The Weighted Integrity Problem is Polynomial for Interval Graphs

Sibabrata Ray
Assistant Professor
Dept. of Computer Science
University of Alabama
Tuscaloosa, AL 35487
USA

Rajgopal Kannan
Assistant Professor
Dept. of Computer Science
Louisiana State University
Baton Rouge, LA 70803
USA

Danyang Zhang
Assistant Professor
Communications Technology
York College
City University of New York
Jamaica, NY 11451 USA

Hong Jiang
Professor
Computer Science and Engineering
University of Nebraska—Lincoln
Lincoln, NE 68588
USA

Keywords: *graph algorithms, interval graph, computational complexity, weighted integrity problem, graph vulnerability*

Abstract

Network reliability is an important issue in the area of distributed computing. Most of the early work in this area takes a probabilistic approach to the problem. However, sometimes it is important to incorporate subjective reliability estimates into the measure. To serve this goal, we propose the use of the weighted integrity, a measure of graph vulnerability. The weighted integrity problem is known to be NP-complete for most of the common network topologies including tree, mesh, hypercube etc. It is known to be NP-complete even for most perfect graphs, including comparability graphs and chordal graphs. However, the computational complexity of the problem is not known for one class of perfect graphs, namely, co-comparability graphs. In this paper we give a polynomial time algorithm to compute the weighted integrity of interval graphs, a subclass of co-comparability graphs.

ARS COMBINATORIA 79(2006), pp. 77-95

1 Introduction

The advent of affordable but powerful workstations and the improved network cost-performance ratio have meant that much more powerful computing systems can now be constructed by interconnecting a large number of such units in a distributed working environment, Of paramount importance in the design and maintenance of such system is the knowledge of and the ability to maintain a certain level of sustainable computational power. Thus the study of system reliability in general and network reliability in particular is critical to achieving performance goals. Previous work in this area has been mostly on a probabilistic basis. However, sometimes it is important to take subjective reliability estimates into consideration. Among the relevant issue of importance we are particularly interested in one of vulnerabilities. That is, in an unfriendly external environment, how vulnerable is such a distributed system to certain external destruction and how much computing power can be sustained in the face of destruction. A graph theoretic approach is taken in addressing this problem in this paper.

The concept of network vulnerability is motivated by the design and analysis of networks under a hostile environment. Several graph theoretic models under various assumptions have been proposed for the study and assessment of network vulnerability. Graph integrity, introduced by Barefoot et al. [1, 2], is one of these models that has received wide attention [6, 7, 11, 14]. Barefoot et al. studied two measures of network vulnerability, the integrity and the edge integrity of a graph. Bagga et al. has introduced a similar measure called pure edge integrity [4].

In the integrity model, the basic assumption is that some intelligent enemy is trying to disrupt the network by destroying its elements. The cost on his part is measured by the number of elements he would destroy, and his success in incapacitating the network is measured by the order (i.e. number of nodes) of the largest connected component in the remaining network. The enemy of course wants both to be small. Therefore, the minimum attainable sum of these two quantities is considered as a measure of vulnerability of the network. This measure is called graph integrity.

The integrity *I(G)* of a graph *G* is defined as

$$I(G) = \min_{S \subseteq V(G)} \{|S| + m(G-S)\},$$

where $m(G-S)$ denotes the order (the number of vertices) of a largest component of $G-S$. The edge-integrity $I'(G)$ of a graph G is defined as

$$I'(G) = \min_{S \subseteq E(G)} \{|S| + m(G-S)\}.$$

Both *I(G)* and $I'(G)$ turn out to have interesting properties.

The above definitions have led to a number of interesting results. It is to be noted that all nodes are of equal importance in determining the integrity of a

graph. In a distributed computing system, however, it is usually the case that different components of the system assume different importance, by virtue of difference in computing power, cost, jobs being executed, protection mechanism, etc. Therefore, assigning equal importance to all nodes is neither desirable nor realistic. In this paper, we develop a weighted system. This gives rise to the following definition of weighted integrity.

Definition 1.1 *The weighted integrity of a graph* $G=(V,E)$ *is defined as*

$$I_\omega(G)=\min_{X\subseteq V}\{\omega(X)+m_\omega(G-X)\}$$

where $\omega: V \to R \geq 0$ *is vertex weight function.* $m_\omega(G-X)$ *is the maximum sum of the vertex weights of the connected components of* $G-X$. *Note that* $\omega(X)=\sum_{v\in X}\omega(v)$.

Clearly, this weighted model reduces to the original integrity model when all nodes have equal weight. While the weighted integrity problem in its generality is NP-complete, as shown in the next section, we have identified a particular class of graphs, namely the interval graphs, whose integrity can be found in polynomial time. As can be seen, the proposed model, the first of its type to the best of our knowledge, is of theoretical as well as practical significance.

The rest of this paper is organized as follows. In the next section we present a brief literature survey about weighted integrity problem. A polynomial time algorithm for computing the weighted integrity of interval graphs is described in section 3. Section 4 presents a subroutine used in section 3. Finally, concluding remarks are given in section 5.

2. Background and NP-Completeness of the Problem

A number of researchers have investigated various integrity related problems [6, 7, 13]. It is known that the integrity problem is NP-complete on planar graphs [7]. However, the problem is known to be polynomial on several common families of graphs [6].

It can be easily seen that integrity is monotonic under subgraph containment (if H is a subgraph of G, then the integrity of H is less than or equal to that of G). Further, the integrity is always bounded by the edge integrity [6]. Among the connected graphs of given order, the star graph has the maximum possible edge integrity and the minimum possible integrity [8].

The common graphs on which the explicit expression for the integrity is known include complete graphs, stars, paths, cycles, comets, complete bipartite graphs, complete multipartite graphs, etc. Though there exists an explicit expression for the edge integrity of hypercube, computing the integrity of hypercube is an open problem [6, 2, 5].

The relation between edge integrity, integrity and other graph parameters is

well investigated [6, 14, 1, 13]. There are results available connecting integrity, edge integrity, maximum degree, size of a graph, connectivity etc.

Some extremal graph theoretic results are also known about integrity. Results about extremal graphs with maximum edge integrity are well known [6].

However, very few algorithmic and complexity theoretic results are known about integrity. Bagga et al. gave a nice polynomial time algorithm for computing the edge integrity of a tree [9]. That algorithm can be extended very easily to compute the integrity of a tree. To the best of our knowledge, no other algorithmic result is known about integrity. Interested researchers may refer to the survey paper [6] for more information on integrity.

The concept of the weighted integrity was introduced by Ray and Deogun [3]. They gave a set of sufficient conditions for the weighted integrity problem to be NP-complete on a class of graphs. The results in [3] are relevant to our work and we describe those here briefly. To state the results we need the following definition.

Definition 2.1 Dense set. *Let $n_0 < n_1 < \cdots$ be an infinite sorted list of positive integers. The set $\{n_i \mid i = 0,1,\ldots\}$ is called dense if there exists a positive real α such that $n_{i+1} \le n_i^{\alpha}$ for all i.*

The main result in [3] is the following theorem.

Theorem 2.1 *Let S be an infinite class of graphs such that $\{|G| : G \in S\}$ is a dense set. If for all $G \in S$, there exists an induced subgraph H' of G satisfying the following conditions, then the weighted (vertex) integrity problem is NP-complete on S.*

1. *There exists an independent set $\{v_0,\cdots,v_k\}$ of H' such that $\deg_{H'}(v_i) = 1$ for $i = 0,\ldots,k$ and $k \ge |G|^{\alpha}$.*
2. *For each v_i, there exists a distinct $u_i \in H'$ adjacent to v_i for $i = 0,\ldots,k$.*
3. *All v_i in the set $\{v_i \mid i \notin I\}$ belong to the same connected component of $\{v_j \mid j \in I\}$, for all $I \subset \{0,\ldots,k\}$.*

The proof of the theorem is by reduction to the 0-1 knapsack problem. Theorem 2.1 may be described informally as follows. Let *S* be an infinite class of graphs. If every $G \in S$ has a "big enough" Type 1 induced subgraph (see Figure 1), then the weighted integrity problem is NP-complete on the class *S*.

From Theorem 2.1 it may be seen that the weighted integrity problem is NP-complete for most of the common graphs including trees, meshes, hypercubes, bipartite graphs, series parallel graphs, regular graphs etc. It may be noted that

Theorem 2.1 is not applicable to the class of cocomparability graphs (refer to Definition 2.2). This is simply because cocomparability graph can not have a Type 1 induced subgraph with more than two leaves. Let $G=\{V,E\}$ be a cocomparability graph where $V=\{v_1,\ldots,v_n\}$. Further,

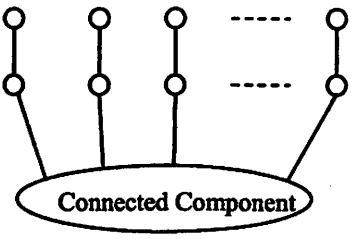


Figure 1. Type 1 induced subgraph

let us assume that $v_1,\ldots,v_n$ is a linear extension of some orientation of the comparability graph $\overline{G}=\{V,\overline{E}\}$. Note that, if $(v_i,v_j)\in E$ then either $(v_i,v_k)\in E$ or $(v_k,v_j)\in E$ for $k=i+1,\ldots,j-1$. Choose p_1 and p_2 such that both v_{p_1} and v_{p_2} are not adjacent to v_i and $p_1<i$, $p_2>i$. If $\{v_i\}\cup N(v_i)$ is removed from G, v_{p_1} and v_{p_2} will belong to different connected components. If possible, let G have a Type 1 induced subgraph with more than three leaves. Let v_{i_1}, v_{i_2}, v_{i_3} be three of the leaves. Without loss of generality, $i_1<i_2<i_3$. Now, if $\{v_{i_2}\}\cup N(v_{i_2})$ is removed, v_{i_1} and v_{i_3} will belong to different connected components. But that cannot happen for a Type 1 subgraph.

Definition 2.2 (Cocomparability graph) *Cocomparability graph is the complement graph of the comparability graph. A simple graph G is a comparability graph if it has a transitive orietation, which is an orientation such that if x→ y and y→z, then also x→z, where x→ y means there is an edge from x to y.*

3. Algorithm

The NP-completeness result described in the previous section shows that the weighted integrity problem is NP-complete for a simple and very basic graph structure. That leads to the NP-completeness of the problem for many common classes of graphs including trees, meshes and hypercubes. However, the computational complexity issue for cocomparability graphs remains unresolved. To our knowledge, the class of cocomparability graphs is the only well-known

class of graph such that no graph in the class contains a large type 1 subgraph as an induced subgraph. In this section we describe a polynomial time algorithm for solving the weighted integrity problem for interval graphs. It is known that the interval graphs constitute a subclass of cocomparability graphs. More precisely, interval graphs are chordal cocomparability graphs [10].

Some definitions are given below in preparation for our algorithm.

Definition 3.1 (Achieving (vertex) cut) *For a graph G and weight function ω, $X \subseteq V(G)$ is called an achieving (vertex) cut of G if $I_\omega(G) = \omega(X) + m_\omega(G - X)$.*

In other words, an achieving (vertex) cut is a set of vertices for which the weighted integrity measure is obtained. From now on, an achieving (vertex) cut will be called an *achieving cut* throughout the rest of the paper.

Definition 3.2 (Achieving component) *For a graph G and achieving cut X, an achieving component is a connected component of $G - X$ with maximum weight.*

Therefore, if G_1 is an achieving component for the achieving cut X, the sum of the vertex weights of G_1 is $m_\omega(G - X)$. Note that, neither an achieving cut nor an achieving component needs to be unique. An example of an achieving cute and an achieving component is shown in figure 2 where $\{v\}$ is the achieving cut and $\{u\}$ is the achieving component.

The number of possible vertex cuts for a graph can be exponentially large. Therefore, no enumerative scheme could possibly lead to a polynomial time algorithm. However, for an interval graph one need not consider all vertex cuts exhaustively to compute the weighted edge integrity. The following definitions and lemmas help us to reduce the size of the search space.

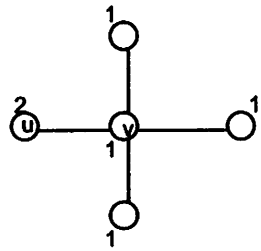


Figure 2. An example of an achieving cut and an achieving component

Definition 3.3 (Strongly minimal (vertex) cut) *For a graph G, $X \subseteq V(G)$ is called a strongly minimal (vertex) cut of G if for every $Y \subset X$, the number of*

connected components in $G-X$ is strictly more than the number of connected components in $G-Y$.

Note that the definition of strongly minimal (vertex) cut differs from the standard definition of minimal (vertex) cut. A subset of a strongly minimal (vertex) cut may still remain a vertex cut, but the number of connected components is less. From now on, a strongly minimal (vertex) cut will be referred to as a strong cut.

Lemma 3.1 *For any graph G there is an achieving cut $X \subseteq V(G)$ such that X is a strong cut.*

Proof. Let Y be an achieving cut of the graph G that is not a strong cut, Let $u \in Y$ be a vertex such that the number of connected components in $G-Y$ is same as the number of connected components in $G-Y+\{u\}$. Let $Y_1 = Y-\{u\}$. Now, $m_\omega(G-Y_1) \le m_\omega(G-Y)+\omega(u)$. Moreover, $\omega(Y_1)+\omega(u)=\omega(Y)$. Therefore, $\omega(Y)+m_\omega(G-Y) \ge \omega(Y_1)+m_\omega(G-Y_1)$.

If Y_1 is a strong cut, then the theorem is proved. If not, the same construction may be repeated on Y_1 until a strong cut is obtained. Obtaining a strong cut is guaranteed because any vertex cut consisting of one vertex is a strong cut.

Definition 3.4 (Interval Graphs) *Given intervals, $[a_i,b_i]$, $i=1,\ldots,n$, an intersection graph $G=(V,E)$ is constructed with $V=\{1,\ldots,n\}$ and $E=\{(i,j)|[a_i,b_i]\cap[a_j,b_j]\neq\phi\}$. An undirected graph is an interval graph if it is isomorphic to some intersection graph constructed from the intervals following the procedure described.*

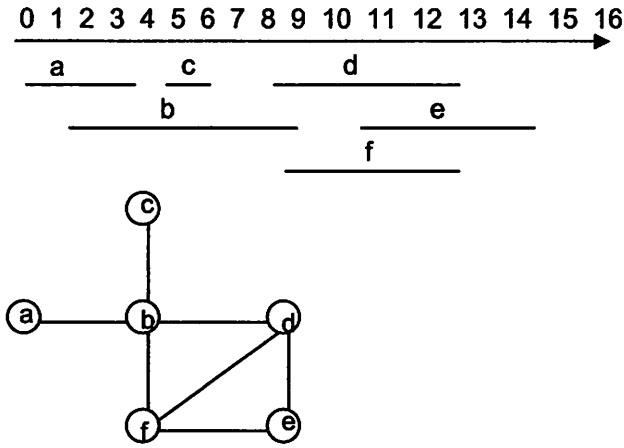
We mentioned earlier that the class of interval graphs is precisely the class of chordal cocomparability graphs. For the rest of the paper we shall adopt the following notation. Given an interval graph $G=(V,E)$ where $V=\{v_1,\ldots,v_n\}$, and G is isomorphic to the intersection graph of the intervals $[a_i,b_i]$ for $i=1,\ldots,n$ The interval $[a_i,b_i]$ will be called the interval attached to the vertex v_i for $i=1,\ldots,n$ and a_i will be called the left end point and b_i the right end point. Further, we shall assume that no two of these intervals have same end points, i.e., $a_i \neq a_j$, $b_i \neq b_j$ for $i \neq j$ and $a_i \neq b_j$ for all i,j. For a detailed discussion about interval graphs, the interested reader may refer to [10].

Given an interval graph $G=(V,E)$, consider a point x on the part of the real line covered by the intervals attached to the vertices of G. Let $C(x)=\{v_i|x\in[a_i,b_i]\}$. Obviously, if $\min_i b_i < x < \max_i a_i$, then $C(x)$ defines a

vertex cut of G. The following definition and lemma establish a relation between strong cuts and $C(x)$'s for an interval graph.

Definition 3.5 (Minimal local (vertex) cut) *Let G be an interval graph with $V = \{v_1, \ldots, v_n\}$. Consider a point x such that $\min_i b_i < x < \max_i a_i$. If the end point immediately to the left of x is a right end point and the end point immediately to the right of x is a left end point, then $C(x)$ is called a minimal local (vertex) cut.*

Figure 3 gives examples of minimal local (vertex) cuts.



Local cuts are *C(4.5)={b}, C(7.5)={b}, C(10)={d, f}*

Figure 3. An example of an interval graph and minimal local (vertex) cuts

In the rest of the paper, minimal local (vertex) cuts will be called local cuts.

It is to be noted that minimal cuts are not dependent on the particular interval representation of G. However, this fact has no bearing on our problem. The following lemma and discussion will show that.

Lemma 3.2 *Any strong cut for an interval graph can be expressed as a union of local cuts.*

Proof. Let C be a local cut. Let $v_s \in C$. There exist v_i, v_j both adjacent to v_s that belongs to different components of $G - C$. Without loss of generality, it may be assumed that all intervals intersecting (b_i, a_j) belong to C. Choose v_p, v_q such that

1. v_p, v_q are either v_i, v_j or they intersect (b_i, a_j).
2. No interval begins or ends in (b_p, a_q).

Construct a cut C_s by taking vertices represented by intervals intersecting (b_p, a_q). Then C_s contains v_s and is a strong cut. Note that $C = \bigcup_{s \in C} C_s$.

Let C_1^G be the class of all possible subsets of vertex set of the graph G. Note that the integrity of graph G is the minimum of $\omega(X) + m_\omega(G - X)$ where X belongs to C_1^G. Let C_2^G be the class of all minimal local cuts of G.

Let C_1^G be the class of all minimal vertex cuts of the interval graph G. Let C_2^G be the class of all vertex cuts which can be expressed as a union of minimal local cuts and C_3^G be class of all possible vertex cuts of G. From Lemma 3.2 we know that $C_1^G \subseteq C_2^G \subseteq C_3^G$. Further, it follows from Lemma 3.1 that

$$\min_{X \in C_1^G} \{\omega(X) + m_\omega(G - X)\} = \min_{X \in C_3^G} \{\omega(X) + m_\omega(G - X)\}$$

Therefore,

$$\min_{X \in C_2^G} \{\omega(X) + m_\omega(G - X)\} = \min_{X \in C_3^G} \{\omega(X) + m_\omega(G - X)\}$$

Therefore we shall try to compute $\min_{X \in C_2^G} \{\omega(X) + m_\omega(G - X)\}$.

Algorithm 3.1 computes all local cuts of an interval graph G. It is to be noted that a naïve implementation of Algorithm 3.1 executes it in $O(n^2)$ time.

Algorithm 3.1

Algorithm Locally-minimal-cuts(G);

Input: An interval graph G in its interval representation.

Let $V(G) = \{v_1, \ldots, v_n\}$ with v_i represented by $[a_i, b_i]$ for $i = 1, \ldots, n$.

Further, let all a_i's and b_i's be distinct.

Output: Local cuts $C(\alpha_1), \cdots, C(\alpha_k)$

Data structure: end-points of type record containing two fields,
value: real and att $\in$ {left, right}

Step 1: Construct records of type end-points $x_1, \ldots, x_{2n}$ as follows,

for $i = 1, \ldots, n$ do

$x_{2i-1}.value \leftarrow a_i$; $x_{2i-1}.att \leftarrow left$;

$x_{2i}.value \leftarrow b_i$; $x_{2i}.att \leftarrow right$;

enddo

Step 2: Sort $x_1, \ldots, x_{2n}$ in non-decreasing order using value as key.

Let the sorted form be $y_1,\ldots,y_{2n}$.

Step 3: $k \leftarrow 0$; $i \leftarrow \min_{y_j.att=right} j$; $v \leftarrow y_i.value$

Step 4: for $i \leftarrow 2$ to $2n$ do

if $y_i.att = left$ and $y_{i-1}.att = right$ and $y_i.value > v$ then

$k \leftarrow k+1$;

$\alpha_k \leftarrow \dfrac{y_i.value + y_{i-1}.value}{2}$;

$C(\alpha_k) \leftarrow \{v_p \mid y_i.value \in (a_p, b_p]\}$;

endif

enddo

Now the problem of finding the weighted vertex integrity of an interval graph G reduces to finding a vertex cut X of G such that

1. X can be expressed as union of local cuts generated by Algorithm 3.1, and
2. $\omega(X)+m_\omega(G-X)$ is minimum among all vertex cuts of G satisfying 1.

Note that, Algorithm 3.1 generates a linear order among the vertex cuts that it computes. This order is important and will be used in our Algorithms later.

Let the cuts generated by Algorithm 3.1 be $C(\alpha_1),\cdots,C(\alpha_k)$ (in the order they are generated). Note that α_i's are sorted in non-decreasing order. Define $comp(i,j) = \{v_p \mid a_p > \alpha_i, b_p < \alpha_j\}$. Note that any connected component generated by a cut that can be expressed as the union of some of $C(\alpha_1),\cdots,C(\alpha_k)$ can be expressed in the form $comp(i,j)$ for some i,j. Further, note that $C(\alpha_i) - C(\alpha_{i+k}) = C(\alpha_i) - (C(\alpha_{i+k}) \cup C(\alpha_{i+k+j}))$ for all i,k,j.

To solve the weighted vertex integrity problem, we need to solve the following problem first.

Definition 3.6 (Restricted weighted vertex integrity) *Given an interval graph G, $C(\alpha_1),\cdots,C(\alpha_k)$ cuts of G generated by Algorithm 3.1, weight function $\omega: V \to R \geq 0$ and $\alpha_i \in R \geq 0$ for $i = 1,\ldots,k$, find a vertex cut X of G such that*

1. *X can be expressed as the union of some of $C(\alpha_1),\cdots,C(\alpha_k)$;*
2. *$m_\omega(G-X) \leq K$, and*
3. *for any X' satisfying 1 and 2, $\omega(X) \leq \omega(X')$*

Let us consider two local cuts $C(\alpha_i)$ and $C(\alpha_j)$. Let all intervals beginning after α_i and ending before α_j from a connected graph. If the weight of this

connected graph exceeds K, then at least one of $C(\alpha_{i+1}),\cdots,C(\alpha_{j-1})$ should be a subset of the vertex cut X. Let $A_{i,j}=\{C(\alpha_{i+1}),\cdots,C(\alpha_{j-1})\}$. Further, if $A_{i,j}\subseteq A_{i',j'}$, then any local cut that is an element of $A_{i,j}$ also belongs to $A_{i',j'}$. Therefore, we need to consider only those $A_{i,j}$'s such that, no $A_{i,j}$ is a subset of another, Obviously, there exists only $O(n)$ such $A_{i,j}$'s . Further, at most one $A_{i,j}$ begins with $C(\alpha_i)$, so we do not need two *subscripts* to describe $A_{i,j}$. We denote $A_{i,j}$ by A_i.

Clearly, the achieving cut X should be such that at least one local cut from each A_i should be a subset of X and the weight of X should be minimum, Formally the problem is described as follows, Given a set V, a positive weight function ω on V. $C(\alpha_1),\cdots,C(\alpha_r)$ subsets of V, $A_1,\ldots,A_k$ classes of sets $C(\alpha_1),\cdots,C(\alpha_r)$, find a class of sets $C(\alpha_1),\cdots,C(\alpha_r)$, A, such that, $A\bigcap A_i\neq\phi$ and the weight of the union of all sets belonging to A is minimum. The **NGWHSP** algorithm developed in the next section solves this problem. Algorithm 3.2 describes how we use that subroutine to solve the restricted weighted integrity problem.

Algorithm 3.2

Algorithm restricted-weighted-vertex-integrity(G, ω, $C_1,\cdots,C_k$, $\alpha_1,\cdots,\alpha_k$, K)

Input: G an interval graph;

Weight function $\omega: V(G)\rightarrow R\geq 0$

Cuts $C(\alpha_1),\cdots,C(\alpha_k)$ generated by Algorithm 3.1;

$K\in R\geq 0$, threshold;

Output: $C\subseteq V(G)$, solution of restricted weighted vertex integrity problem.

Step 1: $\alpha_0\leftarrow -\infty$; $\alpha_{k+1}\leftarrow\infty$;

$C(\alpha_0)\leftarrow 0$; $C(\alpha_{k+1})\leftarrow 0$;

$p\leftarrow 0$

Step 2: for $i\leftarrow 1$ to k do

Step 2a: for $j\leftarrow i$ to k do

Step 3: for $p'\leftarrow i$ to j do

if $C(\alpha_{p'})\subseteq C(\alpha_{i-1})\bigcup C(\alpha_{j+1})$ then

goto step 2a;

endif

enddo

Step 4: if $\omega(comp(i-1,j+1))>K$ then

$p\leftarrow p+1$;

$A_p \leftarrow \{C(\alpha_i),\cdots,C(\alpha_j)\}$;
goto step 2;
endif
enddo
enddo
Step 5: return **NGWHSP**($C_1,\cdots,C_k,A_1,\cdots,A_p,\omega$)

Note that Step 1 can be computed in $O(1)$ time. The amortized cost for Step 3 is $O(n^4 \log n)$. For Step 4, the total cost is $O(n^3)$. Step 5 takes time $O(n^3)$. Therefore, the time complexity of the algorithm is $O(n^4 \log n)$.

To find the weighted vertex integrity, we consider all possible $comp(i,j)$ as a possible achieving component. Then the restricted weighted vertex integrity algorithm is used to find a minimum weight cut such that the particular $comp(i,j)$ becomes the maximum weight component.

Algorithm 3.3
Algorithm weighted –vertex-integrity(G, ω, $C_1,\cdots,C_k$, $\alpha_1,\cdots,\alpha_k$)
Input: G is an interval graph;
Weight function $\omega: V(G) \rightarrow R \geq 0$;
Cuts $C_1,\cdots,C_k$ generated by Algorithm 3.1;
Representatives of cuts $\alpha_1,\cdots,\alpha_k$;
Output: $C \subseteq V(G)$, an achieving cut;
Step 1: $A\_cut \leftarrow C_1 \cup \cdots \cup C_k$;
max $v \leftarrow m_\omega(G - A\_cut)$;
int *egrity* $\leftarrow \max v + \omega(A\_cut)$;
$C_0 \leftarrow \phi$; $C_{k+1} \leftarrow \phi$;
$\alpha_0 \leftarrow -\infty$; $\alpha_{k+1} \leftarrow \infty$;
Step 2: for $i \leftarrow 0$ to k do
Step 2a: for $j \leftarrow i+1$ to $k+1$ do
Step 3: for $p \leftarrow i+1$ to $j-1$ do
if $C_p \subseteq C_i \cup C_j$ then
goto step 2a;
endif
enddo
Step 4: if $\omega(comp(i,j)) \geq \max v$ then
$t\_A\_cut \leftarrow C_i \cup C_j \cup$
restricted-weighted-vertex-integrity($G[-\infty,\alpha_i]$, ω,
$C_1 - C_i,\ldots,C_{i-1} - C_i,\alpha_1,\ldots,\alpha_{i-1}$, $\omega(comp(i,j))) \cup$

restricted-weighted-vertex-integrity$(G\lfloor\alpha_j,\infty\rfloor\omega,$

$C_{j+1}-C_j,\ldots,C_k-C_j,\alpha_{j+1},\ldots,\alpha_k,\ \omega(comp(i,j)))$;

if $\omega(t\_A\_cut)+\omega(comp(i,j))<$ int*egrity* then

$A\_cut \leftarrow t\_A\_cut$;

$integrity \leftarrow \omega(t\_A\_cut)+\omega(comp(i,j))$;

endif

endif

enddo

enddo

Step 1 can be computed in $O(n)$ time. The amortized cost of Step 3 is $O(n^4 \log n)$. Step 4 costs $O(n^6 \log n)$. Therefore, the time complexity of the algorithm is $O(n^6 \log n)$.

4. NGWHSP algorithm

The hitting set problem is the problem of finding a minimum cardinality set that has non-empty intersection with each of the sets in a given collection. Formally,

Definition 4.1 (Hitting Set Problem or HSP) *Given a collection of sets* $A_1,\ldots,A_k \subseteq U$, *find a set satisfying following conditions,*

(1). $B \cap A_i \neq \phi$ *for all* $i=1,\ldots,k$, *and*

(2). For all $C \subseteq U$ *satisfying (1),* $|B| \leq |C|$.

The following theorem about HSP is taken from [12].

Theorem 4.1 *Hitting set problem is NP-complete even when* $|A_i| \leq 2$ *for all* $i=1,\ldots,k$.

In an attempt to generalize HSP, we replace the set of atomic elements in HSP by a set of sets. The structure of a generalized HSP can be described as follows. Let U be a set. A non-negative weight function w is defined on U. Let $B_1,\ldots,B_n$ be subsets of U. Let $\{A_1,\ldots,A_k\}$ be a collection of sets, each of which is a subset of U. In other words, each A_i is a collection of some B_j's $(1 \leq j \leq n)$. The weight of A_i is defined as the weight of the set obtained by taking the union of all B_j's belonging to A_i. The generalized weighted hitting set problem can then be described as finding a minimum weight subset of $\{B_1,\ldots,B_n\}$ such that each has non-empty intersection with that set.

The motivation behind this generalization is as follows. The sets $B_1,\ldots,B_n$ are local cuts computed in section 3. The $A_{i,j}$'s described in section 3 are A_i's here. Obviously, the solution to the problem will give a vertex cut X such that there exists at least one local cut from each $A_{i,j}$ which is a subset of X and weight of X is minimum.

For convenience of further discussion, we have adopted index set notation, An index set I is subset of $\{1,\ldots,n\}$. Thus I denotes a subset of $\{B_1,\ldots,B_n\}$, viz., the set $\{B_i | i \in I\}$. Further, any subset of $\{B_1,\ldots,B_n\}$ has an index set. Define the weight of an index set I as the weight of the union of the B_i's such that $i \in I$. A formal definition of generalized weighted hitting set problem is the following:

Definition 4.2 (Generalized weighted hitting set problem or GWHSP) *Let U be a set, $w: U \to R \geq 0$ $A_1,\ldots,A_k \subseteq P(U)$, where $P(U)$ is the power set of U, and $A = \bigcup A_i = \{B_1,\ldots,B_n\}$. Find $I \subseteq \{1,\ldots,n\}$ satisfying following two conditions,*

(1). $I_i \cap I \neq \phi$ for all $i = 1,\ldots,k$, where $I_i = \{j | B_j \in A_i\}$, and

(2). $w(I) \leq w(I')$ for all I' satisfying (1), where $w(I)$ is defined by $w(I) = w(\bigcup_{i \in I} B_i)$.

An instance of GWHSP is denoted by $GWHSP(A_1,\ldots,A_k,A,U,w)$. Any I satisfying property (1) is called a covering of $A_1,\ldots,A_k$.

Clearly, GWHSP is NP-complete even when $|A_i| \leq 2$ and the A_i's are mutually disjoint. However, in this paper a polynomial time algorithm is presented fore a class of GWHSP, called Nice GWHDP or NGWHSP.

Definition 4.3 *The collection of sets $A_1,\ldots,A_k \subseteq A$ is said to satisfy the consecutive retrieval property if the members of A can be linearly ordered in such a way that all members of A_i occur consecutively in the linear order for all $i = 1,\ldots,k$. The corresponding linear order is called a consecutive ordering of A.*

Definition 4.4 (Nice GWHSP or NGWHSP) *$GWHSP(A_1,\ldots,A_k,A,U,w)$ is called nice if*

1. *$A_1,\ldots,A_k \subseteq A$ satisfies the consecutive retrieval property, and*
2. *if $B_1,\ldots,B_n$ is a consecutive ordering of A, then $B_i \cap B_j \subseteq B_k$ for all $i \leq k \leq j$.*

Without loss of generality, we assume that $A_i \not\subseteq A_j$ for $1 \le i \ne j \le k$, because if $A_i \subseteq A_j$ for some $i \ne j$, then solving GWHSP over the class $\{A_1,\ldots,A_{j-1},A_{j+1},\ldots,A_k\}$ is equivalent to solving GWHSP over the class $\{A_1,\ldots,A_k\}$.

Let us assume $|A| = n$ and $B_1,\ldots,B_n$ is the consecutive ordering of A. Let $left(A_i)$ be the index of the first member of A_i in this ordering and right $right(A_i)$ be the index of the last element of A_i. Without loss of generality, we can assume that $A_1,\ldots,A_k$ are arranged in increasing order of $left(A_i)$'s. In other words, we assume that $left(A_1) < \cdots < left(A_k)$. Further, define $be(B_i) = \max_{right(A_j)<i} j$.

Definition 4.5 *Given a NGWHSP* $(A_1,\ldots,A_k,A,U,w)$, a restriction of a set A_i *by* $B_j \in A$ *is defined as*

$$A_i / B_j = \{B - B_j | B \in A_i\}$$

Definition 4.6 *Given a NGWHSP* $(A_1,\ldots,A_k,A,U,w)$, a restricted *NGWHSP is defined by*

$$w(I / B_j) = w(\cup_{i \in I} B_i - B_j)$$

Note that if $j > right(A_i)$ then *NGWHSP* $[j,i]$ is a nice *GWHSP*. Let $B_{n+1} = \phi$ and let us extend *NGWHSP* $(A_1,\ldots,A_k,A,U,w)$ by replacing A with $A \cup \{B_{n+1}\}$. Let S_i^j be a solution of *NGWHSP* $[j,i]$ $(A_1,\ldots,A_k,A,U,w)$ for $j = right(A_i)+1,\ldots,n+1$, for $i = 0,\ldots,k$. Then the following recursive relation leads to a polynomial time algorithm for *NGWHSP* $(A_1,\ldots,A_k,A,U,w)$.

$$S_0^j = \phi \text{ for } j = 0,\ldots,n+1.$$

$$S_i^j = \{p\} \cup S_{be(B_p)}^p$$

where,

$$left(A_i) \le p \le right(A_i)$$

and

$$w(\{p\} \cup S_{be(B_p)}^p / B_j) = \min_{left(A_i) \le r \le right(A_i)} w(\{r\} \cup S_{be(B_r)}^r / B_j)$$

for $i = 1,\ldots,k$ and $j = right(A_i)+1,\ldots,n+1$.

Clearly S_0^j contains a solution of $NGWHSP[j,0](A_1,\ldots,A_k,A,U,w)$. Let us assume S_i^j contains the solution of $NGWHSP[j,i](A_1,\ldots,A_k,A,U,w)$ for $i=1,\ldots,p$ and $j=right(A_i)+1,\ldots,n+1$. We need to show that same holds for S_{p+1}^j for $j=right(A_i)+1,\ldots,n+1$. Let I' be a solution of $NGWHSP[j,p+1](A_1,\ldots,A_k,A,U,w)$. Let β be the minimum index in I' such that $B_\beta \in A_{p+1}$.

Need to show that $w(I'/B_j) \geq w(S_{p+1}^j/B_j)$. Note that $J \cup \{\beta\} \subseteq I'$. Therefore,

$$w(I'/B_j) \geq w(J \cup \{\beta\}/B_j) \tag{1}$$

Further,

$$\begin{aligned} w(J \cup \{\beta\}/B_j) &= w(B_\beta \cup (\cup_{p\in J} B_p) - B_j) \\ &= w((\cup_{p\in J} B_p - B_\beta - B_j) \cup (B_\beta - B_j)) \\ &= w((\cup_{p\in J} B_p - B_\beta) \cup (B_\beta - B_j)) \\ &= w(\cup_{p\in J} B_p - B_\beta) + w(B_\beta - B_j) \\ &= w(J/B_\beta) + w(B_\beta - B_j) \end{aligned} \tag{2}$$

Note that, J covers $A_1,\ldots,A_{be(B_\beta)}$. Therefore,

$$w(J/B_\beta) \geq w(S_{be(B_\beta)}^\beta/B_\beta) \tag{3}$$

Therefore,

$$\begin{aligned} w(J \cup \{\beta\}/B_j) &\geq w(S_{be(B_\beta)}^\beta/B_\beta) + w(B_\beta - B_j) \\ &= w(\{\beta\} \cup S_{be(B_\beta)}^\beta/B_j) \\ &\geq w(S_{p+1}^j/B_j) \end{aligned} \tag{4}$$

To write the algorithm, the following variables are required besides those already defined.

1. *W(i, j)* containing $w(B_i \cap B_j)$.
2. *SW(i, j)* containing $w(S_i^j/B_j)$.

Algorithm 4.1

Algorithm NGWHSP $(B_1,\ldots,B_n,A_1,\ldots,A_k,w)$;

Input: $B_1,\ldots,B_n$ a collection of sets;

$A_1,\ldots,A_k \subseteq \{B_1,\ldots,B_n\}$;

$(A_1,\ldots,A_k,\{B_1,\ldots,B_n\},\cup_{i=1}^n B_i,w)$

forms an instance of nice GWHSP.

Output: S_k^{n+1}, a solution of
NGWHSP$\left(A_1,\ldots,A_k,\{B_1,\ldots,B_n\},\bigcup_{i=1}^{n}B_i,w\right)$.

Step 1: Initialize *W(i,j)* for all *i.j*.
Initialize *SW(0,j)* by 0 for all *j*;
Initialize S_0^j by ϕ for all *j*;

Step 2: for $i \leftarrow 1$ to k do
for $j \leftarrow right(A_i)+1$ to n+1 do
$SW(i,j) \leftarrow \infty$;
for $r \leftarrow left(A_i)$ to $right(A_i)$ do
Step 3: $t\_weight \leftarrow w(B_r)-W(r,j)+SW(be(B_r),B_r)$;
if $t\_weight < SW(i,j)$ then
$SW(i,j) \leftarrow t\_weight$;
Step 4: $S_i^j \leftarrow \{r\}\bigcup S^r_{be(B_r)}$;
endif
enddo
enddo
enddo

Step 1 can be performed in $O(\left|\bigcup_{i=1}^{n}B_i\right|n^2)$ time. Under the assumption that $\left|\bigcup_{i=1}^{n}B_i\right|=O(n)$, step 1 can be performed in $O(n^3)$ time. Steps 3 and 4 can be computed in *O(1)* time. Hence, the body of the algorithm can be performed in $O(kn^2)$ time. If $k=O(n)$, then the time complexity of the algorithm is $O(n^3)$.

For an example about NGWHSP algorithm, refer to our technical report [15].

5. Conclusion

The main contribution of this paper is a polynomial time algorithm for computing the weighted integrity of interval graphs. We have discussed how the algorithm may be used for computing the vulnerability of some real life networks and how the weighted integrity may be useful in the area of soft real time computation.

One of the important features of the weighted integrity is that the measure may be used to incorporate the subjective measure of reliability of the nodes in a measure of the network vulnerability. It happens too often that system managers have some idea about the reliability of each system but neither any idea nor any

measure of the probability of failure. Yet most of the classical research on reliability is based on the probabilities of failure of individual nodes.

However, a lot more work needs to be done in this area. One important possible future direction of research is how to determine the weights of the nodes effectively. Further, the weighted integrity problem is NP-complete for most of the commonly known network topologies. Therefore, it is important to find approximation algorithm. Though the weighted integrity problem is NP-complete in strong sense for general graphs, it is not the case for trees and some other commonly used network topologies. Therefore, it may be possible to find pseudo-polynomial algorithms for those special classes.

Some interesting investigation may be carried out from graph theoretical standpoint also. The discussion in Section 2 shows that the weighted integrity problem may be polynomial for the class of cocomparability graphs. In this paper we have shown that it is so for interval graphs, a subclass of that class. However, the problem remains open for the whole class of cocomparability graphs. Even if the problem is hard for this family, it may be answered relatively easily for another important subclass, permutation graphs.

References

[1] C. A. Barefoot, R. C. Entringer and H. Swart, Integrity of trees and powers of cycles. *Congressus Numerantium* 58(1987), 103-114.

[2] C. A. Barefoot, R. C. Entringer and H. Swart, Vulnerability in graphs – a comparative survey, *J. Combin. Math. Combin. Comput.* 1(1987), 13-22.

[3] S. Ray and J. S. Deogun, Computational Complexity of Weighted Integrity, *J. Combin. Math. Combin. Comput.* 16(1994) pp. 65-73.

[4] K. S. Bagga and J. S. Deogun, A variation on the edge integrity, *Congress. Numer.* 91(1992), 207-211.

[5] L. W. Beineke et al., The integrity of the cube is small, *J. Combin. Math. Combin. Comput.* 9(1991).

[6] K. S. Bagga, L. W. Beineke, W. D. Goddard, M. J. Lipman, and R. E. Pippert, A survey of integrity, *Discrete Appl. Math.* 37/38(1992), 13-28.

[7] L. H. Clark, R. C. Entringer and M. R. fellows, Computational complexity of integrity *J. Combin. Math. Combin. Comput.* 2(1987), 179-191

[8] K. S. Bagga et al., On the honesty of graph complements, *Discrete Math.*, 122(1993), no. 1-3, 1-6.

[9] K. S. Bagga et al., Some bounds and an algorithm for the edge-integrity of trees, *Ars Combinatoria* 35(1993), A, 225-238.

[10] M.C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press, New York. 1980.

[11] M. R. Fellows and S. Stueckle, The immersion order, forbidden subgraphs and the complexity of network integrity, *J. Combin. Math. Combin.*

Computing. 6(1989), 22-32.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Fransisco, 1979.

[13] W. Goddard, On the Vulnerability of Graphs, Ph.D. Thesis, Univ. of Natal, Durban, South Africa, 1989.

[14] W. Goddard and H. C. Swart, On the integrity of combinations of graphs, *J. Combin. Math. Combin. Computing*. 4(1988), 3-18.

[15] S. Ray, R. Kannan, D. Zhang and H. Jiang, The Weighted Integrity Problem is Polynomial for Interval Graphs, Technical Report, Department of Computer Science, University of Alabama, TR-2004-01.