SANDIA REPORT

SAND2009-4494
Unlimited Release
Printed July 2009

Algebraic Connectivity and Graph Robustness

R. H. Byrne, J. T. Feddema and C. T. Abdallah

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2009-4494
Unlimited Release
Printed July 2009

Algebraic Connectivity and Graph Robustness

R. H. Byrne, J. T. Feddema
Data Analysis and Data Exploitation Department

C. T. Abdallah
University of New Mexico, ECE Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1243

Abstract

Recent papers have used Fiedler's definition of algebraic connectivity to show that network robustness, as measured by node-connectivity and edge-connectivity, can be increased by increasing the algebraic connectivity of the network. By the definition of algebraic connectivity, the second smallest eigenvalue of the graph Laplacian is a lower bound on the node-connectivity. In this paper we show that for circular random lattice graphs and mesh graphs algebraic connectivity is a conservative lower bound, and that increases in algebraic connectivity actually correspond to a decrease in node-connectivity. This means that the networks are actually less robust with respect to node-connectivity as the algebraic connectivity increases. However, an increase in algebraic connectivity seems to correlate well with a decrease in the characteristic path length of these networks - which would result in quicker communication through the network. Applications of these results are then discussed for perimeter security.

Contents

List of Figures

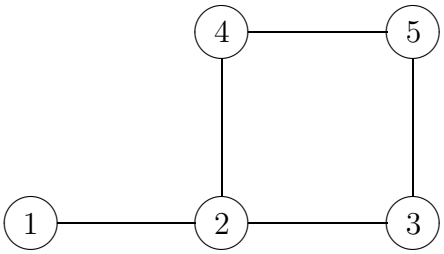

Figure 1: Graph Example

1 Introduction

Graph theory is a powerful tool for modeling the structure of different systems. A graph is represented as $G = (V, E)$ where V is the set of vertices and E is the set of edges [1]. An example of a simple graph is shown in Figure 1. For the graph shown in Figure 1 $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (2, 3), (2, 4), (3, 5), (4, 5)\}$. In a graph with no loops the degree of a vertex is the number of edges adjacent to that vertex. The diagonal degree matrix D of G is defined as $d_i = |N_i|$ where d_i is the degree of node i. The degree matrix D for the graph shown in Figure 1 is

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{1}$$

The adjacency matrix A of a graph G with n vertices is an $n \times n$ matrix with the following structure

$$a_{ij} = \begin{cases} i \neq j & \text{the number of edges joining vertex } i \text{ and vertex } j \\ i = j & \text{the number of loops at vertex } i \end{cases} \tag{2}$$

The adjacency matrix A of a graph G is always a symmetric matrix. A complete graph K_n is a graph with n vertices with exactly one edge joining every pair of vertices. The adjacency

matrix A for the graph shown in Figure 1 is

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{3}$$

The Laplacian matrix $L(G)$ of a graph G is defined as $L = D - A$ where D is the degree matrix and A is the adjacency matrix. The Laplacian matrix always has a zero eigenvalue $\lambda_1 = 0$. If G is connected, the second smallest eigenvalue λ_2 is greater than zero. This eigenvalue is named the algebraic connectivity of the graph [2] because it serves as a lower bound on the degree of robustness of the graph to node and edge failures. This follows from the following inequality [2]

$$\lambda_2(G) \leq \nu(G) \leq \eta(G) \tag{4}$$

where $\nu(G)$ is the node-connectivity and $\eta(G)$ is the edge-connectivity of a graph. Therefore, a network with high algebraic connectivity is robust to both node and edge failures.

The edge-connectivity and node-connectivity may also be calculated directly from the graph. For a graph $G = (V, E)$ and a set of edges denoted by F, the graph $G - F$ represents the graph obtained from G by deleting all of the edges in F. If a connected graph G becomes disconnected after removing the set of edges F, the set F is called a disconnecting set. A graph is *k-edge connected* if every disconnecting set has at least k edges. The edge-connectivity number $\eta(G)$ is the minimum size of the disconnecting set in G. For a complete graph with N vertices, $\eta(G) = N - 1$. Node or vertex connectivity is similarly defined. A set of vertices W in a graph $G = (V, E)$ is defined as a separating set if $G - W$ has more than one component. The connectivity number $\nu(G)$ of a graph G is defined as the minimum size of the separating set. A graph is *k-connected* if $\nu(G) \geq k$. In other words, k is the minimum number of vertices that must be removed in order to break a connected graph into two or more components.

The edge-connectivity number $\eta(G)$ for graph shown in Figure 1 is $\eta(G) = 1$. By removing the edge between nodes 1 and 2 the graph becomes disconnected. The node-connectivity number $\nu(G)$ is $\nu(G) = 1$. By removing node 2 the graph becomes disconnected. The algebraic connectivity for the same graph is $\lambda_2(G) = 0.83$, which satisfies the inequality.

2 Literature Review

Graph theory was originally developed by the famous mathematician Leonhard Euler in 1736. The problem that motivated Euler was whether it was possible to walk a route that crosses each of the seven bridges in Königsberg, Prussia, exactly once and return to the starting

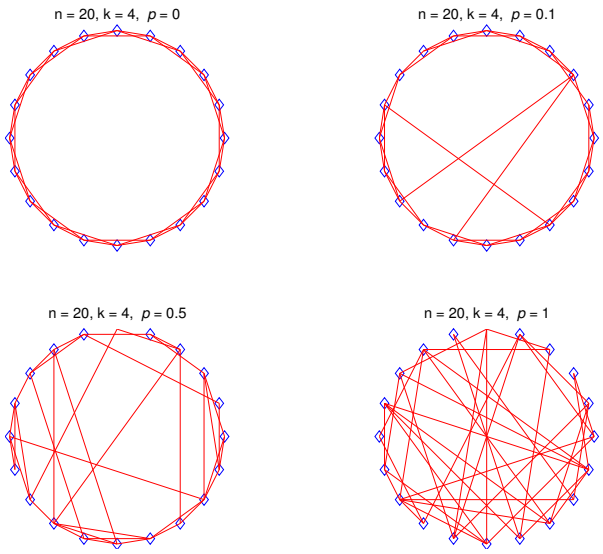

Figure 2: Random ring lattice graph $G = C(n, k)$ with $n = 20$, $k = 4$, for $p = 0, 0.1, 0.5, 1$

point. Using graph theory, Euler proved that no such path exists. Because graph theory looks at pairwise relationships between objects in a collection, graph theory is well suited for modeling and analyzing different types of networks. An overview of complex networks appears in [3]. A description of the world-wide web's scale-free characteristics appears in [4].

In [5], Watts and Strogatz look at the phase transitions that occur between regular and random graphs. They show that the small-world phenomenon (popularly known as six degrees of separation) can occur in sparse networks with many vertices. They show this effect by starting with regular ring lattice networks with n vertices and k edges per vertex and then rewire each edge with probability p. The effects of random rewiring of links for regular ring lattice networks are shown in Figure 2. Olfati-Saber then showed that it is possible to greatly increase the algebraic connectivity in regular complex networks without adding new links or nodes by using the same type of random rewiring [6]. He used this result to show that the consensus problem can be solved more quickly on certain small-world networks. The paper also states that "a network with relatively high algebraic connectivity is necessarily robust to both node-failures and edge-failures" [6]. We show that the contrary is true. While there is an increase in algebraic connectivity, there is a decrease in node- and edge-connectivity (but Fiedler's inequality still holds, it is just a conservative bound). Research has also been conducted on maximizing the algebraic connectivity for a given graph [7]. A reference on Laplacian matrices of graphs appears in [8].

A $k - separator$ or $k - shredder$ of a $k-$node connected graph is defined as the set of k nodes whose removal results in an unconnected graph. When analyzing network robustness, it is important to identify the nodes that are most vulnerable to bringing down network

connectivity. Papers that discuss algorithms for $k - shredders$ include [9, 10].

3 Algebraic Connectivity and Network Robustness

A graph G that consists of a set V of vertices (or nodes) and a collection of E edges is said to be connected if and only if there is a path between every pair of vertices in it. The node-connectivity number $\nu(G)$ of a graph G is defined as the minimum size of a separating set, or in other words the minimum number of nodes that may be removed to separate the graph into more than one component. Similarly, the edge-connectivity number $\eta(G)$ is defined as the minimum number of edges that may be removed to separate the graph into more than one component. Algebraic connectivity is of great interest because of the following inequality developed by Fiedler:

$$\lambda_2(G) \leq \nu(G) \leq \eta(G) \tag{5}$$

which states that the algebraic connectivity of a graph G (defined as the second smallest eigenvalue $\lambda_2(G)$ of the Laplacian) is less than or equal to the node-connectivity which is less than or equal to the edge-connectivity [2]. Although increasing the algebraic connectivity increases the lower bound on node-connectivity our simulation results show that for circular and mesh lattice graphs an increase in algebraic connectivity often corresponds to a decrease in node-connectivity and edge-connectivity.

The *small-world network* introduced by Watts and Strogatz [5] was based on a one-dimensional lattice on a ring with n nodes where each node is connected to its k nearest neighbors. They showed that random rewiring of nodes with a small probability p greatly reduces the characteristic path length resulting in a small-world network. Figure 2 exhibits the effects of random rewiring for a network with 20 nodes and $k = 4$. Olfati-Saber showed that this random re-wiring also results in a large increase in algebraic connectivity for ring lattices [6].

Unfortunately large increases in algebraic connectivity for certain types of networks often correspond to a decrease in node-connectivity and edge-connectivity. As an example, the results for a circular random graph with 100 nodes are shown in Figure 3. For this case, we start with a ring lattice with $n = 100$ vertices and $k = 4$ edges per vertex and then rewire each edge at random with a probability p. As p increases from 0 to 0.9 there is a large increase in algebraic connectivity and a decrease in the mean path length of the network. However, the node-connectivity and edge-connectivity of the network decrease as the probability p increases. Similar results can be shown for a regular mesh lattice like the one shown in Figure 4 where there are 100 nodes and each node has a communication radius $R = 1$. The results for this mesh lattice are summarized in Figure 5.

Olfati-Saber noted that the algebraic connectivity gain $\gamma_2 = \lambda_2(p)/\lambda_2(0)$ has an S-shape that remains the same for various network parameters [6]. The algebraic connectivity gain for our simulation results are shown in Figure 6. These agree with the results in [6]. The more

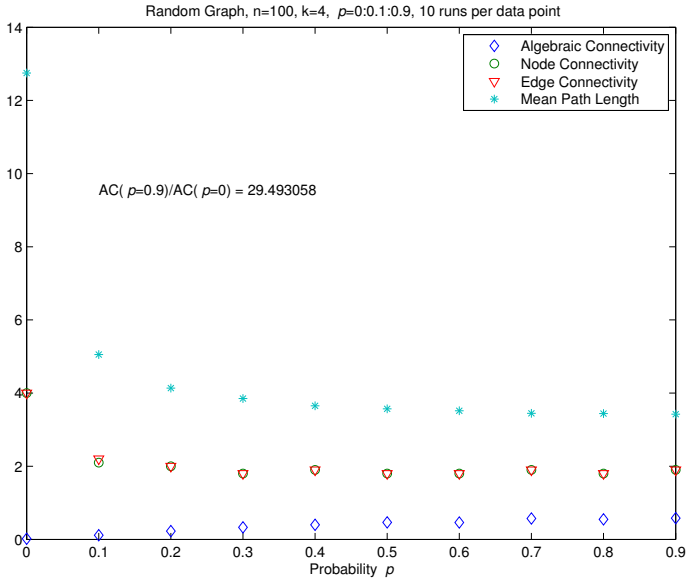

Figure 3: Results for a ring lattice random graph, N=100, k=4

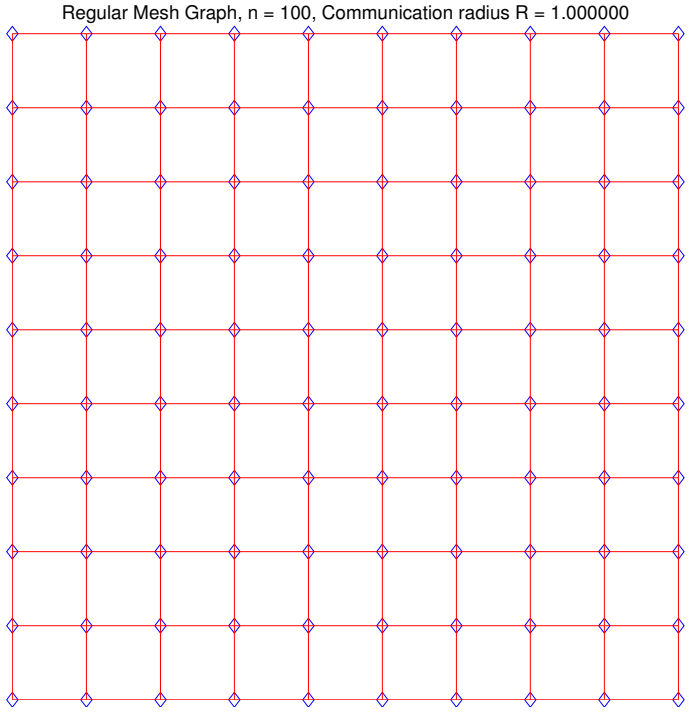

Figure 4: Regular mesh lattice graph, N=100, Communication radius R=1.

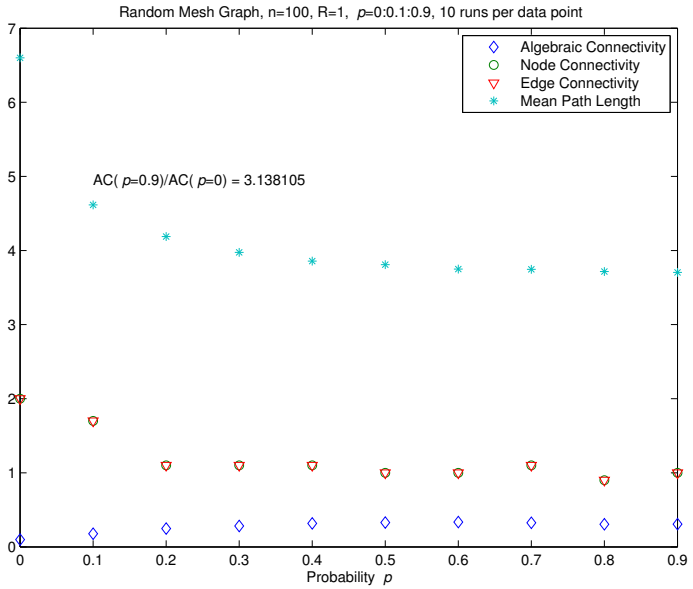

Figure 5: Results for a Mesh Lattice Graph, N=100, R=1

jagged behavior can be attributed to less monte carlo simulations (10 versus 20), a slightly smaller network (100 versus 200 nodes), and sparser connections (4 nearest neighbors versus 6 nearest neighbors). To look at the decrease in node-connectivity and edge-connectivity the subsequent figures show the ratios $\nu(p)/\nu(0)$ and $\eta(p)/\eta(0)$. Although the algebraic connectivity gain has increased, Figures 7-8 show how the node-connectivity and edge-connectivity have decreased significantly as a result of the random rewiring. The mean path length gain is shown in Figure 9. Random rewiring of the circular lattice improves (decreases) the mean path length. For larger circular lattices, the improvement will be even more pronounced.

In a system where nodes are redundant or dispensable, improving algebraic connectivity can improve the overall robustness of the network by reducing the characteristic path length. However, in systems where each node is critical, concepts like node-connectivity and edge-connectivity are important parameters for assessing robustness. This highlights the fact that there are often tradeoffs when assessing robustness to different parameters. In addition, there are computational tradeoffs. Computing algebraic connectivity is much quicker than computing node-connectivity or edge-connectivity for large networks. The next section identifies an application where these tradeoffs are directly applicable.

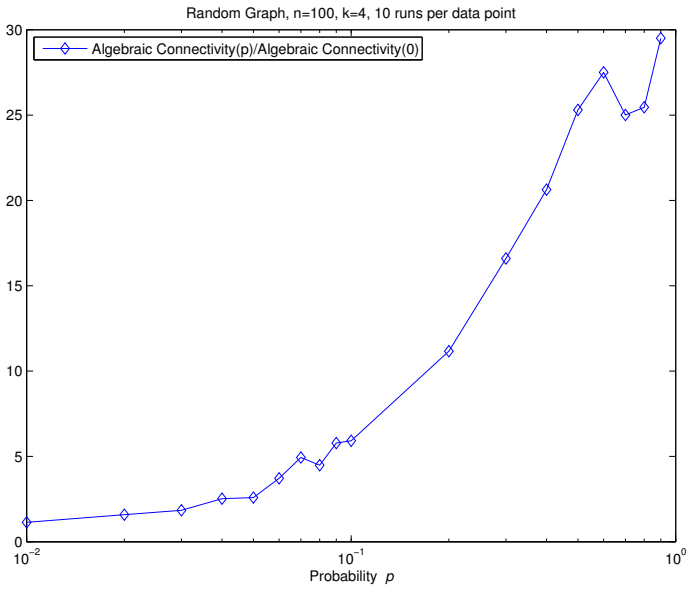

Figure 6: Algebraic Connectivity Gain for a Ring Lattice Random Graph, N=100, k=4

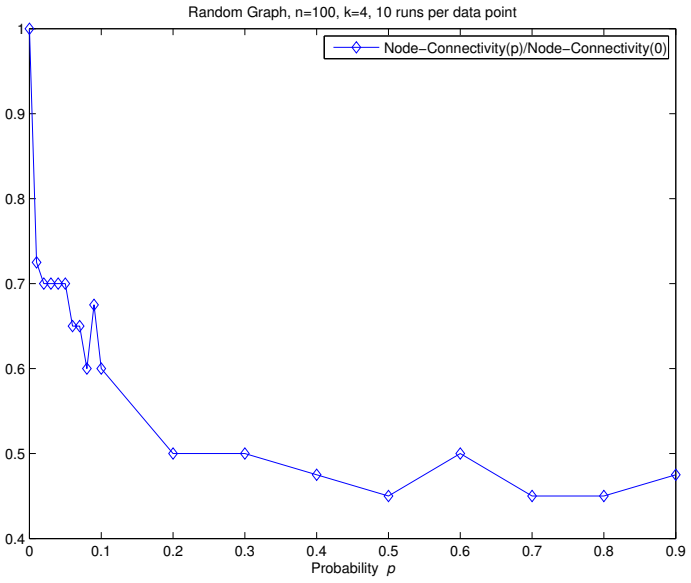

Figure 7: Node-Connectivity Gain for a Ring Lattice Random Graph, N=100, k=4

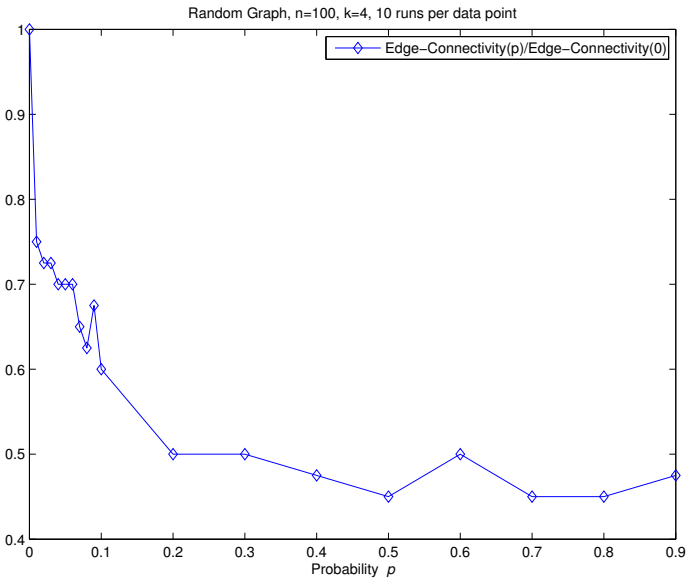

Figure 8: Edge-Connectivity Gain for a Ring Lattice Random Graph, N=100, k=4

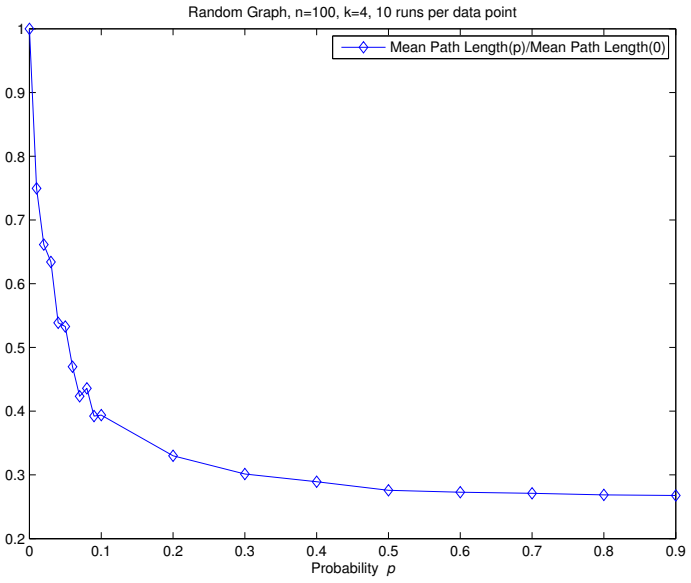

Figure 9: Mean Path Length Gain for a Ring Lattice Random Graph, N=100, k=4

	Figure 10	Figure 11	Figure 12
algebraic connectivity	0.268	1.00	1.27
node-connectivity	2	3	4
edge-connectivity	2	3	4
mean path length	3.00	1.92	1.75

Table 1: Network Parameters, Physical Security Example

4 Physical Security Application

A typical perimeter security application is shown in Figure 10. Here there is a building that is protected by fixed sensors located around the perimeter. Node 1 is the control node/response center, while the other nodes represent security sensors. The sensors are connected in a simple ring topology with two paths back to the control node. There are several ways to analyze the network. The first approach is to treat all nodes as equal and to look at parameters like node-connectivity, edge-connectivity, mean path length, and algebraic connectivity. However, the control node is more critical than the other nodes. Therefore another approach is to just look at the sensor nodes as a separate network of homogeneous nodes and then to make sure that the robustness of the links to the control node is adequate. Using this methodology, if the sensor network is still connected, then there should be a path back to the control node. If latency is critical, then the mean path length back to the control node becomes an important parameter.

Three different examples of topologies are presented in Figures 10 - 12. Figure 10 has a minimal ring topology with two connections to the control node. Figure 11 has the same topology with cross connects through the network. Figure 12 keeps the ring topology and adds a third connection to the control node for symmetry. The network parameters for these three graphs (sensors only) are listed in Table 1.

The cross-connections in Figure 11 reduce the mean path length as expected. In a larger size network, this effect would be even more pronounced. Because the ring is relatively small, the network in Figure 12 has the smallest mean path length. An increase in algebraic connectivity corresponds with a decrease in the mean path length of the network. In designing a large perimeter physical security system, one of the many tradeoffs will involve robustness (e.g. node and edge-connectivity) and mean path length, which will correspond to latency. The topology of many perimeter security systems also directly maps to the circular lattice.

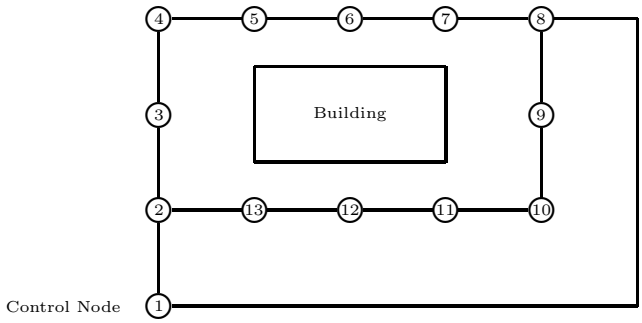

Figure 10: Physical Security Graph Example

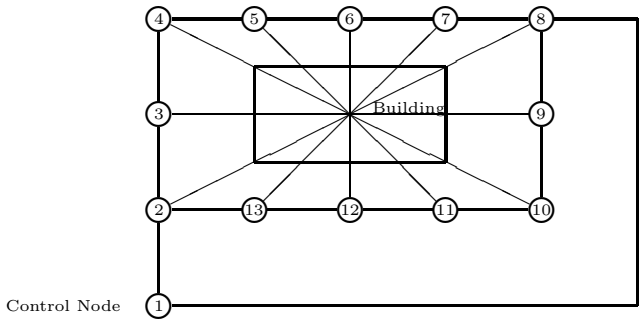

Figure 11: Physical Security Graph Example, Cross Connections

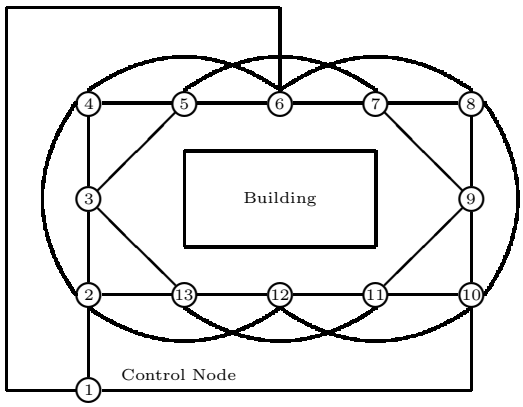

Figure 12: Physical Security Graph Example, 2^{nd} Neighbor Connections ($G = C(12, 4)$ plus control node)

5 Summary and Conclusions

This paper has shown that for circular random lattices and mesh lattices an increase in algebraic connectivity does not necessarily result in an increase in node-connectivity. Therefore, Fiedler's definition of algebraic connectivity is a conservative lower bound. In a system where nodes are redundant or disposable and the speed of the network is important, improving algebraic connectivity can improve the performance of the network by reducing the characteristic path length. However, in systems where connectivity is critical, concepts like node-connectivity and edge-connectivity are important parameters for assessing robustness. In addition, there are computational tradeoffs. Computing algebraic connectivity is much quicker than computing node-connectivity or edge-connectivity for large networks. The perimeter security application is an example where all of these parameters are important. When cost constraints are included which limit the number of nodes or connections, it should be possible to perform an optimization to arrive at the network configuration which maximizes robustness (node/edge-connectivity) and speed (maximizes algebraic connectivity). This will be an area for further research.

Acknowledgments: This work was supported by the United States Department of Energy under Contract DE-AC04-94AL8500. Funding for this research was provided by the "Design Tools for Complex Dynamic Security Systems" Laboratory Directed Research and Development (LDRD) effort.

References

[1] B. Bollobás, *Modern Graph Theory*, vol. 184. New York: Springer-Verlag, 1998.

[2] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.

[3] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, pp. 268–276, March 2001.

[4] A. Barabási, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: The topology of the world-wide web," *Physica A*, vol. 281, pp. 69–77, 2000.

[5] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, June 1998.

[6] R. Olfati-Saber, "Ultrafast consensus in small world networks," in *Proceedings of the 2005 American Control Conference*, (Portland, OR), pp. 2371–2378, June 8-10 2005.

[7] Y. Kim and M. Mesbani, "On maximizing the second smallest eigenvalue of a state-dependent graph laplacian," in *Proceedings of the 2005 American Control Conference*, (Portland, OR), pp. 99–103, June 8-10 2005.

[8] R. Merris, "Laplacian matrices of graphs: A survey," *Linear Algebra and Its Applications*, vol. 197, no. 198, pp. 143–176, 1994.

[9] J. Cheriyan, "Fast algorithms for k-shredders and k-node connectivity augmentation," *Journal of Algorithms*, vol. 33, pp. 15–50, October 1999.

[10] R. Hegde, "Finding 3-shredders efficiently." citeseer.ist.psu.edu/hegde02finding.html.

6 Appendix A - MATLAB M-FILES

```
% algebraic_connectivity.m
% Ray Byrne
% August 2, 2005
% function returns the algebraic connectivity of the
% adjacency matrix A

function AC = algebraic_connectivity(A)

EV = eig(degree(A) - A);

EV = sort(EV,'descend');

N = length(A);

eps = 0.01; % tolerance for 0

if (N >= 2) && (abs(EV(N)) < eps)
    AC = EV(N-1);
    return;
else
    AC = -1;
    return;
end  % for

return;
```

```
% edge_connectivity.m
% Ray Byrne
% 7/29/05
% function edge_connectivity(A) returns the edge connectivity of a
% graph G described by the adjacency matrix A

function k_edge = edge_connectivity(A)

N = length(A);

% check 0 connectivity
if (graph_connected(A))
else
    k_edge = 0;
    return
end  % else

% count edges, store edges in I,J
I = [];
J = [];

for i = 2:N  % i is row index
    for j = 1:(i-1)  % j is column index
        if (A(i,j) ~= 0)   % check neq 0 to allow weighted edges
            I = [I; i];  % store list of edges
            J = [J; j];  % store list of edges
        end  % for
    end  % for
end  % for

edge_count = length(I);

for K = 1:edge_count  % % check K connectivity
    C = nchoosek( 1:edge_count, K);  % calculate all combinations
    num_tests = length(C);
    for j = 1:num_tests
        del_list = C(j,:);  % pull out a row of C
        A_temp = A;
        for k = 1: length(del_list)
            A_temp(I(del_list(k)),J(del_list(k))) = 0;  % delete edges
```

```
            A_temp(J(del_list(k)),I(del_list(k))) = 0;  % delete edges
        end
        if ( ~graph_connected(A_temp) )
          k_edge = K;
          for k = 1: length(del_list)  % list out set of edges that breaks connectivity
              broken_I = I(del_list(k))
              broken_J = J(del_list(k))
          end % for
        return;
        end % if
    end % for
end  % for

return;
```

```
% graph_connected.m
% Ray Byrne, July 29, 2005
% 15234
% the function graph_connected(A) checks the adjacency matrix
% to verify that the graph is connected.  Uses a Depth-First
% Search Algorithm (DFS) to determine if every node is
% reachable from node 1.  Function returns 1 if the graph is
% connected

function [C] = connected_graph (A)

N = length(A);

visited = zeros(1,N);  % stores whether the node has been visited (1 == visited)

DFS = zeros(1,N);  % store DFS number
DFS_number = 0;  % stores the last number assigned


[DFS_number, DFS, visited] = search_graph(1, DFS_number, DFS, visited, A);


if ( sum(visited) == N)
    C = 1;
    %disp('Graph is Connected');
else
    C = 0;
    %disp('Sorry, Graph is not Connected!');
end  % if

return;
```

```
% degree.m
% Ray Byrne
% August 2, 2005
% degree.m returns the D matrix where the (i,i) term is
% the degree of the ith node.  The input is the adjacency matrix A

function D = degree(A)

degrees = sum(A);
N = length(degrees);
D = eye(N);
for i=1:N
    D(i,i) = degrees(i);
end % for
return
```

```
% gen_mesh.m
% Ray Byrne
% August 1, 2005
% The function mesh.h returns the X,Y locations of an
% m by n mesh with unit spacing of 1

function [X,Y] = gen_mesh(m,n)
X = [];
Y = [];

for i=1:m  % number of rows
    for j=1:n  % number of columns
        X = [X; j-1];
        Y = [Y; i-1];
    end
end

return;
```

```
% gen_adjacency.m
% Ray Byrne
% August 1, 2005
% generates adjacency matrix for a set of vertices with
% locations [X,Y] that have a communication range of R

function [A] = gen_adjacency(X,Y,R)

N = length(X);

A = zeros(N,N);

for i=1:N,
  for j=1:N
        distance = sqrt( (X(i)-X(j))^2 + (Y(i)-Y(j))^2 );
        if (distance <= R) && (i ~=j)
            A(i,j) = 1;
            A(j,i) = 1;
        end
  end
end

return
```

```
% dijkstra.m
% Ray Byrne
% August 6, 2005
% Function performs dijkstra's algorithm on the adjacency matrix A
% from the vertex numbered V.  Assumes that for an NxN adjacency
% matrix there are vertices numbered 1-N.  Returns V, the start
% vertex, the shortest distance SD where SD(i) is the shortest distance
% from V to vertex i.  Also returns the labeled arcs in ARCS that
% constitute a shortest path arborescence rooted at vertex V.

function [V, SD, ARCS] = dijkstra(A,V)

N = length(A);
if V > N
    V = -1;
    SD = -1;
    ARCS = [-1 -1];
    return;  % case of invalid Vertex number
end

ARCS = [];  % store labeled arcs here

P = V
T = [1:(V-1) (V+1):N]

L=inf*ones(1,N);
L_prime = inf*ones(1,length(T));
L_prime_min_vertex = zeros(1,length(T));
L(V) = 0;

for i=1:length(T)
    if (A(V,T(i)) ~= 0)
        L_prime(i) = A(V,T(i))  % initialize distances from start vertex V
        L_prime_min_vertex(i) = V;
    end  % if
end  % for

start_min = 1; %

while length(P) < N
```

```
%L_prime = L_prime  % display

[Y, I] = min(L_prime);
[Z, I] = max(L_prime);
if ( Y == 0)  && (Z == 0)
    return  % no finite minimum paths in L_prime, stop
else
    minimum = L_prime(1);
    index = 1;
    if (length(T) > 1)
      for i=2:length(T)
        if (L_prime(i) < minimum) && (L_prime(i) > 0)
            minimum = L_prime(i)
            index = i
        end  % if
      end % for
    else
        % do nothing
    end

    P = [P, T(index)];  % add shortest path to T
    store_T = T(index);
    L(T(index)) = minimum; % label vertex
    store_length = minimum;
    ARCS = [ARCS; L_prime_min_vertex(index) T(index)];  % label arc


    T(index) = [];  % remove from temporary
    L_prime(index) = [];
    L_prime_min_vertex(index)=[];

    if length(T) > 0
      for i=1:length(T)
          t1=L_prime(i);
          if A(store_T,T(i)) > 0
              t2=store_length+A(store_T,T(i));
          else
              t2 = inf;  % no path
          end
        L_prime(i) = min([t1 t2]);
```

```
            if t2 < t1
                L_prime_min_vertex(i) = store_T;
            end  % if
          end  % for
        end  % if


    end % if
end

SD =  L;
return
```

```
% vis_SD.m
% Ray Byrne
% helps visualize the number of hops required to reach
% any node in the network from vertex V
function vis_SD(V,SD,X,Y)

if length(SD) ~= length(X)
    return
end

clf;

axis([min(X)-1 max(X)+1 min(Y)-1 max(Y)+1]);
for i=1:length(SD)
    s = sprintf('%i', SD(i));
    text(X(i),Y(i),s);
    hold on;
end % for

xlabel('X-Axis')
ylabel('Y-Axis')
s = sprintf('Shortest Path from Node %i in Hops', V);
title(s);

return
```

```
% search_graph.m
% Ray Byrne, 7/29/05
% Dept 15234
% the function search_graph.m is a recursive function
% used in a DFS search of a graph to determine if
% all nodes are connected

function [DFS_number, DFS, visited] = search_graph(I, DFS_number, DFS, visited, A)
  DFS_number = DFS_number + 1;
  DFS(I) = DFS_number;
  visited(I) = 1;
  A_row = A(I,:);
  N = length(A_row);
  for i=1:N
      if (A_row(i) > 0) && (visited(i) == 0)  % check > 0 to allow weighted connections
          [DFS_number, DFS, visited] = search_graph(i,DFS_number, DFS, visited,A);
      end % if
  end  % for
```

```
% plot_arcs
% Ray Byrne
% August 8
% function plots arcs in ARCS given the X,Y location of the vertices
function plot_arcs(ARCS,X,Y)

axis([min(X)-1 max(X)+1 min(Y)-1 max(Y)+1])
hold on

for i=1:length(ARCS)
    plot([X(ARCS(i,1)) X(ARCS(i,2))],[Y(ARCS(i,1)) Y(ARCS(i,2))],'r')
end

return
```

```
% plot adjacency.m
% Ray Byrne
% August 8, 2005
% function plots the edges of the adjacency matrix A
function plot_adjacency(A,X,Y)

clf;

N = length(A);
ARCS = [];

for i=1:N,
    for j=1:N,
        if A(i,j) ~= 0
            ARCS = [ARCS; i j];
        end % if
    end % for
end % for

plot_arcs(ARCS,X,Y);
xlabel('X-Dimension');
ylabel('Y-Dimension');


return
```

This page intentionally blank.

Distribution

1	MS1104	Rush Robinett, 6000
1	MS1003	Barry Spletzer, 6470
1	MS0576	John Feddema, 5535
10	MS1003	Ray Byrne, 5535
1	MS 108	David Wilson, 6332
1	MS1007	Larry Shipers, 6471

1		Chaouki Abdallah ECE Department University of New Mexico Albuquerque, NM 87131

1	MS0899	Technical Library, 9536 (electronic copy)