JOURNAL OF COMPUTER AND SYSTEM SCIENCES 26, 139–152 (1983)

Network Structure and the Firing Squad Synchronization Problem

JOHN J. GREFENSTETTE

Computer Science Department, Vanderbilt University, Nashville, Tennessee 37235

Received July 2, 1982

The *firing squad synchronization problem*, or *fssp*, requires that a network of automata, limited to finite memory and local communications only, cooperate in a global task. Previous solutions to the fssp usually assume a certain network topology. This paper presents *embeddable* solutions which exploit the existing network topology without relying on a priori assumptions. A uniform lower bound on the firing time of embeddable solutions is derived, and optimal embeddable solutions are presented for several classes of networks, including rings, star graphs, flower graphs, and n-dimensional rectangular arrays. In addition, we address the question, to what extent can solutions to the fssp for subnetworks contribute to the overall solution?

1. INTRODUCTION

Given the current trends in hardware costs, it is safe to anticipate the proliferation of large networks of processing elements whose logical configurations (number of processors, interconnection patterns) may change dynamically. Under such conditions, it may not be desirable, or even possible, to notify each node of the current network configuration. These considerations motivate the question: How much does each node in a network need to know about the global structure of the network? We pursue this question with respect to a well-known model of distributed computation comprising a finite graph in which each node has an identical copy of a finite automaton. Each node is connected to no more than d neighbors. The network operates as follows: at each time step, each local copy of M performs a state transition based on its own state and the state of its d neighbors. All nodes perform state transitions synchronously. (The preceding assumptions do not, in general, hold for current computer networks. They do hold, however, for networks of processing elements within a VLSI circuit, or within a network of VLSI chips [8].) Early work along these lines appears in Rosenstiehl *et al.* [11], which shows that local finite state automata can solve many network problems, such as finding spanning trees and Hamiltonian cycles and solving the firing squad problem, without any prior knowledge of the network topology. Wu and Rosenfeld [14] present networks of automata which measure properties of the underlying communications graph, such as radius, diameter. Rosenstiehl *et al.* [11] and Angluin [1] show that networks of finite state automata cannot answer certain questions about the underlying graph, or perform certain networks tasks, e.g., finding a distinguished "center" node, unless the topology is restricted, e.g., trees.

This paper asks a particular version of the general question with respect to the well-known *firing squad synchronization problem*, or *fssp*: At time $t = 0$, all nodes in the network are in a quiescent *soldier* state except one, the *General*. It is assumed that quiescent soldiers whose neighbors are all quiescent soldiers remain quiescent. It is required that at some time $t > 0$, each node enters a special *firing* state simultaneously and for the first time. The fssp for a class of networks $\mathscr{G}$ is to design a finite state automaton M such that, if a copy of M is placed at the nodes of any network in $\mathscr{G}$, then that network behaves according to the problem specification. The fssp abstracts features from many practical network problems. For example, it may be desirable to dynamically switch protocols in order to respond to a network condition (e.g., a fault is detected in the central node), and, in order to maintain consistent queues, it may be crucial that all nodes begin exercising a new protocol simultaneously.

The fssp has been solved many times for different classes of networks. Balzar [2] and Waksman [13] present minimum time solutions for the class of one-dimensional arrays with the General at one end. The solution for one-dimensional arrays with the General at an arbitrary position appears in Moore and Langdon [9]. Beyer [3] and Shinahr [12] provide solutions for two- and three-dimensional rectangular arrays. Herman *et al.* [6] synchronizes growing cellular arrays (L systems). Romani [10] and Rosenstiehl *et al.* [11] extend the solutions to arbitrary degree-bounded networks. It is not the primary purpose of this paper to add to the literature of fssp solutions. The fssp was chosen precisely because of the volume and variety of published solutions. It is our intention to explicate the extent to which the various solutions depend on a priori assumptions about the network topology, and thus to impose some structure on this body of literature.

Even solutions which do not *assume* a given network topology should *detect* special structures which may allow more efficient solutions. Since a solution to the fssp is required to be a finite state automaton, the number of neighbors which any node can have is bounded by a constant which depends on the local automaton. An automaton M which solves the fssp for all networks with degree d or less is called a *d-universal* solution. The d-universal solutions in [10, 11] are relatively insensitive to the topology of the networks to which they are applied. For any network with n nodes, both of these solutions always provide $O(n)$ firing time, even when applied to symmetrical networks such as rectangular or cubic arrays. In contrast, a solution for rectangular arrays in [3] takes $O(n^{1/2})$ steps, and the solution for cubic arrays in [12] takes $O(n^{1/3})$ steps. Clearly, the d-universal solutions of [10] and [11] might take much longer than necessary on any given network. This paper explores the conditions under which d-universal solutions to the fssp can detect the properties of the underlying network and exploit these properties to obtain fast yet general solutions.

The remaining sections are organized as follows: Section 2 contains the definition of our network model. Also, we motivate the need for a careful design strategy by showing that the general analysis of proposed solutions to the network fssp is intractable. Section 3 describes the embedding technique and characterizes the embeddable solutions to the fssp (Theorem 1). In Section 4, Theorem 2 provides a uniform lower bound on the time required to synchronize any network by an embeddable solution. Embeddable solutions for several classes of networks, including rings, star graphs, flower graphs, and n-dimensional rectangular arrays, appear in Section 5. All of these solutions achieve the lower bound firing time specified by Theorem 2. A shortcoming of many of the solutions to the fssp is that small differences in network structure may result in large differences in synchronization time. Section 6 addresses this problem by examining ways in which solutions for subnetwork structures provide an upper bound for global solutions. Section 7 contains concluding remarks.

2. Definitions and Preliminaries

An *automaton network* is a 5-tuple $N = \langle d, G, \omega, \alpha, M\rangle$, where d is the *degree* of the network; G is a finite undirected, connected graph, called the *communication graph*, $G = \langle V, E\rangle$ of degree d, with nodes V and edges E. Let $E_u = \{(u, v) \in E\}$ be the set of edges having u as the first term; note $|E_u| \leqslant d$. ω is a special *null node* not in V, let $R = V \cup \{\omega\}$. α is an *adjacency function* $\alpha: V \to R^d$ such that for any $v \in V$, if $\alpha(v) = \langle u_1, u_2,..., u_d\rangle$, then $\{(v, u_i) | u_i \neq \omega, 1 \leqslant i \leqslant d\} = E_v$. Then M is a finite state automaton of degree $dM = \langle \Sigma, \delta, \lambda\rangle$ such that Σ is a finite set of *states*, λ is a special *null state* not in Σ, and δ is a *transition function*

$$\delta: \Sigma \times Q^d \to \Sigma,$$

where $Q = \Sigma \cup \{\lambda\}$.

A *configuration* C of an automaton network N is a mapping $C: R \to Q$ such that $C(u) = \lambda$ iff $u = \omega$. The mapping δ is extended to configurations as follows: Let C and C' be two configurations of N. Then $\delta(C) = C'$ iff for each $v \in V$, if $q = C(v)$ and $\alpha(v) = \langle u_1, u_2,..., u_d\rangle$, and $q_i = C(u_i)$ for $1 \leqslant i \leqslant d$, then $\delta(q, q_1,..., q_d) = C'(v)$.

Informally, an automaton network N comprises a graph G such that each node contains a copy of a finite state machine M. Each node is connected to no more than d neighbors. We can think of α as a function which labels the d I/O ports of each machine. (All automata described in this paper will work correctly for any legal adjacency function.) If a node has fewer than d neighbors, some of its ports will be labeled "ω." The state of such a "virtual" neighbor is λ. The network operates as follows: at each time step, each copy of M performs a state transition based on its own state and the state of its d neighbors. All nodes perform state transitions synchronously, which produces another network configuration. In the sequel, we will usually use this informal description of the operation of an automaton network.

Note that, except for the parameter d, the automaton M of an automaton network is independent of the underlying communication graph G. We are interested in how a particular automaton M behaves on a number of networks, and, in particular, whether a given automaton solves the network fssp for an entire class of communication graphs. The following notation will be useful: A *synchronization machine* is an automaton M in which the set of states Σ contains three distinguished states, the *quiescent state* I, the *firing state* F, and the *fire-when-ready state* X, and in which δ obeys the constraint

$$\delta(I, q_1, ..., q_d) = I$$

if $q_i = I$ or $q_i = \lambda$, for $1 \leqslant i \leqslant d$. A synchronization machine meets the constraints specified in the statement of the fssp that a quiescent node which is surrounded by quiescent nodes must remain quiescent. Given a class of graphs $\mathscr{G}$ and a synchronization machine S, we say that S *solves the fssp for* $\mathscr{G}$ iff, for any automaton network N with communication graph $G \in \mathscr{G}$, automaton S, and any legal adjacency function α, when N is put into an initial configuration with all nodes quiescent except one, which is in the fire-when-ready state, then at some time $t > 0$, N will enter a configuration in which all nodes are in the firing state F and, furthermore, no node enters state F before time t. If S solves the fssp for $\mathscr{G}_d$, where $\mathscr{G}_d$ is the class of all connected graphs with degree d or less, we say that S is a *d-universal solution* to the fssp.

A primary motivation for studying design techniques for solutions to network problems is the lack of general analytic techniques. It has been known since the early work of Hennie [5] that many simple questions about the behavior of systems of parallel automata are undecidable. For example, it is undecidable whether an arbitrary synchronization machine solves the fssp for all linear networks. This is easily shown by embedding an arbitrary Turing machine (TM) computation into a solution of the fssp such that the solution fires correctly on all linear networks iff the TM halts on an empty tape [4].

Given that there is no general procedure for analyzing proposed solutions to the network fssp, the designer of a new solution must assume the responsibility for finding a proof that the proposed solution is correct. One reasonable strategy is to extend a known solution to a more general class of networks and modify the previously known proof so as to apply in the more general case. The next several sections explore this strategy with respect to the fssp.

3. Embedded Solutions to the fssp

Let $S_1, S_2, ..., S_n$ be solutions to the fssp for the network classes $\mathscr{G}_1, \mathscr{G}_2, ..., \mathscr{G}_n$, respectively. For example, S_1 might be a solution to the fssp for the class $\mathscr{G}_1$ of linear arrays. And S_2 might be a solution for the class $\mathscr{G}_2$ of star graphs. We would like to embed these solutions into one solution S, which (a) solves the fssp for all networks in $\bigcup_{i=1}^{n} \mathscr{G}_i$ and (b) fires as quickly as S_i for any network in $\mathscr{G}_i$. A straightforward attempt to form S is as follows: Define the states of S to be n-tuples $\langle a_1, a_2, ..., a_n \rangle$, where a_i is a state of S_i. Define the state transition function of S so that each step simulates the transition functions of $S_1, S_2, \ldots, S_n$ in parallel. Whenever any one of the simulated solutions fires, S also fires.

There are two problems with this approach. First, since S is intended to apply to a more general class of networks than S_i, the transition function for S_i may not be defined for some networks to which S is applied. We can avoid this difficulty by extending the transition function for S_i: Let d be any positive integer. Let d_i be the degree of $S_i = \langle \Sigma_i, \delta_i, \lambda \rangle$. The *d version of* S_i, denoted $S_{i,d}$, is defined to be a finite state automaton $S_{i,d} = \langle \Sigma_{i,d}, \delta_{i,d}, \lambda \rangle$, such that $\Sigma_{i,d} = \Sigma_i \cup \{q'\}$, where $q' \notin \Sigma_i$, and $\delta_{i,d}$ is defined as follows:

(a) if $d \leqslant d_i$, then $\delta_{i,d} = \delta_i$. Otherwise,

(b) $\delta_{i,d}(I, q_1, \ldots, q_d) = I$ if, for $1 \leqslant i \leqslant d$, $q_i = I$ or $q_i = \lambda$, where λ is the null state of S_i,

(c) if $\delta_i(q, q_1, \ldots, q_{d_i}) = q_j$, then $\delta_{i,d}(q, \bar{q}_1, \ldots, \bar{q}_d) = q_j$ for all d-tuples $\langle \bar{q}_1, \ldots, \bar{q}_d \rangle$ which can be obtained from $\langle q_1, \ldots, q_{d_i} \rangle$ by inserting $d - d_i$ λ's at arbitrary positions,

(d) $\delta_{i,d}(q, q_1, \ldots, q_d) = q'$, otherwise.

Informally, the *d version of* S_i is a finite state automaton such that (1) a quiescent node surrounded by quiescent nodes remains quiescent, (2) at any node which has no more than d_i non-null neighbors, the d version of S_i behaves just like S_i, and (3) any nonquiescent node which has more than d_i non-null neighbors enters a special state, which reflects the fact that S_i does not specify what to do in this case. By using the parallel simulation technique on the d versions of $S_1, \ldots, S_n$, we can construct a well-defined solution S which simulates any set $\{S_1, S_2, \ldots, S_n\}$ of solutions to the fssp.

The second problem with this approach arises if any of the simulated solutions *misfires* (enters a network state in which some, but not all, nodes enter firing states). This possibility occurs when some of the known special case solutions to the fssp are applied to network structures for which they were not designed. For example, the solution for rectangular arrays in [3] will misfire if a single extra node is added to the lower right corner of the array. This motivates the following definition: A solution S to the fssp is *embeddable* iff, for any network N with degree d, when the d version of S is applied to N, either each node of N fires simultaneously, or no node of N ever fires. Embeddable solutions can be used to solve the problem posed at the beginning of this section.

THEOREM 1. *Given any finite set* $\{S_1, S_2, \ldots, S_n\}$ *of embeddable solutions to the* fssp, *there is an embeddable solution S such that, for* $1 \leqslant i \leqslant n$, *if* S_i *causes a network N to fire in t steps, then S also causes N to fire in no more than t steps.*

Proof. Let d be the maximum degree of S_i, for $1 \leqslant i \leqslant n$. Let each state of S consist of an n-tuple $\langle a_1, a_2, \ldots, a_n \rangle$, such that a_i is a state of the d version of S_i. Define the transition function δ of S to simulate the transitions of the d versions of $S_1, \ldots, S_n$ in parallel. If any of the simulated solutions enters a firing state, then so does S.

Suppose S_i causes a network N to fire in t steps. By the construction of S, S also causes N to fire by time t. Suppose that S causes some node A in network N to fire. Then, for some i, $1 \leqslant i \leqslant n$, S_i causes A to fire in N. Since S_i is embeddable, S_i causes every node in N to fire at the same time as A. Hence, if S causes any node in N to fire, S causes every node in N to fire simultaneously. Hence, S is embeddable. ▮

It should be noted that the increased speed and flexibility of the solution S in Theorem 1 comes at a cost of an increase in memory for the individual nodes which is linear in n, the number of embedded solutions.

4. Minimal-Time Embeddable Solutions

The additional assumption of embeddability allows us to prove a uniform lower bound for the firing time of embeddable solutions to the firing squad synchronization problem. The *firing radius* r is the maximum distance between the general G and any other node. If $|GA| = r$, then A is a *radial node.* The *radial diameter* s is the maximum distance between any radial node and any other node. (Note that no solution to the fssp can synchronize a network in fewer than r steps.)

Theorem 2. *For any network N with firing radius r and radial diameter s, the minimal firing time on N for any embeddable solutions to the* fssp *is $r + s$.*

Proof. Let S be a embeddable solution to the fssp. Suppose that S fires in time $t < r + s$ for some d-degree network N, where r is the firing radius of N and s is the radial diameter of N. Let A and B be nodes of N such that $|GA| = r$ and $|AB| = s$. Now, obtain a network N' from N by replacing A by a node A' which, in addition to the neighbors of A, in connected to a string of new nodes of length s. See Fig. 1. Note that N' has degree at most $d + 1$. Now apply the $(d + 1)$-degree version of S to the network N'. Since $|GA'| = r$ and $|A'B| = s$, B is unaffected by the new neighbors of A' until time $r + s$. That is, B performs the same transitions in N' as it does in N at least until time $r + s$. Hence B fires in N' at time t. However, since $|GC_s| = |GA'| + |A'C_s| = r + s$, node C_s is still quiescent at time t. Hence, C_s does not fire at time t. This contradicts the assumption that S is an embeddable solution to fssp. ▮

This result provides some needed structure to the theory of synchronization. For certain classes of networks, such as linear arrays [9], rings [11], star graphs [10, 11], and flower graphs [10], the published solutions all fire in time $r + s$. For other classes of networks, such as rectangular arrays [3, 12], the minimal firing time is shown to be between r and $r + s$. Theorem 2 says that the essential difference between those

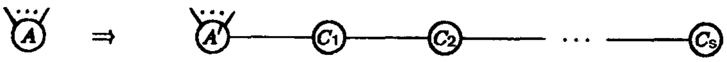

Figure 1

solutions which fire at $r+s$ and those which fire in less time is that the latter inherently rely on a priori knowledge of the network topology.

5. Examples of Optimal Embeddable Solutions

In this section, we describe minimal firing time embeddable solutions for several classes of networks. Our descriptions of solutions are on an intuitive level; the interested reader should have no difficulty in providing the necessary details of any particular solution. For several classes of networks, such as linear arrays [9], rings [10], star graphs [11], and flower graphs [12], the previously published fssp solutions have firing time of $r+s$, where r is the firing radius and s is the radial diameter. All such solutions can be made embeddable by having the original signal from the General vanish at any node which indicates that the network does not have the expected structure. For example, the Romani solution for rings [10] can be made embeddable by modifying the original signal such that it vanishes if it encounters a node which does not have exactly two non-null neighbors. Similar modifications can be made in the case of linear arrays, star graphs, and flower graphs. The modified solutions still achieve the firing time of $r+s$ for the appropriate networks, so the modified solutions are optimal embeddable solutions by Theorem 2.

For some classes of highly symmetrical networks, such as n-dimensional rectangular arrays, fssp solutions are known which fire in less time than $r+s$. We present two approaches to finding optimal embeddable solutions for such networks. First, we can delay initiation of the standard solutions until the network topology is verified. Second, we can use a slower synchronization procedure which runs in parallel with a procedure to verify the network structure.

Square $m \times m$ Arrays with General at Upper Left

Beyer [3] and Shinahr [12] provide a minimal time solution for square arrays. The idea is that the node in the diagonal position (i, i) acts as General for two identical linear firing squads, one consisting of all nodes in the same row to the right of the General, the other consisting of all nodes in the same column below the General. The Generals at the diagonal nodes delay initiation of the firing squad activity so that all firing squads fire at time $2m-2$. For a square array $r+s=4m-4$, so the solution is not embeddable. It can be made embeddable by delaying any firing squad activity until the topology of the network is verified. This can be done as follows: At time $t=0$ the General at the upper left corner sends out checking signals which verify that the network is a square array. There are two components to the checking signals. One component travels to every node, verifying that the network is a rectangular array. The second component travels down the diagonal, one diagonal element every two steps. The network is a square array iff both components reach the bottom right corner simultaneously at time $2m-2$. If the network is a square array, the bottom right corner acts like the General in the Beyer solution for the square array. Once initiated, the square firing squad fires in $2m-2$ steps. Therefore, total firing time is

$r+s=4m-4$. If the network is not a square array, the bottom right corner never initiates firing squad activity and no node ever fires. Hence, this is an optimal embeddable solution for square arrays.

Rectangular $m \times n$ Arrays with the General at Upper Left

Beyer [3] proves that the minimal time needed to synchronize an $m \times n$ rectangular array which is not square is $m+n+\max\{m,n\}-3$ steps, and also provides a minimal time solution. We can make this solution embeddable as follows: The General propagates checking signals which verify the rectangular structure of the network, as above. If the network is a rectangular array, the bottom right corner is so notified at time $r=m+n-2$. That node then acts as General for the rectangular firing squad, using Beyer's solution. This gives us an embeddable solution with firing time $2m+2n+\max\{m,n\}-5$. But $s=r=m+n-2$, so an optimal embeddable solution would fire in time $r+s=2m+2n-4$. Thus the modified solution is not optimal unless $m=n=1$.

In order to obtain an optimal embeddable solution, we use a slower synchronization procedure, also proposed by Beyer, which fires in time $r+s=2m+2n-4$. When used concurrently with a checking phase, this provides an optimal time embeddable solution for rectangular arrays.

Run the following phases in parallel:

Checking phase. At time $t=0$, the General in the upper left corner sends out checking signals to its two neighbors. (If the General does not have exactly two neighbors, it does nothing.) These signals propagate through the network, verifying that the network is a rectangular array. If it happens to be a rectangular array, the bottom right corner receives an OK signal from each of its two neighbors at time $r=m+n-2$. This node then generates a GO signal, which reaches every node in the network by time $r+s=2m+2n-4$. If the network is not a rectangular array, the bottom right corner does nothing.

Synchronization phase. A linear firing squad is activated in the top row by the General in the upper left corner. At time $2n-2$, each soldier in the top row, instead of firing, becomes a General. These new Generals, along with the old General, now activate linear firing squads in their respective columns. The soldiers in these firing squads fire only in the presence of GO signals from the checking phase.

Suppose that the network is in fact a rectangular array. Since the column firing squads all begin at time $2n-2$ and all columns have length m, the entire array will fire at time $2n-2+2m-2=2n+2m-4=r+s$. Notice that in this case, all nodes will receive GO signals by time $r+s$. If the network is not a rectangular array, then no GO signals are generated and no node will ever fire. Hence, this solution is an optimal embeddable solution.

In addition, both the checking and synchronization phases described are easily adapted to higher dimensional networks. For example, for 3-dimensional $m \times n \times p$ arrays, the synchronization phase is: At time $2n-2$ all nodes in the top front row become Generals to synchronize the columns; $2m-2$ steps later, all nodes in the front $m \times n$ array become Generals to synchronize the front to back rows; $2p-2$ steps later, all nodes fire. Total firing time is $2n+2m+2p-6=r+s$. Since the checking phase also runs in time $r+s$, the modified solution is an optimal embeddable solution for 3-dimensional rectangular arrays. Solutions for higher dimensions are similar.

Since the solution is an optimal embeddable solution for all rectangular arrays, it is also an optimal embeddable solution for the special case of square arrays. Recall that the optimal nonembeddable firing time obtained for $m \times m$ square arrays is $2m-2$ [12]. However, if the Beyer solution for rectangular arrays, which is optimal for nonsquare arrays, is applied to a square array, the firing time is $3m-3$. Thus, it is interesting to note that the embeddable solution provides the optimal embeddable synchronization time for square arrays as well.

In this section, we have described optimal embeddable solutions to the fssp for a variety of classes of networks, including linear arrays, rings, star graphs, flower graphs, and n-dimensional rectangular arrays. By Theorem 1, we can incorporate all of these embeddable solutions into one embeddable solution for the fssp. Finally, we can embed this solution in one of the known d-universal solutions, e.g., Rosenstiehl's, to obtain a faster d-universal solution, call it S_U. Rosenstiehl's solution fires in $2n$ steps for every network of n nodes. The solution S_U takes no more time to fire than Rosenstiehl's solution. If, however, S_U is applied to, say, a 3-dimensional $m \times m \times m$ cubic array, then S_U will fire in time $r+s=6m-6$, whereas Rosenstiehl's solution takes $2n=2m^3$ steps.

6. Component-Based Solutions

A shortcoming of many of the known solutions to the fssp is that small differences in network structure may result in large differences in synchronization time. Consider an $m \times m \times m$ cubic array. The solution S_U can synchronize this network in $r+s=6m-6$ steps. Now, suppose we add a single extra node to the cubic network. In this case, the solution S_U must fall back to the general case solution, which takes at least $2m^3$ steps to synchronize the modified network. Such a huge increase in synchronization time is intuitively undesirable. In this section, we show how an efficient solution for a subnetwork can contribute to the efficiency of the overall solution. Our first result shows that, for a network consisting of a collection of components connected at a single node, the sum of the times required to synchronize the components forms an upper bound on the time required to synchronize the entire network. The construction preserves the embeddability of the component solutions.

THEOREM 3. *Let a network N consist of d components $N_1, N_2, \ldots, N_d$, joined at a single node. The common node acts as General. If FS_j is a solution to the* fssp *which synchronizes N_j in T_j steps for $1 \leqslant j \leqslant d$, then there exists a solution S_1 which synchronizes N in $\sum_{j=1}^{d} T_j$ steps. Moreover, if FS_j is a embeddable solution, for $1 \leqslant j \leqslant d$, then S_1 is also embeddable.*

Proof. Each node in the network has d tracks, each track capable of supporting a separate firing squad. We will refer to the firing squad FS_j on track i of component N_j as $fs_{i,j}$. The General keeps a $d \times d$ bit table, the *timing table*, which records timing information about the various firing squads. (Note that this construction depends upon the bounded degree of interconnections.) Initially, the diagonal elements (j,j) of the timing table are set to 1; the other elements are 0. The entries are set one at a time, according to rule (2). Entry (i,j) is set to mark the passage of T_j time steps. The General performs according to the following rules:

(1) At time $t = 0$, initiate $fs_{1,2}$ and $fs_{i,1}$ for $i = 2, 3, \ldots, d$.

(2) When $fs_{i,j}$ fires, set element (i,j) in the timing table to 1. If there is a 0 element in row i, say (i, k), initiate $fs_{i,k}$. Otherwise, initiate $fs_{i,i}$.

For component N_i, $fs_{i,i}$ is the *true* firing squad. When $fs_{i,i}$ fires, all nodes in N_i enter the *fire* state.

For $i = 1, 2, \ldots, d, fs_{i,i}$ is initiated by the General at time $\sum_{j=1, j \neq i}^{d} T_j$. Hence, for all $i = 1, 2, \ldots, d, fs_{i,i}$ fires at time

$$T = \sum_{j=1}^{d} T_j.$$

Suppose strategy FS_j fails to fire in component N_j. Then S_1 never initiates true firing squad activity in any component N_i, for $i \neq j$. Furthermore, if S_1 does initiate true firing squad activity in component N_j (because all other simulated firing squads have fired), component N_j will never fire. It follows that if FS_j is embeddable, for $j = 1, 2, \ldots, d$, then S_1 is also embeddable. ▮

As an application of this theorem, consider a network which consists of two components N_1 and N_2 joined at a single node. Let N_1 be a linear array of n nodes. Let N_2 be an $m \times m \times m$ cubic array. The common node acts as General. Using known solutions for linear arrays and cubic arrays [12, 13], component N_1 can be synchronized in $T_1 = 2n - 2$ steps, and N_2 can be synchronized in $T_2 = 6m - 6$ steps. Suppose $T_1 < T_2$. The solution S_1 of Theorem 3 will cause the following sequence of events in N:

$t = 0$:	General start $fs_{1,2}$ and $fs_{2,1}$,
$t = T_1$:	$fs_{2,1}$ fires, General starts $fs_{2,2}$,
$t = T_2$:	$fs_{1,2}$ fires, General starts $fs_{1,1}$,
$t = T_1 + T_2$:	both $fs_{1,1}$ and $fs_{2,2}$ fire simultaneously.

The overall firing time is $T = 6m - 6 + 2n - 2$. Comparing this firing time with the time required to synchronize a cubic array ($T_2 = 6m - 6$), we see that a small deviation in network structure does indeed yield a small increase in firing time.

Our next result provides another upper bound for this class of networks, again in terms of the time required to synchronize the components. This result requires that the fssp solution use the network itself as "memory" to store timing information, using a technique developed by Rosenstiehl [11]. In [11], it is shown that there exists an automaton M such that, for any network N with a distinguished node and degree bounded by d, if a copy of M is placed at each node in N, then M induces a depth-first traversal in N, with each node storing the local directions for the path. If N has n nodes, then the path is constructed by M in $2n$ steps ($2n$ is also the length of the path).

THEOREM 4. *Let a network N consist of d components $N_1, N_2, \ldots, N_d$, joined at a single node. The common node acts as General. If FS_i is a solution to the* fssp *which synchronizes N_i in T_i steps, for $i = 1, 2, \ldots, d$, then there exists a solution S_2 which synchronizes N in $2 \times \max_{1 \leqslant i \leqslant d}\{T_i\}$ steps. Moreover, if FS_i is a embeddable solution, for $i = 1, 2, \ldots, d$, then S_2 is also embeddable.*

Proof. We assume without loss of generality that $T_d \geqslant T_i$, for $i = 1, 2, \ldots, d-1$. Each node in the network has $d+1$ tracks. One track is capable of supporting firing squad activity. One track supports the automaton M which induces a depth-first traversal in N_i. The remaining $d-1$ tracks are used to pass timing signals, as explained here.

The General performs according to the following rules:

(1) At time $t = 0$, the General initiates *simulated* firing squads fs_i, for $i = 1, 2, \ldots, d$. Also, the General acts as the distinguished node in initiating the depth-first traversal in N_i.

(2) At time $t = T_i$, for $i = 1, 2, \ldots, d-1$, the simulated firing squad fs_i fires in component N_i. The General responds by sending out A_i signals (speed $= \frac{1}{3}$) into components N_j, for $j \neq i$. A_i travels along the traversal path in N_j. If A_i ever completes the path, or if a true firing squad fires in N_j, then A_i is destroyed.

(3) At time $t = T_d$, the last simulated firing squad fs_d fires. The General sends out B_i signals (speed $= 1$) into N_d, for $i = 1, 2, \ldots, d-1$. Also, the General initiates a true firing squad FS_d in N_d. Then B_i travels along the traversal path in N_d. At time $T_d + (T_d - T_i)/2$, B_i overtakes A_i. At this time, A_i and B_i are destroyed. A signal C_i (speed $= 1$) is sent back along the path toward the General.

Note that signals A_i and B_i never overtake the signals that set up the traversal path. Also note that, in order for B_i to overtake A_i before A_i completes the path through N_d, we must assume that $(T_d - T_i)/2 < 2 \times |N_d|$, where $|N_d|$ is the number of nodes in component N_d. This assumption involves no loss of generality, however, since we may always assume that $T_d \leqslant 2 \times |N_d|$.

(4) At time $t = T_d + (T_d - T_i)$, the signal C_i reaches the General. The General responds by initiating the true firing squad FS_i in component N_i.

(5) At time $t = 2 \times T_d$, all true firing squads FS_i fire simultaneously, for $i = 1, 2, \ldots, d$.

Note that, if any of the strategies FS_i fails to fire, then S_2 never initiates any true firing squad activity. It follows that, if FS_i is embeddable, for $i = 1, 2, \ldots, d$, then S_2 is also embeddable. ∎

The two solutions S_1 and S_2 are complementary in the sense that neither one is uniformly superior to the other. For example, if N consists of two components such that one takes very much longer to synchronize than the other, solution S_1 would synchronize the entire network faster than S_2 would. On the other hand, if the network N consists of many evenly balanced components, then S_2 would be more efficient than S_1. Since both S_1 and S_2 preserve the embeddability of the component solutions, they may, of course, be applied in parallel, and the overall firing time will be the lesser of the two.

Solution S_2 can be modified so that later synchronizations can take advantage of the timing information produced by the first synchronization. The modified solution S_3 is as follows: A_i, B_i, and C_i are signals as in S_2. When the B_i signal overtakes the A_i signal, a wall W_i is formed. In future synchronizations, the General acts as follows:

(1) At time $t = 0$, the General initiates a true firing squad FS_d in component N_d. The General also sends B_i signals into N_d, for $i = 1, 2, \ldots, d-1$. When the B_i signal reaches the wall W_i, the B_i is destroyed, and a C_i signal is reflected to the General.

(2) At time $t = T_d - T_i$, the General receives the C_i signal and initiates the true firing squad FS_i in component N_i.

(3) At time $t = T_d$, all true firing squads FS_i fire, for $i = 1, 2, \ldots, d$.

In this modification, S_3 stores the timing information in the network itself. If repeated synchronizations are required (as is typically the case in real applications), S_3 seems to offer the optimal synchronization time using the given component solutions, since it synchronizes the entire network in the maximum number of steps it takes to synchronize each of its components.

7. Concluding Remarks

We might summarize this paper as an attempt to explicate the difference between solutions to the fssp which *exploit* the structure of the underlying communications network, and solutions which *rely* on a priori assumptions about the structure of the network. It is usually desirable to exploit the structure of the network whenever possible. However, as networks become more complex, with nodes failing or coming on-line dynamically, it may be undesirable to inform all nodes of the current topology. In Theorem 2, we provide a uniform lower bound on how fast any network can be synchronized by local automata which do not initially know the network topology.

Kung [7] has shown that many VLSI networks can be classified according to a rather small number of simple communication geometries. We anticipate that such networks will often be linked together to form larger networks. This suggests the second line of questions addressed by this paper: To what extent can solutions for the subnetworks contribute to tasks which involve the global network? We have shown that networks which consist of subnetworks joined at a single node can be synchronized in a way in which efficient solutions for the subnetworks can lead to efficient solutions for the overall network. The proof of Theorem 4 involves an application of the idea that a network of finite state automata can use the underlying network as auxiliary storage for large numbers—in this case, information about the maximum time required to synchronize the subnetworks.

Future research might address the problem of synchronizing arbitrary trees, since an arbitrary network may be synchronized by synchronizing a spanning tree (see [11]). The techniques in Section 6 can be generalized to arbitrary tree-connected networks, but appear to require a number of states in each node which increases linearly with the height of the tree. Therefore, these schemes do not appear to provide an optimal solution to the general tree synchronization problem.

ACKNOWLEDGMENTS

The author thanks Robert P. Daley for his help and encouragement, and Patrick C. Fischer for his comments on an early version of this paper.

REFERENCES

1. D. ANGLUIN, Local and global properties in networks of processors, *in* "Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing, 1980," pp. 82–93.
2. R. M. BALZAR, An 8-state minimal time solution to the firing squad synchronization problem, *Inform. Control* **10** (1967), 22–42.
3. W. T. BEYER, "Recognition of Topological Invariants by Iterative Arrays," Ph.D. Dissertation, MIT, Cambridge, Mass., 1969.
4. J. J. GREFENSTETTE, "Automaton Networks and Parallel Rewriting Systems," Ph.D. Dissertation, University of Pittsburgh, Pittsburgh, Pa., 1980.
5. F. C. HENNIE, "Iterative Arrays of Logical Circuits," MIT Press, Cambridge, Mass., 1961.
6. G. T. HERMAN, W. H. LIU, S. ROWLAND, AND A. WALKER, Synchronization of growing cellular systems, *Inform. Control* **25** (1974), 103–122.
7. H. T. KUNG, "Let's Design Algorithms for VLSI Systems," Technical Report CMU-CS-79-151, Computer Science Department, Carnegie-Mellon University, Pitssburgh, Pa., 1979.
8. C. MEAD AND L. CONWAY, "Introduction to VLSI Systems," Addison-Wesley, Reading, Mass., 1980.
9. F. R. MOORE AND G. G. LANGDON, A generalized firing squad problem, *Inform. Control* **12** (1968), 212–220.
10. F. ROMANI, Cellular automata synchronization, *Inform. Sci.* **10** (1976), 299–318.
11. P. ROSENSTIEHL, J. R. FIKSEL, AND A. HOLLIGER, Intelligent graphs: networks of finite automata capable of solving graph problems, *in* "Graph Theory and Computing" (R. C. Read, Ed.), pp. 219–265, Academic Press, New York, 1972.

12. I. Shinahr, Two- and three-dimensional firing squad synchronization problems, *Inform. Control* **24** (1974), 163–180.
13. A. Waksman, An optimal solution to the firing squad synchronization problem, *Inform. Control* **9** (1966), 66–78.
14. A. Wu and A. Rosenfeld, Cellular graph automata, I and II, *Inform. Control* **42** (1979), 305–353.

Printed in Belgium