INFORMATION AND CONTROL 42, 330–353 (1979)

Cellular Graph Automata. II. Graph and Subgraph Isomorphism, Graph Structure Recognition*

ANGELA WU AND AZRIEL ROSENFELD

Computer Science Center, University of Maryland, College Park, Maryland 20742

This paper deals with cellular automata in which the intercell connections define a graph of bounded degree. It discusses acceptance tasks that involve the detection of graph or subgraph isomorphism in time proportional to the diameter of the given graph. In some of the algorithms, special assumptions are made about the "homogeneity" of the graph; these assumptions hold for many important classes of graphs, including trees and arrays. The paper also examines types of graph structures that can be recognized by these automata. Diameter-time algorithms are presented for the recognition of cycles, strings, trees, cliques, rectangular and square arrays, Eulerian graphs, bipartite and complete bipartite graphs, stars, and wheels. The recognition of planar graphs is also discussed.

1. INTRODUCTION

A class of generalized cellular automata in which intracell connections define a graph of bounded degree (a "d-graph") is introduced in (Wu and Rosenfeld, 1979), where the terminology and notation used in the present paper are defined. Sections 2 and 3 of this paper discuss graph acceptance tasks for such automata that depend on the concept of d-graph isomorphism—in particular, the task of deciding whether a d-graph has a d-subgraph isomorphic to a given d-graph. Section 4 discusses types of graph structures that can be recognized by such automata.

Given two node-labeled graphs $\gamma_1 = (N_1, A_2, f_1)$ and $\gamma_2 = (N_2, A_2, f_2)$ where f_1, f_2 are the node-labeling functions, γ_1 is *isomorphic* to γ_2 if there exists a bijection b from N_1 to N_2 such that $f_1(n) = f_2(b(n))\ \forall n \in N_1$ and $(m, n) \in A_1$ iff $(b(m), b(n)) \in A_2$. A d_1-graph $\Gamma_1 = (N_1, A_1, f_1, g_1)$ and a d_2-graph $\Gamma_2 = (N_2, A_2, f_2, g_2)$ are *isomorphic* (denoted by $\Gamma_1 \simeq \Gamma_2$) iff their underlying graphs $U(\Gamma_1)$ and $U(\Gamma_2)$ are isomorphic. Here we allow $d_1 \neq d_2$. A *subgraph* of a

* The support of the U.S. Air Force Office of Scientific Research under Grant AFOSR-77-3271 is gratefully acknowledged, as is the help of Ms. Kathryn Riley in preparing this paper.

0019-9958/79/090330-24$02.00/0

d-graph $\Gamma = (N, A, f, g)$ is denoted by $(N', A', f \mid N', g \mid A')$, where $N' \subseteq N$ and $A' \subseteq A$ and if $(m, n) \in A'$ then $m \in N'$ and $n \in N'$. Note that $(N', A', f \mid N', g \mid A')$ is not necessarily a d-graph, since some of the nodes may not have exactly d neighbors. However, we can always attach # nodes so as to make it into a d-graph. A labeled graph α is *isomorphic* to Γ if $\alpha \simeq U(\Gamma)$, and α is isomorphic to a subgraph of Γ if $\alpha \simeq U(\Gamma')$ for some subgraph Γ' of Γ. In the following, we will consider only connected d-graphs, and deterministic cellular d-graph acceptors.

2. Graph Isomorphism

In this section, we will consider acceptance tasks that depend on graph isomorphism. Specifically, given a labeled graph α of degree $\leqslant d$, we will find a finite-state acceptor M_α such that (Γ, M_α, H) accepts Γ if α is isomorphic to Γ and rejects Γ otherwise.

We first need

Proposition 1. *For every integer $r > 0$, there exists a cellular d-graph acceptor $\mathcal{M}_r = (\Gamma, M_r, H)$ with distinguished node D that accepts all d-graphs Γ whose nodes are all within distance r from D in $2r + 1$ steps, and when it accepts, every node is in a different state.*

Proof. Given any d-graph Γ, the cellular d-graph acceptor (Γ, M_r, H) operates as follows: The distinguished node D sends out a message S which propagates to the nodes at distance r from D. The paths traveled by S define a spanning tree of Γ, and each node is identified uniquely by marking each node's state with a sequence of arc end numbers which define a shortest path from D to the node. Specifically, when a neighbor of D receives S, its state is marked with the number i if it is the ith neighbor of D. It then sends the message (S, i) to its neighbors. When an unmarked node m receives the message $(S, i_1, \ldots, i_k)$, $k \geqslant 1$, from node n, and m is the jth neighbor of n, then m marks its state with $(i_1, \ldots, i_k, j)$ and sends $(S, i_1, \ldots, i_k, j)$ to its neighbors. If a node receives a message from more than one neighbor simultaneously, it can choose to accept one of them, say the one sent by the lowest-numbered neighbor.

If a node m_1 is marked with $(i_1, i_2, \ldots, i_r)$ and one of its neighbors, say m_2, is still unmarked, then m_2 is at distance $r + 1$ away from D. A rejection signal is thus sent to D because the graph contains nodes more than distance r away. If no rejection signal is received after $2r + 1$ steps, Γ is accepted. ∎

Given a node-labeled graph α of degree $\leqslant d$, we can find its diameter r and construct its spanning tree T_α. The height of the spanning tree is $\leqslant r$ and associated with each node is a level number. The level numbers of a node and

its neighbors differ by at most 1; this follows from the way the tree is constructed. Now we can prove

PROPOSITION 2. *For any labeled graph α of degree $\leqslant d$, there exists a cellular d-graph acceptor $\mathcal{M}_\alpha = (\Gamma, M_\alpha, H)$ with distinguished node D that accepts Γ if $\alpha \simeq \Gamma$ and rejects Γ otherwise.*

Proof. (1) $\mathcal{M}_\alpha$ makes sure that all the nodes of Γ are within distance r from node D and assigns a unique "color" to each node (recorded in its state) by simulating $\mathcal{M}_r$ of Proposition 1 in the first r steps.

(2) $\mathcal{M}_\alpha$ finds all the subgraphs of Γ isomorphic to T_α in the next $h + 1$ steps, where h is the height of T_α. At step $r + 1 + i$ $(0 \leqslant i \leqslant h)$, the nodes having the correct neighbors to serve as the sons of a level $h - i$ node of T_α are identified. Recorded in the state of the node are (a) the level $h - i$ node, call it m, of T_α that it qualified to be, and (b) the assignment of the nodes of Γ (represented by their colors found in (1)) to the subtree of T_α at m. Note that a node n of Γ may qualify to be more than one level $h - i$ node to T_α, and that different sets of neighbors of n may qualify n to be the same node of T_α. Therefore the state of a node may contain many assignments. However, the numbers of level $h - i$ nodes of T_α and of possible assignments are bounded. Therefore the size of the states and the number of states depend only on T_α, and may be large but are bounded. At the end of step $r + 1 + h$, the nodes of Γ corresponding to the root of T_α and the assignments of the nodes of T_α are known. Each of the assignments gives a subgraph of Γ isomorphic to T_α. For a more detailed description see (Wu, 1978).

(3) Starting at step $r + h + 2$, each qualified root node initiates signals to check each assignment recorded in its state to make sure that all the ards in α exist and no other arcs are present. This is done by transmitting the assignment to each node. If a node not in the assignment receives the assignment signal, this means that Γ has more nodes than α and it cannot be isomorphic to α; thus a rejection signal is sent to the distinguished node to reject Γ. If a node is in the assignment, when it receives the signal, it makes sure that all the arcs incident upon it are connected to the nodes with the correct identities as in α. Any time a node finds an arc out of order, it sends a cancellation signal to report to the root node of this assignment to delete the assignment. If after $2h$ steps, the qualified root node finds that it still has uncanceled assignments, then it can send a success signal to the distinguished node D. When D gets a success signal, it accepts. However, if at the end of step $r + 1 + h + 2h + r$ no success signal is received by D, this means there is no successful assignment. This is because either the assignments made at step $r + 1 + h$ are all canceled or there is no assignment at all at step $r + 1 + h$, since there may be too few nodes in Γ, or it is not possible even to find a subgraph of Γ isomorphic to T_α. In any case, Γ is not isomorphic to α and Γ is rejected. ∎

3. Subgraph Isomorphism

In this section we will consider acceptance tasks that depend on subgraph isomorphism.

Proposition 3. *Let α be a labeled graph and let T_α be a spanning tree of α with root node R_α. Then there exists a cellular d-graph acceptor $\mathcal{M}$ with a distinguished node D such that $\mathcal{M}$ accepts a d-graph Γ if Γ has a subgraph isomorphic to α, where D corresponds to R_α, and rejects Γ otherwise, in time proportional to k, the height of T_α.*

Proof. First $\mathcal{M}$ assigns a unique color to each node within distance k from D. The color of a node n can be represented by the sequence of arc end numbers which constitute a path from D to n; thus uniqueness is guaranteed. Each node also has a max-distance number which is initially zero for each node and the max-distance number of D is always 0. A non-D node n changes its max-distance number to $i + 1$ if i is the maximum of its neighbors' max-distance numbers and if $i < k$. After k steps, all the possible level k nodes of T_α have max-distance number k. A node with max-distance k records the level k node of T_α it can be, if any, in its state as in Section 2. In general, a colored node having the correct level i neighbors of T_α records in its state the level $i - 1$ node of T_α it qualifies to be and all the nodes in its subtree, After $2k$ steps, if there is a subtree at D isomorphic to T_α, D will have the subtree recorded. Then $\mathcal{M}$ can proceed to check if the arcs in $\alpha - T_\alpha$ exist. $2k$ more steps later, D knows if there is a subgraph of Γ isomorphic to α with D corresponding to the root of T_α. ∎

Wu and Rosenfeld (1979) show that a depth-first spanning tree (DFST) of a d-graph can be constructed in area time (i.e., time proportional to the number of nodes). Combining the DFST construction and Proposition 3 we have:

Proposition 4. *For any labeled graph α, there exists a cellular d-graph acceptor $\mathcal{M}$ such that $\mathcal{M}$ accepts a d-graph Γ if Γ has a subgraph isomorphic to α and rejects Γ otherwise in time proportional to the number of nodes in Γ.*

Proof. During the DFST construction, whenever a new node n is reached, the following is done before signal P is passed to another node: If n does not have the same label as the root R_α of T_α, a spanning tree of α, or n has fewer non-# neighbors than R_α, then P goes to the next node. Otherwise, we test whether there is a subgraph of Γ isomorphic to α with n corresponding to R_α, treating n as D in Proposition 3; if such a subgraph is found a success signal is sent to the distinguished node to accept Γ; otherwise all the colors of the nodes within distance k from n are erased. When all the nodes have been tested and no success signal has been received, then Γ does not have a subgraph isomorphic to α and Γ is rejected.

The above subgraph matching process takes time proportional to $k \cdot \text{area}(\Gamma)$, where k is the height of T_α. ∎

It should be pointed out here that each uncanceled assignment represents a subgraph isomorphic to α. Moreover there are redundancies because the same subgraph may be specified by different assignments which correspond to different automorphic images of the subgraph.

Since the time complexity of the subgraph isomorphism algorithm depends on the height of the spanning tree T_α of α that is used, it is desirable to find a T_α with minimal height. It is evident that if a central point of α is used as the root to build a breadth-first spanning tree then the height of T_α will be minimal. The number of states of the cellular d-graph acceptor is a function of α and d and is in general large. Proposition 4 does not contradict the fact that subgraph isomorphism in an *NP*-complete problem because our d-graphs have bounded degree d. This makes the nondeterminism at each step bounded by a function of d and α, and therefore they can be represented by a bounded, though large, number of states.

The cellular d-graph acceptor of Proposition 4 takes area time to decide subgraph isomorphism. We have been unable to find a diameter time cellular d-graph acceptor that can decide subgraph isomorphism for general d-graphs. Unlike the graph isomorphism case, the number of nodes of the d-graph Γ we are interested in may be arbitrarily large. However, the definition of M_α in the cellular d-graph acceptor (Γ, M_α, H) of Proposition 2 depends only on the labeled graph α and not on Γ; therefore it is not possible to give each node of Γ a unique identification as part of its state. Attempts to simulate the action of $\mathcal{M}$ of Proposition 3 from many nodes simultaneously create confusion among the signals since there are many signals and some of them are not distinguishable from each other. These problems seem to be due to the existence of cycles in a general d-graph (for examples, see (Wu, 1978)). In the following subsections we will show that for certain general classes of d-graphs, subgraph matching can be done in diameter time.

3.1. *Trees*

Trees have the special properties that there are no nontrivial cycles and there is exactly one simple path from one node to another. Any labeled graph isomorphic to a subgraph of a d-graph which is a tree must also be a tree. This makes the subgraph matching tasks for trees easier.

PROPOSITION 5. *For any labeled tree Z, there is a cellular d-graphs acceptor $\mathcal{M}_Z$ such that for any d-graph Γ which is a tree, $\mathcal{M}_Z$ accepts Γ iff $Z \simeq$ a subgraph of Γ and otherwise it rejects Γ, in time proportional to the diameter of Γ.*

Proof. Let h be the height of Z. At each step i $(1 \leqslant i \leqslant h + 1)$ each node n of Γ looks at its neighbors to decide if they can correspond to the sons of a level

$h + 1 - i$ node m in the tree Z. If they can, n declares itself to qualify as the node m of Z. Instead of recording the assignments of nodes as $\mathcal{M}_\alpha$ of Proposition 2, each node n's state simply indicates the level $h - i + 1$ node it can be and the arc end numbers leading to its sons. Note that a node can correspond to several sets of sons. In the next step, the knowledge of the sons prevents those nodes from serving both as n's father and son when n corresponds to a particular node m_0 of Z since n's neighbor can see from n's state whether it was used as n's son to qualify n as m_0 of Z. It is easy to see that at the end of step i ($1 \leqslant i \leqslant h + 1$), if a node n's state indicates that it qualifies as a level $h - i + 1$ node m of Z and its $i_1, i_2, \ldots, i_j$th neighbors $n_1, n_2, \ldots, n_j$ correspond to the sons $m_1, m_2, \ldots, m_j$ of m in Z, then n is the root of a tree isomorphic to the subtree of Z at m. (This tree is an acyclic subgraph of Γ.)

At the end of step $h + 1$, all the nodes that correspond to the root node of Z are identified. These nodes just send a success message to the distinguished node D for acceptance. If after $h + 1 +$ height(Γ) steps, no success meassage is received by D, it rejects. Let $\mathcal{M}_Z$ send out a special signal from D as in the breadth-first spanning tree construction during the first step. When the return signal reaches D, $2 \cdot$ height(Γ) steps have passed and therefore $\mathcal{M}_Z$ knows that it can reject Γ if no success signal has reached D. ∎

In Section 4.3 we show how a cellular d-graph automaton can recognize trees in diameter time. Combining this with the above proposition, we have the result that for any tree Z, there is a cellular d-graph automaton that recognizes all the d-graphs which are trees and have subgraphs isomorphic to Z.

3.2. *k-Level-Colored d-Graphs*

Suppose a labeled graph α has diameter r; if a node n is part of a subgraph S isomorphic to α, then all the other nodes of S must be within distance r from n. We will show that if every node of Γ has a different state from any node within distance r from it, then we can discover whether α is isomorphic to a subgraph of Γ in time proportional to the diameter of Γ.

A *d-graph* Γ will be called *k-level-colored* if the nodes of Γ are colored and any two nodes within distance k from each other have different colors. For any node n in a d-graph Γ, the number of nodes within distance k from it is at most $c(k) = d + d(d-1) + d(d-1)^2 + \cdots + d(d-1)^{k-1}$. The number of colors needed for Γ to be k-level-colored is not more than $1 + c(k)$. We can assume that the color at each node is part of the label and thus becomes part of the initial state of the automaton at the node.

PROPOSITION 6. *Given a labeled graph β with diameter r, there is a cellular d-graph acceptor $\mathcal{M}_\beta = (\Gamma, M_\beta, H)$ with a distinguished node D such that for any k-level-colored ($k \geqslant r$) d-graph Γ, $\mathcal{M}_\beta$ accepts Γ if $\beta \simeq$ a subgraph of Γ and rejects Γ otherwise, in time proportional to the diameter of Γ.*

Proof. Given β we can find its spanning tree T_β. $\mathcal{M}_\beta$ works in almost the same way as $\mathcal{M}_\alpha$ in Proposition 2 with some obvious modifications. It first finds subgraphs isomorphic to T_β, then checks the existence of the arcs in $\beta - T_\beta$. For a more detailed proof, see (Wu, 1978). Note that this can be done in diameter time since the signal from a node never needs to travel more than distance r away and Γ is k-level-colored with $k \geqslant r$. ∎

In general, it is not an easy task to k-level-color a d-graph. However, once a d-graph is colored, the coloring can be used for solving many problems, so that the time needed for the coloring process may be well spent. The following propositions shows that a tree can be k-level-colored efficiently.

PROPOSITION 7. *For any $k > 0$, there is a cellular d-graph acceptor $\mathcal{A}_k$ with a distinguished node D such that for any d-graph Γ which is also a tree, $\mathcal{A}_k$ k-level-colors the nodes of Γ in time proportional to dimaeter(Γ).*

Proof. Let $l = \lfloor k/2 \rfloor$ = (the largest integer $\leqslant k/2$). Each node n's state will have a component of the form $(j, j_1, \ldots, j_l)$, where j is the distance between D and n modulo $k + 1$, and $j_1, \ldots, j_l$ are the last l elements of the path from D to n specified by arc end numbers such that if the distance between D and n is $p < l$ then $j_1, \ldots, j_{l-p}$ are all zeros. At the first step, D writes $(0, \ldots, 0)$ in its state. When an uncolored node receives a message $(i, i_1, \ldots, i_l)$ from its neighbor n, m writes in its state $(i + 1 \pmod{k+1}, i_2, \ldots, i_l, j)$ if m is the jth neighbor of n.

Suppose nodes m, n having colors $(i, i_1, \ldots, i_l)$ and $(j, j_1, \ldots, j_l)$ are within distance k from each other. If $i = j$ then their distance from D must be the same and their closest common ancestor (if D is considered to be the root) must be within distance l from both m and n, hence $(i_1, \ldots, i_l) \neq (j_1, \ldots, j_l)$.

The time it takes for the coloring signal from D to reach a node n equals the distance between D and n. Therefore the coloring process takes diameter(Γ) time. ∎

3.3. *k-Locally-Homogeneous d-Graphs*

As pointed out in Section 3, the difficulty in obtaining diameter-time algorithms for subgraph isomorphism seems to be due to the existence of the cycles. The k-locally homogeneous d-graphs defined in this section are a class of d-graphs in which knowledge about the cycles is available.

From the definition of a cellular d-graph automaton, at each node n, $H(n) = (t_1, \ldots, t_d)$ tells n that it is the t_ith neighbor of its ith neighbor. If we consider a sequence of numbers $a_1, \ldots, a_j$ $(1 \leqslant a_i \leqslant d$ for $1 \leqslant i \leqslant j)$ at n as a path $n = n_0, n_1, \ldots, n_j$ such that n_i is is the a_ith neighbor of n_{i-1} $(1 \leqslant i \leqslant j)$, then $H(n)$ tells node n the inverse of any path of length 1. Define $H^j(n)$: $D^j \to D^j$ such that if the image of $(a_1, \ldots, a_j)$ is $(b_1, \ldots, b_j)$ then the inverse of the path $a_1, \ldots, a_j$ is $b_j, b_{j-1}, \ldots, b_1$. It is obvious that knowing H^k at a node n implies knowing H^j at n for any $1 \leqslant j \leqslant k$.

It is straightforward to show that for any k, there is a cellular d-graph automaton that finds H^k and records $H^k(n)$ in node n's state in time proportional to k.

In a d-graph Γ, a *node n knows all cycles of length up to k* if given a sequence of arc end numbers $a_1, \ldots, a_j$ $(1 \leqslant a_j \leqslant d,\ 1 \leqslant j \leqslant k)$, hence a path of length $j \leqslant k$ starting from n, node n knows whether or not this path is a cycle. A *node n knows all equivalent paths of total length up to k* if given any two sequences of arc end numbers, hence two paths from n, with sum of their lengths $\leqslant k$, node n knows if they lead to the same node. Γ is said to know all cycles of length up to k or know all equivalent paths of total length up to k iff every node in Γ knows the respective information. It is easily seen that knowing H^k does not imply knowing all cycles of length j or knowing all equivalent paths of total length j for some $j > 2$.

PROPOSITION 8. *A node n knows all cycles of length up to k iff n knows all equivalent paths of total length up to k.*

Proof. The knowledge of H^k allows one to find the inverse of a path. Given two paths from node n of total length $\leqslant k$, if one path is appended to the inverse of the other, the result is a cycle iff the two paths reach the same node. Given a sequence of arc ends of length up to k, breaking it at some point and finding the inverse of one of them gives two paths. These two paths reach the same node iff the sequence is a cycle. ∎

Having the equivalence given by Proposition 8, we can define a d-graph Γ to be *k-locally-homogeneous* if at each node n of Γ, $H(n)$ and all cycles of length up to k or all equivalent paths of total length up to k are known. Clearly, a d-graph that is a tree is k-locally-homogeneous, since there are *no* cycles, and no two distinct paths can be equivalent.

PROPOSITION 9. *For any labeled graph ω with degree $\leqslant d$ and diameter r, there exists a cellular d-graph acceptor $\mathcal{M}_\omega$ such that for any k-locally homogeneous d-graph Γ $(k > 2r)$, $\mathcal{M}_\omega$ accepts Γ if $\omega \simeq$ a subgraph of Γ, and rejects Γ otherwise, in time proportional to the diameter of Γ.*

Proof. $\mathcal{M}_\omega$ first identifies the subgraphs of Γ isomorphic to T_ω, a spanning tree of ω, in a way similar to the operation of $\mathcal{M}_\beta$ and $\mathcal{M}_Z$ defined earlier. However, the nodes are not colored distinctly and Γ is not necessarily a tree. Therefore, each node n records in its state the nodes of T_ω that it qualifies to be and the arc end numbers leading to the neighbors corresponding to the sons of n, together with the information recorded in each of those neighbors. After $h + 1$ steps, where h is the height of T_ω, all the nodes that can possibly correspond to the root of a subgraph isomorphic to T_ω have that subgraph recorded in their states. However, this subgraph is not necessarily a tree.

It is easy to see (by an induction proof) that if the subgraph S obtained at a

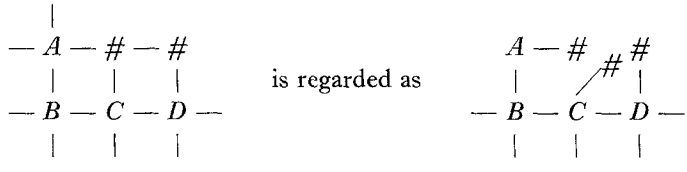
node n at step $h + 1$ is a tree, then S is isomorphic to T_ω. S is a tree iff it contains no cycles. Hence, since Γ is k-locally-homogeneous, n can tell which subgraph in its state is not a tree, and can delete that subgraph from its state at step $h + 2$. All the subgraphs that remain are trees isomorphic to T_ω. At step $h + 3$, node n checks to make sure the edges of ω not in T_ω exist, using the knowledge it possesses about k-local homogeneity. If any non-tree edge does not exist, that subgraph is deleted from the state of node n.

If there is a tree left in the state of any node, then the node sends a success meassage to the distinguished node to signal acceptance. If the distinguished node receives no success meassage after step $h + 3 +$ diameter(Γ) it rejects Γ. Again the distinguished node can tell that $h + 3 +$ diameter(Γ) steps have passed by the same method that $\mathcal{M}_Z$ used in the proof of Proposition 5. ∎

k-Local-homogeneity seems somewhat artificial; however, a special case of it, namely, homogeneity, holds for many important classes of d-graphs, as we will see in the next subsection.

3.4. *Homogeneous d-Graphs*

A two-dimensional array may be regarded as a 4-graph, provided we assume the boundary (#) nodes are distinct so that each # node has only one neighbor, i.e.,



The arc ends at each node are labeled with 1 (=N), 2 (=W), 3 (=S), 4 (=E). Each node n knows the inverse of any path starting from n, since 1 and 3, 2 and 4 are always inverses of one another. Each node also knows when a path is a cycle by checking if the number of 1's = the number of 3's and the number of 2's = the number of 4's. Therefore a two-dimensional array is a k-locally-homogeneous d-graph for any $k \geqslant 1$. Moreover, all the k-local-homogeneity conditions at each node are the same in the sense that if a path from a node exists (no # node is encountered) then the same criterion determines, for all nodes, whether or not the path is a cycle.

A d-graph will be called *k-homogeneous* if all the k-local-homogeneity conditions are the same for every node of Γ. If the d-graph is k-homogeneous for every $k \geqslant 1$, we call it simply *homogeneous*. As indicated above, the two-dimensional arrays are homogeneous 4-graphs. It is easy to see, analogously, that any n-dimensional array is a homogeneous $2n$-graph.

A natural way to specify the homogeneity conditions of a d-graph is in terms of group generators and relations. We can regard the d arc end numbers at each

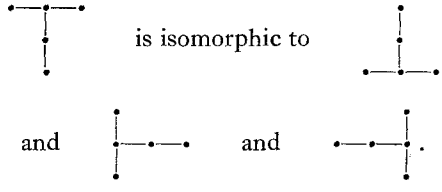
node as the generators. A relation $s_1 s_2 \cdots s_t = e$ (where e is the identity) says that at each node n, the path $s_1 s_2 \cdots s_t$ is a cycle. Thus knowing the relations implies knowing all the cycles. Moreover, the cycles of length 2 at each node are the same. If $s_1 s_2 = e$ then when one end of an arc is numbered with s_1, the other end of the same arc must be numbered with s_2; thus $s_1 = s_2^{-1}$. This shows that the d generators must form a group.

Mylopoulos and Pavlidis (1975) described a number of graphs corresponding to different finitely presented Abelian groups. Any finite subgraph of one of these graphs (with the appropriate # nodes added to make the degree exactly d at each non-# node) is a homogeneous d-graph. Examples include the three regular tessellations (square, hexagonal, and triangular) and eight semiregular (Archimedean) tessellations of the plane. Some of these are illustrated in (Wu, 1978), where t-ary trees (for any $t \geqslant 1$) and cliques are also shown to be homogeneous d-graphs.

Since every homogeneous d-graph is k-locally-homogeneous for all k, the results of Section 3.3 imply

PROPOSITION 9. *For any homogeneous d-graph, subgraph matching can be done in diameter time by a cellular d-graph automaton.*

It should be pointed out here that when we consider arrays as homogeneous d-graphs, the notion of direction in an array is not important in graph isomorphism, namely,



Homogeneous d-graphs may be considered as a natural generalization of both arrays and trees. The arc end numbering of a homogeneous d-graph is consistent. The description of the homogeneity conditions is the same at each node. In general, the description is also finite and compact (for example, using group presentation), so that it can easily be stored in the finite-state automaton at each node of the d-graph. It would be of interest to study homogeneous d-graphs further.

3.5. *Application: Clique Finding in d-Graphs*

In a clique all the nodes are neighbors of each other. Since every node of a d-graph has at most d non-# neighbors, the size of any clique in a d-graph is $\leqslant d + 1$. This makes the clique finding problem in a d-graph much easier than that in a general graph.

Let us first consider the simple problem of finding the largest clique that a given node n belongs to. At the first step, each non-# ith neighbor of n marks its state with i ($1 \leqslant i \leqslant d$). At the next step, each neighbor m of n writes in its state a list $i_1, \ldots, i_t$ if the i_1th,..., i_jth neighbors of n are also neighbors of m (the list may be empty). Thus at the third step, n can tell from its neighbors' states which neighbors form cliques with it, and the size of the cliques. It is easy for n to record in its state the size of a largest clique and the numbers of the neighbors which are nodes of the largest clique. It is also not hard to mark the largest clique or even to mark all the cliques that n belongs to, because the number of such cliques is at most

$$\binom{d}{1} + \binom{d}{2} + \cdots + \binom{d}{d} < 2^d.$$

Therefore the time required to find the cliques at a node is constant.

Now consider the problem of finding the size of a maximal clique in a d-graph Γ in diameter (of Γ) time. If we find the size of the largest clique at each node of Γ, one node at a time, and then transmit the maximum of the sizes to the distinguished node, this takes area time. But largest clique finding at many nodes simultaneously will involve difficulties, since the signals from different nodes are not distinguishable. A better approach is to try to find subgraphs of Γ isomorphic to C_{d+1}, i.e., cliques of size $d + 1$; if none exist, then we try subgraphs isomorphic to C_d, $C_{d-1}, \ldots, C_3$, C_2 in order (there are always subgraphs isomorphic to C_2 if Γ is connected and has more than one non-# node). When for some i, a subgraph isomorphic to C_i is found, i is transmitted to the distinguished node D as the size of the maximal clique in Γ. When attempting to find subgraphs isomorphic to C_i ($2 \leqslant i \leqslant d + 1$) in diameter time, the difficulties of subgraph matching in a general d-graph as discussed in Section 3 also arise. However, if Γ is homogeneous, or 3-locally-homogeneous or 1-level-colored, we can detect the existence or nonexistence of C_i in diameter (of Γ) time. Therefore the size of a maximal clique can be transmitted and recorded in the state of the distinguished node of Γ in time proportional to the diameter of Γ, since there at most d C_i's to be checked.

When a subgraph isomorphic to C_i is identified, the node n of Γ corresponding to a special node (say A, the root of a spanning tree T_{c_i}) of C_i can be identified and the subgraph isomorphic to C_i can be recorded in n's state. Therefore it is easy to mark the cliques of Γ isomorphic to C_i. Note that if the nodes of C_i have the same label, then when node n identifies itself as corresponding to node A of C_i, there are $(i - 1)!$ different correspondences of C_i to the same $i - 1$ neighbors of n in Γ, since each qualifies as any one of the $i - 1$ nodes of C_i. n can get rid of these redundant assignments by just specifying which $i - 1$ of its neighbors belong to C_i. It is also straightforward to see that each node can record in its state the size of the largest clique it belongs to and which of its neighbors form such largest cliques.

4. Acceptance of Graph Structures

In the remaining sections of this paper we will study various basic graph structures accepted by cellular d-graph acceptors. Such acceptance must take time $t \geqslant$ radius from the distinguished node, since otherwise the part of the graph $>t$ away can be arbitrary. Most of the algorithms presented below have time complexity proportional to the radius, which is on the order of the diameter of the given d-graph.

The recognizers for cycles, strings, trees, and cliques have transition functions δ that are independent of the neighbor vector of the given d-graph; thus they are weak cellular d-graph automata, in which δ may be considered to be a function from Q^{d+1} to Q.

In the following we are interested only in the structure of a graph, independent of its labels and of the way the arc ends are numbered. Without loss of generality, we will assume that all the nodes of the graph initially have label S_0 except for the distinguished node which has label D, since we can always design the recognizer so that at the first step, each $q \in Q_I$ is changed to state S_0 and the distinguished node is changed to state D.

Whenever no confusion can arise, we will simply use n or the label of n to refer to the automaton located at node n.

The following subsections deal, respectively, with the acceptance or recognition of cycles, strings, trees, cliques, rectangular and square arrays, Eulerian graphs, bipartite and complete bipartite graphs, stars and wheels, and planar graphs by cellular d-graph acceptors. In this paper we only sketch the definitions of these acceptors; the details can be found in (Wu, 1978). In addition, acceptance of some other types of graph structures, including biconnected graphs, line graphs, and graphs having property values in a given set of numbers, is briefly discussed in Section 4.10.

4.1. *Cycles*

To accept cycles, the automaton M_C at node D sends a signal to its neighbors and then the neighbors transmit the signal to the other neighbors. The first time a node receives the signal from one but not both of its neighbors, it changes to state S_1, and then transmits the signal to its other neighbor if that neighbor has not received the signal. After passing the signal to its neighbor, it changes to state S_2. If a node receives the signal from both of its neighbors (in this case, the number of nodes in the cycle is even), or if a node has the signal, so is in state S_1, but finds that its other neighbor has also just received the signal and is also in state S_1 (in this case, the number of nodes in the cycle is odd), then the node changes to a new state S_3 signifying that the signals from D have crossed, and starts to send the new signal indicating acceptance back to the distinguished node. When the distinguished node receives the new signal from its two neighbors, it goes into a final accepting state A. If a node finds that it does not have

exactly two non-# neighbors (this must happen at the first step, so that the node is either in state S_0 or D), then the node changes to state K, signifying a rejection, if it is the distinguished node; otherwise it changes to state S_4 and transmits S_4 to its neighbors. If D receives the signal S_4 from one or both neighbors, it goes into the rejecting state K.

It is easy to prove the following preliminary proposition:

PROPOSITION 10. *Every connected 2-graph is a string or a cycle.*

Let G^k be the set of labeled graphs of degree at most k. $\gamma \in \mathcal{G}^d$ is a cycle iff every node has exactly two neighbors. If γ is not a cycle, then at the first step, M_C at a node having only one or more than two non-# neighbors will be in state S_4, and thus D will eventually go into state K to reject any $\Gamma \in U^{-1}(\gamma)$. The time it takes for the distinguished node to reject is not more than the diameter, i.e., the maximum of the distances between any two nodes of the graph. If γ is a cycle, then each node immediately finds that it has two neighbors. This fact must be transmitted to D. It is straightforward to see that $\mathcal{C}(M_C)$ accepts any $\Gamma \in U^{-1}(\gamma)$, and the number of steps it takes is the number of nodes in the cycle.

To recognize a cycle of size k in this way, $\mathcal{C}(M_C)$ takes order k (= twice diameter) time, which is not any faster than a sequential machine. This is not surprising since the information that every node has exactly two neighbors has to be received by the distinguished node D, which is a distance equal to the graph diameter away from the farthest node. In order for the distinguished node to know that it has received the information from all the nodes, it sends out a signal to identify the farthest node and makes sure that this signal comes back. The acceptance time does not depend on the position of the distinguished node, since the radius from any node is the same—in fact, is equal to the diameter. Note that if we had defined acceptance by every node going into a final state, then a cycle could be accepted in one step.

4.2. *Strings*

Let M_S be an automaton that behaves the same as M_C except that it allows two nodes to have one non-# neighbor and it sends a rejection signal S_4 to D if the signal from D meets itself. A node with one non-# neighbor is at the end of the string and it changes to state S_3, an accepting signal, when it receives the original signal from its only non-# neighbor. Then the signal S_3 is propagated back to D. When D receives S_3 from all of its non-# neighbors, it accepts by going into state A. If D receives an S_4 signal from any of its neighbors, it rejects.

Clearly, $\mathcal{C}(M_S)$ rejects all graphs having nodes with more than two neighbors and all the cycles. Thus, the only graphs it can possibly accept are strings. When the ends of the string receive the signal from D, they change state to the accepting signal, so that every string is accepted by $\mathcal{C}(M_S)$. To reject a graph γ,

the number of steps required is at most twice the diameter of the graph. To accept a graph requires twice as many steps as there are nodes between node D and the end of the string farthest away from D. In any case, at most twice diameter time is required. Note that in the case of a string, the acceptance time is shortest if the distinguished node is in the middle of the string.

4.3. *Trees*

We first describe a tree acceptor with base automaton M_A. Informally, we consider the distinguished node D as the root of the tree. First, the leaves of the tree identify themselves as nodes having exactly one non-# neighbor, and change to state S_1. If the graph is a tree, then the non-D immediate ancestors of the leaves will have only one S_0 or D neighbor, the rest being S_1's or #'s. Thus these nodes can identify themselves and change to state S_1. In this way, the signal S_1 propagates up toward the root one level at a time. When the root receives the signal from all of its non-# neighbors, i.e., from its descendants, it accepts. Therefore any tree is accepted by $\mathscr{C}(M_A)$.

Suppose γ is not a tree; then γ must contain a cycle $n_0, n_1, n_2, \ldots, n_j, n_{j+1} = n_0$. Initially, all n_i's $(0 \leqslant i \leqslant j)$ are in state S_0 or state D. Each node n_i cannot go into state S_1 because it has more than one neighbor in state S_0 or state D. Hence the nodes on a shortest path between D and the cycle cannot reach state S_1. Therefore D always has a non-# neighbor not in S_1 so that acceptance cannot occur.

The time it takes for $\mathscr{C}(M_A)$ to accept a tree is equal to the height of the tree using D as the root. Thus, the time complexity is never more than the diameter of the tree, even though it depends on the choice of D.

Since a graph is not a tree iff it has a cycle, we can define a tree recognizer with base automaton M_T using M_C of Section 4.1 to identify cycles in the graph for rejection and M_A above to identify the trees for acceptance.

Similarly, it is easy to define an M_B such that $\mathscr{C}(M_B)$ recognizes binary trees. Each non-leaf node which is not the distinguished node D changes to state S_1 when it has an S_0 or a D neighbor, at most two S_1 neighbors, and no other non-# neighbors. D accepts when it has at most three S_1 neighbors and no other non-# neighbors, since D need not be the root of the binary tree. Any non-D node having more than two S_1 or three non-# neighbors initiates a rejection signal. The existence of cycles is also checked.

The time it takes for $\mathscr{C}(M_T)$ and $\mathscr{C}(M_B)$ to recognize trees and binary trees is at most twice the radius of the graph (as measured from D), hence no more than twice the diameter.

4.4. *Cliques*

In this section we describe a d-graph automaton M such that $\mathscr{C}(M)$ recognizes the complete graph with j nodes for $i \leqslant j \leqslant d + 1$.

We can easily take care of the trivial case when there is only one node in the graph. At the first step, D changes to state q_i, where q_i is used to denote the number of neighbors D has; if the graph is complete, it must have $i + 1$ nodes. At the same time, all the neighbors of D are changed to state S_1. At the second step, each neighbor of D changes to state S_2 if it has $i - 1$ neighbors that are also in state S_1; and the distinguished node changes to state S_3. Finally, at the third step, the distinguished node accepts if all its non-# neighbors are in state S_2, but rejects if it has non-# neighbors in state S_1.

It is straightforward to show that $\mathscr{C}(M)$ recognizes cliques in three steps. This is substantially faster than any sequential automaton. However, note that the radius and diameter of a complete graph are 1, so that the algorithm takes three times the diameter. Note also that the size of the complete graph is identified at the first step by q_i.

4.5. *Rectangular and Square Arrays*

In this section we will show that there is a cellular 4-graph recognizer which recognizes rectangular arrays in diameter time. This recognizer can be easily modified to recognize square arrays in diameter time.

PROPOSITION 11. *Let Γ be a d-graph with a distinguished node D. If $U(\Gamma)$, the underlying graph of Γ, is a rectangular array, then the node farthest away from D is one of the four corner nodes.*

Proof. Suppose $U(\Gamma)$ has r rows and s columns. A node on the ith row and jth column may be identified by (i, j) for $1 \leqslant i \leqslant r$, $1 \leqslant j \leqslant s$. If D is (a, b) then the distance between D and the node farthest away from D is

$$\max_{1 \leqslant i,j \leqslant r} (|\, i - a \,| + |\, j - b \,|) = \max_{1 \leqslant i \leqslant r} |\, i - a \,| + \max_{1 \leqslant j \leqslant s} |\, j - b \,|.$$

This maximum occurs when $i = 1$ or r, and $j = 1$ or s. ∎

The states of the basic automaton M of the cellular 4-graph recognizer that we will construct have a direction component $\delta = (a_1, a_2, a_3, a_4)$, where $a_i \in \{0, \mathrm{N}, \mathrm{E}, \mathrm{W}, \mathrm{S}\}$ and $a_i \neq a_j$ if $a_i \neq 0$ for $1 \leqslant i < j \leqslant 4$. This quadruple at each node n specifies the directions of the arcs at n. $\delta = (a_1, a_2, a_3, a_4)$ says that if $a_i \neq 0$ then the ith arc end points to direction a_i, otherwise its direction is not yet determined. Initially, the direction component δ is $(0, 0, 0, 0)$ for every node. The ith arc end is said to have assigned direction S if a_i is changed from 0 to S. δ is said to be completed if $a_i \neq 0$ for $1 \leqslant i \leqslant 4$.

Proposition 11 allows us to define a cellular 4-graph automaton $\mathscr{M} = (\Gamma, M, H)$ to first identify one of the four corner nodes, call it D', if Γ is a rectangular array. Each non-# node indicates in its state its being a C (corner), B (border) or I (interior) node if it has two, three, or four non-# neighbors. Depending on the number of # neighbors D' has, there are four cases:

Case 1. D' has four # neighbors. Γ has only one non-# node, and $\mathcal{M}$ accepts Γ.

Case 2. D' has three # neighbors. To be a rectangular array, Γ must be a string. Therefore the string recognition automaton of Section 4.2 is used.

Case 3. D' has less than two # neighbors. By Proposition 11, Γ is rejected.

Case 4. D' has two non-# neighbors, i.e., it is a C node. If D' has an I neighbor then $\mathcal{M}$ rejects Γ. If D' has a C neighbor, for Γ to be a rectangular array, it must have two rows (or columns). The two C nodes can then send signals traveling along the upper and the lower border at the same speed. $\mathcal{M}$ accepts Γ iff the two nodes receiving the two signals at each step are neighbors and the signals reach two neighboring corner nodes at the same time. If D' has two B neighbors, then $\mathcal{M}$ starts to assign directions to the arc ends of Γ treating D' as the northwest corner. The two # arc ends of D' are assigned directions N and W, while the non-# ends are assigned E and S. This direction assignment of D' gives Γ an orientation and determines the direction assignments of the other nodes in Γ. D' also sends out two signals, "top" and "left," to identify the two borders. The top (left) signal passes from a node n to n's east (south) neighbor after direction E (S) of n is assigned, provided n is a B or C node and the neighbor to receive the signal is a B or C node. If the neighbor to receive the signal is a # node, then the top (left) signal changes to a "right" ("bottom") signal to be passed to S (E) neighbors as in the case of the left (top) signal. When the right and the bottom signals meet at a node, call it D'', transmission of the two signals stops. A rejection signal is initiated whenever a situation different from the above arises.

Each node n assigns directions to its arc ends according to the following four rules:

RULE 1. If one end of an arc is assigned E or S then in the next step the other end of the arc is assigned W or N, respectively. When a node n assigns direction N to any of its arc ends, a mark N' which lasts for only one step is made on its state, and n starts to count. If the assignment of δ at n is not completed in four steps then a rejection signal is sent to D.

RULE 2. If n receives the top or left signal, then n must be a B (or C) node and (one of) the # arc end(s) is assigned direction N or direction W, respectively.

RULE 3. A node can assign direction E or direction S only after both of its N and W directions have been assigned. Any time a node makes an assignment of direction S, it sends a message to its direction W neighbor n if n is not a # node, so that at the next step, n has a mark S', which lasts for one time step.

(3a) A top (right) border node assigns direction E (S) to its unassigned arc end leading to a B or C neighbor, and S (E) is assigned to the other arc end. If no such unassigned arc end exists, a rejection signal is sent to D.

(3b) A C node receiving the top signal assigns E to the unassigned # arc end and S to the arc end leading to a B neighbor. Again a rejection signal is sent to D if there are no such arc ends.

(3c) The C node D'' where right and bottom signals meet assigns E and S to the two # arc ends arbitrarily and a signal F is sent to D'.

(3d) A node n assigns directions E and S when both of its N and W directions are assigned and its N neighbor has the mark S′ and one of the unassigned arc end leads to a neighbor with the mark N′. This unassigned arc end is assigned direction E, and thus forces the only other unassigned arc end to have direction S. A node remains in the same state after its N and W directions are assigned until the conditions above are satisfied; or if four time steps have passed and the conditions are not satisfied, then a rejection signal is sent to D. A rejection signal is also initiated when any of the following is true: (i) more than one neighbor has the mark N′; (ii) a neighbor is marked with N′ but the north neighbor does not have the mark S′; (iii) one if its unassigned arc ends leads to a node with N assigned but the mark N′ had already disappeared; or (iv) the north neighbor has the mark S′ but it does not have an unassigned arc end leading to a neighbor with mark N′.

RULE 4. Any time a direction assignment gives a conflict or a situation not described in the above three rules, then a rejection signal is sent to D.

The above direction assignment rules attempt to make sure that if node m is (e.g.) the W neighbor of node k, then m's N neighbor is also the W neighbor of k's neighbor, as would be the case for an array in the plane. However, $\mathcal{M}$ can successfully assign directions to all arc ends without conflict even when Γ is not a rectangular array. Therefore after the completion of direction assignment, $\mathcal{M}$ starts the construction of a special (canonical) breadth-first spanning tree which cannot be completely constructed unless Γ is a rectangular array as follows: When the signal F from D'' reaches D', D' sends out a signal P which propagates from neighbor to neighbor as in the spanning tree construction. If n is a top (or left) node, it receives P from its W (or N) neighbor only, and then passes P to its other neighbors. If n is neither a top nor a left node, it must receive P from its W and N neighbors simultaneously and it always chooses to take P from its N neighbor and then passes P to its south and east neighbors. If a node receives P in any other way, then a rejection signal is sent to D. When D'' receives P it sends a return signal back to D', which then sends an acceptance signal to D. Since D' is a node farthest away from D, if D receives the acceptance signal without getting any rejection signal, then Γ is accepted.

PROPOSITION 12. *If Γ is a 4-graph such that $U(\Gamma)$ is a rectangular array, $\mathcal{M}$ accepts Γ in diameter (Γ) time.*

Proof. If the underlying graph of Γ is a rectangular array with r rows and s columns, it is easy to see that $\mathcal{M}$ accepts Γ. We only need to show that the acceptance takes diameter (Γ) time.

After the northwest corner of Γ is identified, $\mathcal{M}$ proceeds to assign directions to the nodes in the top row of $U(\Gamma)$. Instead of doing the direction assignment row by row, which will take area time, $\mathcal{M}$ starts the assignment of the $(i+1)$st row as soon as it has enough information—before the ith row is completely assigned. At each row, the leftmost node is the first one with completed δ, and the set of such nodes grows one node at a time. The N direction of a node n in the $(i+1)$st row can be assigned when it has a neighbor n_1 in the ith row with a completed δ. The W direction of n is assigned when it has another neighbor whose δ is completed and n is its E neighbor, or if n is a left border node. Similarly, n can assign the E and S directions when it has a neighbor m identified as the S neighbor of a node m_1 in the ith row (this places m on the $(i+1)$st row) and m_1 is the E neighbor of n_1.

Now the set of nodes with completed δ's in the $(i+1)$st row is extended by one. Hence the direction assignments of Γ take time proportional to the diameter of Γ.

After the direction assignment, the propagation of the signal P from D' defines a spanning tree of a special form rooted at D'. It contains all the S–N arcs in Γ and the E–W arcs between nodes in the first row. All the propagation of signals can be accomplished in order (diameter) time. Thus Γ is accepted by $\mathcal{M}$ in diameter(Γ) time. ∎

PROPOSITION 13. *If a connected 4-graph Γ is accepted by $\mathcal{M}$, then $U(\Gamma)$ is a rectangular array.*

Proof. Suppose Γ is a connected 4-graph accepted by $\mathcal{M}$. Clearly $U(\Gamma)$ is a rectangular array if it has only one node or if it is a string. For the remaining cases, the direction assignment at each node is successful. A careful examination of the process shows that there can only be two situations:

(1) When all the arcs join the nodes the same way as in a rectangular array. In this case Γ is a rectangular array.

(2) There is an arc joining a node m to another node n, but the north neighbors of m and n are not neighbors. At some step m assigns E to its arc end leading to node n if node n has the mark N′ and the N neighbor of m has the mark S′. For convenience of notation, let us denote D' by $n(1, 1)$. Starting with the first step of the direction assignment, for each node $n(i, j)$, its E neighbor is denoted by $n(i, j+1)$ and its south neighbor is denoted by $n(i+1, j)$ until the conflicting situation as above arises. It is not hard to see that if two nodes

have the mark N' at the same time step and one of the nodes is $n(i, j)$ then the others must be $n(i - k, j + 2k)$ for some $0 < k < i$. When the conflicting situation arises, nodes $n(i, j)$ and $n(i - k, j + 2k)$ both have the mark N′, and node $n(i - 1, j)$ is joined to node $n(i - k, j + 2k)$ by an arc for some $0 < k < i$. The distance from node $n(1, 1)$ to node $n(i - 1, j)$ is $i + j - 3$. The distance from node $n(1, 1)$ to node $n(i - k - 1, j + 2k)$ is $i + j - 3 + k > i + j - 3$ since $k > 0$. When the direction assignment phase is completed, signal P is sent from $D' = n(1, 1)$. After $i + j - 3$ steps, P reaches node $n(i - 1, j)$ but not node $n(i - k - 1, j + 2k)$. Thus at the next step, node $n(i - k, j + 2k)$ receives P from its W neighbor but not its N neighbor, and a rejection signal is generated. This is impossible because Γ is accepted by $\mathcal{M}$. Since this is the only discrepancy that can occur in the direction assignment phase, and it would be detected by the canonical spanning tree construction, the nodes with completed δ's do indeed form a rectangular array.

Moreover, Γ does not have nodes with incomplete δ's, since Assignment Rule 1 assures that no nodes can have partially assigned δ if Γ is accepted by $\mathcal{M}$, and connectedness of Γ assures that every node's arc ends are assigned. For more details, see (Wu, 1978).

The direction assignment of the nodes of Γ takes diameter time (because starting from D', all the arcs at each node are used and thus D'' is reached from a shortest path). All the other parts also take diameter time. ▮

COROLLARY 1. *$\mathcal{M}$ recognizes rectangular arrays in diameter time.*

COROLLARY 2. *There is a cellular 4-graph recognizer that recognizes square arrays in diameter time.*

Proof. $\mathcal{M}$ can be modified slightly to an $\mathcal{M}'$ which recognizes square arrays. When D' gets signal F from D'', it sends another signal that zigzags down by going in directions E and S alternatively. If this signal does not reach D'' from its N neighbor, then a rejection signal is sent to D; otherwise an acknowledge signal is sent back to D'. D' sends an acceptance signal to D only when both this acknowledge signal and the returning signal are received. ▮

For any $d > 4$, $\mathcal{M}$ can be modified in the obvious way so that $\mathcal{M}$ can recognize the d-graphs which are rectangular arrays. The results of this section can be generalized to n-dilensional rectangular arrays.

4.6. *Eulerian Graphs*

An Eulerian graph is a connected graph such that starting from any node, it is possible to traverse each arc exactly once and pass through all points. It is well known that a graph G is Eulerian iff every node of G has even degree. A connected d-graph Γ is Eulerian iff its underlying graph $U(\Gamma)$ is Eulerian.

Eulerian graphs can be recognized by a cellular d-graph recognizer $\mathcal{M} =$

(Γ, M, H) with a distinguished node D in radius time as follows: Each non-# node sends a rejection signal to D if it has an odd number of non-# neighbors. Whenever D receives the rejection signal from any node, it rejects Γ. Note that if no rejection signal is received by D within $r + 1$ steps, where r is the radius of Γ as measured from D, then Γ is Eulerian. In (Wu and Rosenfeld, 1979), it is shown that it takes twice radius time for D to know that the spanning tree has been constructed. Therefore at the first step, $\mathscr{M}$ also starts the spanning tree construction. When D receives the message that the spanning tree is constructed and no rejection signal has arrived, Γ is accepted. Clearly the recognition process takes twice radius time.

4.7. *Bipartite and Complete Bipartite Graphs*

A bipartite graph G is a graph whose nodes can be partitioned into two subsets V_1 and V_2 such that every arc of G joins V_1 with V_2. If G contains every possible arc joining V_1 and V_2 then G is a complete bipartite graph, and is denoted by $K_{a,b}$ where a, b are the cardinalities of V_1 and V_2. Clearly, a graph is bipartite iff it is bicolorable since the nodes of V_1 can be given one color and the nodes of V_2 the other color.

Let Γ be a d-graph. A cellular d-graph recognizer recognizes bipartite graphs by making sure that Γ is bicolorable. The distinguished node D first colors itself (by having a special mark in its state to denote the color) with one color, say blue. A non-# node with one or more blue neighbors colors itself red and a non-# node having one or more red neighbors colors itself blue. A non-# node with all neighbors uncolored does not change its state. If a node has both red and blue neighbors, then Γ cannot be bicolorable and a rejection signal is sent to D. At the end of $r + 1$ steps, where r is the radius of Γ as measured from D, either all the nodes of Γ are colored or one of the nodes has sent a rejection signal to D. At the end of $2r + 1$ steps if D has not received a rejection signal then Γ is bicolorable. Again, by starting the construction of the spanning tree at the first step, D knows that $2r$ steps have passed when it receives the meassage that the tree has been constructed. The time it takes for recognizing a bipartite graph is twice the radius plus 1 time steps.

In a d-graph every node has at most d non-# neighbors. The number of complete bipartite d-graphs is very limited; in fact, it is at most $d(d + 1)/2$. If Γ is a d-graph such that its underlying graph $U(\Gamma)$ is $K_{a,b}$ then $a \leqslant d$ and $b \leqslant d$.

To recognize a complete bipartite d-graph, the distinguished node D at the first step colors itself blue and records the pair of numbers i, i_0 in its state where i is the number of non-# neigbors D has and i_0 is the lowest-numbered arc end at D leading to a non-# node. At the next step, all D's neighbors color themselves red; D_0, the i_0th neighbor of D, also records the numbers j, i in its state where j is the number of non-# neighbors D_0 has, and i is the first of the pair of numbers in D's state. At the third step, i is sent from D_0 to all its neigbors, which

also color themselves blue; the number j is sent to D from D'. At the fourth step, each blue node with the number i makes sure that it has i non-# neighbors and all of them are red; otherwise a rejection signal is sent to D. D also sends the number j to all of its neighbors. At the fifth step, each red node with the number j makes sure that it has j non-# neighbors and all of them are blue; otherwise a rejection signal is sent to D. At the sixth step, if no rejection signal has reached D then Γ is accepted. Every complete bipartite d-graph has diameter 2. Such graphs are recognized in six steps, which is three times the diameter.

4.8. *Stars and Wheels*

A *star* is a special complete bipartite graph $K_{1,n}$ consisting of a center node which has n neighbors. Therefore the cellular d-graph recognizer in Section 4.7 can easily be modified to recognize stars.

An *extended star* consists of a center node with n strings of nodes emanating from it, instead of n neighbors. It is easy to see that a cellular d-graph recognizer with distinguished node D recognizes extended stars in time proportional to the diameter of the graph. It first sends a signal T from D to find the center D' of the star as the only node having more than two non-# neighbors. Then a signal is sent from D' to make sure that only strings emanate from it. Note that if signal T from D reaches # nodes without finding the center then the d-graph is a string which is a degenerate extended star.

A wheel is obtained from a star by having the non-center nodes form a cycle. Let $\mathcal{M} = (\Gamma, M, H)$ be a cellular d-graph recognizer that first identifies all the nodes having three non-# neighbors as B nodes. Then the only non-B node is the center node C. $\mathcal{M}$ makes sure that no other non-B nodes exist; and that each B node is a neighbor of C and has two B neighbors and no other non-# neighbors. If every node is a B node, so that the centered node cannot be identified, $\mathcal{M}$ accepts the d-graph if it has four nodes each of which is joined to the other three by an arc. Clearly $\mathcal{M}$ recognizes wheels in time proportional to the diameter of the d-graph.

More details on the recognition of stars and wheels can be found in (Wu, 1978).

4.9. *Planar Graphs*

Planarity of graphs has been studied quite extensively. Many of the sequential algorithms use the approach of constructing a representation of a planar embedding of the given graph, such that a graph is planar iff such a representation can be completed. Hopcroft and Tarjan (1974) discovered a sequential planarity algorithm that is linear in the number of nodes of the graph. Their measure of time complexity is based on using a random access computer model, and a memory cell is allowed to hold integers with absolute values as large as kV for some constant k, where V is the number of nodes in the given graph. Hence it is possible to associate a different integer with each node. Our cellular d-graph automata have fixed memory regardless of the size of the input graph. This

makes distinguishing the nodes and thus testing the embeddability of a graph in the plane difficult.

There are three other criteria for planarity. The earliest characterization was given by Kuratowski (1930). He proved that a graph is planar iff it has no subgraph homeomorphic to the complete graph C_5 or the complete bipartite graph $K_{3,3}$. This is equivalent to saying that a graph has no subgraph contractible to C_5 or $K_{3,3}$. In view of the results on subgraph isomorphism in Section 3, one might expect to be able to design a cellular d-graph automaton to test planarity using Kuratowski's criterion. However, the length of the sequence of contractions depends on the graph given.

Whitney (1932, 1933) expressed planarity in terms of the existence of a dual graph, which is difficult for a cellular d-graph automaton to verify. MacLane (1937) expressed planarity in terms of cyclic structure; a graph G is planar iff every nontrivial block of G has a cycle basis $Z_1, \ldots, Z_m$ and one additional cycle such that every arc occurs in exactly two of these $m + 1$ cycles. The value of m is determined by the number of arcs and nodes in the graph, hence is dependent on the graph given. It is not clear how a cellular d-graph automaton can use this criterion.

An extremely slow algorithm for planarity testing would be for a cellular d-graph automaton to test the existence of subgraphs contractible to C_5 by systematically choosing five nodes, and testing if there are 10 nonintersecting paths, one path for each pair of nodes. Similarly, the existence of $K_{3,3}$ can be checked by selecting two sets of three nodes each and testing for the correct nonintersecting paths.

It is interesting that in spite of the existence of linear sequential algorithms for planarity, we can only find an extremely slow cellular d-graph algorithm to recognize planar graphs. One reason is that planarity is a very global property which is difficult for a cellular d-graph recognizer to verify.

4.10. *Some Other Examples*

Some of the results in (Wu and Rosenfeld, 1979) allow us to define cellular d-graph recognizers for certain graph structures. In this section we mention some of these.

A graph is *biconnected* if it has no cut nodes. All cut nodes of Γ can be identified by a deterministic cellular d-graph automaton $\mathcal{M}$ in time proportional to area(Γ) times diameter(Γ), where area(Γ) is the number of nodes in Γ. and $\mathcal{M}$ knows when the identification process is finished. Therefore there exists such an $\mathcal{M}$ that recognizes biconnected graphs.

The *line graph* of a graph G is the graph obtained by creating a node for each arc of G and joining together those nodes corresponding to arcs that are adjacent in G. In (Wu and Rosenfeld, 1979) we show the existence of a deterministic cellular d-graph recognizer for line graphs. The time required for recognition is proportional to $2^d \cdot \text{area}(\Gamma)$.

Suppose a number representing some property or measurement μ of Γ (e.g., radius, diameter) can be stored in unary notation along some path of length l in Γ. This path can then act like a DBCA (deterministic bounded cellular automaton), where the nodes with zeros stored are treated as boundary nodes. Thus any DBCA language (predicate) $\mathscr{L}$ of numbers defines a cellular d-graph language (predicate) $\hat{\mathscr{L}}$ such that the value of property μ is in $\mathscr{L}$. Let $T(n)$ be the DBCA acceptance time of $\mathscr{L}$ for inputs of length n. Then the acceptance time of $\hat{\mathscr{L}}$ is proportional to the maximum of $T(l)$ and the time needed to obtain and store the number representing property μ along the path.

We have shown how to obtain the radius of a d-graph (from the distinguished node) in diameter time. Therefore the set of d-graphs having radii in a given linear time DBCA language (predicate) is a diameter-time cellular d-graph language (predicate). For example $\{a^m \mid m \text{ is a prime}\}$ is a linear-time DBCA language; hence the set of d-graphs having prime radius is a diameter-time cellular d-graph language.

We have also shown that the traversal of a DFST of Γ can be represented by a sequence of arc end numbers. The length of the traversing path is $l = 2 \cdot \text{area}(\Gamma) - 2$, since starting from the root, the arc from each node to its ancestor in the DFST is traversed twice, and the root has no ancestor. Each arc end number in the path may be considered as a cell, represented in the node it is at. Hence a node may have as many as d cells in it. The neighbor relation is well defined. The lowest-numbered cell at each node has state a and the others have state b. Therefore we have a DBCA with the number of a-cells equal to area(Γ). It is easy to see that the a's can be packed so that the DBCA has the form $a^{\text{area}(\Gamma)}b^{l-\text{area}(\Gamma)}$ in l steps. Now the b-cells can be treated as boundary cells and the first a-cell as the general cell in a one-dimensional firing squad so that all the a-cells fire simultaneously in time proportional to area (Γ). This then allows the DBCA to test whether area(Γ) belongs to some DBCA language of numbers. Therefore the set of d-graphs having areas in a given linear time DBCA language (predicate) is an area-time cellular d-graph language (predicate).

We have described a cellular d-graph automaton $\mathscr{M}$ that counts and outputs the area of the given graph in base d, least significant digit first, in diameter time, and knows when it is finished. It is easy to modify $\mathscr{M}$ so that it recognizes the set of d-graphs whose areas are numbers in some finite-state language. We can also define recognizers for sets of areas that are not finite state. For example, let us modify $\mathscr{M}$ to output the area in base 2 in diameter time. We can then define a cellular d-graph recognizer $\hat{\mathscr{M}}$ to accept the given d-graph Γ iff its area is a power of 2. To do this, $\hat{\mathscr{M}}$ simulates $\mathscr{M}$ to count the area of Γ in binary; whenever the output is 1, $\hat{\mathscr{M}}$ sets a flag. $\hat{\mathscr{M}}$ rejects Γ if another 1 is generated after the flag is set, and $\hat{\mathscr{M}}$ accepts Γ if no more 1's are generated when the counting is done.

RECEIVED: June 23, 1978; REVISED: May 29, 1979

REFERENCES

HOPCROFT, J., AND TARJAN, R. (1974), Efficient planarity testing, *J. Assoc. Comput. Mach.* **21**, 549–568.

KURATOWSKI, K. (1930), Sur le problème des courbes gauches en topologie, *Fund. Math.* **15**, 271–283.

MACLANE, S. (1937), A structural characterization of planar combinatorial graphs, *Duke Math. J.* **3**, 340–372.

MYLOPOULOS, J. P., AND PAVLIDIS, T. (1975), On the topological properties of quantized spaces. I. The notion of dimension, *J. Assoc. Comput. Mach.* **18**, 239–246.

WHITNEY, H. (1932), Non-separable and planar graphs, *Trans. Amer. Math. Soc.* **34**, 339–362.

WHITNEY, H. (1933), Planar graphs, *Fund. Math.* **21**, 73–84.

WU, A. (1978), "Cellular Graph Automata," Ph. D. dissertation, Department of Computer Science, University of Maryland, College Park, Md.

WU, A., AND ROSENFELD, A. (1979), Cellular graph automata. I. Basic concepts, graph property measurement, closure properties, *Inform. Contr.* **42**, 305–329.