

# Methodologies and Metrics for Assessing the Strength of Relationships between Entities within Semantic Graphs

T. L. Hickling, W. G. Hanley

September 29, 2005

Lawrence Livermore National Laboratory

UCRL-TR-216074



This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

## Abstract

Semantic graphs are becoming a valuable tool for organizing and discovering information in an increasingly complex analysis environment. This paper investigates the use of graph topology to measure the strength of relationships in a semantic graph. These relationships are comprised of some number of distinct paths, whose length and configuration jointly characterize the strength of association. We explore these characteristics through the use of three distinct algorithms respectively based upon an electrical conductance model, Newman and Girvan's measure of betweenness [5], and cutsets. Algorithmic performance is assessed based upon a collection of partially ordered subgraphs which were constructed according to our subjective beliefs regarding strength of association.

# Contents

<b>1 INTRODUCTION</b> .....	<b>1</b>
<b>2 PARTIAL ORDERING</b> .....	<b>3</b>
<b>3 CONDUCTANCE</b> .....	<b>5</b>
3.1 CONDUCTANCE MODEL CONSIDERATIONS .....	6
3.2 GRAPH MODIFICATION .....	7
3.3 IMPLEMENTATION.....	9
<b>4 CUTSETS</b> .....	<b>11</b>
4.1 SIMULATION APPROACH.....	12
4.2 HIDDEN MARKOV MODEL .....	13
<b>5 BETWEENNESS</b> .....	<b>14</b>
<b>6 INCORPORATING SOURCE DOCUMENT CONFIDENCE</b> .....	<b>18</b>
<b>7 RESULTS</b> .....	<b>19</b>
7.1 CONDUCTANCE RESULTS.....	19
7.2 CUTSET RESULTS.....	21
7.3 BETWEENNESS (CRITICALITY) RESULTS.....	23
7.4 SOURCE DOCUMENT CONFIDENCE RESULTS .....	24
<b>8 APPLYING THE METRICS TO REAL WORLD SEMANTIC GRAPHS</b> .....	<b>26</b>
<b>9 CONCLUSION</b> .....	<b>27</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>27</b>
<b>REFERENCES</b> .....	<b>28</b>
<b>APPENDIX – HMM</b> .....	<b>29</b>

# 1 Introduction

In practical informatic applications, semantic graphs are rapidly becoming more valuable for representing massive quantities of relational data based upon a defined ontology. Methods must be developed for extracting and interpreting information from these massive graphs in order for them to be useful in the real world. In this paper, we address the need to quantify, or rank, our confidence in the relationships between entities within a semantic graph.

We begin by providing some terminology that will be used extensively in this discussion. A *graph*  $G = (V, E)$  is composed of a set of vertices, or nodes,  $V = \{v_1, v_2, \dots, v_n\}$ , together with a set of edges, or links,  $E \subseteq V \times V$ . A *subgraph*  $G'$  of  $G$  is defined by  $G' = (V', E')$  where  $V' \subseteq V$  and  $E' \subseteq (V' \times V') \cap E$ . We will generally denote an edge from vertex  $v_a$  to  $v_b$  as  $v_a v_b$ , and any quantity  $q$  which applies specifically to the edge  $v_a v_b$  will be denoted by  $q_{ab}$ .

A *path*  $p$  in  $G$  from  $v_a$  to  $v_b$  can then be defined as the set of  $l$  coincident edges given by

$$p = \{v_a v_{i_1}, v_{i_1} v_{i_2}, \dots, v_{i_{l-2}} v_{i_{l-1}}, v_{i_{l-1}} v_b\}, \quad (1)$$

or equivalently,

$$p = (v_a v_{i_1} v_{i_2} \dots v_{i_{l-2}} v_{i_{l-1}} v_b), \quad (2)$$

where  $l$  is considered to be the *length* of  $p$ . The topological strength of association metrics between any two nodes will depend heavily upon the nature of the paths which link them. Therefore, we will differentiate here between *simple* and *composite* paths, which behave differently under certain circumstances. Let  $P$  denote the set of all distinct paths  $\{p_1, p_2, \dots, p_k\}$  from  $v_a$  to  $v_b$ . Then  $p_j$  is a *simple* path if  $\forall v_n \in p_j, v_n \notin p_i, i \neq j$ , i.e., every node in  $p_j$  is unique to  $p_j$ . The simple paths  $p_i$  and  $p_j$  may also be described as *internally disjoint*. Any path which is not a simple path will be known as a *composite* path. Such paths may intersect in various ways with other composite paths, and the extent of their coincidence will be instrumental in determining the strength of association.

*Semantic graphs* are graphs in which nodes represent real-world entities, and edges represent relationships between them. Semantic graphs are governed by *ontologies*, which define the permissible contents of the graph. Depending upon how the semantic graph is constructed, we may also apply a level of *confidence* or *degree of belief* to a node or link in the graph. While this high-level description of semantic graphs will suffice for the purposes of this paper, they are discussed in greater detail in [1].

We begin our discussion by constructing a definition of the nebulous concept of *strength* as it applies in the semantic graph context. The concepts presented in this paper will be based upon the assumption that the strength of a given relationship is affected to some degree by all of the following features, not necessarily exhaustive:

- I. Subgraph topology, i.e., number and configuration of paths between two nodes of interest.
- II. The level of confidence associated with nodes and links connecting two nodes of interest. Such confidence may be based upon the reliability of information sources, information extraction, and/or data fusion. The development and use of an appropriate confidence metric would require the participation of an expert who is knowledgeable of the information extraction techniques used to build the graph, as well as the nature and reliability of supporting document/data sources.
- III. The *stability* of the subgraph topology between two nodes, where *stability* describes in some fashion the robustness of a relationship to the removal or failure of data.
- IV. The *criticality* of components which link two nodes. This is a local stability metric, applied to an individual node or link, which is based upon the degree to which a relationship relies on the component in question.
- V. *Semantic interpretation* of information in the semantic graph [1]. Such a metric attempts to determine the usefulness of information based upon its semantics and relevance to the question of interest.
- VI. Underlying document/data support. Data based upon diverse information sources may be considered stronger or more reliable.

We will not address (V) in detail, as this is discussed at length in [1], nor will we comprehensively discuss techniques for designing a confidence metric based on information extraction for specific nodes and links (II), since such a metric depends heavily upon the nature of the information system and the problem domain. However, both of these play a significant role in the interpretation of semantic graph contents, so we will discuss appropriate ways to incorporate them into the topological strength metrics we have developed, and we will also address an initial approach to creating data confidence values from those of document sources.

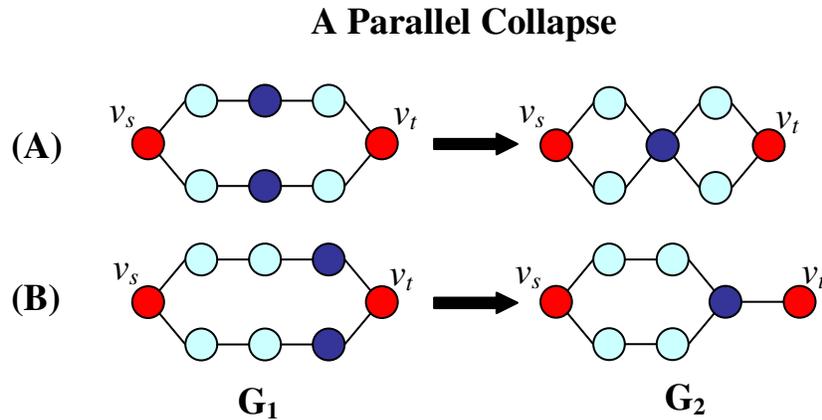
Note also that the above measures depend upon a neighborhood within the semantic graph containing paths which connect the two nodes of interest. Clearly, any two nodes linked by at least one path reside in the same connected graph component. In a real-world application, such a connected component is potentially massive, and examining *every* path between two nodes may be an intractable problem. Later in this paper, we will discuss in more detail the conditions under which these metrics should be applied. For the purpose of clarity, however, we will assume in the meantime that our example graphs comprise the relationships between two entities in their entirety.

We will begin by introducing partially ordered sets due to graph configuration in Section 2, followed by detailed discussions of our strength of association, global stability, and criticality algorithms in Sections 3, 4, and 5, respectively. Section 6 introduces a method by which we incorporate document/data source confidence into these three algorithms. We present notable results on sample graphs in Section 7, and finally, Section 8 discusses the appropriate conditions for applying these metrics, along with suggestions for preprocessing graphs prior to implementation.

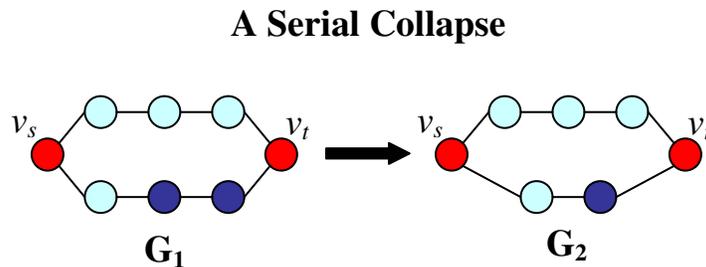
## 2 Partial Ordering

In order to provide a basis by which to judge the performance of our various algorithms, we define a *partially ordered set*, or “*poset*”, of graphs we wish to examine. Let us denote the strength of association between two nodes,  $v_s$  and  $v_t$  (frequently referred to as the *source node* and *terminal node*, respectively), in a graph  $G$  as the function  $\Psi(s, t | G)$ . We make the following initial assumptions regarding strength of association as the foundation for a poset:

- (1)  $G' \subseteq G \Rightarrow \Psi(s, t | G') \leq \Psi(s, t | G)$
- (2) A *parallel collapse* of nodes in  $G$  causes  $\Psi(s, t | G)$  to decrease (Figure 2-1).
- (3) A *serial collapse* of nodes in  $G$  causes  $\Psi(s, t | G)$  to increase (Figure 2-2).



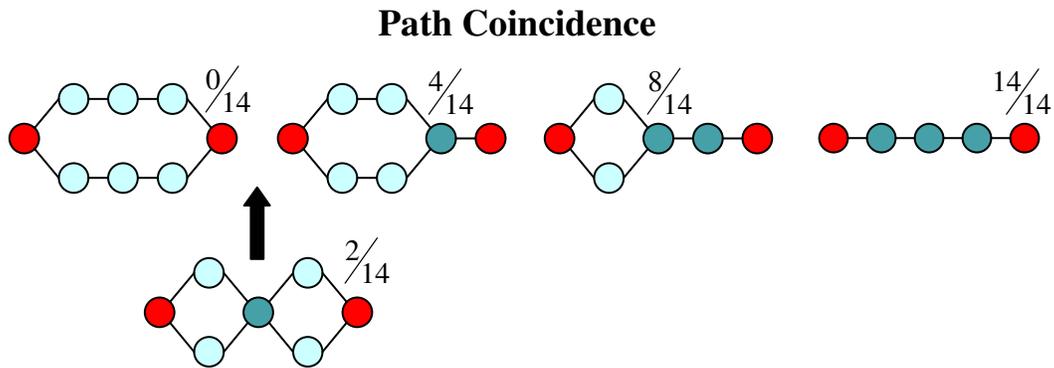
**Figure 2-1: A parallel collapse occurs when two nodes from distinct paths (shown in blue) in  $G_1$  fuse in  $G_2$ . (A) A parallel collapse may cause a single node dependency between paths, or (B) link dependencies may form. In a multigraph setting, (B) may exhibit a single node dependency.**



**Figure 2-2: A serial collapse occurs when two adjacent nodes (shown in blue) in  $G_1$  fuse in  $G_2$ .**

Assumption (1) is clearly true given that any subgraph of  $G$  connecting  $v_s$  and  $v_t$  contains less supporting information than  $G$  for the relationship and must thus represent a weaker association (we assert that the strengths are equal only for the trivial case where  $G' = G$ ).

Assumption (3) is similarly obvious in that it illustrates the basic assumption that a short path is always topologically stronger than a long path (this is not necessarily true semantically, as described in [1]). The underlying assumption which gives rise to (2) is that two simple paths represent a stronger association than one simple path. A single simple path can be envisioned as two paths which intersect at every point, while two simple paths are completely independent. Therefore if two paths only partially coincide (forming a composite path), their combined strength should lie between these two extremes, dictated by the extent of the coincidence between them. If we quantify the degree of coincidence as the proportion of nodes and links shared between the two, we get the ordering shown in Figure 2-3.



**Figure 2-3: Dark nodes indicate the portions of shared information between two paths. The proportion of shared data determines the ordering for paths collapsed in parallel. Thus we can properly order the graph from Figure 2-1(A) with its single-node dependency.**

While most subgraphs of semantic graphs encountered in real world applications will be far more complex than those shown here, we can use these three assumptions to qualitatively evaluate the performance of our algorithms in ranking various relationships.

### 3 Conductance

First we present a promising algorithm which is based upon the concept of an electrical circuit [2, 3, 7]. We may imagine the graph as a series of pathways along which an electrical current might travel, where the links of our graph represent wires, and each node is a connection between those wires. We associate with each node an *electric potential*, which acts as a sort of “electrical pressure”, pushing current through the graph from high to low potential. Thus, current will only flow through a link if there is a *potential difference* between the two nodes it connects. The magnitude of the potential difference between  $v_a$  and  $v_b$  is determined both by the amount of current flowing between them, and the amount of opposing force on the link  $v_a v_b$ . This opposing force is called the *resistance*. We can formalize our definition of potential difference, denoted  $V_{ab}$ , as the amount of energy required to move a current,  $\omega_{ab}$ , across a resistance  $r_{ab}$ . This relationship is described by *Ohm’s Law* (3), and it holds true for any resistor or component in an electrical network, as well as the network itself, as long as the resistors are *ideal* (resistances remain constant).

$$V_{ab} = \omega_{ab} r_{ab} \quad \forall v_a, v_b \quad (3)$$

The *conductance* on a resistor between  $v_a$  and  $v_b$ , denoted  $k_{ab}$ , is defined as the inverse of the resistance, i.e.,

$$k_{ab} = \frac{1}{r_{ab}} \quad \forall v_a, v_b, \quad (4)$$

so that conductance on a component of the graph increases as its resistance decreases. The graph conductance is the metric we will use to measure the strength of association between two nodes in a semantic graph.

To see why this is appropriate, we must first consider the behavior of resistance and conductance due to graph configuration. In an electrical circuit, resistors along a single path or path fragment are connected in series and have an additive dampening effect on a current; hence longer pathways will carry a higher resistance (i.e., lower conductance). On the other hand, conductance is additive across resistors in parallel paths between two points in the electrical network, which implies that conductance will increase across path multiplicity. In terms of our strength of association metric, increasing numbers of parallel paths (e.g., multiple simple paths, or disjoint portions of composite paths) will boost strength, while increasing the length of simple paths will decrease the strength.

The behavior of conductance in the electrical network mimics in some sense the effect we would like to see from a confidence metric, if such a metric were available, for the data underlying the graph. Low confidence, as with low conductance, should lower the strength of association. Let us assume, then, for the sake of argument, that we have such a confidence metric,  $c_{ab} \in (0, 1]$ , where  $c_{ab} = 1$  represents certainty, and that this metric

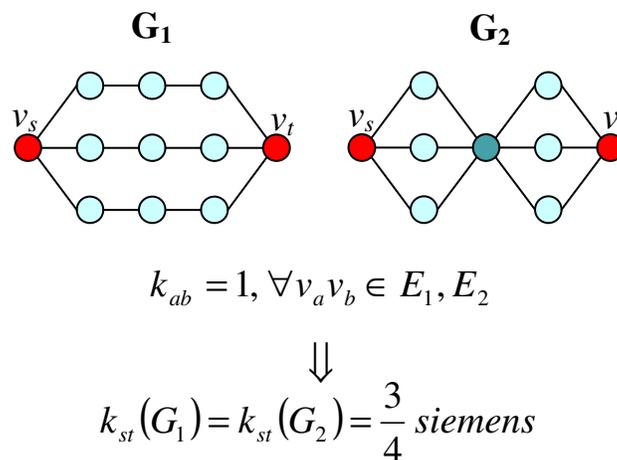
may be applied to a link or a node<sup>1</sup>. Standard methods of computing graph conductance expect the conductance values to be applied only to the links of the graph. Therefore, to explicitly incorporate nodes and their effects, we must make some modifications to our graph to set the stage for the implementation.

### 3.1 Conductance Model Considerations

As we have previously discussed, the graph conductance model is a promising candidate for a strength of association metric. However, there are three issues we must address before we can apply the model properly.

#### Single-Node Dependencies

There are certain properties of the conductance model that are inconsistent with our interpretation of the strength of association metric in a semantic graph context. One such property is that *single-node dependencies* may often be ignored by the model. By single-node dependency, we mean an intersection of two or more distinct paths at exactly one node. In Figure 3-1, we compare such a graph to its counterpart, which is composed of the same number of simple, i.e., internally disjoint, paths of the same length.



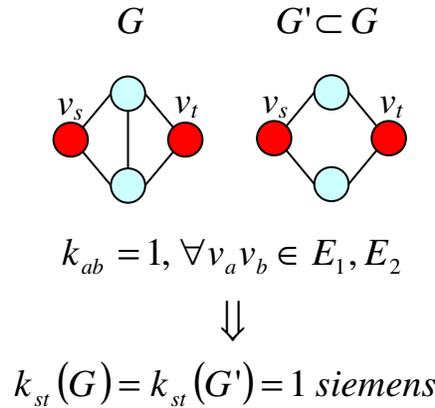
**Figure 3-1: Single node dependencies, as shown above, are often ignored in the conductance model. In the above graphs, all link conductances are fixed at 1.**

In the context of strength of association, however, we would like to see a lower graph conductance in the presence of single-node dependencies, as asserted in assumption (2) regarding a parallel collapse.

<sup>1</sup> Note that we exclude 0 from the range of  $C_{ab}$  and assume that data with zero confidence has been pared from the graph. This is not necessary from an electrical standpoint (zero confidence simply produces infinite resistance, preventing the flow of current along a pathway), but may be implementationally desirable to avoid potential divide-by-zero errors and to allow more efficient computation.

### Zero potential difference

There are some circumstances in which a link may have zero potential difference between its nodes. Such links carry no current and will have no effect on the graph conductance. Due to assumption (1) regarding strength of subgraphs, however, we expect such links to strengthen the graph, even if only to a small degree. Figure 3-2 shows a simple example, where the center vertical link of  $G$  has zero potential difference.



**Figure 3-2: The center vertical link in  $G$  has zero potential difference, and thus has no effect on the graph conductance. All link conductances are fixed at 1 in each graph. Note that in real world applications where confidences have been applied, zero potential differences may be rare.**

It is also interesting to note that from an electrical standpoint, single-node dependencies and zero potential difference produce the same result. Though they may be electrically identical, our solution to these issues, in the context of semantic graphs, must treat them quite differently.

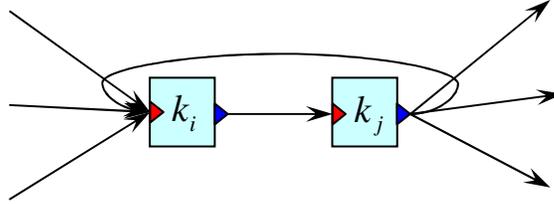
### Applying Confidence to Nodes

Perhaps the most important issue of the three listed here is that we would like to be able to apply confidence, ergo conductance, to nodes as well as links. Unfortunately, standard graph conductance computation algorithms expect nodes to act as merely connection points between the links. We have found this problem to be nontrivial in that graph configurations can become quite complex, giving rise to great difficulty in encapsulating the behavior of nodes relative to the links between them in terms of conductance.

### 3.2 Graph Modification

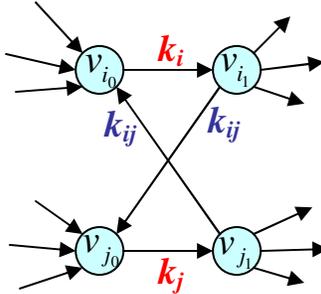
We begin by conceptualizing our nodes as “objects” in the graph, to which conductances can be applied, without yet attempting to precisely define their nature. We would like to force any current entering a node object to cross a resistance within the node before it can move on to another node object. Thus we assert that our node object has an entry point, where all incoming links must enter, and an exit point “on the other side”, where all outgoing links leave (See Figure 3-3).

Undirected links between node objects in the graph are represented through the use of two directed links which must go from exit to entry. Directed graphs and multigraphs may also be represented in this fashion, through the use of an appropriate number of directed links.



**Figure 3-3: Node objects  $v_i$  and  $v_j$  with conductances  $k_i$  and  $k_j$ , respectively. Red arrows denote entry points and blue arrows denote exit points for the node objects. Note that the undirected link between the nodes is represented here by two directed links, where each must go from exit to entry.**

For our implementation, we have chosen to represent node objects as a pair of dummy nodes connected by a link which carries the node conductance. Figure 3-4 illustrates the structure for two adjacent node objects,  $v_i$  and  $v_j$ . One dummy node acts as the entry point for the node object, and we refer to it as the *entry node* for the node object. The other node is the *exit node*.



**Figure 3-4: A node object represented as two dummy nodes connected by a link which carries the node conductance. Node object conductances are given in red, link conductances in blue. Note that the link belonging to the node object is the only link permitted to flow from entry node to exit node.**

As shown in Figure 3-4, links belonging to node objects are the only links in the graph “permitted” to flow from an entry node to an exit node. In general, we only convert *internal* nodes, i.e., all nodes other than the source node  $v_s$  and the terminal node  $v_t$ , into node objects. The source and terminal nodes each act as their own entry and exit points. As a consequence, a link connecting the source directly to the terminal node, for the sake of consistency, should become two links (multigraph). The absence of this second link, in contrast to the two links between other pairs of node objects, may tend to weaken graphs which already rely heavily upon the direct link between these nodes, potentially skewing graph rankings. Having made these modifications, we are prepared to compute the graph conductance.

### 3.3 Implementation

To compute the graph conductance, we will make use of *Kirchoff's Current Law* [2], which states that the sum of currents flowing into a node must equal the sum of currents flowing out. Mathematically, we represent this as

$$\sum_j \omega_{ij} = 0, \quad \forall v_i, \quad v_i v_j \in E. \quad (5)$$

It should be noted here that  $\omega_{ij} > 0$  if current flows from  $v_i$  to  $v_j$ , and  $\omega_{ij} < 0$  if it flows in the opposite direction. Now, by Ohm's Law (3) and (4), we can rewrite (5) as

$$\sum_j k_{ij} V_{ij} = 0, \quad \forall v_i, \quad v_i v_j \in E. \quad (6)$$

Since  $V_{ij} = V_i - V_j$  is the potential difference between  $v_i$  and  $v_j$ , we can solve (6) for  $V_i$  to obtain an expression for the potential at any node  $v_i$ .

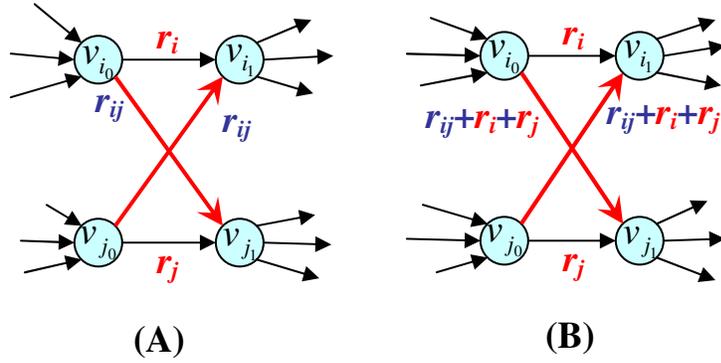
$$V_i = \frac{\sum_j k_{ij} V_j}{\sum_j k_{ij}}, \quad \forall v_i, \quad v_i v_j \in E. \quad (7)$$

The linear system of equations introduced in (7) can be solved using the Gauss-Seidel method [6], an iterative approach. The steps of the algorithm are given below.

- (i) Set  $V_s = 1, V_t = -1$ . Initialize  $V_i = 0$  for all internal nodes.
- (ii) For each internal node  $v_i$ , compute  $V_i = \frac{\sum_j k_{ij} V_j}{\sum_j k_{ij}}$ , where  $v_j$  is adjacent to  $v_i$ , and previous results are used as soon as they are available.
- (iii) Repeat (ii) until the change in potential is less than some tolerance.

When the above algorithm converges, we will have computed potentials for each node in the graph. The resulting potential differences indicate the direction of current flow on each link.

Recall that our graph modifications included directed links flowing in each direction between two node objects. It can be easily seen that the above algorithm ignores link direction in computing node potentials. Frequently, if not always, we will find that the resulting current flow along at least one of these links is flowing in the opposite direction than we intended (See Figure 3-5A). We can determine the direction of the current by examining the potentials at the end nodes of a link (recall that current flows from high to low potential).



**Figure 3-5: (A) Frequently, current flows in the opposite direction than intended (red links), bypassing the node object resistance. (B) We mitigate this situation by adding the node object resistances to such links.**

If either link between two node objects flows from an entry node to an exit node (i.e., the wrong direction), we can mitigate this situation by adding the node object resistances to the offending link resistance. In effect, we are “forcing” the current to flow across an equivalent resistance, since it does not follow the path we intended.

Once we have made the necessary adjustment to these link resistances, we run the Gauss-Seidel algorithm again (without reinitializing the internal node potentials) until it converges.

### **Computing a Final Result**

Having computed potentials on all of the nodes using the Gauss-Seidel algorithm, we still must compute the conductance. Ohm’s Law (3) tells us the graph conductance is given by

$$k_{st} = \frac{\omega_{st}}{V_{st}}. \quad (8)$$

Since  $V_{st} = 1 - (-1) = 2$ , we need only compute the total current flowing into the graph to determine the conductance. Due to Kirchoff’s Current Law (5), we know that the total current flowing into the graph at  $v_s$  equals the total current flowing out of  $v_s$ . Thus, by (3), (4), and (5),

$$\omega_{st} = \sum_j k_{sj} V_{sj} = \sum_j k_{sj} (1 - V_j), \quad v_s v_j \in E. \quad (9)$$

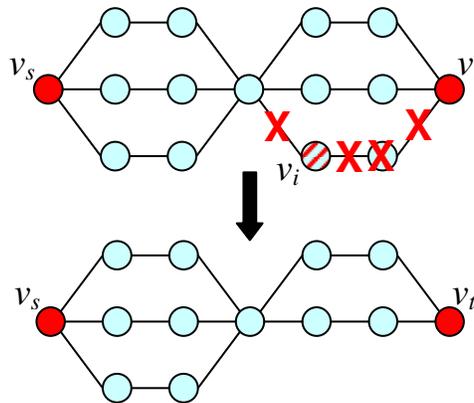
The graph conductance, given by (8), will be our strength of association measure,  $\Psi(s, t | G)$ .

## 4 Cutsets

In this section we discuss an appropriate method for determining a global stability measure for a graph. The foundation for this metric is the assumption that a graph with multiple redundancies will be more robust to breakage due to the random removal, or failure, of links or nodes. Our approach is to incrementally remove graph components (links or nodes) according to some probability of removal applied to each component at a given time  $\tau$ . We also incorporate a probability that no component fails at time  $\tau$ . We wish to determine the expected time  $E[\tau]$  at which the association “breaks”, i.e. there is no longer a path from  $v_s$  to  $v_t$  existing in the graph.

Due to the fact that some composite paths may share only a single node, the removal of links alone is insufficient to quantify the distinction between such paths and multiple simple paths. Therefore, we must also consider the potential removal of nodes in our procedure. We could implement this adjustment in the algorithm by converting the nodes to node objects as described in Section 3, but it is more straightforward to apply removal probabilities to the nodes as well as the links.

We also must consider the fact that multiple simple paths will be penalized for shorter length if we remove only one component at each stage of the algorithm. We correct for this bias by removing the entire *subpath* which depends upon the link removed (see Figure 4-1).



**Figure 4-1:** The removal of node  $v_i$  necessitates the removal of the other components in its subpath, as they are no longer involved in any path connecting  $v_s$  and  $v_t$ .

We loosely define *subpath* as referring to the set of components which, as a result of a given node or link removal, no longer exist on any path in the graph from  $v_s$  to  $v_t$ .

While minimal cutsets are widely discussed in the literature, determining the expected cutset size in a procedure such as this is not so straightforward. That the effect of removing a given component depends upon prior removals suggests a Markov process, which may be approached in different ways. We have explored two possible methods for

performing this computation – a Hidden Markov Model [7], which is discussed in detail in the Appendix, and simulation.

### 4.1 Simulation Approach

The simulation approach is relatively straightforward. At a given time  $\tau$ , each component has a probability of removal,  $p(i|\tau)$ , associated with it, and additionally, there is a probability  $p(\emptyset|\tau)$  that no component is removed. At each step we randomly choose a component (or no component at all) according to these probabilities. If a component is chosen, it is removed along with its subpath, and the graph is examined to determine whether a path from  $v_s$  to  $v_t$  remains. Each value of  $\tau$  where  $v_s$  and  $v_t$  are finally separated is stored, and, for  $n$  trials, the expected value of the number of steps required is computed in the usual way

$$\hat{E}(\tau) = \sum_{i=1}^n \hat{p}_i \tau_i, \quad \hat{p}_i = n_\tau / n, \quad (10)$$

where  $n_\tau$  is the number of failures that occurred at time  $\tau$ . The slightly more complex aspect to this algorithm is determining the appropriate removal probabilities at a given time  $\tau$ . These removal probabilities provide a good vehicle for incorporating confidence on the nodes and links. Assuming, as for the conductance algorithm, that confidence  $c_j \in (0,1]$  is defined and available, low-confidence components should have a higher probability of being removed, decreasing the global stability of the graph. One way to accomplish this is by first assigning unscaled likelihoods of removal at time  $\tau$  as shown in (11),

$$\lambda(\emptyset|\tau) = \min(\{c_i\}), \quad \lambda(i|\tau) = c_i^{-1}, \quad \forall i \quad (11)$$

where  $\lambda(\emptyset|\tau)$  is the likelihood that no component is removed at time  $\tau$ , and  $\lambda(i|\tau)$  represents the likelihood that component  $i$  is removed. Note that the likelihood for component removal is dependent on  $\tau$  only insofar as whether it exists in the graph at time  $\tau$ , i.e., confidence values are held constant over time. Given the likelihoods in (11), we compute the removal probabilities as shown in (12).

$$p(j|\tau) = \frac{\lambda(j|\tau)}{\sum_i \lambda(i|\tau) + \lambda(\emptyset|\tau)}, \quad j = \emptyset, \{i\} \quad (12)$$

A key property of this metric is that it will be bounded below by the maximum number of pairwise-internally-disjoint paths in the graph (Menger's Theorem) [4]. In terms of strength, this would be undesirable, since path influence on strength should approach zero as path length approaches infinity. However, in terms of path redundancy and robustness to breakage, we consider this property appropriate. Although very long paths are easily broken, they still provide a “safety net” of sorts for the association in question. We will discuss these and other considerations when we discuss our results in Section 7.

## **4.2 Hidden Markov Model**

We have also investigated an implementation for the cutset algorithm using a Hidden Markov Model (HMM) [7]. Although our favored algorithm is the simulation approach due to its simplicity, the HMM provides an interesting, if complex, alternative. In an HMM, we have a finite set of *states*, which are related to each other by *transition probabilities*, i.e., the probability that one state will transition to another through a sequence of events. Each state is associated with some observable feature or value, and these observable features are the only part of the event sequence visible to an external observer. This technique produces a precise result, in contrast to the simulation approach described in Section 4.1, and it is discussed in detail in the Appendix.

## 5 Betweenness

Newman and Girvan [5] proposed a metric called *betweenness*, which measures the intensity of “traffic” along a link in a graph. It is computed using the number of shortest paths in the graph which traverse a given link. Thus, a link with high betweenness plays a large part in holding a relationship together. We call this metric *criticality*. It is based on betweenness, and it measures the relative vulnerability of a link or node in the graph.

Betweenness is generally computed for the links in a graph<sup>2</sup>, so in order to apply the metric to both nodes and links, we transform the graph as described in Figure 3-4. In this case, we require the links to be directed.

### **Implementation**

The algorithm begins by selecting either the source node,  $v_s$ , or terminal node,  $v_t$ , as a root and computing betweenness for all links relative to that root node. Each node in the graph is initially assigned a *node weight* and *node distance* from the root node by implementing the algorithm below as presented by Newman and Girvan [5]:

1. The root node,  $v_r$ , is given a distance of  $d_r = 0$  and a weight of  $w_r = 1$ .
2. Each node  $v_i$  adjacent to  $v_r$  is given distance  $d_i = d_r + 1 = 1$ , and weight  $w_i = w_r = 1$ . Denote this set of nodes by  $I$ . Let  $J = \emptyset$
3. For each node  $v_j$  adjacent to one the above nodes in  $I$ , we do one of three things:
  - a. If  $v_j$  has not been assigned a distance, then it is assigned distance  $d_j = d_i + 1$  and weight  $w_j = w_i$ . Set  $J = J \cup \{j\}$ .
  - b. If  $v_j$  has already been assigned a distance and  $d_j = d_i + 1$ , then the weight of  $v_j$  is increased by 1, i.e.  $w_j \leftarrow w_j + w_i$ . Set  $J = J \cup \{j\}$ .
  - c. If  $v_j$  has already been assigned a distance and  $d_j < d_i + 1$ , do nothing.
4. Repeat from step 3, where  $I = J$ , until no nodes remain that have been assigned distances, but whose neighbors do not have assigned distances.

The node weight,  $w_i$ , assigned in the algorithm above provides the number of distinct shortest paths from  $v_r$  to the node  $v_i$ , while the node distance,  $d_i$ , is the length of those shortest paths. We found it useful in our implementation to construct a *shortest path tree* during the assignment of node weights which contains only those links belonging to at least one shortest path. Only these links will be assigned weights<sup>3</sup>. We can use the node

---

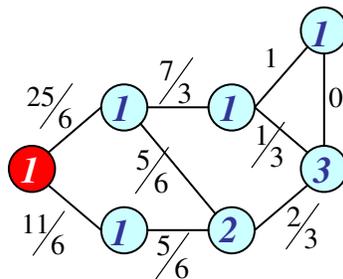
<sup>2</sup> Some versions of betweenness compute the betweenness of nodes in a similar fashion, but the algorithm is not usually applied simultaneously to both nodes and links.

<sup>3</sup> Note that links in the original graph which *do not* appear in the shortest path tree from the node weight algorithm will have a betweenness value of 0 (Figure 5-1).

weights and distances computed above, along with the shortest path tree, to compute the link weight (betweenness) for each link as presented in Newman and Girvan [5]:

1. Find every “leaf” node  $v_n$ , i.e., a node such that no shortest paths from  $v_r$  to other nodes pass through  $v_n$ . In a shortest path tree from  $v_r$ , all leaf nodes will qualify.
2. For each node  $v_i$  neighboring  $v_n$ , i.e. moving up in the shortest path tree, assign a score to the edge from  $v_i$  to  $v_n$  of  $w_i/w_n$ .
3. Now, starting with the edges that are farthest from the root node,  $v_r$ , continue the above process, working upward toward  $v_r$ . To the edge  $v_i v_j$  (with  $v_j$  further from  $v_r$  than  $v_i$ ), assign a score that is 1 plus the sum of the scores on the neighboring edges immediately below it in the shortest path tree (i.e., those flowing away from  $v_r$ ), all multiplied by  $w_i/w_j$ .
4. Repeat step 3 until the node  $v_r$  is reached.

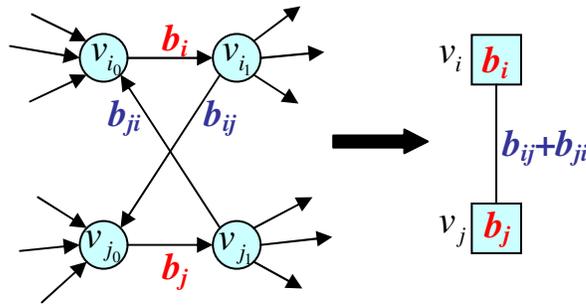
Figure 5-1 shows the results for a simple graph containing a link with zero betweenness.



**Figure 5-1: An example of betweenness from source node (in red), from Newman and Girvan. Values within the nodes are node weights.**

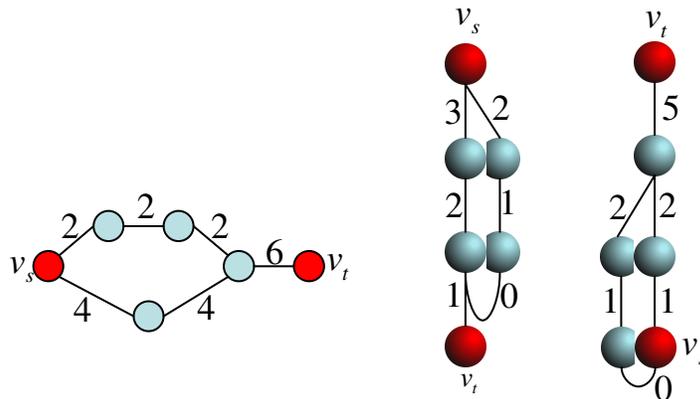
Newman and Girvan use their measure of betweenness to drive a graph decomposition algorithm, so they compute betweenness using each node as a root node, and the resulting betweenness values on each link are summed to obtain a final value. For our purposes, we are interested in characterizing the relationship between  $v_s$  and  $v_t$ . To that end, we implement the betweenness algorithm twice, first using  $v_s$  as the root node, and then using  $v_t$ . We add the resulting values to obtain a final measure of betweenness on the links.

Now, recall that in the transformed graph, each pair of connected node objects is linked by two directed links. The betweenness values which result on these directed links should be summed to obtain the true betweenness value for the original link (see Figure 5-2).



**Figure 5-2: Betweenness values on the directed links between two node objects should be summed to obtain the true value.**

As written, the above implementation produces values of betweenness that have meaning only within their graph, i.e., useful comparisons cannot be made between betweenness values arising from different graphs. However, we can mitigate this problem. To motivate this, let us consider a physical interpretation of the betweenness metric. We may imagine a graph as a network of spheres suspended by wires which connect them. Each of these wires is of equal length, and the spheres are of equal weight, say 1. If we suspend the network of wires from the sphere representing  $v_s$ , we represent the tension on each wire by the amount of weight it supports due to the spheres hanging below it. If multiple wires support the same set of spheres, we consider the weight equally distributed among the wires. An illustration of this concept is provided in Figure 5-3 for a simple, unmodified graph.



**Figure 5-3: Betweenness values represent the total weight (tension) on each link when the network is suspended from its end nodes.**

The resulting betweenness values represent the total amount of node weight stressing the links from both directions. Therefore, we can scale the betweenness values by the number of nodes in the network, so that each value represents the *proportion* of the total network weight supported by the link. This will allow us to compare the vulnerability of links from different graphs.

In a real world application of betweenness as a measure of criticality, there may be circumstances under which the component with maximum betweenness has extremely high confidence, i.e., there is little to no risk that the component will fail. Under such circumstances, then, the metric provides little help in highlighting weaknesses in the relationship. So we would like to incorporate, in some fashion, the confidence on the data in the graph. In order to draw attention to true weak points in the graph, we need to inflate the criticality measure for data with low confidence. Therefore we update the criticality metric,  $\kappa_{ab}$  on link  $v_a v_b$ , to account for confidence as indicated in (14),

$$\kappa_{ab} = \frac{b_{ab}}{c_{ab}(2N-2)}, \quad \forall v_a v_b, \quad c_{ab} \in (0,1] \quad (14)$$

where  $N$  is the number of nodes in the original, unmodified, graph, including  $v_s$  and  $v_t$ .<sup>4</sup>

---

<sup>4</sup> Recall that we modified the graph to use node objects, within which there are two dummy nodes. Thus the modified graph upon which the betweenness values are based has  $2N-2$  nodes.

## 6 Incorporating Source Document Confidence

All of the metrics presented in this paper can in some fashion incorporate data confidence to account for potentially false or unreliable information. While we cannot specifically address the impact of information extraction reliability and data fusion confidence (these depend upon the information system in use), we can provide an initial attempt at applying source document confidence to the previously defined metrics for the assessment of strength of association, global stability of a graph, and criticality of a node or link.

Suppose we have a graph,  $G$ , such that each link  $v_a v_b$  has associated with it a set of  $n_{ab}$  source documents  $\{d(i | a, b), i = 1, \dots, n_{ab}\}^{**}$ . Then the set of source documents associated with a node  $v_a$  in the graph will be given by

$$\{d(j | a)\}_{j=1}^{n_a} = \bigcup_{i=1}^{n_{ab}} \{d(i | a, b)\}, \quad \forall v_b \ni v_a v_b \in E. \quad (15)$$

For the purposes of this discussion, we assume that these source documents are independent with regard to confidence.<sup>5</sup> Let us denote the confidence on a source document  $d(k | a, b)$  by  $c_{kla,b}^* \in (0, 1]$ . We interpret confidence in general as the probability that the information in question is true, i.e.

$$\begin{aligned} c_{kla,b}^* &= P(d(k | a, b) \text{ true}) \\ c_{ab} &= P(v_a v_b \text{ true}) \end{aligned} \quad (16)$$

If at least one source document underlying a link in the graph is true, then the information encapsulated by the link is also true. Thus,

$$c_{ab} = P(\text{at least one } d(k | a, b) \text{ true}) = 1 - P(d(k | a, b) \text{ false } \forall k), \quad \forall a, b \quad (17)$$

and hence, under the assumption of independence,

$$c_{ab} = 1 - \prod_{i=1}^{n_{ab}} (1 - c_{ila,b}^*). \quad (18)$$

We may compute  $c_{ab}$  for all links in the graph, along with  $c_a$  for all nodes, so that these confidence values may be incorporated into the strength and reliability metrics presented here.

---

\*\* We only consider the source documents supporting the links, which represent relationships between entities; i.e., we assume that the presence of an entity in a document is insignificant unless a relationship is formed, or “discovered” in the document.

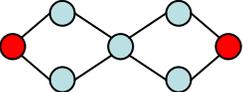
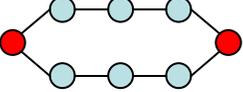
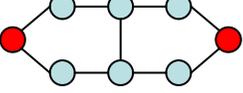
<sup>5</sup> The appropriateness of this assumption should be revisited in future efforts.

## 7 Results

### 7.1 Conductance Results

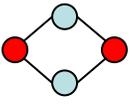
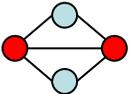
Performance evaluation for the conductance algorithm is based primarily upon adherence to the early assumptions we made regarding partial orders for strength of association. We tested more than forty small graphs with well-defined features, such as path length, multiple paths, single-node dependencies, symmetry-induced zero potential difference, etc., for comparison. In all cases, our assumptions were upheld. Standard conductance algorithms on unmodified graphs produced rank *groups* from these test cases, where as many as eight graphs exhibited the same conductance. Our modified conductance algorithm ranked all test cases consistently with standard conductance rank groups, and managed to produce rankings within these rank groups which were defensible using our basic ordering assumptions.

We examined the performance particularly closely for graphs with single-node dependencies and links with zero potential difference. Recall that a standard conductance algorithm on an unmodified graph does not differentiate between these two conditions, which is undesirable for our strength of association metric. The modified algorithm presented in this paper produces the results shown in Figure 7-1.

	<u>Modified</u>	<u>Standard</u>
	0.625	0.5
	0.63529	0.5
	0.63898	0.5

**Figure 7-1: Standard conductance measure on the displayed graphs is identical. Our modification allows the conductance to vary a small amount in these special circumstances. The single-node dependency is weaker; the link with zero potential difference makes the graph slightly stronger.**

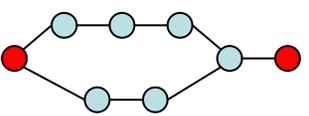
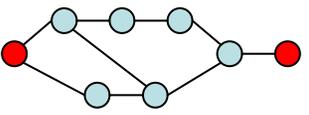
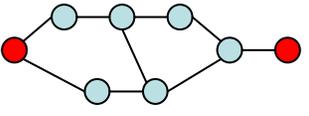
We have found another interesting property of this algorithm, which intuitively makes sense in regard to strength of association. The strength of a graph is, in some sense, the sum of its parts, i.e., the strength is the aggregate of the individual strengths for each independent (simple) path or set of composite paths (Figure 7-2). This is consistent with our intuitive belief that the existence of one path should in no way affect the strength of another path, so long as they do not intersect at any point in the graph.

	<u>Modified</u>	<u>Standard</u>
	<b>1.4286</b>	<b>1</b>
	<b>2</b>	<b>1</b>
	<b>3.4286</b>	<b>2</b>

**Figure 7-2: The strength of the third graph is the sum of the strengths on the simple paths comprising it. This holds true for both the standard and modified conductance algorithms. It should not be surprising that each path of length two in the first graph has modified conductance of 0.7143.**

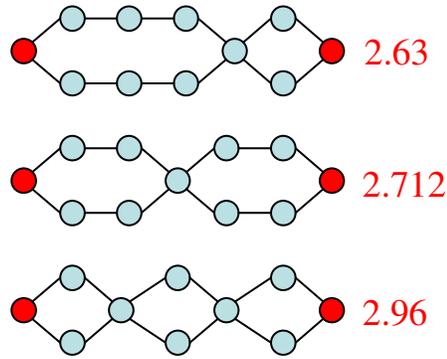
We also noticed that the effect of a direct link between  $v_s$  and  $v_t$  is boosted in the modified conductance algorithm, due to the fact that these nodes are not transformed in the graph modification. The trivial graph consisting of this link alone has its conductance doubled in the modified algorithm due to our assertion in Section 3-2 that consistency in the graph modification is a necessary requirement. While this behavior was not explicitly sought in the design of the modified conductance algorithm, it is not necessarily undesirable, considering that a direct link between  $v_s$  and  $v_t$  is the strongest possible link or simple path in any graph. Thus, graphs containing this direct link will frequently be considered stronger than other graphs in which it is lacking.

Figure 7-3 presents a graph to which a link has been added, but in this case we have more than one possible configuration for the new link. As should be expected, the orientation of this link does make a difference in the strength metric.

	<u>Modified</u>	<u>Standard</u>
	<b>0.47134</b>	<b>0.36842</b>
	<b>0.50977</b>	<b>0.39583</b>
	<b>0.4778</b>	<b>0.37255</b>

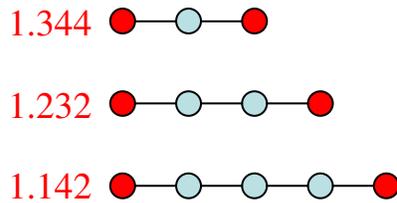
**Figure 7-3: The second and third graphs are stronger than the first due to the addition of a link. The orientation of this link is significant - in the second graph, the link creates a new path of length 4, a more significant improvement than in the third graph.**



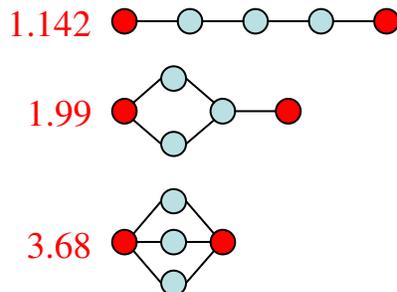


**Figure 7-5: The cutset algorithm favors single node dependencies. Note that the first graph is considered most vulnerable due to the number of components which are dependently linked, i.e., the longer subpaths in the graph are more vulnerable.**

As should be expected, longer paths and subpaths are more likely to be broken by random failure (Figure 7-6). However, as mentioned in the discussion of the algorithm, the cutset value is bounded below by the maximum number of pairwise-internally-disjoint paths in the graph. Theoretically then, the addition of an independent path of infinite length, though certain to fail, will add 1 to the cutset value. Hence, we conjecture that this algorithm is most useful for comparing the configurations of graphs of similar size (Figure 7-7). When this recommendation is followed, the cutset value will depend upon topological configuration alone rather than confounding it with influences due to size.

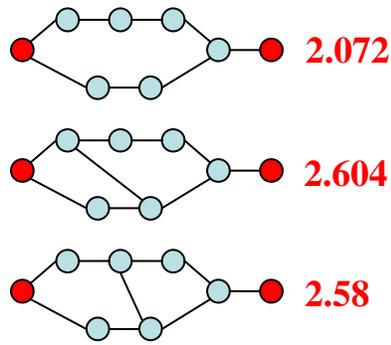


**Figure 7-6: Longer paths are more susceptible to failure. Menger's Theorem implies that cutset value approaches one as a single simple path length increases without bound.**



**Figure 7-7: This metric can compare configuration vulnerability between graphs of similar size (here, five nodes) in the absence of multigraphs.**

We must make note of the fact, however, that results from this algorithm should be examined with care. When implementing the algorithm through simulation, there will be some variability in the result we obtain. Therefore in comparing two or more cutset values, it is wise to simultaneously consider the configuration differences between the graphs in question. For example, in Figure 7-8, the topmost graph is clearly less stable than the two below it, due to the absence of the center link. So the difference in cutset values is quite likely to be significant and representative of a true disparity between the graphs. However, the second and third graphs are similar enough in configuration that the difference in their cutset values may be simply an artifact of the simulation approach to computing the measure. In such cases, if a more precise comparison is required, the HMM or some other comparable approach may prove to be a more appropriate computational method.



**Figure 7-8:** Due to configuration similarity in the second and third graphs, the difference between their cutset values may be strictly due to variability from the simulation approach.

### 7.3 Betweenness (Criticality) Results

We have previously described criticality as a component vulnerability measure. In practice, this metric can provide an analyst with a target for further investigation, or perhaps indicate the most reliable relationship between two nodes of interest. We can see an example of this in Figure 7-9, where we have imposed confidence values on the nodes and links and indicated the resulting criticality values.

The highest criticality values (shown in red) indicate the most prominent obstacles in making this a reliable relationship. Interpretation of the criticality metric is really dependent upon the needs of an analyst. The critical value may indicate pieces of information which should be initially targeted for further research or investigation, in order to gain greater confidence in the association as a whole. There may also be circumstances where only certain pathways contain solely low-criticality components, so that only these paths should be used as a basis for determining the validity of a relationship.

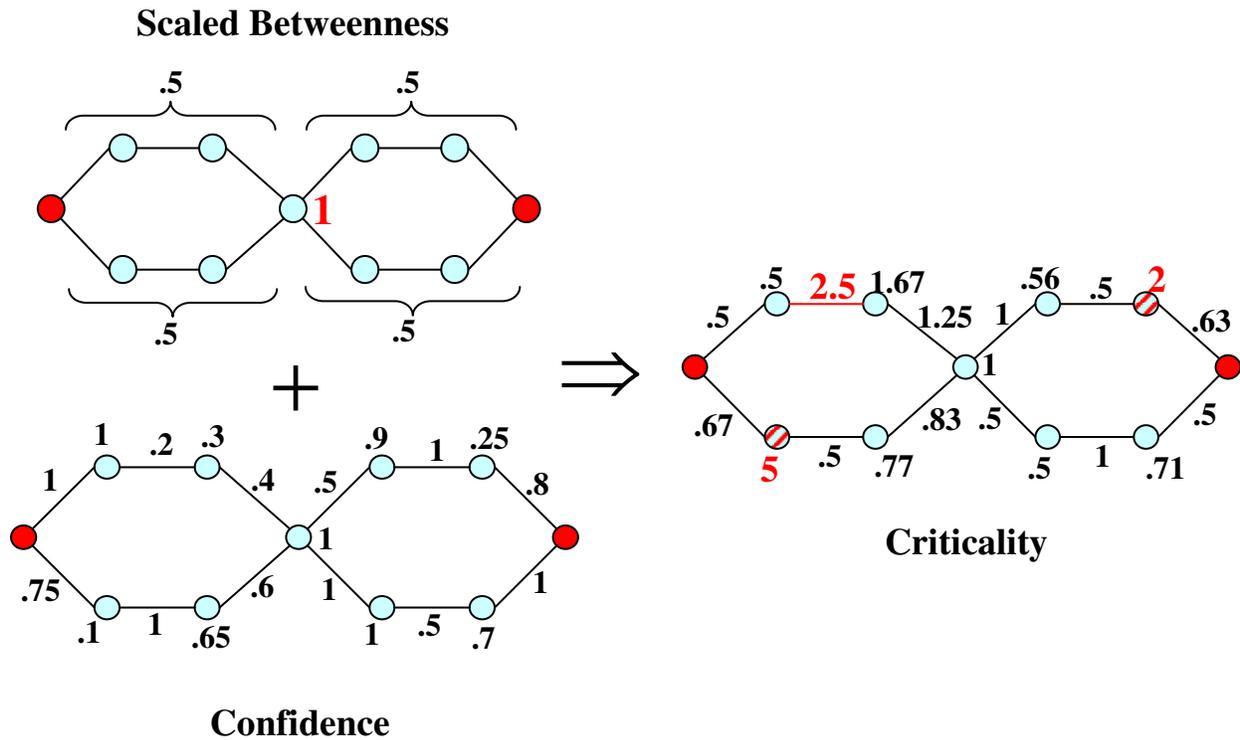


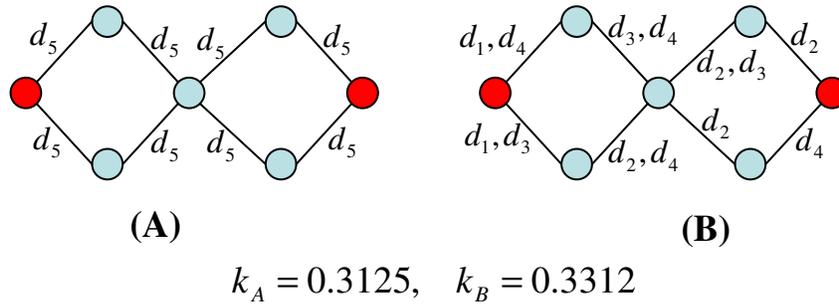
Figure 7-9: Criticality values for a sample graph. Highest critical values shown in red.

### 7.4 Source Document Confidence Results

We applied source document confidences to the graph shown in Figure 7-10, where the document confidences are given by  $c_1^* = 0.3$ ,  $c_2^* = 0.45$ ,  $c_3^* = 0.1$ ,  $c_4^* = 0.4$ ,  $c_5^* = 0.5$ , respectively, for the document set  $\{d_1, d_2, d_3, d_4, d_5\}$ . We have assigned these hypothetical source documents to links so that graph (A) is supported by only one document,  $d_5$ , and the links in graph (B) are supported by the remaining four documents.

The strength results for (A) and (B) are  $k_A = 0.3125$  and  $k_B = 0.3312$ . It is interesting to note that graph (A), though supported by a document with higher confidence, ultimately has a lower strength. Based upon this metric, additional document support cannot make a relationship less likely to be true, no matter how low the document confidence might be. Thus, each new source document which corroborates a relationship improves the probability that the relationship is true. We make no assertion that this metric is ideal; indeed, we hypothesize that documents are likely not independent with regard to confidence, which must be taken into consideration. It is, however, an encouraging start to developing an effective confidence metric based upon information source reliability,

which may ultimately incorporate a higher-order model to handle document dependencies.



**Figure 7-10: The conductance results for a graph incorporating source document confidence. (A) is supported by a single document. (B) is supported by multiple documents with lower confidence.**

## 8 Applying the Metrics to Real World Semantic Graphs

Many information systems which use semantic graphs to organize and interpret data may potentially employ graphs of massive size, some of which may exceed  $10^9$  nodes. Algorithms run on graphs of this size can easily become impractical, if not impossible, to use in a realistic time-frame, particularly for applications which require near real-time decision-making. Even for graphs of a more manageable size, metrics such as the conductance-based strength, which can theoretically achieve arbitrarily large values, may become less useful if neighborhoods of interest grow too large. Such neighborhoods may tend to obscure the true strength of the relationship between two nodes of interest.

For these reasons, we recommend that the metrics presented here be run on peeled communities. The community decomposition algorithm described by Newman and Girvan [5] attempts to partition the graph into clusters of closely-related nodes. Its stopping criterion, called modularity, is designed to seek the “best” partition so as to minimize the breakage of tight clusters. This implies that not only will communities be smaller and more manageable than the semantic graph as a whole, but each community should contain the core of the relationship between any two of its member nodes. In addition, the community decomposition algorithm is hierarchical, which allows the decomposition to be “rewound” in some sense to obtain larger neighborhoods, if necessary, while remaining faithful to the modularity criterion.

We also suggest that peeling the community once a pair of nodes has been selected for analysis is advantageous. Since tendrils (not including the two nodes of interest) do not lie on any path between these nodes, they do not contribute to the strength of the relationship, and they decrease the efficiency of the analyses.

There are preprocessing options which may also be considered in an analysis environment using these tools. For example, one might wish to filter the graph of semantically undesirable information [1] prior to performing the analysis, ensuring that the data upon which the relationship is based is significant, or useful. The removal of data with zero confidence is also advisable, given that such data cannot assist (and in fact, may complicate) implementation, the interpretation of results and the decision-making process.

There may also be ways of improving the algorithms in terms of efficiency through preprocessing or other means. For example, the conductance algorithm may be improved by computing conductances on serial connections beforehand. In the case of the cutset algorithm, the HMM implementation discussed in the Appendix may be improved by determining the minimum and maximum possible steps to break the graph.

## **9 Conclusion**

Our exploration of strength of association metrics has merely scratched the surface of an extensive range of possible methods to exploit the information stored in a semantic graph. As for our metrics in particular, there may be many ways to build on them in order to provide the greatest possible benefit to programs in need of inference techniques for semantic graphs. As the use of semantic graphs becomes more widespread, the need for metrics such as these will become paramount. Perhaps those presented here will provide a foundation for the development of increasingly more efficient and valuable algorithms for knowledge discovery based upon semantic graphs.

Finally, we must note that these metrics and results are the product of a partially funded effort, and as a consequence may be inconsistent in some ways with the initial proposal and statement of work.

## **Acknowledgements**

We would like to thank Dr. John Nitao, Ph.D. for his helpful suggestions and insight.

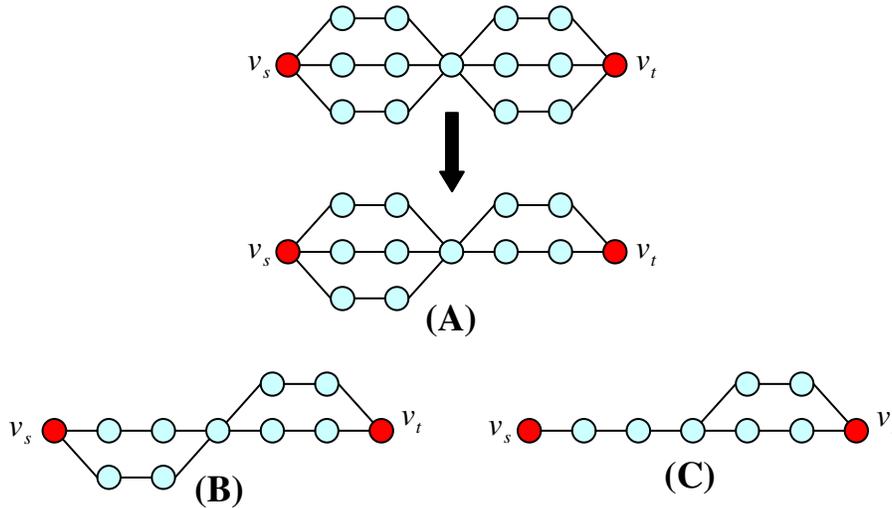
## References

- [1] Barthélemy, Marc, Edmond Chow, and Tina Eliassi-Rad. Knowledge Representation Issues in Semantic Graphs for Relationship Detection. Technical Report, Lawrence Livermore National Laboratory.
- [2] Bollobás, Béla. Modern Graph Theory. New York: Springer-Verlag New York, Inc., 1998.
- [3] Lessons in Electric Circuits. Tony R. Kuphaldt. 2000-2005. ibiblio. Sept. 2005. <<http://www.ibiblio.org/obp/electricCircuits/>>
- [4] Menger's Theorems and Max-Flow-Min-Cut. S.C.Locke. July, 1996. Sept. 2005. <<http://www.math.fau.edu/locke/Menger.htm>>
- [5] Newman, M.E.J. and M. Girvan. Finding and evaluating Community Structure in Networks. *Phys. Rev. E* **69**, 026113 (2004).
- [6] Nitao, John J. Personal Correspondence. 01 Sept. 2005.
- [7] Wikipedia, The Free Encyclopedia. Sept. 2005. <<http://en.wikipedia.org/>>

## Appendix – HMM

We investigated an implementation for the cutset algorithm using a Hidden Markov Model (HMM). Though our favored algorithm is the simulation approach due to its simplicity, the HMM provides an interesting, if complex, alternative. In an HMM, we have a finite set of *states*, which are related to each other by *transition probabilities*, i.e., the probability that one state will transition to another through a sequence of events. Each state is associated with some observable feature or value, and these observable features are the only part of the event sequence visible to an external observer.

Our graph is composed of some combination of distinct paths, some or all of which may overlap to some degree, between two nodes. It can be easily shown that the removal of a node or link will remove *at least one* of these distinct paths. Thus, the maximum number of steps that can be taken to break the association is no greater than the number of distinct paths. For our implementation of the HMM, we allow each unique combination of distinct paths to be a hidden state (Figure A).



**Figure A:** The original graph contains nine distinct paths. (A), (B), and (C) are hidden states with three, five, and seven distinct paths removed, respectively, from the larger graph. Similarly, we would provide hidden states for other possible configurations, along with the trivial states (all paths and no paths).

There will be a maximum of  $\|H\| = \sum_{i=0}^k \binom{k}{i} = 2^k$  hidden states, where  $k$  is the number of distinct paths, and each state has an observable value of 0 or 1 indicating failure (failing to break the association between  $v_s$  and  $v_t$ ) or success, respectively. We must explicitly determine the transition probabilities between each pair of states, as well as the probability of the observable value for each state. The observable value is easy to determine – exactly one hidden state, the one with all distinct paths removed, will have a status of 1. Since there are only two observable states, our observation probability matrix will be  $\theta_{\|H\| \times 2}$  where  $\theta_{ij}$  is the probability that hidden state  $H_i$  exhibits the observed state

$j, j = 0$  or  $1$ . Transition probabilities are more complex to compute. Suppose  $H_1$  and  $H_2$  are hidden states, where  $h_k$  represents the set of distinct paths remaining in the state  $H_k$ , and the transition probability between states is given by  $T$ . Then

$$T(H_1, H_2) = \begin{cases} 0 & \text{if } h_2 \not\subset h_1 \\ \sum_x p(x | H_1) & \text{otherwise} \end{cases}, \forall x \ni \begin{cases} h_1^i \in h_1 - h_2 \Rightarrow x \in h_1^i \\ h_1^i \in h_2 \Rightarrow x \notin h_1^i \end{cases}, \quad (19)$$

$$T(H_i, H_i) = p(\emptyset | H_i), \quad i = 1, \dots, \|H\|$$

where  $x$  represents components present in  $H_1$ ,  $p(x | H_1)$  is the removal probability of  $x$  as defined in Section 4.1, and  $h_1^i$  is the  $i^{\text{th}}$  path in  $h_1$ . In essence, we sum the removal probabilities of all components in  $H_1$  which could have given rise to  $H_2$  when removed, i.e., all components *not* in  $H_2$  which belong to *every* path removed from  $H_1$ .

Suppose we have specified the transition probability matrix,  $T$ , along with a probability matrix,  $\theta$ . We set the initial state probability  $\pi = \{\pi_j\}, j = 1, \dots, \|H\|$ , where, for our purposes,  $\pi_i = 1$  if the  $i^{\text{th}}$  state is the original graph, and 0 otherwise. We use this HMM, given by  $\lambda = (T, \theta, \pi)$ , to determine the probability of observing the status sequence  $O = \{o_1, o_2, \dots, o_{Q-1}, o_Q\} = \{0, 0, \dots, 0, 1\}$ , where  $Q$  is the number of states required to achieve “success”. We do that by using the *recursive* algorithm given below:

- Let  $\alpha_1(j) = \pi_j \theta_{j o_1}, j = 1, \dots, \|H\|$
- Recursively,  $\alpha_{q+1}(j) = \theta_{j o_{q+1}} \sum_{i=1}^{\|H\|} \alpha_q(i) T_{ij}, j = 1, \dots, \|H\|, q = 1, \dots, Q-1$
- Then  $P(O | \lambda) = \sum_{i=1}^{\|H\|} \alpha_Q(i)$

In the above algorithm,  $\alpha_\tau(j) = P(\text{state } H_j | \text{time } \tau)$ , and since this probability depends only upon the previous state, we can define  $\alpha_\tau(j)$  recursively. The complexity of this algorithm is  $\|H\|^2 Q$ , which is linear with respect to the length of the observation sequence, and we run the HMM for  $Q = 2, \dots, k$ , where  $k$  is the number of distinct paths in the original graph<sup>6</sup>. We can then use (10) to compute the expected value for  $Q$ .

<sup>6</sup> In reality,  $k$  need only be as large as the number of distinct path memberships of the components in the graph. For example, in Figure 10,  $k < 6$  is appropriate, since any five component removals will break the association. Computing the upper (and lower) bound for  $k$  may significantly decrease runtime for many graphs.