F_i = false \wedge |\{j | j \in N[i] \wedge F_j = true\}| < |\{j | j \in N[i] \wedge F_j = false)\}| \\ \textbf{then } F_i := true \end{cases}$$

$$\textbf{R2:} \quad \begin{cases} \textbf{if } F_i = true \wedge |\{j | j \in N(i) \wedge F_j = true\}| \geq |\{j | j \in N(i) \wedge F_j = false)\}| + 1 \\ \textbf{then } F_i := false \end{cases}$$

Figure 2. Minimal Global Offensive Alliance Algorithm

Consider the scenario when node i executes rule R1: before the move there are $|N(i) \cap S|$ many x-edges incident on node i, and after the move, there are $|N(i) \cap (V - S)|$ many x-edges incident on node i. Since $|N[i] \cap S| < |N[i] \cap (V - S)|$, $|N(i) \cap S| \leq |N(i) \cap (V - S)|$. So execution of rule R1 does not decrease X. Similarly, when a node i executes rule R2, there are $|N(i) \cap (V - S)|$ many x-edges incident at node i before the move, and there are $|N(i) \cap S|$ many x-edges after the move. Since $|N(i) \cap S| \geq |N(i) \cap (V - S)| + 1$, execution of rule R2 always increases X. If m denotes the number of edges in the graph G, there can be at most m executions of rule R2. Also note that each execution of R2 decreases $|S|$ by 1 and each execution of R1 increases $|S|$ by 1. Since $|S| \leq n$, so there can be at most $m + n$ number of R1 executions. Consequently, total number of moves made by the algorithm in the worst case is $2m + n$. □

3. Global Defensive Alliance

Definition 2 *Given a graph $G = (V, E)$, a global defensive alliance S is a dominating subset of V, s.t. $\forall v \in S, |N[v] \cap S| \geq |N[v] \cap (V - S)|$.*

A global defensive alliance S is 1-minimal when for any $v \in S$ the set $S - \{v\}$ is not a global defensive alliance. A global defensive alliance S is called minimal when no proper subset of S is a global defensive alliance.

Remark 2 *A global defensive alliance can be 1-minimal but not minimal.*

For example, consider $C_3 \times P_5$ (Figure 3). The set of nodes $\{1, 4, 7, 8, 9, 10, 13\}$ is a 1-minimal global defensive alliance; however, this set is not minimal, because its subset $\{1, 4, 7, 10, 13\}$ is also a global defensive alliance.

Our objective is to develop a self-stabilizing algorithm for a 1-minimal global defensive algorithm. Even when we do not require the set to be minimal, there seems to be no simple way to design a self-stabilizing algorithm where the nodes take action based on local knowledge of its immediate neighbors. By definition, a global defensive alliance needs to be dominating and defended; thus, a node cannot enter into a locally legitimate state by changing its own state alone. Consider a node where the nodes in its closed neighborhood is not in the global defensive alliance set. If this node stays out of set, it is not dominated. If this node moves into the set, it is not defended. Hence the node cannot make decision by only the distance one information. We use the interesting lock/unlock protocol proposed in [14] to design our algorithm for 1-minimal global defensive alliance.

3.1. Algorithm

Each node i maintains following variables:

- A boolean flag F_i; $F_i = 1$ indicates that node i belongs to the defensive alliance S [When the algorithm terminates, nodes with $F_i = true$ gives the 1-minimal global defensive alliance set S.
- A pointer variable P_i that serves as an exclusive lock. A node i gets the lock iff all nodes in its neighborhood is pointing to it.
- An integer variable a_i that stores the number of defensive alliance nodes in the neighborhood of node i, i.e., the number of nodes $j \in N(i)$ with $F_j = 1$. We use $b(i)$ to denote $|\{j | j \in N(i) \wedge F_j = true\}|$. At any system state, the stored variable a_i at node i may not be equal to $b(i)$; thus, when node i makes a move, it updates a_i to $b(i)$ to reflect the most recent changes.

Definition 3 *For any node i, the function $minn(i)$ is defined to be the minimum node in the open neighborhood of node i that points to itself, i.e., $minn(i) = min\{j | j \in N(i) \wedge P_j = j\}$. Also, $min\ \phi = null$.*

Definition 4 *A Boolean function $h(i)$ is defined to be*

the anticipated value of F_i as

$$h(i) \stackrel{\text{def}}{=} \begin{cases} 1 & \textbf{\textit{if}}\ (F_i = 0\ ) \bigwedge ((\forall j \in N(i) | F_j = 0) \\ & \vee\ (\exists j \in N(i) | F_j = 1\ \wedge a_j + 1 \\ & <\ deg(j) - a_j)) \\ 0 & \textbf{\textit{if}}\ (F_i = 1\ ) \wedge (\exists j \in N(i) | F_j = 1\ ) \\ & \wedge\ (\forall j \in N(i) | F_j = 1\ \wedge a_j \\ & \geq deg(j) - a_j + 1) \\ F_i & \textbf{\textit{otherwise}} \end{cases}$$

Remark 3 *Consider the example shown in Figure 4, where the black nodes are those with flag $F = 1$ and the white nodes are those with flag $F = 0$. Assume that for each node i, a_i has been updated to $b(i)$, the correct number of black nodes in the neighborhood $N(i)$ of node i.*

- *If $F_i = 0$, and none of the nodes in the close neighborhood of node i is in the global defensive alliance, i is anticipated to enter the alliance (since global defensive alliance is a dominating set). Node 6 in Figure 4 is an example.*
- *If $F_i = 0$, and some neighbors of node i are already in the alliance but do not have sufficient nodes to protect them, i is also anticipated to enter the alliance; node 3 is an example (note that node 2 is the neighboring node that is not defended).*
- *f $F_i = 1$, and i is adjacent to some other nodes in the alliance, and removing i from the alliance set will not leave its neighboring alliance node unprotected, then i is anticipated to leave the alliance (to strive for minimality). In the example, if node 4 is switched to white color, its black neighbor, node 7, is still defended, hence $h(4) = 0$*

Definition 5 *We define a Boolean function $\psi(i)$ as follows:*

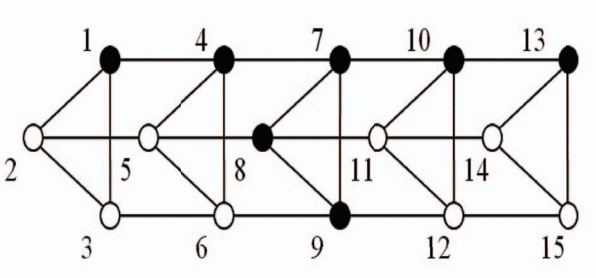

Figure 3. Example: 1-minimal global defensive alliance

$$\psi(i) \stackrel{\text{def}}{=} \begin{cases} |\{j | j \in N(i) \wedge F_j = 1\}| \geq 1 & \textbf{\textit{if}}\ F_i = 0 \\ |\{j | j \in N(i) \wedge F_j = 1\}| + 1 & \\ \quad \geq deg(i) - |\{j | j \in N(i) & \\ \quad \wedge F_j = 1\}| & \textbf{\textit{if}}\ F_i = 1 \end{cases}$$

Remark 4 *When node i is not in the alliance set, i.e., $F_i = 0$, $\psi(i)$ is defined as whether i is dominated by some nodes in the alliance set; similarly, when i is in the alliance set, i.e., $F_i = 1$, $\psi(i)$ is defined as whether i is protected by its alliance node neighbors and itself.*

Definition 6 *The global legitimate system state is defined to be a system state when $\psi(i) = true$ for all nodes in the graph, i.e., the set of nodes with $F_i = 1$ defines the global defensive alliance set of the graph.*

The self-stabilizing algorithm is shown in figure 5. We make the following observations.

- The rule R1 updates the variable a_i so that i reveals the most recent changes of the status of its neighboring nodes. This update is performed every time a node makes a move.
- A node executing rule R2 asks for the lock and gets it only when all its neighbors are pointing to null.
- When a node i is not holding the lock (i.e., not pointing to itself, it executes rule R3 to point to the minimum possible node in its neighborhood that is holding the lock, if any.
- If a node i holds the lock (i.e., $P_i = i$) but there exists a lower neighbor with a lock, node i releases the lock and points to that neighbor by executing rule R4.

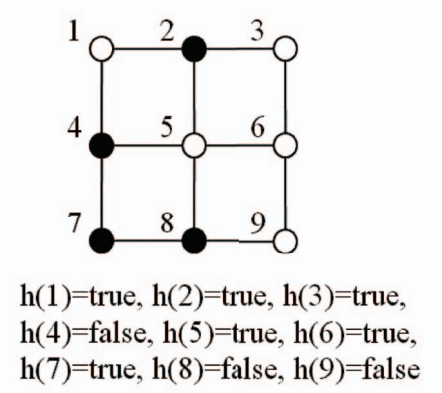

Figure 4. Example: Anticipated value of F_i

$$
\begin{array}{ll}
\textbf{R1:} & \left\{ \begin{array}{l} \textbf{if } a_i \neq b(i) \\ \textbf{then } a_i := b(i) \end{array} \right. \\
\textbf{R2:} & \left\{ \begin{array}{l} \textbf{if } F_i \neq h(i) \wedge P_i = null \wedge \forall j \in N(i) \rightarrow P_j = null \\ \textbf{then } P_i := i, a_i := b(i) \end{array} \right. \\
\textbf{R3:} & \left\{ \begin{array}{l} \textbf{if } P_i \neq i \wedge P_i \neq minn(i) \\ \textbf{then } P_i := minn(i), a_i := b(i) \end{array} \right. \\
\textbf{R4:} & \left\{ \begin{array}{l} \textbf{if } P_i = i \wedge \exists j \in N(i) \rightarrow P_j = \ell < i \\ \textbf{then } P_i := minn(i), a_i := b(i) \end{array} \right. \\
\textbf{R5:} & \left\{ \begin{array}{l} \textbf{if } P_i = i \wedge \forall j \in N(i) \rightarrow P_j = i \\ \textbf{then } F_i := h(i), P_i := null, a_i := b(i) \end{array} \right.
\end{array}
$$

Figure 5. 1-Minimal Global Defensive Alliance Algorithm

- By executing rule R5, node i changes its F_i only when i has the lock, i.e. $P_i = i \wedge \forall j \in N(i) \rightarrow P_j = i$.
- If node i makes a move by R3 or R4 and changes its pointer P_i to j, then $j = minn(i) = min\{k | k \in N(i) \wedge P_k = k\}$. Therefore, only the nodes that point to themselves can get the lock.
- After i executes R5, $P_i = null$ and $\forall j \in N(i)$, we have $P_j = i$. So for P_i to change again, either (1) all j's make move and set $P_j = null$, or (2) some j's make move and set $P_j = j$, so that $minn(i)$ is no longer null. In the first case, since $minn(j) \neq null$, j has to make move by R5 to set $P_j = null$. In the second case, j has to first make move by R5, then make move by R2 to set $P_j = j$. In both cases, after i moves by R5, node i cannot make the next execution of rule R5 until all its neighbors have executed R5 at least once.

3.2. Correctness

We use similar arguments as used in [14], to prove the correctness of the proposed defensive alliance algorithm.

Lemma 3 *When the protocol terminates, $\psi(i)$ is true at each node, i.e., the system is in a legitimate state.*

Proof : The flag variable F_i and consequently $\psi(i)$ at any node can be changed only by execution of rule R5. If $\psi(i)$ is true on all the nodes, $h(i) = F_i$ is also true on all the nodes. So no node will change F_i by R5, hence $\psi(i)$ will not be changed. □

Theorem 3 *When the protocol terminates, the set $S = \{j | F_j = 1\}$ is a 1-minimal global defensive alliance.*

Proof : By Lemma 3, S is a global defensive alliance. If S is not 1-minimal, there exists a node $i \in S$, such that setting $F_i = false$ still gives a global defensive alliance. i.e. i is adjacent to some nodes in S and for any $j \in S$ adjacent to i, $|N(j) \cap (S - \{i\})| + 1 \geq |N(j) \cap (V - S + \{i\})|$. Since every move updates a_i to $b(i)$, this can be rewritten as $a_j \geq deg(j) - a_j + 1$. Thus $\psi(i) = false$ on this node i, system is not converged. □

3.3. Complexity

Lemma 4 *Starting from any illegitimate state, rule R5 will be executed at most $2n$ times.*

Proof : Only R5 will change F_i. For a given node i, if it sets F_i to false, all its adjacent nodes j that have $F_j = false$ will not change F_j to true again. Hence each node can make R5 move at most twice. G contains n nodes, therefore at most $2n$ R5 moves. □

Theorem 4 *The algorithm converges in $O(n^3)$ time steps.*

Proof : Between two consecutive R5 moves, a node i need to point to it self to get the lock. However, if some neighbor is pointing to smaller id node, i will change its pointer to $minn(i)$. Hence, i can change its point to different $minn(i)$ at most n times. Therefore between two consecutive R5 moves, there can be at most $2n$ moves on any node i. Since all the rules update a_i, R1 will only be executed at the very first time, hence at most once. Therefore the complexity of the algorithm is $O(2n \times 2n \times n) = O(n^3)$. □

4. Conclusion

We have proposed two self-stabilizing algorithms for global defensive and offensive alliances in a network graph. Graph theoretic optimization problems are useful for such dynamic networks; fault tolerant distributed protocols for such problems provide the key resources for designing wireless, sensor and ad hoc networks and they offer new insight into the fundamental role of discrete distributed algorithms in developing these real life applications [15].

5. Acknowledgment

The work was supported by a NSF grant # ANI-0073409.

References

[1] O. Favaron, G. Fricke, W. Goddard, S. M. Hedetniemi, S. T. Hedetniemi, P. Kristiansen, R. C. Laskar, and D. Skaggs. Offensive alliances in graphs. In *17th Internatioanl Symposium of Computer Information Science*, pages 298–302, Orlando, FL, USA, October 2002.

[2] G. H. Fricke, L. M. Lawson, T. W. Haynes, S. M. Hedetniemi, and S. T. Hedetniemi. A note on defensive alliances in graphs. *Bulletin of the Institute of Combinatorics and its Applications*, 38:37–41, 2003.

[3] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. Global defensive alliances in graphs. *The Electronic Journal of Combinatorics*, 10, 2003.

[4] P. Kristiansen, S. M. Hedetniemi, and S. T. Hedetniemi. Introduction to alliances in graphs. In *17th International Symposium of Computer Information Science*, pages 308–312, October 2002.

[5] S. M. Hedetniemi, S. T. Hedetniemi, and P. Kristiansen. Alliances in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 48:157–177, 2004.

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

[7] L. Lamport. Solved problems, unsolved problems, and non-problems in concurrency. In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*, pages 1–11, 1984.

[8] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25(1):45–67, March 1993.

[9] T. Herman. A comprehensive bibliograph on self-stabilization, a working paper. *Chicago J. Theoretical Comput. Sci.*, http://www.cs.uiowa.edu/ftp/selfstab/bibliography.

[10] SC Hsu and ST Huang. Analyzing self-stabilization with finite-state machine model. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 624–631, 1992.

[11] W. Goddard, S.T. Hedetniemi, D.P. Jacobs, and P.K. Srimani. The b-matching paper. Preprint.

[12] SK Shukla, DJ Rosenkrantz, and SS Ravi. Observations on self-stabilizing graph algorithms for anonymous networks. In *Proceedings of the Second Workshop on Self-Stabilizing Systems*, pages 7.1–7.15, 1995.

[13] S. M. Hedetniemi, S. T. Hedetniemi, D. P. Jacobs, and P.K. Srimani. Self-stabilizing algorithms for minimal dominating sets and maximal independent sets. Presented at SouthEastern, 2001.

[14] M. Gairing, W. Goddard, S. T. Hedetniemi, P. Kristiansen, and A. A. McRae. Distance-two information in self-stabilizing algorithms. *Parallel Processing Letters*, 2004.

[15] Christian Boulinier, Franck Petit, and Vincent Villain. When graph theory helps self-stabilization. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing (PODC 2004), St. John's*, pages 150–159 , 2004.