ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-4, 781 – 788

A NEW ALGORITHM FOR RECOGNITION OF A SPHEROID

A Kazmierczak

Northwest Arkansas Community College, One College Drive, Bentonville, AR 72712, ***akazmierczak@nwacc.edu***

Abstract

A sphere mesh, hereafter called simply a spheroid, is a rectangular array of processors with the rows wrapped around to form a ring, and the columns extended and meeting at a node at both the top and bottom. The appearance is that of a ball. The top node is the north pole and the bottom node is the south pole. A new algorithm is presented to recognize a spheroid. A program was written to determine if the new spheroid recognition algorithm is valid. The program was written in C and tested using a representation of a network's nodes and their neighbors. The program tests several spheroids of different dimensions and different spheroid-like structures. The program found the spheroids and rejected the non-spheroids on the test data. The algorithm presented here is fairly simple and straightforward. However, its true importance lies in the fact that it forms the foundation for algorithms for recognition of more complex mesh like structures, such as, torus, and 3-dimensional mesh.

Keywords*: sphere-mesh, spheroid, algorithm*

----------***----------

1. INTRODUCTION

K.V.S. Ramarao [1] presented the problem of recognizing networks. Networks recognized were bipartite graphs, complete graphs, rings, stars, or trees. This problem was explored for two reason: because a considerable number of different algorithms were being designed for use on networks, and because it was expensive to create and use many different algorithms for these networks. It was worth the investment of time and money to characterize the network structure before using an algorithm, because the cost of an algorithm for an individual network is less expensive than algorithms for general networks. The cost to determine the type of network plus the cost of an individual algorithm is less than the cost of a general algorithm [1].

Recognition algorithms were later developed to find outerplanar graphs [2], strict 2-threshold graphs [3], graphs with threshold of dimension-of-two [4], bipartite-permutation graphs [5], tree connected meshes [6] and mesh networks .

The new spheroid recognition algorithm recognizes spheroids networks just as the algorithm in [7], but uses a different technique. The algorithm presented here proceeds around the ring rows of the spheroid using a coloring scheme and assigning coordinates to the nodes.

2. RELATED WORKS

Much of the work related to graph recognition comes from the graph isomorphism problem. "The graph isomorphism problem can be easily stated: check to see if two graphs that look differently are actually the same. Many sub-disciplines of mathematics, such as topology theory and group theory, can be brought to bear on the problem, and yet only for special classes of graphs have polynomial time algorithms been discovered" [8]. The problem of spheroid recognition is a restricted case of the more general graph isomorphism problem.

A spheroid is defined as a rectangular array of nodes with the rows wrapped around to form a ring and the columns extended and meeting at a node at both the top and bottom. The appearance is that of a ball. The top node is the north pole and the bottom node is the south pole.. Any structure that appears like a spheroid but is not a spheroid is referred to as a spheroid-like structure. A spheroid-like structure is a structure that is not graph isomorphic to a spheroid. One of the first works in this area is due to Hopcroft and Wong [9] who provided the first linear time algorithm for isomorphism of planar graphs. This lead to a great many other studies on graph isomorphism and the complexity of graph isomorphism from a variety of approaches.. Many works focused on algorithms for graph isomorphism, such as Faizullin&Prolubnikov [10], Kikina&Faizullin [11], Faizullin&Prolubnikov [12] Schmidt &Druffel [13] Gazit&Reif [14] and Wolter, Woo &Volz [17]. Other researchers have addressed the complexity of graph isomorphism, Toda [15], Boucher &Loker [16], Toran [18] and Golubchik [20, 21]. Many works on practically all aspects of graph isomorphism can be found in Bengoetxea [29] and the references therein.

3. STATEMENT OF THE PROBLEM

The statement of the problem can be stated as: given a spheroid looking structure, devise an algorithm to test whether the structure is graph isomorphic to a spheroid.

ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-4, 781 – 788

Necessary Conditions. – All interior spheroid nodes must have four neighbors (degree of 4). Let D4 denote the number of nodes with four neighbors and Dndenote the number nodes of degree n. Let N denote the number of nodes in this network and let m and n denote the number of rows and columns, respectively. The m rows are wrapped around to form the ring.D4, and N are determined first. Dn is obviously 2. Then m, which must be a positive integer, is calculated as:

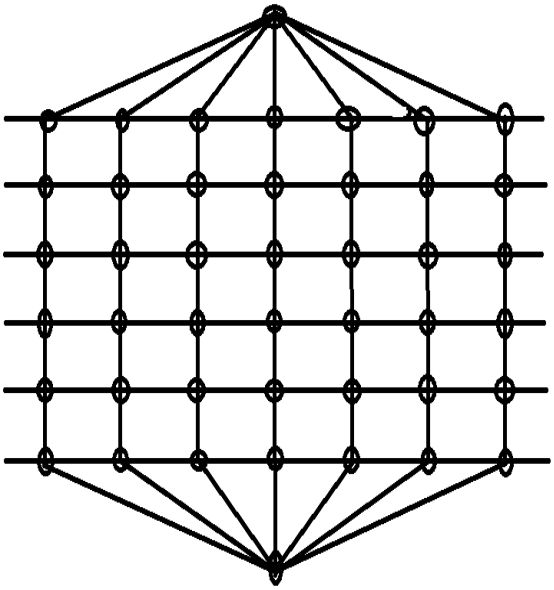

Fig 1 – Spheroid

$$N = m * n + 2$$

$$m = (N - 2) / n$$

$$D_n = 2$$

If the following equations hold, the structure might be a spheroid:

$$D_4 = N - D_n$$

$$D_4 = N - 2$$

$$D_4 = m * n + 2 - 2$$

$$D_4 = (m * n)$$

$$m = D_4 / n$$

Fig. 1 shows a structure that meets these necessary conditions and is a spheroid. By studying the graph, it can be observed that the necessary conditions hold.

Fig. 2 shows a structure that meets the necessary conditions, but is not a spheroid.

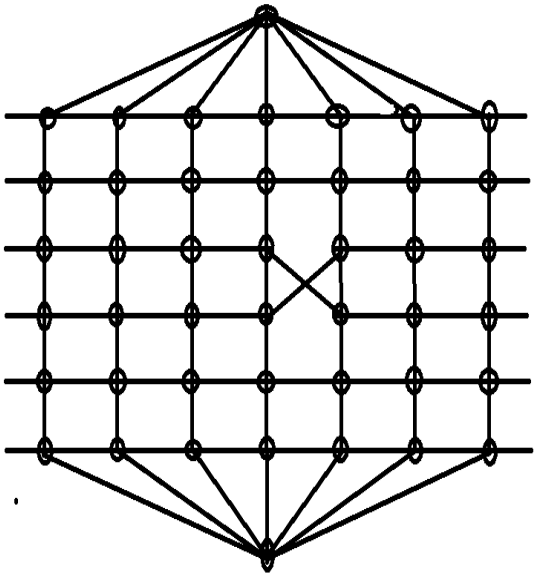

Fig 2 – Spheroid Like

Sufficient Condition. - An (m , n) spheroid-looking structure is a spheroid if for every two adjacent nodes the following equation is true.

$$|r_1 - r_2| + |c_1 - c_2| = 1$$

r1and c1 are the coordinates of one node and r2and c2are the coordinates of the adjacent node [7].

A spheroid can now be defined as any structure that meets the necessary and sufficient conditions.

4. INFORMAL DESCRIPTION OF THE AGORITHM

The algorithm consists of two phases. In the first phase, the necessary conditions for a network to be a spheroid are verified. If these conditions are met, the second phase peels the spheroid row by row, using a coloring scheme as a guide. During this

ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-4, 781 – 788

phase, nodes are assigned coordinates. A successful assignment of coordinates, which satisfies the sufficient condition, ensures the recognition of a spheroid. The algorithm rejects any other structure.

Phase I. During phase I of the algorithm, the following tasks are performed:

- Compute values for D_4, *N, m, and n.*
- Check necessary conditions
- Select a pole to initiate phase II
- Color all nodes white
- Set row = 0, column = 0

This phase determines the values of the network parameters,, D4, and N. The network parameters are initialized to zero before starting. As each node is encountered, the appropriate parameter is updated. After the last node is reached, the network parameters are used to verify the necessary conditions. If any condition is violated, the algorithm terminates indicating the network is not a spheroid. Otherwise, a pole is selected to initiate the second phase of the algorithm. Thus, at the conclusion of phase I, necessary conditions have been verified and the structure is spheroid-looking.

Phase II. Phase II of the algorithm assigns coordinates to the nodes and determines if the network is a spheroid. There are many stages, each processing one row of the spheroid. Three algorithms are needed. The first and last rows have boundary conditions that must be tested, so separate algorithms are necessary. All internal rows are processed the same way, so one general algorithm is needed. A coloring scheme is used to guide the processing along the row. As the nodes are colored, the colors of adjacent nodes must satisfy specific coloring requirements for the network to be a spheroid. These requirements are also verified. The network is rejected as not being a spheroid as soon as any requirement is violated. As a final test, the sufficient condition is verified using the coordinates assigned to each node. If the sufficient conditions are verified, the structure is a spheroid.

4.1 Processing the First Row: Algorithm Firstrow

The start node is one of the poles. Processing this node is trivial. Color the pole black, and assign coordinates (0, 0) Select any neighbor node. This node is a node with four neighbors. Reference Figs 3. This node, called S1(a1)., is then colored black. The row and column coordinates are set to one so the coordinates of S1(a1). are (1, 1). The subscripts are used to indicate row and column. The subscript for S is the row number and the subscript for a is the column number of the current node being processed.

Caution is called for here to make sure that processing proceeds along a row of the spheroid and not up one of the columns. We describe briefly how this is performed.

The start node for the row colors one of its neighbors green. This green node sets its white neighbors to yellow. The start node colors one of the other two nodes to green. If one of this nodes' neighbors is already yellow, then process the start node and proceed to the white neighbor of the start node. If, when setting the second node to green, the greens neighbors are both white, then we would be processing columns instead of rows and that will fail.

The white neighbor of S1(a1). is the next node to be processed. This is S1(a2)

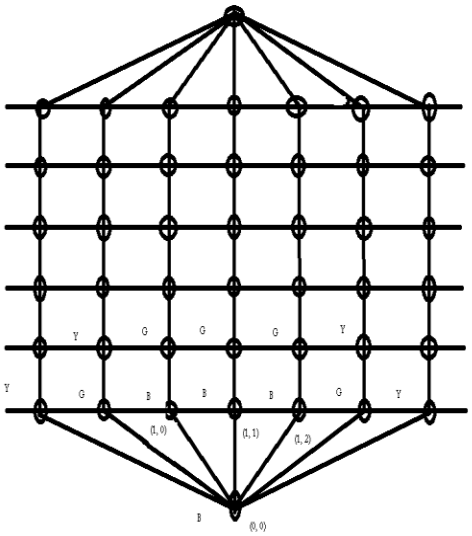

Fig 3 – Processing Middle Node on First Row

Find $S_1(a_1)$

Color $S_1(a_1)$ black

Set row= 1, column = 1

Assign coordinates (row, column)

Change process to $S_1(a_1)$'s white neighbor node and name it $S_1(a_2)$

Node $S_1(a_2)$ has neighbors that are colored black, black, yellow, and white. If this coloring requirement is not true, this is not a spheroid and processing is stopped. Otherwise, node $S_1(a_2)$ is

colored black. The column coordinate is incremented by one and the row coordinate remains zero and the node is assigned coordinates (row, column). The yellow neighbor is colored green. The white neighbors of the new green node are changed to yellow. The white neighbor of $S_1(a_2)$ is the next node to be processed. This node becomes $S_1(a_3)$.

If $S_1(a_i)$'s neighbor nodes are black, black, yellow and white

Do while $S_1(a_i)$'s neighbor nodes are black, black, yellow, and white

Color $S_1(a_i)$ black

Increment column by one

Assign coordinates (row, column)

Color $S_1(a_i)$'s yellow neighbor node to green

Color new green nodes white neighbors to yellow

Change process to $S_1(a_i)$'s white neighbor node and name it $S_1(a_{i+1})$

Else terminate with failure

The processing performed at node S1(ai+1) is exactly the same as the processing performed at node S1(ai). This same processing continues until the next to last node on the row S1(an-1) is encountered. Its neighbors colors are black, black, greenandyellow. Node S1(an-1)is processed the similar to S-1(ai)except for the assignment of the next node to be processed. The next node is the black 4-degree node. Proceed to the green node, which is S1(an).

At S1(an), the neighbors are colored black, black, green and yellow. Color the yellow node green and color the whit neighbors of this green node yellow. Increment the column coordinate and assign the coordinates (row, column. Color S1(an) black

The first row has now been successfully processed. To process the next row, return to node S1(a1), then to the green neighbor of S1(a1). This is S2(a1), the start node of the second row.

If $S_1(a_n)$'s neighbors are black, black, black and green

Color $S_1(a_n)$ black

Increment column by one

Assign coordinates (row, column)

Color $S_1(a_n)$'s yellow neighbor node to green

Color new green nodes white neighbors to yellow

Change processing to $S_1(a_1)$

Change process to green neighbor node of $S_1(a_1)$and call it $S_2(a_1)$

Else terminate with failure

4.2 Processing a Middle Row: Algorithm Middlerow

This section describes the processing for an arbitrary row i, 1 <i< n.

The start node Si(a1) has neighbors colored black, green, green, andyellow. Reference Figs 4. The color of Si(a1)is changed to black. The row coordinate is incremented by one and the column coordinate is set to zero. One of the neighbors of Si(a1) that is green is the next node to be processed. To choose the correct green node to process next, the black node queries both green nodes for the coordinates of its black neighbor. When the green nodes return this information, the black node chooses the green node that returned the lower column coordinate of its black neighbor. This is Si(a2). The neighbor of Si(a1) that is yellow is colored green. The white neighbors of this green node are changed to yellow. Processing proceeds to node Si(a2).

Node Si(a2) has neighbors that are black, black, green, andyellow. Node Si(a2) is then colored black, the column coordinate is incremented by one, the node is assigned the coordinates (row, column). The green neighbor of Si(a2) will be the next node processed. The yellow neighbor of node Si(a2) is colored green. The white neighbors of this green node are colored yellow. Processing proceeds to node Si(a3).

The processing performed at node Si(a3) is the same as performed at node Si(a2). Processing continues until node Si(an-1) is encountered that has neighbors colored black, black, black, and green. Node Si(an-1) is the next to last node on this row.
Node Si(an-1) has neighbors colored black, black, green and yellow. At node Si(an-1), increment the column coordinate, assign the coordinates (row, column) and color the node black. Processing proceeds to the green node, Si(an). This node has neighbors colored black, black, black and green. Si(an) increments the column coordinate, assigns itself the coordinates

ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-4, 781 – 788

(row, column), and colors itself black. Processing proceeds to the black neighbor across the link not yet traversed. This brings processing to node Si(a1), the start node of the row.

This completes the processing for this row. The green neighbor of Si(a1) is the first node of the next row. This becomes Si+1(a1). Processing proceeds to Si+1(a1).

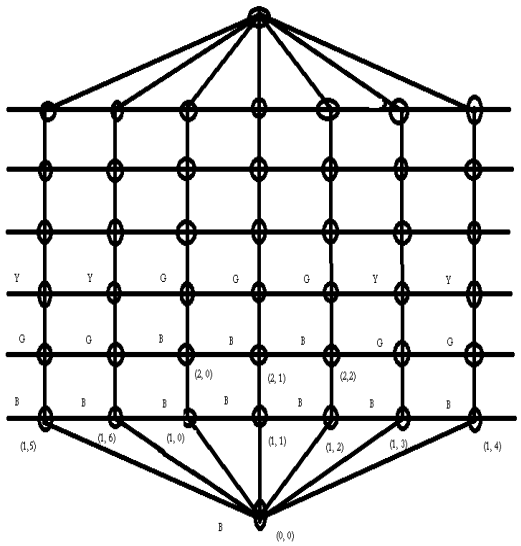

Fig 4 – Processing Middle Node on Middle Row

If $S_i(a_1)$'s neighbor nodes are black, black, yellow and green

Do while $S_i(a_1)$'s neighbor nodes are black, black, green, and yellow

Middle Rows

Beginning of Middle Row

Set $S_i(a_1)$'s to black

Increment row by one

Set column to zero

Assign coordinates (row, column)

Set next process node to $S_i(a_1)$'s green neighbor node (the green node whose black neighbor has the smaller column coordinate)

Color $S_i(a_1)$'s yellow neighbor node to green

Color the new green nodes white neighbors to yellow

Change process to node $S_i(a_2)$

Middle node of Middle Row

Do while $S_i(a_i)$'s neighbor nodes are black, black, green, and yellow

Set $S_i(a_i)$ to black

Increment column by one

Assign coordinates (row, column)

Set next process node to $S_i(a_i)$'s green neighbor node

Color $S_i(a_i)$'s yellow neighbor node to green

Color the new green nodes white neighbors to yellow

Change process to next node $_{Si}(a_{i+1})$

Last of Middle Row

If $S_i(a_n)$neighbor nodes are black, black, black, and yellow

Set $S_i(a_n)$ to black

Increment column by one

Assign coordinates (row, column)

Set next process node to $S_i(a_n)$'s green neighbor node named $S_{i+1}(a_1)$

Color $S_i(a_n)$'s yellow neighbor to green

Color the new green nodes white neighbors to yellow

Change process to node $S_{i+1}(a_1)$

Else terminate with failure

4.3 Processing the Last Row: Algorithm Lastrow

The start node for the last row, node Sn(a1) has neighbors colored black, black, green, and green. Reference Figs. 5. Node Sn(a1) is colored black. The row coordinate is incremented by one, the column coordinate is set to zero and the node is assigned coordinates (row, column). Caution is called for here to make sure that processing proceeds along a row of the spheroid and not up one of the columns. We describe briefly how this is performed.

The start node for the row colors one of its neighbors green. This green node sets its white neighbors to yellow. The start node colors one of the other two nodes to green. If one of this nodes' neighbors is already yellow, then process the start node and proceed to the white neighbor of the start node. If, when setting the second node to green, the greens neighbors are both white, then we would be processing columns instead of rows and that will fail.

One of the green neighbor will be the next node on this row. This becomes node Sn(a2), and processing proceeds to node Sn(a2). To choose the correct green node to process next, the black node queries both green nodes for the coordinates of its black neighbor. When the green nodes return this information, the black node chooses the green node that returned the lower column coordinate of its black neighbor.

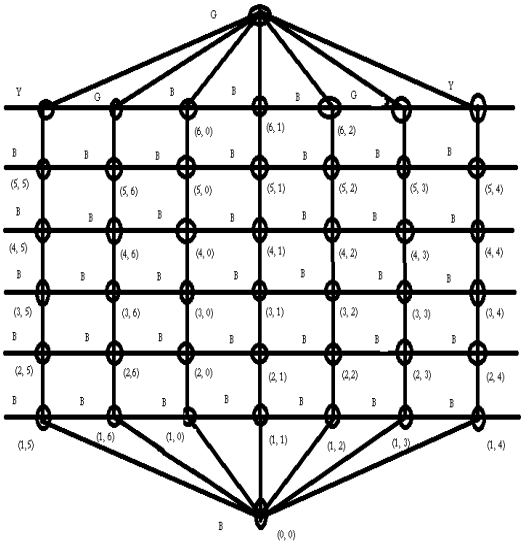

Fig 5 – Processing a Middle Node on Last Row

If $S_n(a_1)$'s neighbor nodes are black, black and green, green

Set $S_n(a_1)$ to black

Increment row by one

Set column to zero

Assign coordinates (row, column)

Set next process node to $S_n(a_1)$'s green neighbor node (the green node whose black neighbor has the smaller column coordinate)

Change process to node $S_n(a_2)$

Else terminate with failure

Node $S_n(a_2)$ has neighbors colored black, black, *black* andgreen. Node $S_n(a_2)$ is colored black, the column coordinate is incremented by one and the node is assigned coordinates (row, column). The neighbor colored green will be the next node on the row. This becomes node $S_n(a_3)$. Processing proceeds to this node $S_n(a_3)$.

If $S_n(a_i)$'s neighbor nodes are black, black, black and green

While $S_n(a_i)$'s neighbor nodes are black, black, black and green

Set $S_n(a_i)$ to black

Increment column by one

Assign coordinates (row, column)

Set next process node to $S_n(a_i)$'s green neighbor node

Change process to node $S_n(a_{i+1})$

Else terminate with failure

Node $S_n(a_2)$ has neighbors colored black, black, blackandgreen. The processing performed at node $S_n(a_2)$ is the same as performed at node $S_n(a_1)$. This same processing continues until some node $S_n(a_n)$ is encountered that has neighbors colored black, black, black and green.

Node Sn(an-1) is the next to last node on the row. Node Sn(an-1)is colored black. The column coordinate is incremented and the node assigned coordinates (row, column).The next node to be processed is the green node of degree 4, which is Sn(an).Processing proceeds to Sn(an). The column coordinate is incremented, the coordinates (row, column) are assigned and the node is colored black. Processing proceeds to the black neighbor across the link not previously traversed. This is node Sn(a1), the start node of the row.

Only one node is left, the pole. Processing proceeds to the green pole. Row and column coordinates are incremented and the node is assigned the coordinates (row, column) and is colored black

At this point spheroid, has been successfully processed. Processing can return to the Opposite pole where processing began. The algorithm tests the sufficient condition. If the sufficient condition does not hold, then there is not a spheroid. If

the sufficient condition holds, then the algorithm will terminate with the successful recognition of a spheroid.

If $S_n(a_n)$'s neighbor nodes are black, black and black

Set $S_n(a_n)$ to black

Increment column by one, and row by one

Assign coordinates (row, column)

If coordinates match sufficient condition

Terminate with success

Else terminate with failure

Else terminate with failure

4.4 Special Mesh Structures

Degenerate Mesh. - A degenerate spheroid is defined as a spheroid with only one rows, that is, m = 1. It is distinguished by having only neighbors; there are four-degree nodes or n degree nodes. The changes to the algorithms are described in very general terms.

This spheroid posed a problem in determining the proper direction to go to be sure of going around the row. We describe briefly how this is performed.

The start node colors one of its neighbors green. This green node sets its white neighbors to yellow. The start node colors one of the other two nodes to green. If one of this nodes' neighbors is already yellow, then process the start node and proceed to the white neighbor of the start node. If, when setting the second node to green, the greens neighbors are both white, then we would be processing columns instead of rows and that will fail.

Once the correct direction is determined, after the pole node is processed, processing proceeds to the first row. Thus, recognition of a degenerate spheroid requires one application of algorithm FirstRow.

Special Mesh. - A special spheroid is defined as a spheroid with only three columns. The only change to the description in the previous sections is that node anis encountered immediately after node a1; that is, there is no node a2 between a1 and an. All other processing is the same.

5. COMPLEXITY

In this section we provide an argument for the complexity of the spheroid recognition algorithm.

Theorem: The algorithm to recognize a spheroid runs in O(N).
Argument: For the argument, we will count the number of operations performed at a 'typical' node. As a 'typical' node, we select any 4-degree node in the center of the mesh. The operations we count are: 'move' processing from one node to the next, 'find' a neighbor of a node, 'color' the neighbor of a node, and 'check' the color of a neighbor of a node as these are the operations actually performed.

Arbitrarily select a 4-degree node somewhere in the middle of the mesh (the node with coordinates 1,1 will do). Start by 'moving' (1 operation) processing to the green neighbor. From here, the green node 'finds' its four neighbors (4 operations) and 'checks their color (4 operations) for the color pattern. The green node 'colors' itself black (1 operation). The algorithm 'finds' the yellow neighbor (1 operation) and 'colors it green (1 operation). This green node must 'find' its white neighbor from its other three neighbors (3 operations) and 'colors' the node yellow (1 operation). Processing 'moves' back to the newly colored black node (1 operation). Processing is now done for this arbitrary 4-degree node. The total number of operations performed at this node is found by counting the number of operations performed. This total is 16 operations.

Since we chose a 4-degree node arbitrarily, the total number of operations performed to process all nodes is bounded above by 16N, where N is the number of nodes in the mesh. THIS IS AN UPPER BOUND BECAUSE 3-DEGREE NODES WILL NEED FEWER THAN 16 OPERATIONS. Therefore, the complexity is 16N, using big-O notation, this is O(N).

6. TEST DATA

Data was selected to test the program. The program was tested on spheroids with dimensions 10x10, 25x25, 50x50 and 100x100. A 2x3 spheroid was tested but a 3x2 mesh was not tested as it is a mirror image of a 2x3 spheroid. The program successfully recognized these as spheroids. Further testing is ongoing.

Data was selected to test spheroid-like structures. The test data was selected to fail necessary conditions. Some conditions could not be tested because an earlier test failed. Example: a node was deleted at the side. The test for number of nodes of degree 4 was never performed because the test for number of nodes of degree 2, which was performed earlier, failed the necessary conditions. All spheroid-like structures were successfully detected and rejected.

Data was also selected to test the coloring technique. The test consisted of a cross-over, as depicted in Fig. 2, at a corner, on a side and in the middle. The program successfully detected the coloring violations and rejected the structure.

7. CONCLUSIONS AND FUTURE RESEARCH

This paper presented a new optimal sequential algorithm to recognize a rectangular mesh. The algorithm has complexity O(N). The program proved that the coloring scheme, that guides its way around the rings of the mesh, works. This technique can now be used, with minor modification, to develop algorithms for the recognition of more complex mesh-like structures, such as torus, and three-dimensional mesh structures. Research is ongoing in this area to prove this assertion.

REFERENCES

1. K. V. S. Ramarao, "Distributed algorithms for network recognition problems". IEEE Transactions on Computers, 38(9), 1989, 1240-1248

2. A. Kazmierczak & S. Radhakrishnan, "Optimal distributed ear decomposition with application to outerplanarity testing". Proc. Fifth IEEE Symposium on Parallel and Distributed Processing, 1993

3. R. Petreschim& A. Sterbini, "Recognizing strict 2-threshold graphs in O(m) Time". Information Processing Letters, 54(4), 1995, 193-198.

4. T. Raschle, & K. Simon, "Recognition of Graphs with Threshold Dimension Two". Proc. of the Twenty-seventh Annual ACM Symposium on the Theory of Computing, 1995, 650-661.

5. C-W. Yu & G-H. Chen, 1996. "An Efficient Parallel Recognition Algorithm For Bipartite-Permutation Graphs". IEEE Transactions on Parallel and Distributed Systems, 7(1), 1996, 3-10.

6. E. Jennings, A. Lingas, L. Motyckova, "Dynamic Detection of Forests of Tree Connected Meshes", ICCP (3), 1991, 300-301

7. R. Subbiah, S. S. Iyengar, S., Radhakrishnan, R. L. Kashyap, "An optimal distributed algorithm for recognizing mesh-connected networks". Theoretical Computer Science, 120(2), 1993, 261-278.

8. "The Graph Isomorphism Problem", http://citeseer.ist.psu.edu/fortin96graph.html

9. J. E. Hopcroft, J. K. Wong, "Linear Time Algorithm for Isomorphism of Planar Graphs", Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, 1974, Pp 172-184

10. R. T. Faizullin, A. V. Prolubnikov, "Heuristic Algorithms for Solving the Graph Isomorphism Problem" http://arxiv.math.GM/02050220v1, 21 May 2002

11. A. Kikina, R. T. Faizullin, "The Algorithm for Testing Graph Isomorphism", VINITI, 21 Jun 1995, pp 1789-95

12. R. T. Faizullin, A. V. Prolubnikov, "A Polynomial Time Algorithm for Solution of the Graph Isomorphism Problem Which is Applicable for the Wide Class of Graphs, http://www.psy.omsu.ons.kreg.ru/session/isomorphism

13. D. C. Schmidt, L. E. Druffel, "A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Metrics", Journal of the ACM, Vol 23, No 3, Jul 1976 Pp 433-445

14. H. Gazit, J. H. Reif, "A Randomized Parallel Algorithm for Planar Graph Isomorphism", ACM 1990, pp 210-219

15. S. Toda, "Graph Isomorphism Problem: Its Complexity and Algorithms", http://www.imsc.res.in/~fsttcs99/inv-abs.html, 1999

16. C. Boucher, D. Loker, "Graph Isomorphism Completeness for Subclasses of Perfect Graphs", Ontario Combinatorics Workshop, Apr 2006

17. J. D. Wolt, T. C. Woo, R. A. Volz, "Optimal Algorithms for Symmetry Detection in Two and Three Dimensions", http://deepblue.lib.umich.edu/bitstream/2027.42/8338/4/bam3304.0001.001.txt

18. J. Toran. "On the Hardness of Graph Isomorphism", SIAM Journal on Computing, 2004, pp1-16

19. E. Bengoetxea, "Inexact Graph Matching Using Estimation of Distribution Algorithms", PhD Thesis, EcoleNationaleSuperieure de Telecommunications, 2002

20. A. Golubchik, "The Graph Isomorphism Problem is Polynomial", http://arxiv.math.co.060777v1/, 29 Jul 2006

21. A. Golubchik, "The Polynomial Algorithm for Graph Isomorphism Testing", http://arxiv.org/abs/math.CO/0202085, Feb 2002

BIOGRAPHY

Dr. Kazmierczak: (Photo unavailable) Dr. K earned his PhD in Computer Science from the University of Oklahoma. He is currently on the faculty of Computer Information Systems at Northwest Arkansas Community College.