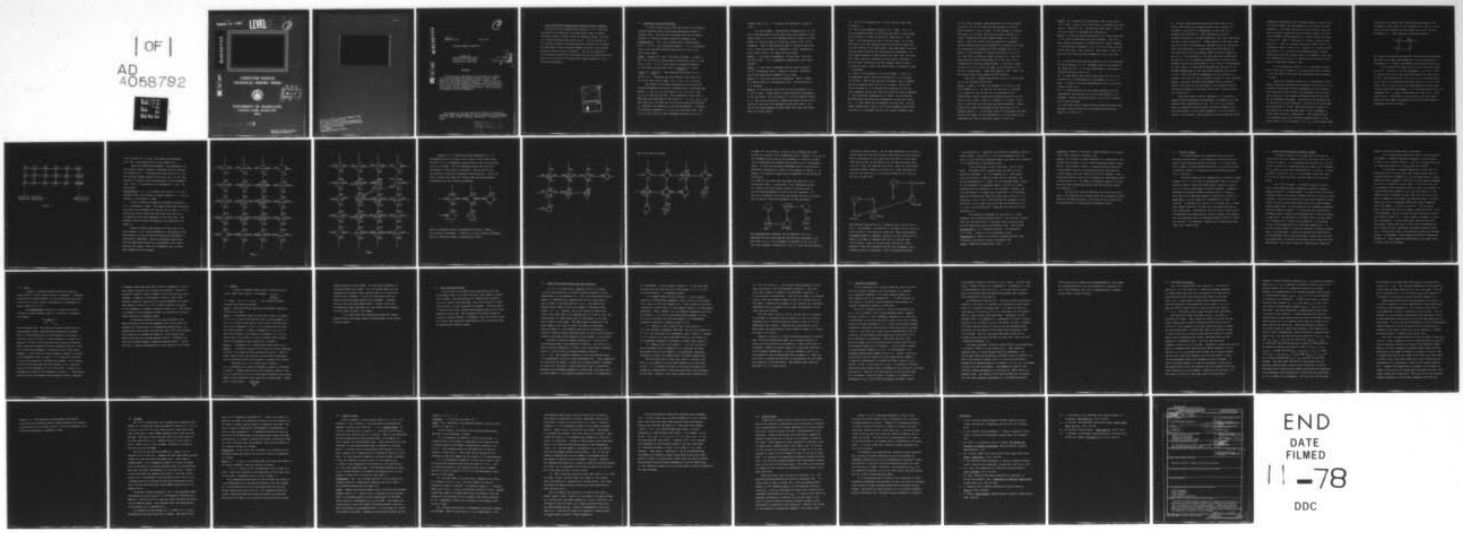
AD-A058 792 MARYLAND UNIV COLLEGE PARK COMPUTER SCIENCE CENTER F/G 9/4
CELLULAR GRAPH ACCEPTORS, 4.(U)
JUN 78 A WU
UNCLASSIFIED CSC-TR-667 AFOSR-77-3271
AFOSR-TR-78-1363 NL



END
DATE FILMED
11-78
DDC

COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

TR-667
AFOSR-77-3271

June 1978

CELLULAR GRAPH ACCEPTORS, 4

Angela Wu
Computer Science Center
University of Maryland
College Park, MD 20742

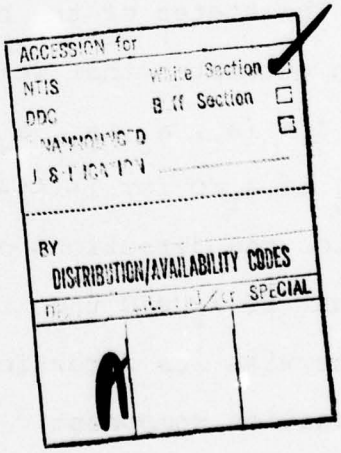
ABSTRACT

Diameter-time algorithms are presented for recognition of rectangular and square arrays, Eulerian graphs, bipartite and complete bipartite graphs, stars, and wheels by cellular d-graph acceptors. Slower algorithms are given for construction of a depth-first spanning tree (area time) and for identification of cut nodes, borders, and central points (diameter·area time). The recognition of planarity is also discussed.

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 Mrs. Virginia Kuykendall in preparing this paper.

1. Graph structures recognized by cellular d-graph automata.

Cellular d-graph automata were defined in [1]. Efficient algorithms for acceptance of various basic types of graphs, including cycles, strings, trees and complete graphs, by cellular d-graph acceptors were given in [2]. In this section we will present diameter time algorithms for recognizing rectangular and square arrays, Eulerian graphs, bipartite and complete bipartite graphs, stars, and wheels. Due to its complexity, the transition function in each case will not be given explicitly. Rather, the actions of the cellular d-graph automata will be described informally.



1.1 Rectangular and Square Arrays

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

Proposition 1: 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 i-th row and j-th column may be identified by (i.j) for $1 \le i \le r$, $1 \le j \le s$. If D is (a,b) then the distance between D and the node farthest away from D is $\max_{1 \le i,j \le r}(|i-a| + |j-b|)$ $= \max_{1 \le i \le r}|i-a| + \max_{1 \le j \le s}|j-b|$. This maximum occurs when i=1 or r, and j=1 or s. Therefore the node farthest away from D is one of the four corner nodes (1,1), (1,s), (r,1) or (r,s).

The states of the basic automaton M of the cellular 4-graph automaton that we will construct have a direction component $\delta = (a_1, a_2, a_3, a_4)$ where $a_i \in \{O, N, E, W, S\}$ and $a_i \neq a_j$ if $a_i \neq O$ for $1 \le i < j \le 4$. 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 $a_i \neq O$ then the i-th arc end points to direction a_i, otherwise its direction is not yet determined. Initially, the direction component δ is (O,O,O,O) for every node. The i-th arc end is said to have assigned direction S if a_i is

changed from O to S. δ is said to be completed if $a_i \neq O$ for $1 \leq i \leq 4$.

For any 4-graph , the cellular automaton $M=(\Gamma, M, 1\text{-}1)$ with a distinguished node D at the first step records in each node's state a C, B or I to indicate that the node is a corner, border or interior node if it has two, three or four non-# neighbors. Then M identifies the node D' farthest away from D [2]. This can be done in diameter(Γ) time. Depending on what kind of node D' is, there are four cases:

Case 1: D' has four # neighbors. In this case Γ has only one non-# node. It is a degenerate rectangular array and M accepts Γ.

Case 2: D' has three # neighbors and one non-# neighbor. To be a rectangular array, Γ must be a string. Therefore the string recognition automaton [2] is used.

Case 3: D' has less than two # neighbors. Thus D' cannot be a corner node of a rectangular array. By Proposition 1, Γ is rejected.

Case 4: D' has exactly two # and two non-# neighbors, i.e. it is a C node. M then starts to make direction assignments to all the nodes. We can think of D' as the northwest corner of Γ. The two arc ends at D' leading to# nodes (we will call them the # arc ends) are assigned directions N and W. For the two arc ends leading to non-# nodes (the non-# arc ends), there are three cases:

(a) One of the neighbors of D' is an interior node; then M rejects Γ.

(b) One of the neighbors, node m, is a C node. For Γ to be a rectangular array, it must have two rows or two columns. We can assign S to the arc end of D' leading to m. This gives Γ an orientation so that it has two rows. It is then easy for n,m to send signals travelling along the upper and the lower border at the same speed. Each step makes sure that the two nodes receiving the two signals are neighbors. If each signal reaches a corner node at the same time, and these two corner nodes are neighbors of each other, M accepts Γ. For all other situations, say the signals cross or reach an interior node or reach the corner nodes at different times, Γ is rejected.

(c) Both of the neighbors of D' are B nodes. If $U(\Gamma)$ is a rectangular array, then its size must be r x s for some $r \geq 3$ and $s \geq 3$. Each B node has one # neighbor, one I neighbor and two neighbors which can be B or C nodes. One of the non-# arc ends of D' is assigned the direction E, say the lower numbered of the two, while the other arc end is assigned 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. The top signal passes from a node n to n's east neighbor after direction E of n is assigned; and the left signal transmits from

n to n's south neighbor after direction S of n is assigned; provided n is a B or C node and the neighbor to receive either signal is a B or C node. If the neighbor to receive the signal is a # node, then the top signal changes to a 'right' signal and the left signal changes to a 'bottom' signal. The right signal will be passed to S neighbors and the bottom signal will be passed to E neighbors provided the directions are assigned and the neighbors are B or C nodes. If the neighbor to receive the signal is not a B or C node then a rejection signal is sent to D. When the right and the bottom signals meet at a node, call it D", transmission of the two signals stops. A rejection signal is sent to D if D" is not a C node, or if any other two of the four signals meet. These four signals, left, right, top and bottom, define the borders of the d-graph Γ.

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

Rule 1: Suppose n is the i-th neighbor of m and m is the j-th neighbor of n. If the i-th arc end of m is assigned E or S then the j-th arc end of n is assigned W or N respectively. When a node first assigns direction N to any one of its arc ends, a mark N' is made on its state. This mark N' lasts for only one step and then disappears. Either W and N of a node are assigned at the same step, or N is assigned before W. In any case, after N is assigned, the node counts four steps; if the assignment of δ at the node is not completed yet, then a rejection signal is sent to D.

Rule 2: If n receives the top signal, then n must be a B (or C) node. (One of) the # arc end(s) is assigned the direction N. Similarly, if n receives the left signal, (one of) the # arc end(s) is assigned the direction W.

Rule 3: A node can assign directions E or 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 in the next step, n has a mark S'. This mark S' lasts for only one time step and then disappears at the next step.

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

(b) A right border node assigns direction E to its # arc end and S to its unassigned arc end leading to a B or C node. If there is no such unassigned arc end, a rejection signal is sent to D.

(c) 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.

(d) The corner 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'.

(e) A node n assigns directions E and S when both of its N and W directions are assigned and its north 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. In particular, if (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 of 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'; then a rejection signal is also initiated.

Rule 4: Any time a direction assignment gives a conflict, namely, more than one arc end needs to be assigned the same direction according to the rules above, then a rejection signal is sent to D.

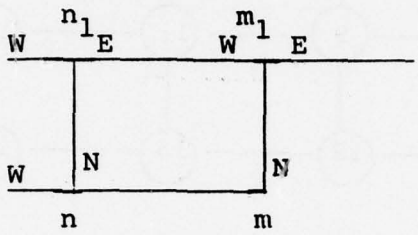
When the signal F from D" reaches D', D' sends out a signal P which propagates from neighbor to neighbor as in the spanning tree construction in [2]. If n is a top (or left) node, it receives P from its west (or north) neighbor only, and then passes P to its other neighbors. If n is neither a top nor a left node, it must receive P from its west and north

neighbors simultaneously and it always chooses to take P from its north neighbor and then passes P to its south and east neighbors. If a node receives P in any other way, say only from its north neighbor but not the west neighbor, or from three neighbors simultaneously, etc., then a rejection signal is sent to D. When D" receives P it sends a return signal back to D'. When D' receives the return signal from D", an acceptance signal is sent 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. Note that the transmission of the F, P, returning and acceptance signals all take order diameter time.

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

After the northwest corner of Γ is identified, 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, M starts the assignment of the second row as soon as it has enough information--before the first 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 second row can be assigned when it has a neighbor n_1 in the first 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 first row (this places m on the second row) and m_1 is the E neighbor of n_1. The situation can be described by



All these are specified by rule 3e of the direction assignment. Now the set of nodes with completed δ's in the second row is extended by one. Then m can assign its W direction and so on.

Let us denote the node in the i-th row and j-th column by n(i,j). Thus n(i,j) is the W neighbor of n(i,j+1). Rule 3e implies that n(i,j) gets its W assignment one step after n(i,j-1) has completed its δ or n(i,j) is a left border node. n(i,j) gets its N assignment one step after n(i-1,j) has completed its δ and gets its E and S assignments one step after n(i,j+1) gets its N assignment, i.e., two steps after n(i-1,j+1) completes its δ. Figure 1 shows when each node in U(I) completes its δ. The number in the circle is the time step at which δ at that node is completed if the north west corner's assignment is done at step 1.

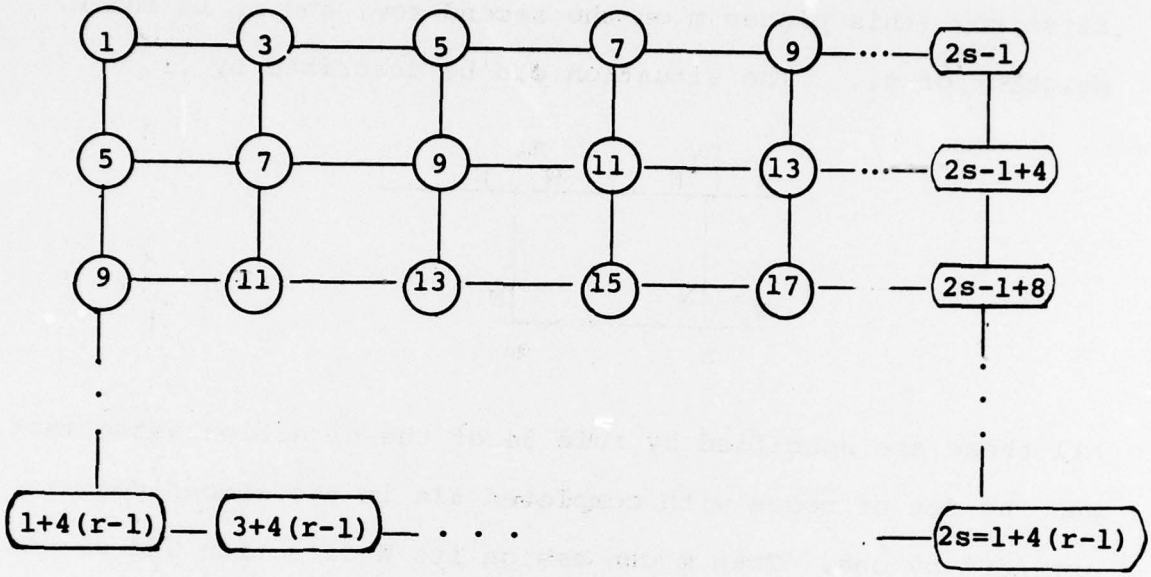


Figure 1

Since $2s-1+4(r-1) \leq 4(r+s)$, 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 M in diameter(Γ) time. We have proved,

Proposition 2: If Γ is a 4-graph such that $U(\Gamma)$ is a rectangular array, the cellular 4-graph automaton $M = (\Gamma, M, 1\text{-}1)$ accepts Γ in diameter(Γ) time.

Figure 2 shows the direction assignment process of a 5 x 4 rectangular array. The number beside each direction shows when the assignment of that arc end is made. S' or N' inside the circles shows when the marks are made and it is understood that they disappear at the next step. The darkened lines indicate the spanning tree defined by the signal P.

Figure 3 differs from Figure 2 in only two of its arcs; however, it no longer represents a rectangular array. From Figure 3 we can see that the direction assignment is successfully completed. The non-rectangular-array-ness will be found when signal P is transmitted, since node k receives the signal P from its W neighbor only, so that this d-graph will be rejected.

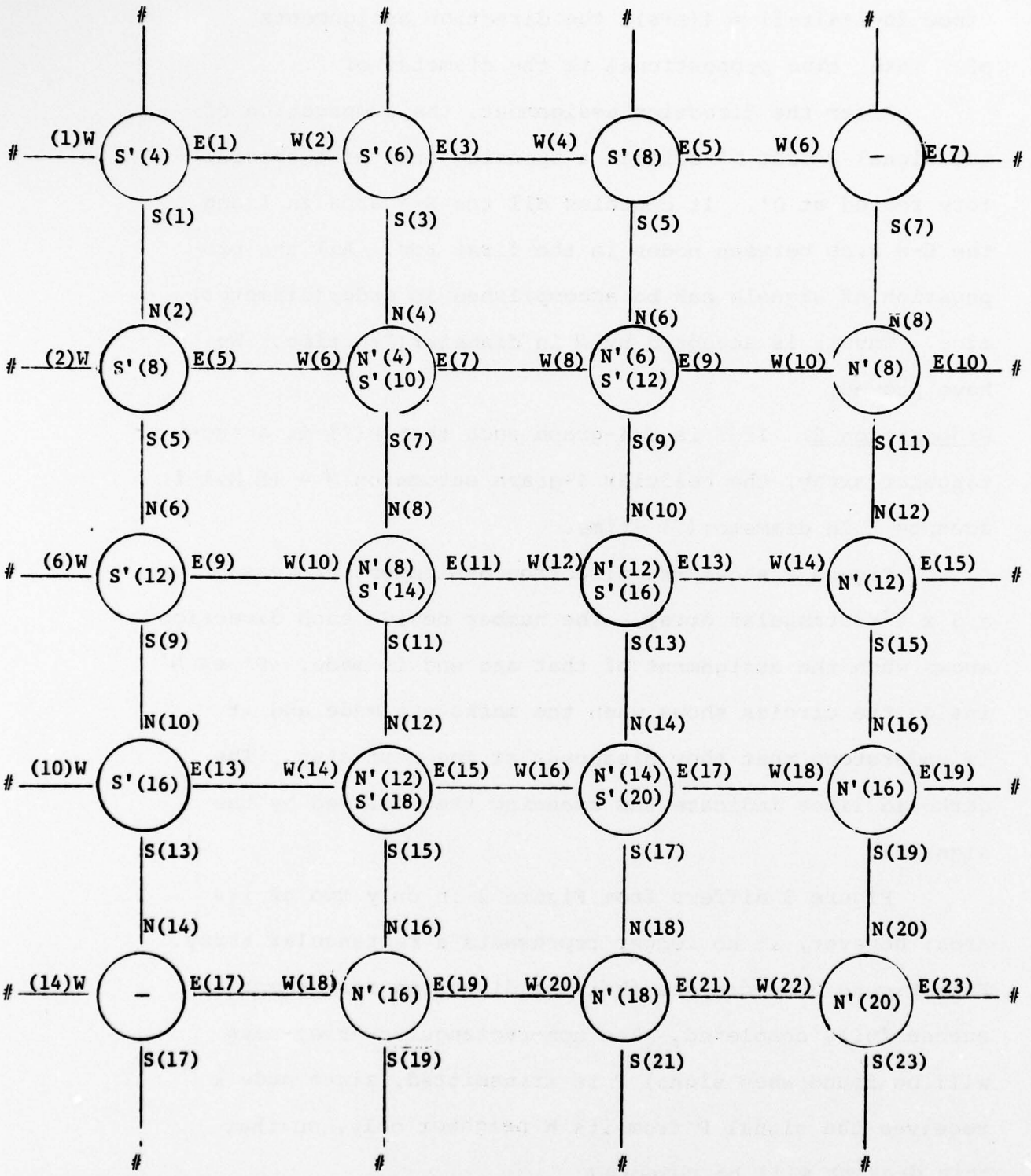
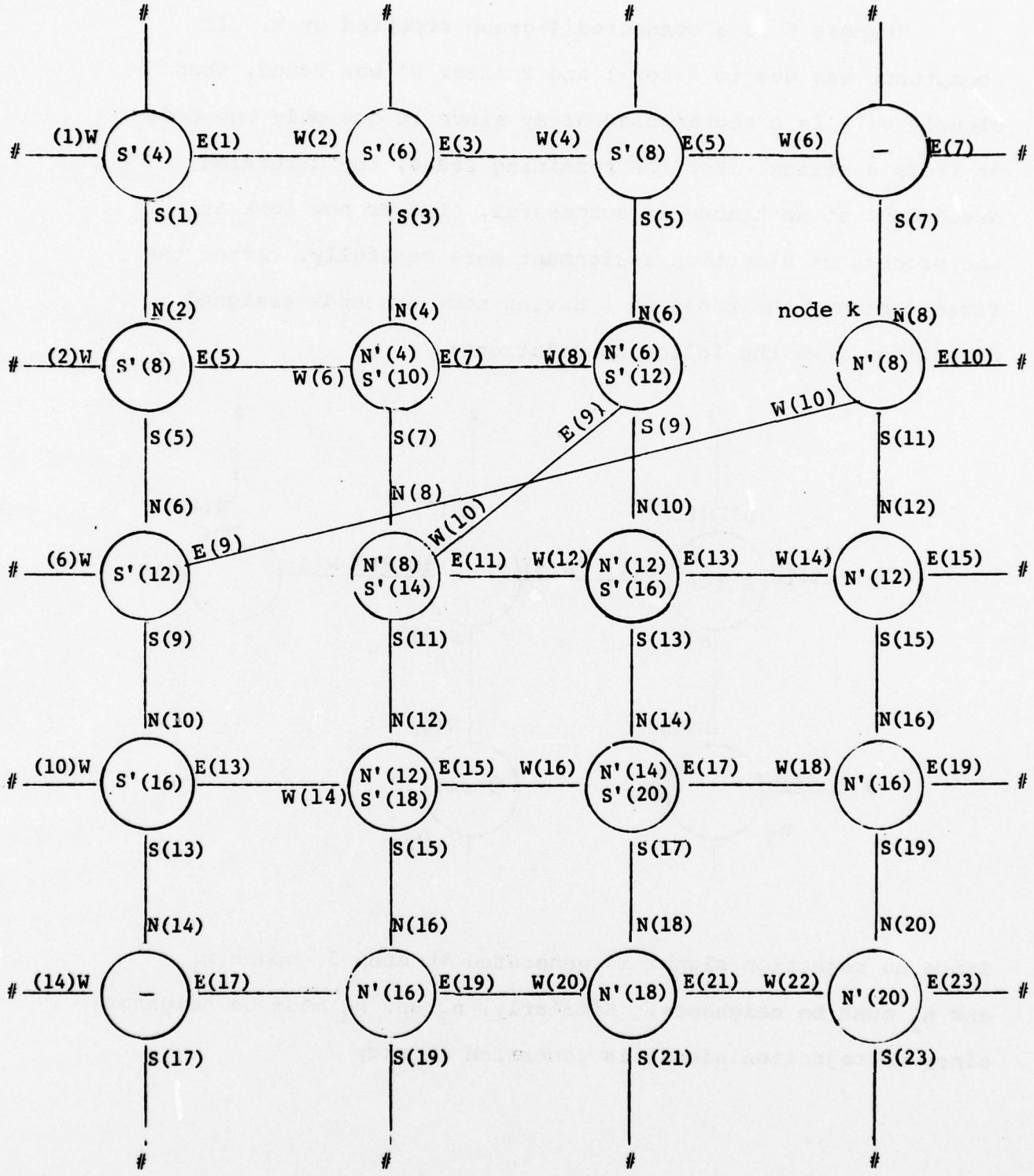
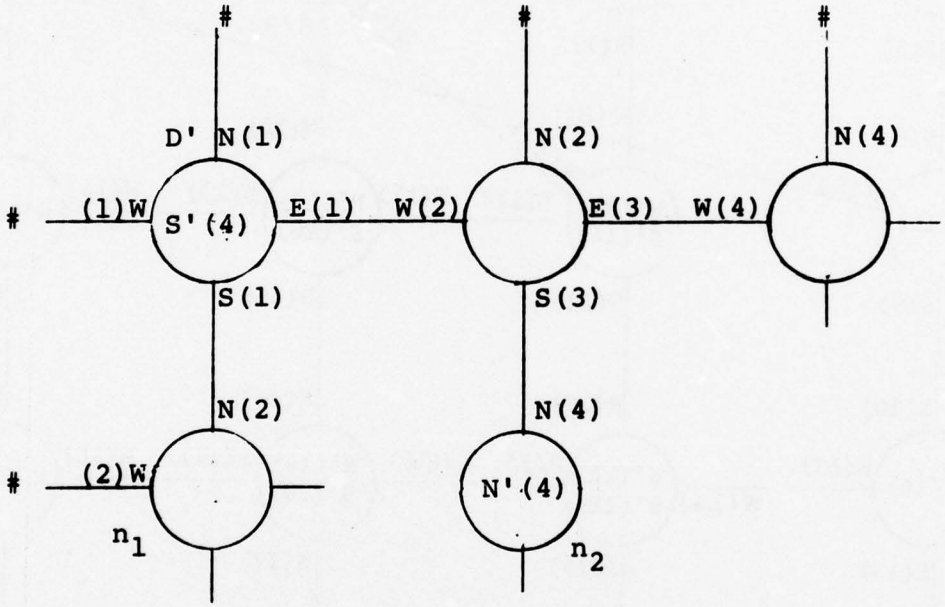

#
(1)W
S'(4)
E(1)
W(2)
S'(6)
E(3)
W(4)
S'(8)
E(5)
W(6)
–
E(7)
S(1)
S(3)
S(5)
S(7)
N(2)
N(4)
N(6)
N(8)
(2)W
S'(8)
E(5)
W(6)
N'(4)
S'(10)
E(7)
W(8)
N'(6)
S'(12)
E(9)
W(10)
N'(8)
E(10)
S(5)
S(7)
S(9)
S(11)
N(6)
N(8)
N(10)
N(12)
(6)W
S'(12)
E(9)
W(10)
N'(8)
S'(14)
E(11)
W(12)
N'(12)
S'(16)
E(13)
W(14)
N'(12)
E(15)
S(9)
S(11)
S(13)
S(15)
N(10)
N(12)
N(14)
N(16)
(10)W
S'(16)
E(13)
W(14)
N'(12)
S'(18)
E(15)
W(16)
N'(14)
S'(20)
E(17)
W(18)
N'(16)
E(19)
S(13)
S(15)
S(17)
S(19)
N(14)
N(16)
N(18)
N(20)
(14)W
–
E(17)
W(18)
N'(16)
E(19)
W(20)
N'(18)
E(21)
W(22)
N'(20)
E(23)
S(17)
S(19)
S(21)
S(23)

Figure 2

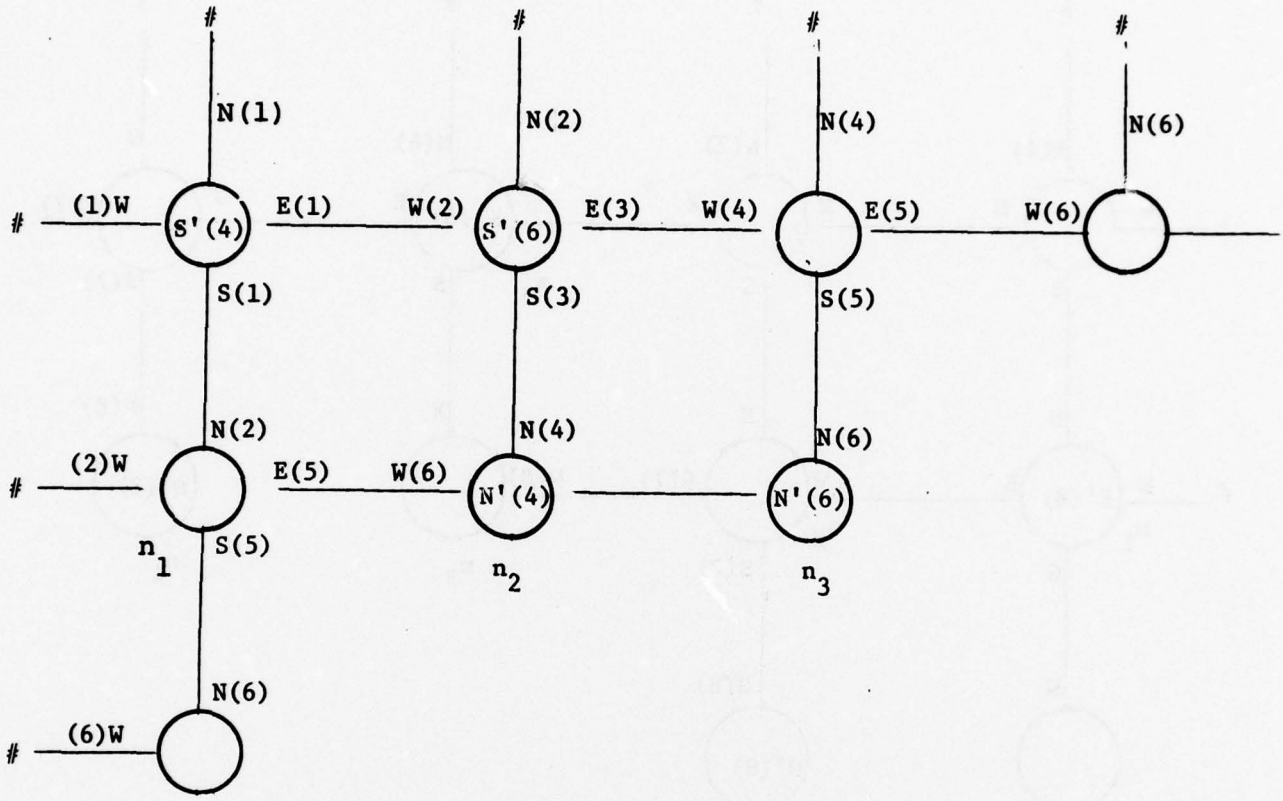

(1)W
S'(4)
E(1)
W(2)
S'(6)
E(3)
W(4)
S'(8)
E(5)
W(6)
–
E(7)
S(1)
S(3)
S(5)
S(7)
N(2)
N(4)
N(6)
node k
N(8)
(2)W
S'(8)
E(5)
W(6)
N'(4)
S'(10)
E(7)
W(8)
N'(6)
S'(12)
N'(8)
E(10)
E(9)
W(10)
S(5)
S(7)
S(9)
S(11)
N(6)
N(8)
N(10)
N(12)
(6)W
S'(12)
E(9)
N'(8)
S'(14)
W(10)
E(11)
W(12)
N'(12)
S'(16)
E(13)
W(14)
N'(12)
E(15)
S(9)
S(11)
S(13)
S(15)
N(10)
N(12)
N(14)
N(16)
(10)W
S'(16)
E(13)
W(14)
N'(12)
S'(18)
E(15)
W(16)
N'(14)
S'(20)
E(17)
W(18)
N'(16)
E(19)
S(13)
S(15)
S(17)
S(19)
N(14)
N(16)
N(18)
N(20)
(14)W
–
E(17)
N'(16)
E(19)
W(20)
N'(18)
E(21)
W(22)
N'(20)
E(23)
S(17)
S(19)
S(21)
S(23)

Figure

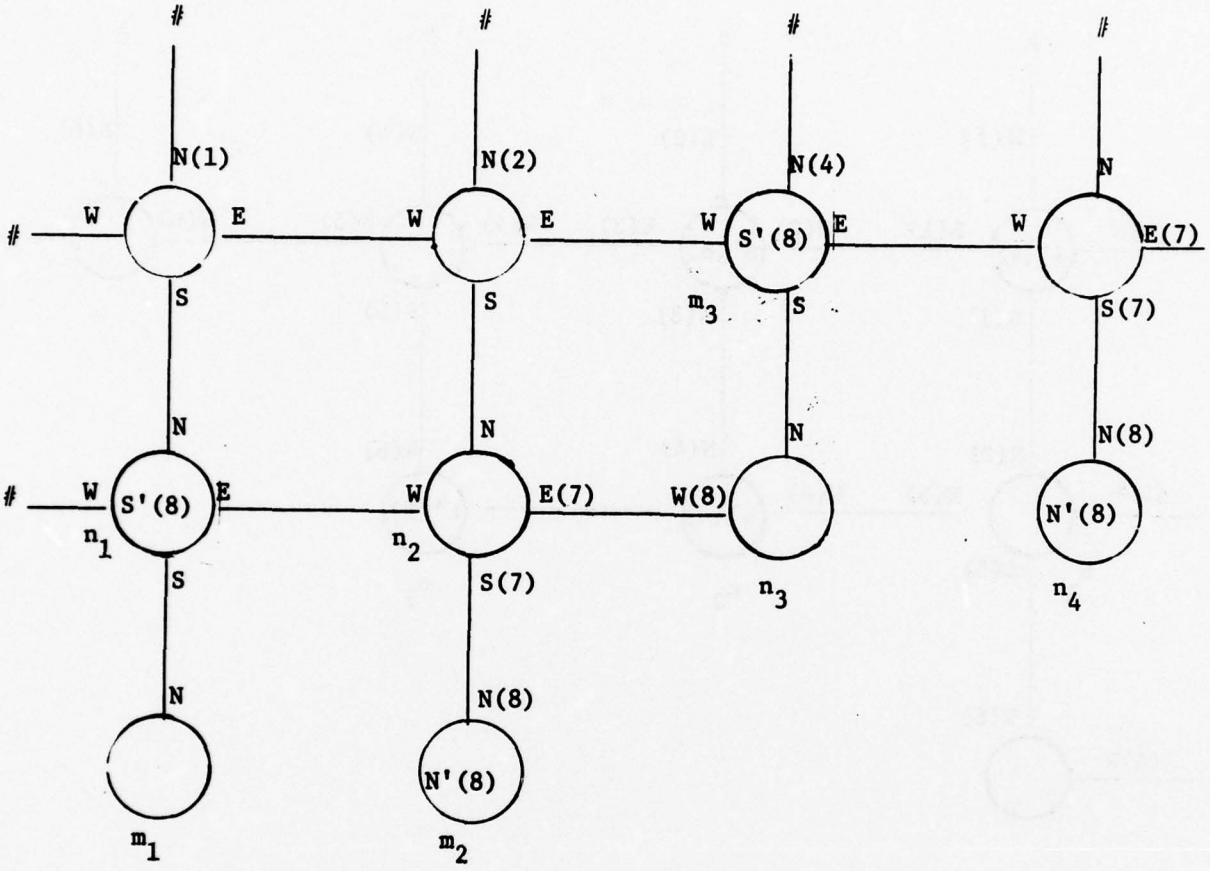
Suppose Γ is a connected 4-graph accepted by M. If acceptance was due to cases 1 and 2 after D' was found, then clearly $U(\Gamma)$ is a rectangular array since it has only one node or it is a string. For the remaining cases, the direction assignment at each node is successful. Let us now look at the process of direction assignment more carefully. After the first 4 steps, the nodes of Γ having some arc ends assigned directions have the following relations:



Since no rejection signal is generated at step 5, nodes n_1 and n_2 must be neighbors. Similarly, n_2 and n_3 must be neighbors since no rejection signal is generated at step 7.


#
N(1)
N(2)
N(4)
N(6)
(1)W
S'(4)
E(1)
W(2)
S'(6)
E(3)
W(4)
E(5)
W(6)
S(1)
S(3)
S(5)
N(2)
N(4)
N(6)
(2)W
E(5)
W(6)
N'(4)
N'(6)
n_1
S(5)
n_2
n_3
N(6)
(6)W

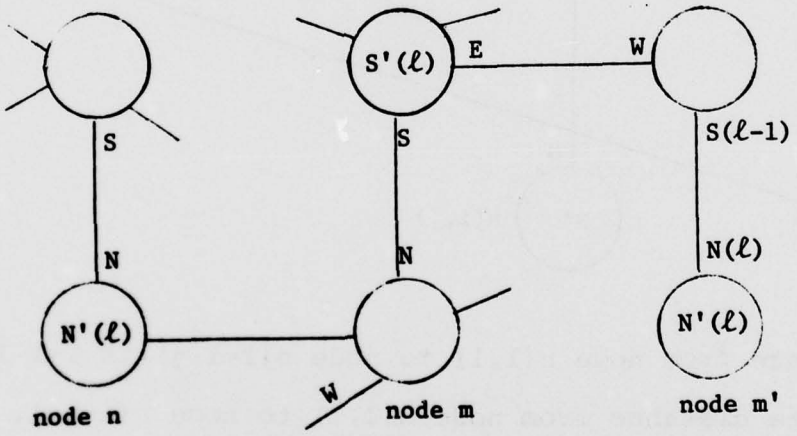
Now in the next two steps,



At step (9), if m_1 and m_2, n_3 and n_4 are neighbors then what we have is a portion of a rectangular array. However, if m_1 and n_4 are neighbors and m_2 and n_3 are neighbors, no rejection signal will be generated even though what we have is not a rectangular array. In fact the direction assignment can be continued and completed successfully as shown in the example of Figure 3. In general, no rejection signal will be generated in the following situations:

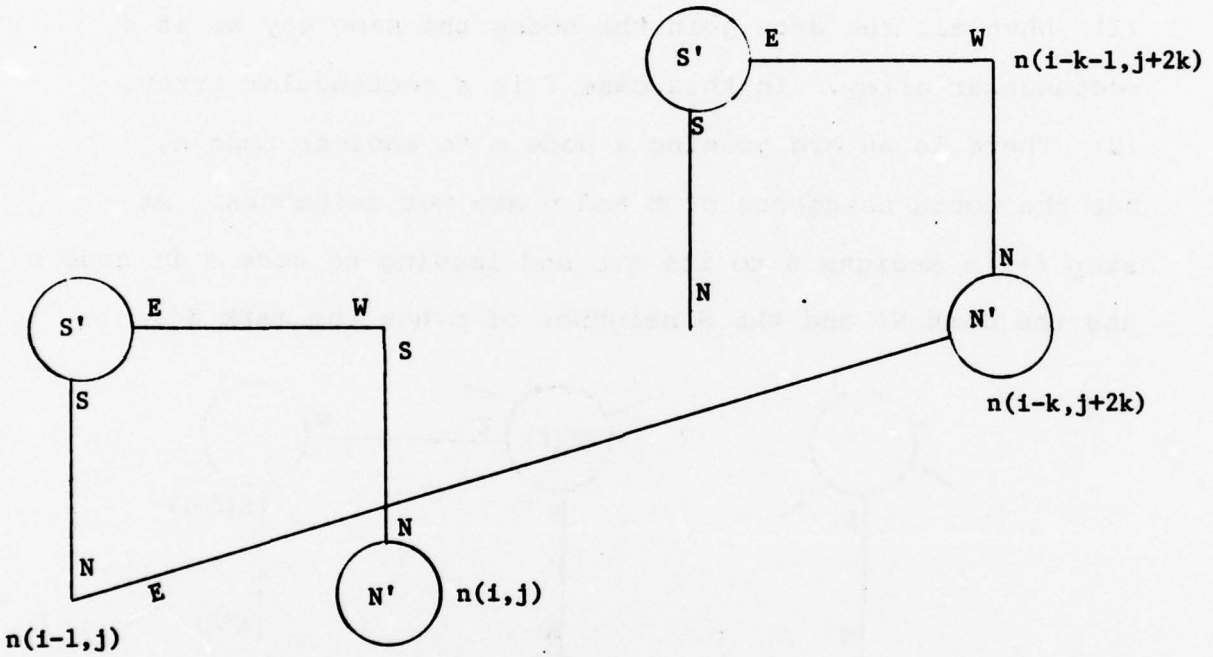
(1) When all the arcs join the nodes the same way as in a rectangular array. In this case Γ is a rectangular array.

(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 step ℓ+1 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. By the same reasoning as in Figure 1, which shows the time step at which each node's direction assignment is completed, 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 $k \neq 0$ and $-i < k < i$. 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(1-1,j) is i-1-1+j-1 = i+j-3. The distance from node n(1,1) to node n(i-k-1, j+2k) is i-k-1-1+j+2k-1 = 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 in 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 M. Therefore the direction assignment conflict cannot happen. Since this is the only discrepency that can occur in the direction assignment phase, the nodes with completed δ's do indeed form a rectangular array.

In the following, we will show that Γ has no other nodes. Assignment Rule 1 assures that no nodes can have partially assigned δ if Γ is accepted by M. Now suppose there are nodes in Γ whose arc ends are not assigned. By the connectedness of Γ, there is an unassigned node n which is the neighbor of an assigned node m. All m's arc ends, including the one leading to n, are assigned. Moreover the arc end leading to n cannot be of direction N or W because they are assigned only after the other end of the arc is of direction S or E. But if the arc end of m leading to n has direction S or E, then n has an arc end with direction N or W. This is a contradiction. Therefore all nodes of Γ have completed δ's.

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. Thus we have

Proposition 3: If a connected 4-graph Γ is accepted by $M = (\Gamma, M, H)$ then $U(\Gamma)$ is a rectangular array.

Proposition 4: There is a cellular 4-graph automaton that recognizes rectangular arrays in diameter time.

Proof: Combining Propositions 3 and 4. //

Corollary: There is a cellular 4-graph automaton that recognizes square arrays in diameter time.

Proof: The cellular 4-graph automaton M in Proposition 4 can be modified slightly to an 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 $d>4$, M can be modified in the obvious way so that M can recognize the d-graphs which are rectangular arrays. Based on the same principle, the result of this section can be generalized to n-dimensional rectangular arrays.

1.2. 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 acceptor $M = (\Gamma, M, H)$ 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 centered at D, then Γ is Eulerian. In Section 1.3.1 of [2], 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, 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.

1.3. 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 acceptor M with a distinguished node D recognizes bipartite graphs by making sure that Γ is bicolorable. D 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 centered at 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 has passed when it receives the message that the tree has been constructed. The time it takes for recognizing a bipartite

graph is twice the radius plus 1 time steps.

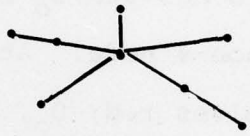
In a d-graph every node has at most d non-# neighbors. The number of complete bipartite d-graphs is very limited; in fact, it is at most $d(d+1)/2$. If Γ is a d-graph such that its underlying graph $U(\Gamma)$ is $K_{a,b}$ then $a \leq d$ and $b \leq d$.

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_o in its state where i is the number of non-# neighbors D has and i_o 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_o, the i_o-th neighbor of D also records the numbers j,i in its state where j is the number of non-# neighbors D_o has, and i is the first of the pair of numbers in D's state. At the third step, i is sent from D_o to all its neighbors, 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.

1.4. Stars

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 acceptor in the last section can easily be modified to have one more requirement for acceptance of stars, namely i=1 or j=1.

An extended star consists of a center node with n strings of nodes emanating from it, instead of n neighbors. For example

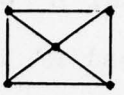


is an extended star. The cellular d-graph acceptor with a distinguished node D described below recognizes extended stars in time proportional to the diameter of the graph. In the first step, if D has ≤ 2 non-# neighbors, D sends out a message T to find a node with more than two non-# neighbors. When a node with exactly two non-# neighbors receives T from one of the non-# neighbors, it passes T to the other non-# neighbor. A node with one non-# neighbor changes T to signal S and transmits S back to node D. If D receives S from all of its non-# neighbors, M accepts the d-graph. If T reaches a node D' with more than two non-# nodes, or if D has more than two non-# neighbors (at the first step), a signal E is transmitted to each of the neighbors of node D'. A node having exactly two non-# neighbors simply passes E along. Whenever

E reaches a node with more than two non-# neighbors, a rejection signal is sent to D to reject the d-graph. Signal E is changed to signal E' when E reaches a node with only one non-# neighbor. Signal E' is sent back to node D. When node D' receives signal E' from all of its non-# neighbors, an acceptance signal is sent to D. If D' receives signal E from any of its neighbors, it sends a rejection signal to D. The d-graph is accepted if the acceptance signal is received by D without first receiving a rejection signal.

D receives S from all of its non-# neighbors only when D has one or two non-# neighbors, and in this case the d-graph is a string which is a degenerate extended star. An extended star has at most one node with more than two neighbors. This condition is checked by signal E. The E signal also makes sure that only strings emanate from D'. Therefore the cellular d-graph acceptor recognizes extended stars. Clearly the time it takes is proportional to the radius of the d-graph.

1.5. Wheels

A wheel is obtained from a star by having the non-center nodes form a cycle. For example  is a wheel. Let $M = (\Gamma, M, H)$ be a cellular d-graph acceptor that works as follows:

Step 1: Each node having three non-# neighbors identifies itself as a B node.

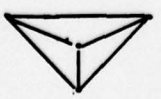
Step 2: A rejection signal is sent from a node n to node D if (i) n is a non-B node having a non-#, non-B neighbor, or (ii) n is a non-B node which is not D and it does not have node D as a neighbor or (iii) n is a B-node with more than one non-B neighbor. A node m is marked C if (iv) m is a non-B node having D (a B-node) as a neighbor, or (v) m is node D and a non-B node, or (vi) m is node D and a B-node with all B neighbors (the B mark of n is erased).

Step 3: B neighbors of C change their marks from B to S.

Step 4: A rejection signal is sent from any B node remaining.

At step 5, the spanning tree construction starts. After 2r steps, when D knows the spanning tree has been constructed and no rejection signal was received, M accepts the d-graph Γ.

Obviously, if Γ is a wheel then M accepts Γ.

If Γ is accepted by M, then no rejection signal is initiated at step 2. Suppose there are no non-B nodes at step 2; then D is a B node and all of its three non-# neighbors are B nodes. Steps 3 and 4 guarantee that there are no other nodes. Therefore Γ is the wheel  .

Suppose there are non-B nodes. All the non-# neighbors of the non-B nodes are B nodes. If D is a non-B node, all its neighbors are B nodes, which implies there is no non-B node having D as a neighbor. If D is a B node then there can only be one non-B node having D as a neighbor. Therefore there is only one node marked with C. Steps 3 and 4 and no rejection signal mean that all the B-nodes are neighbors of the C node, so that Γ is a wheel.

It is clear that the recognition process for wheels takes at most 2r+4 steps, which is proportional to the radius of the d-graph.

2. Some Slower Algorithms

In this section we present some algorithms which take time proportional to at least the area (=the number of nodes) of the graph. The construction of a depth-first spanning tree is a linear time algorithm. Due to the nature of depth-first search, a linear algorithm is optimal. The algorithms to identify cut nodes, bridges and central points are of order diameter * area time. The recognition of planar graphs by cellular d-graph automata is extremely slow, even though there are linear time sequential algorithms to test planarity using a random access computer model.

2.1. Depth First Spanning Tree and Node Ordering

In [2] it is shown that a spanning tree of a d-graph Γ can be constructed by a cellular d-graph automaton in time proportional to the diameter of Γ. The tree constructed corresponds to a breadth-first spanning tree and its importance is seen in some of the applications in [2]. The depth-first spanning tree (DFST) is also very useful in designing algorithms [3 , 4]. However, due to the nature of depth-first search, the nodes need to be explored one at a time so that the construction of a DFST takes time at least equal to the number of nodes in the graph. The DFST gives an ordering of the nodes based on the order in which the nodes are explored. Once the nodes of Γ are ordered, they can be considered as a string of nodes and many algorithms such as firing squad synchronization on strings can be applied to the automata at the nodes.

Consider a cellular d-graph automaton $M = (\Gamma, M, H)$ with a distinguished node D that operates as follows: the states of M have an ancestor component of length 1 and a descendant component which is a set of size $\leq d-1$ whose elements belong to $\{1,2,\ldots,d\}$. The ancestor component of the distinguished node D is considered to be part of the descendant set. These components are initially all empty. A node with an empty ancestor component is said to be unvisited. During the first step, D records the smallest arc end number leading to a non-# node (the least non-# arc end number) in its ancestor component which is considered as

D's descendant. D also creates a signal P. At any time step, one and only one node has one of the signals P or P' (P' will be defined below). In subsequent steps:

(1) Suppose node m has the signal P, i is the largest number in its descendant component, node n is the i-th neighbor of m, and m is the j-th neighbor of n. Thus n must be unvisited (otherwise m cannot have i as the largest number in its descendant component). Then n records j in its ancestor component and takes the signal P. If n has an unvisited non-# neighbor, it puts the least unvisited non-# arc end number in its descendant component; otherwise n changes signal P to P'.

(2) Suppose a node m which is not D has signal P', i is in its ancestor component, and node n is the i-th neighbor of m. If n has unvisited non-# neighbors, then n takes P', changes P' to P and puts the least unvisited non-# arc end number in its descendant component; otherwise it simply takes signal P'.

(3) Suppose node D has signal P'. If D has unvisited non-# neighbors then it changes P' to P and puts the least unvisited non-# arc end number in its descendant component; otherwise D signals the completion of the DFST construction.

Starting with node D, each time signal P reaches a new node, it is passed to an unvisited neighbor of the new node, if any. P' is invoked only when all the non-# neighbors of a node are visited and P' indicates going back to the ancestor of the node. Clearly, this gives a depth first spanning tree

of Γ with the root at D. The ancestor and descendants of each node are stored in the corresponding components. Note that each time a number is added to the descendant component, it is the smallest one qualified. Thus they are added in increasing order so that the largest number in the set is always the last one added. Therefore the numbers in the ancestor and descendant components give an ordering of the nodes corresponding to the order they are visited.

For each node n which is not D, the arc from n's ancestor in the DFST is travelled once by signal P to set n's ancestor component and is travelled once by signal P' when all of its descendants are visited. Therefore the construction of the DFST takes time proportional to the number of nodes of Γ, which we refer to as area(Γ).

Suppose it takes t(Γ) steps to determine whether a specific node (say the distinguished node) has a certain property X. During the construction of the DFST of Γ, when a new node n is reordered, instead of passing signal P to the least unvisited non-# neighbor immediately, n is first checked to see if it has property X and an appropriate mark is made at n. When the last node is checked and the DFST is constructed, all the nodes with property X are identified. The process takes time proportional to t(Γ) times area(Γ).

2.2. Subgraph Isomorphism

It was indicated in [5] that we were not able to find a cellular d-graph automaton that will decide if a d-graph Γ has a subgraph isomorphic to a given labelled graph α in time proportional to the diameter of Γ. In this section, we will exhibit a cellular d-graph automaton that decides the subgraph isomorphism problem in area time.

Let T_α be a spanning tree of α and let the height of T_α be k. For a d-graph Γ with a distinguished node D, suppose we want to know if Γ has a subgraph isomorphic to α such that D corresponds to the root node of T_α; then we can use a method similar to those described in [5]. First the cellular d-graph automaton 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 [5]. 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 M

can proceed to check if the arcs in α-T_α 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. It is easy for D to tell when 4k time steps have passed by the breadth first spanning tree construction (with root D).

As indicated in Section 2.1, 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 or n has fewer non-# neighbors than T_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 the above; if such a subgraph is found a success signal is sent to the distinguished node, 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, we know that Γ does not have a subgraph isomorphic to α.

The above subgraph matching process takes time proportional to $k \cdot \text{area}(\Gamma)$ where k is the height of T_α. Thus finding a spanning tree of α with minimal height is desirable. [It can be seen from the discussion of central points in Section 2.4 that if we choose a central point C as the root and build a breadth first spanning tree as in Section 1.3.1 of [2], the height of such a T_α will be minimal]. The number of states of the cellular d-graph automaton is a function of α and d and is in general large. The result of this section does not contradict the fact that subgraph isomorphism is a NP-complete problem

[6,7] 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.

2.3. Cut Nodes and Blocks

Let Γ be a d-graph and N be a node of Γ. To find if a node N of Γ is a cut node of Γ, a cellular d-graph automaton $M=(\Gamma,M,H)$ partitions the non-# arc ends at N so that two arc ends belong to same class iff the arcs belong to the same block, i.e., there exists a simple cycle containing them. From the definition and properties of cut nodes, N is a cut node iff there is more than one class of non-# arc ends at N [3, 4]. The arcs in each class belong to the same block.

The states of M have a d-tuple $(a_1,a_2\ldots,a_d)$ which is initially all zeros. This d-tuple at node N is called the partition vector and is used to record the partition of the arc ends at N. The d-tuples at the other nodes are used to record arc end equivalences. At the first step, the partition vector is set so that $a_i=i$ if the i-th arc end of N leads to a non-# node, and $a_i=0$ otherwise for $1\leq i\leq d$. Therefore each arc end belongs to a different class. If N has only one non-# neighbor, then N is not a cut node. Otherwise M starts the construction of a breadth-first spanning tree (BFST) rooted at N as in Section 1.3.1 of [2], except that N sends out d signals instead of just one. The i-th neighbor of N gets the signal i which will be propagated from non-# neighbor to non-# neighbor. A node m receives signals i,j from its neighbors iff there are two paths from N to m, one contains the i-th neighbor of N, the other contains the j-th neighbor. Hence the i-th and the j-th arc ends of N belong to the same class of the partition.

Whenever a node m receives signals $i_0,i_1,\ldots,i_k$ from its neighbors where $i_0<i_1<\ldots<i_k$, m chooses a neighbor m' which gives it the smallest signal i_0 as its immediate ancestor in the BFST. Singal i_0 is transmitted to its unmarked neighbors and m sends a classification signal $(i_o,i_1,\ldots,i_k)$ to m' to be transmitted up the BFST to its ancestors. A node m with signal j_0 having neighbors with signals $j_1,\ldots,j_\ell$ such that $j_{i-1}<j_i$ for $1\leq i\leq \ell$, also sends a classification signal $(j_0,j_i,\ldots,j_\ell)$ up to its ancestors. Any node receiving a classification signal passes the signal to its ancestor. A node receiving more than one classification signal merges them and sends the merged classification signal up to its ancestor since they all belong to the same class of the partition. Each classification signal is of length at most d. Hence the d-tuples at the nodes can be used for their transmission. When N receives a classification signal $(i_0,\ldots,i_k)$ and the partition vector of N is $(a_1,\ldots,a_d)$, then $a_{i_0}, a_{i_1},\ldots,a_{i_k}$ are all set to min $\{a_{i_0},a_{i_1},\ldots,a_{i_k}\}$. In case more than one classification signal is received by N, the partition vectors are reset so that if $\{i_0,i_1,\ldots,i_k\}\cap \{j_0,j_1,\ldots,j_\ell\}\neq\emptyset$, then $\min\{a_{i_0},\ldots,a_{i_k},a_{j_0},\ldots,a_{j_\ell}\}$ is used. If $\{i_0,i_1,\ldots,i_k\}\cap\{j_0,j_1,\ldots,j_\ell\}=\emptyset$ then $a_{i_0},a_{i_1},\ldots,a_{i_k}$ are set to their minimum and $a_{j_0},\ldots,a_{j_\ell}$ are set to their own minimum. One step after the BFST is constructed, all the possible classification signals are on their way to node N. Therefore when N receives the message indicating the completion of the spanning tree construction, N knows that the partition of its arc ends is also complete. The i-th and j-th arc ends

of N belong to the same block iff $a_i=a_j\neq 0$ in the partition vector $(a_1,\ldots,a_d)$. The non-zero components of the partition vector have the same value iff N is not a cut node. The time for N to decide if it is a cut node is twice the radius of Γ "centered" at N, which is no more than the diameter of Γ.

Combining the above with the construction of a DFST of Γ, all the cut nodes of Γ can be identified in time proportional to area(Γ) times diameter(Γ). It is easy to tell then if Γ is biconnected (a block, containing no cut points). All the blocks of Γ can easily be identified by the partition vector at each node which tells which arcs at the node belong to the same block. In this case the transmission of the classification signal should not use the d-tuples which are the partition vectors so that their values will not be destroyed. Clearly an extra d-tuple is needed in the states of M.

Note that the identification of the cut nodes is done one node at a time. Attempts to do the identification in parallel lead to possible confusion of signals from different nodes. Tarjan [4] has a sequential algorithm which takes time proportional to the number of edges of a graph, in which a DFST is constructed and the cut nodes ($\neq$ the root) are those nodes n having a son s such that there are no arcs connecting any descendents of s (including s itself) and a proper ancestor of n. However this algorithm is allowed to use integers as large as the size of the graph; thus each node can have a different number for distinction. The state set of the cellular d-graph automaton M is the same regardless how large the

d-graph Γ is. The inability to distinguish the signals originating from different nodes creates problems for parallel identification of cut nodes, just as it did for attempts to do subgraph isomorphism in diameter(Γ) time.

2.4. Bridges

Let Γ be a d-graph and T be a breadth-first spanning tree (BFST) of Γ constructed using the method in Section 1.3.1 of [2]. If an arc (m,n) of Γ is not a tree arc in T, then node m and node n must have a least common ancestor node k such that $k \neq m$ and $k \neq n$. The arcs in the tree path form k to m and those in the tree path from k to n, together with arc (m,n), form a cycle. Hence (m,n) is not a bridge. Therefore a bridge of Γ must be an arc of the BFST.

Let (a,b) be an arc of the BFST of Γ, where a is the ancestor of b in the tree. Suppose the root node sends a signal B down the tree and colors the nodes blue. When signal B reaches node b, it is changed to signal R to color the nodes red so that node b is colored red while node a is colored blue. Note that the other descendents of a are also blue. Signal R is sent only down the subtree with root b. Any time a blue node m has a red neighbor n and $(m,n) \neq (a,b)$ then (a,b) is not a bridge since (a,b) belongs to the cycle containing the arc (m,n) and the paths from the least common ancestor of m,n to nodes m and n.

A cellular d-graph automaton M with a distinguished node D can decide if an arc (p,q) of Γ is a bridge by constructing a BFST of Γ, and if (p,q) is a tree arc, then starting the coloring of nodes as above. This process takes time proportional to the radius of Γ "centered" at D.

To identify all the bridges of Γ, a BFST T of Γ is constructed and then each tree arc is tested. The order of the

arcs of T is induced by the DFST of Γ : when a new node n is added to the DFST, the unique are joining n and its ancestor in the BFST is tested, and the answer is stored in the node. The time it takes to identify all the bridges is proportional to area(Γ) times diameter(Γ). It should be pointed out that not every arc of the BFST needs to be tested since during the BFST construction, if a node n receives the signal (originating from the root) from more than one neighbor, then the arc leading to n's ancestor cannot be a bridge.

Proposition: An arc (m,n) of a d-graph Γ is a bridge iff the subgraph consisting only of the arc (m,n) (with its endpoints) is a block.

Proof (==>): If (m,n) is a bridge then (m,n) does not belong to any cycle; therefore (m,n) by itself is a block.

(<==): If (m,n) is not a bridge then (m,n) is part of a cycle. Thus (m,n) belongs to the same block as the other arcs of the cycle. Therefore {(m,n)} is not a block. //

This proposition shows that if the cut nodes or blocks of Γ are identified as in the previous section, then the bridges of Γ can be identified in one step since they are just the blocks consisting of only one arc. The singleton blocks are readily identified from the partition vectors at each node; each of its arc ends is in a class of the partition by itself.

2.5. Central Points

Given a d-graph Γ, for any non-# node n of Γ, $r(n)$ = the radius of Γ with "center" n is the maximum of the distances between n and any non-# node of Γ. C is a central point of Γ if $r(C) = \min\{r(n) \mid n \text{ is a node in } \Gamma\}$. Note that there may be more than one central point in a d-graph. Many of the algorithms we have presented take time proportional to the radius of Γ with "center" the distinguished node. These algorithms are more efficient if the distinguished node is located at a central point. A central point is interesting also because it is the best location for a communication or emergency center in a network. In this section we exhibit a cellular d-graph automaton M that can identify a central point in time proportional to area(Γ) times diameter(Γ) by finding $r(n)$ for each node n of Γ and then comparing them. The following proposition shows that a longest branch of a depth-first spanning tree is long enough to store $r(n)$ in unary for each node n.

Proposition: Let Γ be a d-graph and let ℓ be the length of a longest branch of a depth-first spanning tree of Γ; then $\ell \geq$ the distance between any two nodes of Γ.

Proof: Suppose there exist nodes m and n such that the distance between them is $>\ell$. Without loss of generality we can assume that n is visited before n in the construction of the DFST. Then m must be a descendent of n in the DFST. The branch containing both n and m has length $\geq$(distance between n and m)$>\ell$. This contradicts the assumption that ℓ is the length of a longest branch of the DFST. Therefore the distance between any two

nodes of Γ is $\leq \ell$. //

Corollary: $\ell \geq r(n)$ for any node n of Γ.

Proof: $r(n)$ =(maximum of the distance between n and any non-# node of Γ) $\leq \max\{\ell\} = \ell$. //

On any d-graph Γ, $M=(\Gamma,M,H)$ with a distinguished node D operates in the following 5 stages:

(1) M constructs the DFST of Γ with D as its root. A longest branch of the DFST is identified in the same way that a longest branch of the BFST is identified in Section 1.3.2 of [2]. This longest branch of the DFST will be used as a counter to store r(n)'s. Each node of the counter has two channels so that two numbers can be stored in it simultaneously.

(2) M constructs the BFST of Γ with D as its root. This tree gives the shortest paths from any node of Γ to D. r(D) is found and stored in the counter in unary using a method similar to that in Section 1.3.2 of [2] and described in more detail in 4(b) below.

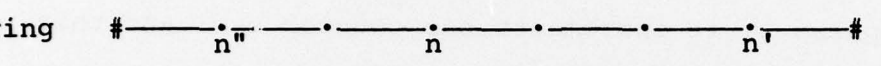
(3) At each node n, the state has a component p(n) which is a number in $\{0,1,\ldots,d\}$. Let Q(n) denote the p(n)-th neighbor of node n if $p(n)\neq 0$. D, $n_1=Q(D)$, $n_2=Q(n_1),\ldots,$ $N=Q(n_h)$, where $p(D), p(n_1),\ldots,p(n_h)$ are non-zero and p(N) is zero, denotes a path from node D to N when node N has the property that the radius of Γ with center at N is stored in the counter defined in (1). Therefore, after r(D) is stored in the counter, p(D) is set to zero.

(4) M visits each node of Γ according to the order induced by the DFST. When M visits node n, it (a) identifies n', the

node farthest away from n; and (b) stores r(n) in unary in the counter by generating a pulse u from node n every other step after a signal S is sent from n to n' until a signal S' reaches n. S' is generated by node n' when signal S reaches node n'. Both S and S' are sent at unit speed. Each pulse u is passed on to node D, the first node of the counter. When D gets the pulse u, it increments the counter by 1 and this pulse u is dissolved. D knows no more pulses u will come from node n if it does not receive any pulse in two consecutive steps. (c) The new number is discarded if it is not smaller than the old number stored in the counter. (d) If the new number is smaller then the old number is discarded and the new number is stored in the old number's channel. Moreover the shortest path from D to n (which is given by the BFST rooted at D from (2)) is recorded in the state as p(m)'s for all the m's in the path and p(n) is set to zero.

(5) When M finishes visiting every non-# node of Γ so that (4) above is done for each node, the number in the counter gives the radius of Γ centered at a central point. The center can easily be identified by following the path specified by p(D) and the p(n)'s.

Let t(i) denote the time for M to perform the task in stage i above, $1 \le i \le 5$. Then $t(1) = C_1 \cdot \text{area}(\Gamma)$ for some constant C_1, $t(2) = C_2 \cdot r(D)$, for some constant C_2, $t(3)=1$, $t(5)=r(D)$, and t(4)=sum of s(n) for each n in Γ where $s(n) = b_1 \cdot r(n) + b_2 \cdot r(D)$, for some constants b_1, b_2. Since $r(n) \le \text{diameter}(\Gamma)$ for every node n in Γ, the time it takes M to identify a central point is proportional to area(Γ) times diameter(Γ).

M can be modified by taking the following into consideration. A node n with only one non-# neighbor m is not a central point when there are other nodes present since the distance between any node $k \neq n$ and node m is 1 less than that between k and n. If n' is a node farthest away from n, n is not necessarily a node farthest away from n'. For example, in the string 

, n" is the node farthest away from n'. However $r(n') \geq r(n)$ because r(n')=maximum of the distance between n' and any non-# node of Γ $\geq$distance between n' and n = r(n) since n' is a node farthest away from n. Therefore n' can be eliminated from consideration of being a central point when we are only interested in finding a central point rather than all central points. The modified cellular d-graph automaton M' will be faster than M, even though the speed is still on the order of area(Γ)$\cdot$diameter(Γ) for many d-graphs.

2.6. 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 [8] 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 embedability of a graph in the plane difficult.

There are three other criteria for planarity [9]. The earliest characterization was given by Kuratowski [10]. 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 our results on subgraph isomorphism in Section 2.2, one may expect to be able to find an efficient cellular d-graph automaton using Kuratowski's criterion to test planarity. However, the length of the sequence of contractions depends on the graph given.

Whitney [11, 12] expressed planarity in terms of the existence of a dual graph, which is difficult for a cellular d-graph automaton to verify. MacLane [13] 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 Z_0 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.

On 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 5 nodes, and testing if there are ten non-intersecting 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 automaton algorithm to recognize planar graphs. One reason is that planarity is a very global property which is difficult for a cellular d-graph automaton to discover.

References

1. A. Wu, Cellular graph acceptors, TR-599, Computer Science Center, University of Maryland, College Park, MD, November 1977.
2. A. Wu, Cellular graph acceptors, 2, TR-621, Computer Science Center, University of Maryland, College Park, MD, December 1977.
3. A.V. Aho, J.E. Hopcroft, and J.D. Ullman, The Design and Analysis of Computer Algorithms, Addition-Wesley, Reading, Massachusetts, 1974.
4. R.E. Tarjan, Depth first search and linear graph algorithms, SIAM J. Computing 1, 1972, 146-160.
5. A. Wu, Cellular graph acceptors, 3, TR-648, Computer Science Center, University of Maryland, College Park, MD, April 1978.
6. S.A. Cook, The complexity of theorem proving procedures, Proc. 3rd STOC, 1971, 151-158.
7. R.M. Karp, Reducibility among combinatorial problems, in Miller and Thatcher, eds., Complexity of Computer Computations, Plenum Press, NY, 1972, 85-104.
8. J. Hopcroft and R. Tarjan, Efficient Planarity Testing, JACM 21, 1974, 549-568.
9. F. Harary, Graph Theory, Addison-Wesley, Reading, Massachusetts, 1969, 102-116.

10. K. Kuratowski, Sur le problème des courbes gauches en topologie, Fund. Math 15, 1930, 271-283.

11. H. Whitney, Non-separable and planar graphs, Trans. Amer. Math. Soc. 34, 1932, 339-362.

12. H. Whitney, Planar graphs. Fund. Math 21, 1933, 73-84.

13. S. MacLane, A structural characterization of planar combinatorial graphs, Duke Math. J. 3, 1937, 340-372.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 78-1363	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CELLULAR GRAPH ACCEPTORS, 4.		5. TYPE OF REPORT & PERIOD COVERED Interim rept.
		6. PERFORMING ORG. REPORT NUMBER CSC-TR-667
7. AUTHOR(s) Angela Wu		8. CONTRACT OR GRANT NUMBER(s) AFOSR-77-3271
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Maryland Computer Science Center College Park, Maryland 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332		12. REPORT DATE June 1978 13. NUMBER OF PAGES 50
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Formal languages
Cellular automata
Graph automata
Graph property recognition

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Diameter-time algorithms are presented for recognition of rectangular and square arrays, Eulerian graphs, bipartite and complete bipartite graphs, stars, and wheels by cellular d-graph acceptors. Slower algorithms are given for construction of a depth-first spanning tree (area time) and for identification of cut nodes, borders, and central points (diameter·area time). The recognition of planarity is also discussed.

403 018

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)