Theoretical Computer Science 593 (2015) 132–138

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A theorem of Ore and self-stabilizing algorithms for disjoint minimal dominating sets

Stephen T. Hedetniemi [a], David P. Jacobs [a,*], K.E. Kennedy [b,1]

[a] School of Computing, Clemson University, Clemson, SC 29634, USA
[b] Department of Computer Science, Southern Wesleyan University, Central, SC 29630, USA

ARTICLE INFO

Article history:
Received 10 February 2012
Received in revised form 1 June 2015
Accepted 2 June 2015
Available online 9 June 2015
Communicated by G. Ausiello

Keywords:
Graph
Self-stabilizing algorithm
Minimal dominating set
Maximal independent set

ABSTRACT

A theorem of Ore [20] states that if D is a minimal dominating set in a graph $G = (V, E)$ having no isolated nodes, then $V - D$ is a dominating set. It follows that such graphs must have two disjoint minimal dominating sets R and B. We describe a self-stabilizing algorithm for finding such a pair of sets. It also follows from Ore's theorem that in a graph with no isolates, one can find disjoint sets R and B where R is maximal independent and B is minimal dominating. We describe a self-stabilizing algorithm for finding such a pair. Both algorithms are described using the Distance-2 model, but can be converted to the usual Distance-1 model [7], yielding running times of $O(n^2 m)$.
© 2015 Elsevier B.V. All rights reserved.

1. Introduction

A distributed system can be modeled as an undirected graph $G = (V, E)$ of order $n = |V|$ having no loops or multiple edges, where V is the set of *nodes* and E is the set of *edges*, $m = |E|$. If $i \in V$, its *open neighborhood* $N(i)$ is the set of nodes adjacent to i, while its *closed neighborhood* is $N[i] = N(i) \cup \{i\}$. Nodes $j \in N(i)$ are called *neighbors* of i. A set $S \subseteq V$ is called *independent* if no two nodes in S are adjacent. A set $S \subseteq V$ of nodes is called a *dominating set* if every node $v \in V - S$ is adjacent to at least one node in S. It is well known that every maximal independent set is a minimal dominating set. It is also well known that a set is maximal independent if and only if it is independent and dominating [11].

Self-stabilization is a paradigm for distributed systems that was introduced by Dijkstra [3,4], that allows the system to achieve a desired global state, even in the presence of faults. A fundamental idea of the paradigm is that no matter what global state the system finds itself in, after a finite amount of time the system will reach a correct and desired global state. In a self-stabilizing algorithm, each node maintains a set of local variables. It can make decisions based only on its local variables and typically on its neighbors' local variables. The values of a node's local variables constitute its *local state*, and the system's *global state* is the union of all local states. A node i may change its local state by making a *move*, i.e., changing the value of at least one of its local variables. Self-stabilizing algorithms are often described as a set of rules of the form

if $p(i)$ **then** S

where $p(i)$ denotes the value of a predicate at node i, and S is a sequence of statements. A node i is said to become *privileged* if at least one predicate $p(i)$ in one rule is true. When a node becomes privileged, it is eligible to execute

* Corresponding author. Fax: +1 864 656 0145.
E-mail addresses: hedet@clemson.edu (S.T. Hedetniemi), dpj@clemson.edu (D.P. Jacobs), Ken.Kennedy@bmwgroup.com (K.E. Kennedy).
1 Current address: BMW Group, Greenville, SC, USA.

http://dx.doi.org/10.1016/j.tcs.2015.06.004
0304-3975/© 2015 Elsevier B.V. All rights reserved.

the corresponding statements. In this paper, our model assumes a serial daemon or scheduler. That is, exactly one node is allowed to move among the set of privileged nodes. It should be noted that other scheduling models exist, allowing simultaneous nodes to move. We say the system has *stabilized* when no nodes are privileged.

Normally, self-stabilizing algorithms are formulated for the Distance-1 model, in which nodes can utilize state information of only their immediate neighbors. Sometimes, however, it is convenient to assume the Distance-2 model, in which nodes can utilize state information of nodes *distance two* away. As shown in [7], an algorithm in this model can be translated to the Distance-1 model, with a slowdown of $O(m)$.

Theorem 1. *If S is a Distance-2 algorithm which stabilizes in A moves under the serial daemon, then S may be implemented as an algorithm in the Distance-1 model which stabilizes in $O(mA)$ moves.*

Thus, the Distance-2 model permits a higher-level description of an algorithm, at a polynomial-time cost. In this paper we formulate our algorithms using the Distance-2 model. A more general treatment involving Distance-k algorithms, for arbitrary k, can be found in [8]. We refer the reader to [5] for a general treatment of self-stabilizing algorithms.

A number of self-stabilizing algorithms have been published for finding minimal dominating sets in graphs. An excellent survey of self-stabilizing algorithms involving domination and independence has been published by Guellati and Kheddouci [10].

Few papers have been published on self-stabilizing algorithms for finding *disjoint* sets R and B, having prescribed properties. This paper is a follow-up to [14], which gave several self-stabilizing algorithms for constructing pairs of disjoint independent sets. In 1962, Ore [20] published this now-classical theorem.

Theorem 2. *In a graph having no isolated nodes, the complement of any minimal dominating set is a dominating set.*

Corollary 1. *A graph having no isolated nodes contains disjoint sets R and B, each of which is a minimal dominating set.*

Corollary 2. *A graph having no isolated nodes contains disjoint sets R and B, where R is a maximal independent set, and B is a minimal dominating set.*

Several papers have been written on disjoint dominating sets in graphs [1,2,6,13,15,18,22]. The study of disjoint dominating sets in graphs has applications to heterogeneous multiagent systems in which one seeks to allocate at most one type of resource to each node of a network so that every node has access to at least one of each type of resource in its closed neighborhood [19]. The maximum number of different resources that can be allocated in this way is called the *domatic number* of a graph, and is denoted $d(G)$. Ore's theorem asserts that for any graph having no isolated nodes, $d(G) \geq 2$, that is, one of two different types of resources can be allocated to each node of a such graph, so that each node has access to both types of resources. The domatic number is therefore a measure of the capacity of a network to accommodate different types of resources so that every node can easily access each type of resource. A thorough discussion of the domatic number of a graph is given by Zelinka in Chapter 13 of [12].

Throughout this paper, we will assume that G is a graph of order $n \geq 2$, with m edges, having no isolated nodes. In Section 2 we give a self-stabilizing algorithm for finding the sets guaranteed in Corollary 2. In Section 3 we give a self-stabilizing algorithm for finding the sets guaranteed in Corollary 1.

2. Maximal independent–minimal dominating

In this section we give Algorithm 1 for finding disjoint sets R and B, where R is a maximal independent set and B is a minimal dominating set. Each node i has two binary variables $r(i)$ and $b(i)$. We denote $R = \{i | r(i) = 1\}$, and $B = \{i | b(i) = 1\}$.

Note that Algorithm 1 has five rules. Let us say a move is *red* if it executes either **RIN**, **RB**, or **ROUT**. We will say a move is *blue* if it executes **BIN** or **BOUT**.

If i makes a red move, it will leave $b(i) = 0$. A node executes rule **RIN** to enter R and executes rule **ROUT** to leave R. Rules **RIN** and **ROUT** require ordinary Distance-1 information, and mirror the standard algorithm for finding a maximal independent set, first proposed in [21], but also studied under other models [9,17]. A node not in R can execute **RIN** if it has no neighbor in R. A node in R can execute **ROUT** if it has a neighbor in R. Rule **RB** ensures that R and B will be disjoint.

A node i can make a blue move only if $r(i) = 0$, and after the move, $r(i)$ will remain zero. A node executes **BOUT** to leave set B. It does so only if, after leaving B, all its neighbors, *and* i itself, will remain dominated by B. A node executes **BIN** to enter set B. It does so if either a neighbor j, or i itself, is not dominated by B. Both blue moves require Distance-2 information because a node must read the state information in the neighborhood of each neighbor.

Lemma 1. *If Algorithm 1 stabilizes, then $R \cap B = \emptyset$.*

Proof. A node $i \in R \cap B$ would be privileged to execute rule **RB**. □

Download English Version:

https://daneshyari.com/en/article/433853

Download Persian Version:

https://daneshyari.com/article/433853

Daneshyari.com